





## SFoG-RPI: A Secured QoS Aware and Load Balancing Framework for FoG Computing in Healthcare Paradigm



Geetha Gunasekar<sup>1</sup>, Anand Krishnamurthy<sup>2</sup>, Tamilvizhi Thanarajan<sup>3</sup>, Surendran Rajendran<sup>4\*</sup>

<sup>1</sup> Department of Networking and Communications, School of Computing, SRM Institute of Science and Technology, Kattankulathur 603203, India

<sup>2</sup> Department of Computer Science and Engineering, Rajalakshmi Engineering College, Chennai 602105, India

<sup>3</sup> Department of Computer Science and Engineering, Panimalar Engineering College, Chennai 600123, India

<sup>4</sup> Department of Computer Science and Engineering, Saveetha School of Engineering, Saveetha Institute of Medical and Technical Sciences, Chennai 602105, India

Corresponding Author Email: [surendran.phd.it@gmail.com](mailto:surendran.phd.it@gmail.com)

<https://doi.org/10.18280/ria.370403>

### ABSTRACT

**Received:** 26 April 2023

**Revised:** 23 July 2023

**Accepted:** 1 August 2023

**Available online:** 31 August 2023

#### Keywords:

*IoT, fog, cloud, Raspberry Pi, security, data compression and encryption*

Diabetes, characterized by persistently high blood glucose levels, has been identified as a hazardous health condition, potentially leading to severe complications such as heart attacks, strokes, and heart failure. This study introduces a fog-based remote health monitoring system designed to mitigate the devastating impacts of diabetes and hypoglycemia. This system persistently monitors health parameters including glucose levels, carbohydrate intake, physical activities, heart rate, and blood pressure. It additionally supports advanced services such as feature extraction, distributed local storage, and enhanced security. The traditional cloud-based architecture, while effective, often results in significant latency due to the processing of vast amounts of data. By bringing computing servers closer to users, Fog computing addresses this issue, reducing latency, and increasing security, resource accessibility, and on-demand scaling. In this context, the proposed system aims to minimize latency and network usage while addressing critical issues such as security, access control, and privacy. It employs lossy data compression at the gateway level to decrease network bandwidth and enhance efficiency. Furthermore, the system introduces a novel Load Balancing mechanism to distribute the load among fog nodes evenly. It utilizes lightweight cryptographic algorithms, efficient key exchange protocols, and digital signatures to ensure confidentiality, authentication, and user privacy. The performance of the proposed framework was evaluated in terms of average processing time, energy consumption management, computational resource distribution, latency, and network usage. When compared with other systems, the proposed framework demonstrated superior results, thus validating its effectiveness.

## 1. INTRODUCTION

The Internet of Things (IoT) paradigm has been increasingly adopted in recent years, finding applications in diverse fields such as healthcare, smart agriculture, industrial automation, and the conception of smart cities. Traditional IoT applications directly oversee the operations of these systems, which include data collection, event detection, analysis, and actuation [1]. However, cloud data centers, which are predominantly used in these applications, are often situated several hops away from the IoT-enabled applications, posing certain challenges. Edge and Fog models have been introduced to address this issue by bringing computational capabilities closer to the applications [2]. As a result, the responsiveness of applications and the time taken for service delivery are substantially improved. Additionally, these models aid in mitigating network congestion by preventing the transfer of large data quantities to cloud data centers. In this context, the present study sets out to further explore the potential of Edge and Fog models in enhancing the efficiency and effectiveness of IoT applications across various domains.

IoT is a platform to connect many things to the internet, and

these things should be integrated with both electronics software and sensors, which generates and exchange data across things. IoT allows humans to contribute, interact and collaborate with the things around us. IoT utilizes the resources efficiently which are available, resources may be in terms of inputs, monetization, etc. The time-sensitive data must be processed and analyzed very quickly and hence it should have happened very near to the devices. Time insensitive data will be analyzed in the cloud [3]. It reduces human efforts and time and this is why the Internet of Things has become more popular and also its ever-growing technology. IoT has certain features: connect the things to the platform, collect and analyze data for business, and integrate all the models for an improved experience. IoT Architecture consists of five layers namely: Perception, transport, processing, application, and business layer [4].

Each layer is intended to perform its assigned task, the perception layer obtains data about the atmosphere and sends it to the transport layer, in turn forwards data to the different layers using a network such as BLE, Zigbee, 6LowPan, Wi-Fi, etc. Processing layer stores, processes and analyzes the sensor data and also employs cloud, fog, and big data analytics

modules to accomplish the task. Next, we have application layers that are responsible for delivering the application-specific services to the end-users. Finally, the business layer which accomplishes the complete IoT organization such as applications, business models, user's privacy, etc.

Cloud computing is already well-known due to its extensive adoption in its target market. During the early phases of its adoption, though, it encountered various challenges like security, privacy, attacks, latency, and bandwidth. As a result, the cloud computing paradigm's primary challenges are privacy and security.

The IoT technology incorporates a fog computing paradigm towards processing power and computing close to the things that are connected. Cloud computing is the method of storing, managing, and processing data on a remote computer contained on the Internet rather a native server or a local system. Fog computing also called edge computing supports computation, storage, and networking capabilities be communal between cloud computing data centers and end devices.

Both fog and cloud deliver identical resources comprising computation, memory, storage, and infrastructure, yet they complement one another [5, 6]. As a result, the fog has evolved into a middleware capable of responding quickly and avoiding redundant data to be sent to the network and other healthcare IoTs, and ensuring that healthcare services have the necessary privacy protections. Their geographical position is a significant distinction. In contrast to the cloud, it offers more general functions and also fog is close to the IoT nodes. These devices will produce big quantities of raw data (for example, from sensors), and rather than sending all of this data to cloud-based servers to be managed, the idea behindhand fog computing is to do most of the processing work as possible by the computing units located near the data-generating devices, which results in processed data will be forwarded to the cloud rather than raw data being forwarded and due to this cause bandwidth requirement being greatly reduced. Another advantage is that the processed data may be required by the same devices that created it, which reduces the latency between input and response time by processing locally rather than remotely.

FoG architecture implemented in IoT gateway consists of security, storage, pre-processing, and monitoring layer deployed between physical and transport layer. The underlying nodes send raw and unstructured data to the fog. The monitoring layer then takes care of the data that has been received. Following that, the data is directed to the preprocessing layer, which structures the unstructured data using various data filtering and transformation techniques. At the middleware level, data storage necessitates privacy and security protections. Such duties are handled by the local storage layer. At this layer, lightweight security procedures and data integrity measures must be implemented. The temporary storage layer then stores the data for a set length of time.

As a result, data is not stored for an extended amount of time. If the data is sensitive, it is eventually securely transported to the cloud. Cloud computing can store data for prolonged periods and conduct computation-intensive activities. Fog device security and privacy is a critical concern that should be considered in Fog devices. We offer a simple, yet general, paradigm for deploying QoS-aware IoT applications on Fog infrastructures in the healthcare domain.

The QoS metrics we consider in our systems are latency,

bandwidth, loss, and jitter. The QoS profiles  $Q$  are made up of the set of pairs  $(l_i, b_i, j_i, lo_i)$  where  $l_i, b_i, j_i, lo_i$  signify the average latency, bandwidth, jitter, and loss of a communication channel, respectively. The bandwidth of a link is represented by a pair  $(b_{upward}, b_{downward})$ , which distinguishes the download and upload bandwidth, and  $\pi$  will be used to represent latency or bandwidth metrics that are unknown or unspecified.

Congestion on the network and route changes are the reason for jitter. Acceptable jitter is the smallest delay in the transmission that we are ready to allow. Milliseconds are used to measure jitter (ms). A delay of 30 milliseconds or longer might cause call distortion and disruption. Opted to prioritize packets over other types of traffic using the quality of service (QoS) setting on our router. If the jitter is caused by traffic congestion, prioritizing packets may be beneficial. Using a jitter buffer is one of the most effective techniques to reduce internet jitter. On a VoIP system, a jitter buffer is a useful component. They work by delaying and storing voice packets as they arrive. Before delivering traffic to the receiver, they buffer it for 30 to 200 milliseconds.

This procedure ensures that data packets arrive in the correct order and with little delay. They can also reorder data packets according to when they were transmitted in some cases, depending on the buffer. When sending data over the internet, there is always the possibility that packets of data will be lost or damaged. To guarantee easier and faster transfers, the packets are frequently encrypted and decreased in size. Your communication will be delayed and sometimes garbled if these packets are missing during transit. Prioritizing traffic to guarantee that the maximum critical data is sent over the network first would improve data flow and reduce congestion.

## 2. LITERATURE SURVEY

FogBus framework practices a group of IoT devices, Fog nodes, and Cloud infrastructure. For task distributions, it solely uses the master-slave method. FogBus is a bare-metal application that does not allow containerization. Nguyen et al a fog-based market-based framework that provides diverse resources to applications based on budget limitations to overcome the resource allocation problem, uses a centralized approach.

In Foggy computing environments [7], the Foggy framework allows for dynamic resource supply and application placement. System isolation and overhead management are aided by the Foggy framework's container-based virtualization. SmartFog is a framework that makes big data processing in Fog settings easier. It extracts and analyses data from wearable IoT devices using machine learning algorithms [8]. SmartFog is a gateway that preserves a continuous link with cloud datacenters.

FogPlan is a framework for dynamically deploying and releasing Fog infrastructure applications depending on latency limitations and resource accessibility [9]. It emphasizes on Quality of Service needs and uses a couple of greedy algorithms towards the reduction of total cost of application execution. The current fog computing literature is divided into three categories: (1) QoS, is supposed to quantify the results of various units within the fog architecture, such as mobile operators; (2) the QoE, which is thought to be used to depict the end-user experience on top of the QoS; (3) A variety of apps from the mobile, cloud, and IoT based contexts were

investigated for security, privacy, and access control problems [10].

This work doesn't allow for sharing of resources between numerous RPIs via containers, and also has security issues [11]. The proposed SFog-RPI addresses these issues by providing APIs that could help consumers with computation infrastructure and external devices in a safe, simple manner. EHR was created and implemented employing digital signatures and file encryption in our work. Not only are digital signatures and file encryption used to tackle the problems of security related concerns but they're correspondingly working to avoid file change and illegal access [12].

Mahmud and Toosi [13] developed a framework for an IoT-enabled healthcare system, highlighting smart gateway architecture for data storage and processing on the local level. In this paradigm, the cloud turns into the backend database system for data processing and analysis. An orchestrator coordinates the deployment of applications on the computational nodes in Ad-Hoc Edge. Ad-Hoc Edge additionally maintains a logical registry to keep track of the system's available edge devices [14].

Riya et al. [15] offer an IoT system for e-health monitoring that includes a smart gateway. The gateway offers interoperability with Bluetooth Low Energy (BLE), Wi-Fi, and IPv6 (6LoWPAN) over LowPower Wireless Personal Area Networks. Countless innovative capabilities, like data storage, data compression, and security, are available through the gateway. These studies [16] explore security issues for IoT systems that monitor health. Lightweight cryptography algorithms safeguard the link between sensor nodes and smart gateways. The sensor node's energy consumption, on the other hand, is not taken into account. The authors [17] suggest an IoT system for continuous glucose monitoring that uses the fog computing technique. For data processing and analysis, the system employs a mobile-based gateway.

### 3. PROPOSED METHODOLOGY

In our proposed work we have implemented Raspberry Pi as the fog node since it has a processor, memory, storage, and display devices and also it is less costly. It also has an input

and output port that can be easily connected to sensors and actuators very easily. Raspberry Pi is typically equipped with moderately fast processors that facilitate peripheral interface and networking. However, the Raspberry Pi is limited in its ability to support multi-tenancy and resource sharing. Hence, we need to perform virtualization of RPIs but it does not allow us to do hypervisor-based virtualization, in this case, containerization of RPIs will be the best possible solution.

The privacy and security issues surrounding the capture and transfer of healthcare data first. Then, to address security problems similar to those raised above, a secure data gathering technique for IoT-based healthcare systems.

### 3.1 Fog architecture

Fog architecture consists of fog nodes, IoT devices, and Cloud nodes and it can be denoted as a tuple (F,I,C), where F is a group of fog nodes, and each node will be represented as  $f(i=1,2,3...n)=(\epsilon, \pi, \mu)$ , where  $\epsilon$  denotes the location of fog node,  $\pi$  refer to software and hardware specification of the fog node,  $\mu$  refers to the IoT devices directly connects with this fog node. I is a collection of Things, each of which is represented by the tuple  $I(i=1,2,3...n)=(t, \epsilon)$  t identifies its type and  $\epsilon$  is the location of the device. C is a collection of existing Cloud data centers, all of which is identified by the tuple  $f(i=1,2,3...n)=(\epsilon, \pi)$ , where  $\epsilon$  defines its location and  $\pi$  software and hardware specification of the cloud data centers  $\mu$  in I are directly connected to the fog node, hence it has negligible latency and sufficient bandwidth. Cloud nodes connect with sensors and actuators using some API's which supports QoS metrics.

A Fog computing infrastructure application can be assumed as a collection of autonomously deployable components that must work composed while meeting particular QoS requirements. An application is represented as a triple  $[\xi, \beta, \mu]$ ,  $\xi$  where is a group of software components,  $\beta$  denotes existing interactions among components in,  $\mu$  and is a group of Internet of Things requests. Fog architecture is represented as (F, I, C). Figure 1 illustrates the Fog-Cloud infrastructure, in which three fog nodes and two data centers have been deployed. It also depicts the average latency and bandwidth of each link.

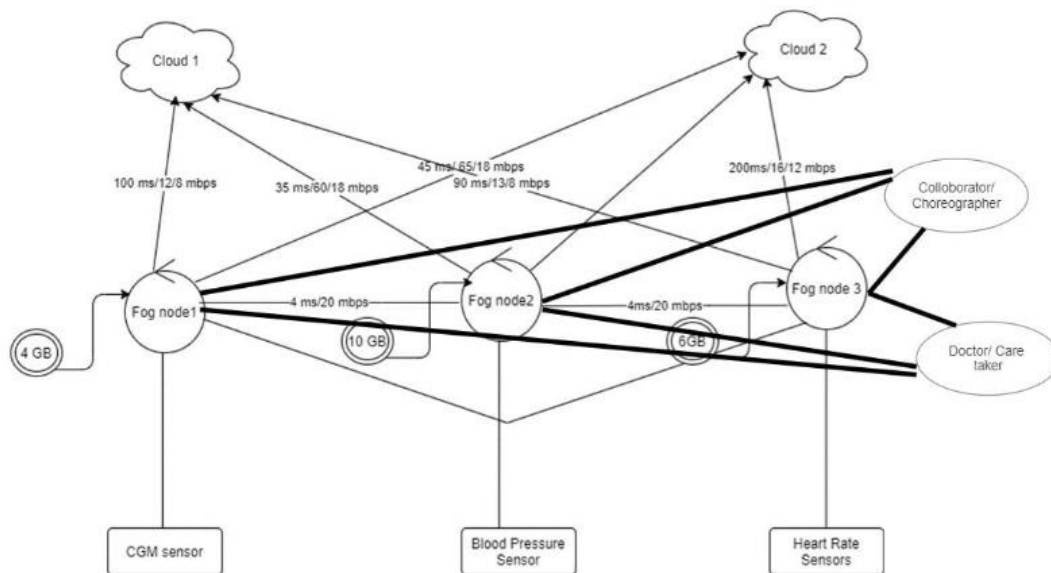


Figure 1. FoG-cloud communication systems

### 3.2 Collaboration between FoG nodes using choreography

One fog node wouldn't be sufficient for some sophisticated queries. However, due to latency or other difficulties, forwarding multiple requests to the cloud may not be a practical answer. In this circumstance, fog resource centers that are co-located with the primary fog can be merged with the primary fog to enable the inter-fog scenario. It will be difficult to combine numerous fogs in cost-effective collaborative-fog scenarios that meet service level agreements (SLAs) and are resource-efficient. Using choreography concept inter-fog communication could be established, it allows the fog to exchange the information.

### 3.3 Higher level organization of SFog-RPis framework

Integrated IoT-Fog-Cloud environment using a high-level detail of SFog-RPis is shown in Figure 2. The SFog-RPis framework is made up of many different components, the items that follow: IoT devices, Fog Gateways, Fog Agent, Fog Computing Nodes, Fog Database, and Cloud Datacenters.

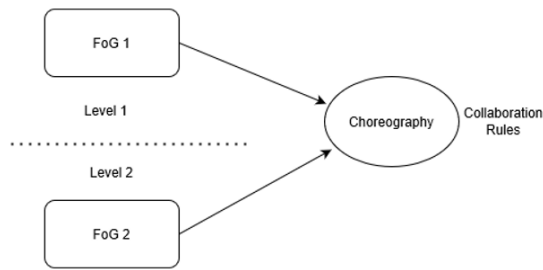


Figure 2. Inter-fog collaboration using choreography

Sensors and devices will use different protocols depending on the kind of facility, data transfer rate as well as technology used, and as an outcome, certain of them might not be capable to pad right into another software interface. Interoperability becomes a huge challenge when there are so several devices and sensors made by so many different developers working together. Protocol conversion, channeling, and other similar events will be required in the fog in this instance. In these situations, fog must be delivered using conventional techniques to enable seamless integration and interoperability.

This software component supports the collecting of generated data for every sensor node linked to the host RPi through API. It can begin sending data to the microservice as quickly as the container is launched, or it can pause till the microservice acts together. The associated sensors have two modes of operation: pull and push. Microservices receive sensor data from Sensor Manager only in the pull mode, in the push mode, Sensor Manager provides collected data to the Services autonomously.

### 3.4 FoG computing node and agent node

Fog Gateway Nodes filter data and convert it into standardized style. They are forwarded to another computer instances in the integrated environment by FGNs. It gives customers the ability to set login credentials, get service responses, control IoT devices, and request resources using application user interfaces. In addition to the standard functions of collecting and transferring data, Fogas-assisted smart gateways offer a variety of sophisticated services aimed

at improving the quality of healthcare. Distributed local storage, data compression, and other Fog services.

If the Fog Computing Node is unable to meet the application's requirements on its own, it acts as an agent node, contacting other Fog Computing Node and Cloud data centers on behalf of the Fog Gateway Nodes to obtain the resources needed to run the back-end application. It distributes computing workloads over several Fog Computing Nodes and monitors, synchronizes, and coordinates their actions in this situation. Concerning capacity and resource architecture, Fog Computing Nodes are diverse. To undertake various SFog-RPI functions, it consists of processing cores, memory, and storage. It provides suitable security characteristics to such broker nodes, allowing them to assure the stability of communication and data exchange between Fog Nodes and Cloud.

The database nodes keep track of diverse applications' meta-data, such as the application model, runtime needs, and dependencies [18, 19]. Furthermore, during application execution, these nodes can save some intermediate data, allowing data processing to resume from any anomaly-driven stop point.

### 3.5 Cloud nodes

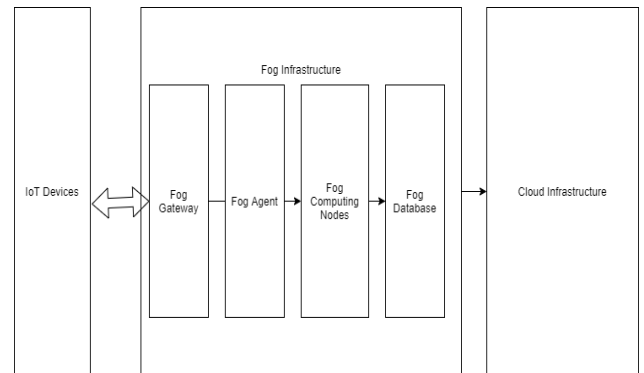


Figure 3. Higher level organization of SFog-RPis framework

Cloud servers offer many advantages, including centralized scalability, data protection, global storage, and processing. Heavy computational activities that cannot be handled by Fog could be handled efficiently by Cloud servers. The Figure 3 depicts how the Raspberry Pi integrated with various sensor nodes and external devices and how this does the job of the Fog node. Raspberry Pi can perform collecting data from sensor nodes, integrating collected data from various sensor nodes, process and analyze the data into meaningful information. We have implemented Raspberry Pi as the fog node since it has a processor, memory, storage, and display devices and it is less costly. Since for a real-time health monitoring system, during an emergency, the patients must be treated immediately, hence processing and analyzing the data very close to the sensing node is necessary. In this situation, if the Fog node does not perform this task, then processing must be done at cloud data centers, and responding to the patients in the critical situation takes lots of time which leads to life-critical. Therefore, Raspberry pi implemented in our proposed system acts as the Fog node, which provides immediate response to the patients in an emergency.

SFog-RPi Controller contains conventional algorithms and strategies for managing an RPi's computational resources. It also communicates with the information registry to obtain other RPis' references for resource and data sharing.

Information Manager sends commands to RPi's Operative to govern the execution of microservices formed on the results of connected algorithms and policies.

RPi's Operator component is in charge of container creation and termination. RPi's Operator uses APIs to inject essential arguments into containers, such as location of the microservice and the unique identifier of the RPi hosting the RPi Service. It also prevents the Sensor devices from forwarding data for newly dismissed containers and keeps informed the Information Registry's executor.

The context information extracted from the host RPi includes resource configuration, network bandwidth, remaining battery lifetime, storage, and current memory, location, and CPU utilization. Information Monitor provides such information to SFog-RPi Controller and participates in resource management decisions based on Information Manager's requirements.

Periodic Signal Generator: It produces the heartbeat data signals periodically to Information Registry of SFog-RPi Controller to alert the occurrence of the host RPi. Periodic Signal Generator additionally notifies the corresponding SFog-RPi Controller of the status of microservice execution to track run-time performance. It aids in the detection of RPi's that have suddenly failed. Figure 4 represents the Raspberry Pi integrated with various sensors.

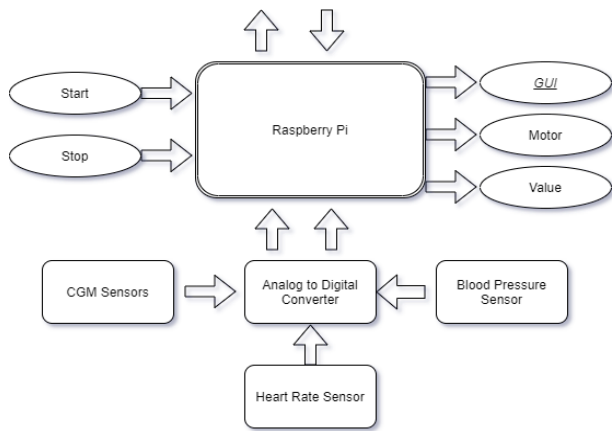


Figure 4. Raspberry Pi integrated with various sensors

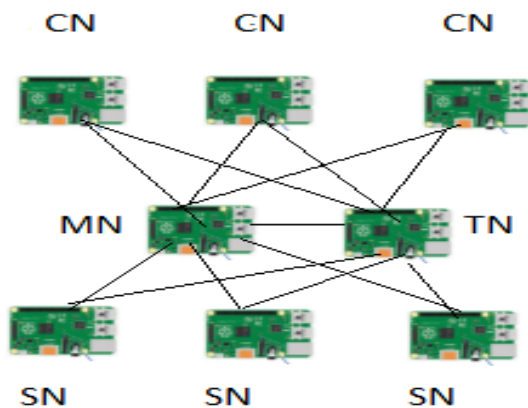


Figure 5. Setup for SFoG-Rpi

The data is sent to a master node via the sensor node, in turn it is distributed by the master node to the number of worker nodes on which the processing is actually to be carried out. Figure 5 shows the Setup for SFoG-Rpi. Parallely, a traffic

monitor observes the data rate of IoT devices based on which amount of data to be processed will be computed and assigned to the respective computing node. CN does the encryption process using a private key and transfers it to the FoG node for processing.

The strong traffic load reveals that dataflow latency is the primary cause of overall response time. There are two types of latency in data flow on IoT devices: first one is communication latency triggered by traffic load, and another is computing latency induced by computing load. We require a load balancing technique to stabilize together the computing and traffic loads of BSs or fog nodes in a direction to lessen the system's total latency and network use [20].

CN-Computing Node, SN-Sensing Node, MN-Master node, TN-Traffic Node. Let  $F_i=(f_1,f_2,f_3\dots f_n)$  be the set of FoG nodes,  $t(x)$  be the transmission power of sensor node,  $c(x)$  is the channel gain of sensor nodes,  $\lambda(x)$  be the arrival rate,  $s(x)$  be the traffic size,  $c_i(x)$  be the capacity of the sensor node,  $c_j(x)$  is the FoG node computing capacity,  $y(x)$  data flow computation size,  $C_{lr}$  is the computational latency ratio,  $CM$  is the communication latency,  $TL$  is the traffic load at the FoG node,  $CL$  is the computational load at the FoG node,  $tr(x)$  is the traffic load density of sensor node,  $ld(x)$  is the computing load density of sensor node.

In Traffic Load and Load Computation, let us assume that, a set of sensor nodes are implanted on the body of the patient and it has the transmission power of  $t(x)$  with a gain of  $c(x)$ , the noise is represented as  $\alpha 2$ . The signal-to-noise ratio is calculated by:

$$STNR(x)=\frac{t(x)Xc(x)}{\alpha 2} \quad (1)$$

where, gain  $c(x)=10\log_{10}\left[\frac{\phi 2}{(4\pi D)^2}\right]$ ,  $\phi$  is the wavelength and  $D$  is the distance between the sensor nodes.

$tr(x)$  is the traffic load density of the sensor node which will be calculated by:

$$tr(x)=\frac{a(x) \times s(x)}{c_i(x)} \quad (2)$$

$TL$  is the traffic load at the FoG node will be calculated by:

$$TL=\sum tr(x) \quad (3)$$

$CM$  is the communication latency will be calculated by:

$$CM=\frac{TL}{1-TL} \quad (4)$$

FoG node Computing latency is directly relative to the latency of the data flow, let  $ld(x)$  is the computing load density of sensor node can be calculated by:

$$ld(x)=\frac{a(x) \times y(x) \times b_j(x)}{c_j(x)} \quad (5)$$

where,  $b_j(x)$  is the binary indicator of the FoG node.

$CL$ -computing load at the FoG node can be calculated by:

$$C_L=\sum ld(x) \quad (6)$$

$C_{lr}$  is the computing latency ratio is calculated by:

$$C_{lr}=(1-C_L)/C_L \quad (7)$$



Patients' confidential information, such as disease type, age, surgery is undergone, history, and so on, is stored in healthcare data. Unauthorized users must not have access to any of this information. With the proliferation of technology in different domains related to healthcare, guaranteeing data integrity, privacy, and security leftovers is a concern [21]. Security in storage and transmission are two distinct issues. The level of data access authorization is context-dependent. For instance, the data can be vital to a nurse in one situation, but the authorization must be confined to the doctor in another.

In the event of an emergency, paramedics may be given provisional access to the patient's private data, although, under other circumstances, same admission may be restricted to doctors exclusively. Fog must be conscious of these limits and to ensure the data it obtains/interconnects with other devices and systems is secure and private.

Fog, a middleware, can show a significant part in bringing multiple parties together to provide robust healthcare services. For example, a pharmacy may preserve a patient's medication record to see if the new medicine the patient is purchasing is compatible with the other medications she or he is already taking for the same or different health concerns. In Lightweight Secured Fog-based framework, nowadays all the PHR data will be stored on the cloud server remotely, which causes security-related issues like privacy, security, and access control. Fog computing techniques relieve the computational and communication stress on remote cloud servers and act as a middleware amongst cloud servers and IoT devices. Fog computing mainly focuses on the quality-of-service (QoS) metrics [22]. Most of the healthcare monitoring systems do not support security-related concerns, hence in this work, we are focusing on security issues also. To sort out these issues, the most direct solution is to use encryption and decryption techniques.

Because the sensor nodes are small and have limited resources, they are unable to implement the sophisticated algorithms to handle security issues. Even while sophisticated algorithms may be effectively performed on sensor nodes, it may not be used since the system's latency requirements may be violated and their battery may be drained. Raw data is frequently supplied in many IoT systems to extend the battery life of sensor nodes. Because data can be listened to by unauthorized persons, this strategy is risky [23, 24]. Sensor nodes must run a lightweight security method as well as the battery life of the sensor node cannot be greatly lowered, the algorithm must provide some levels of security.

AES uses symmetric key encryption, which use same secret key to encrypt and decrypt the data [25, 26]. For encrypted

data, individual or group of sensor nodes owns its private key, whereas a gateway holds all of the private keys of entire sensor nodes. Sensor nodes deliver encrypted data using a private key, and the encrypted data received at a smart gateway is decrypted using the right private key. The AES-256 algorithm is implemented in our proposed system since it's not easy to tamper with anonymous users.

In digital signature, a public-private key pair is created for each individual who uses this system [27, 28]. The signature key acts as the private key used for digitally signing, while the verification key acts as the public key. Data is fed into the hash function by Signer, which generates a hash value of the data correspondingly. The hash value and signature key are fed into the signature algorithm, which generates a digital signature for the supplied hash. After appending the signature to the data, both are submitted to the verifier. The verification key and digital signature are fed into the verifier for the verification process, then the verification algorithm returns a value as an output.

The Verifier uses the same hash function on the supplied data to generate a hash value. The result of the verification process and this hash value are compared based on the comparison result, the verifier assesses whether the digital signature is genuine or not. Figure 6 discusses the Digital signature and encryption/decryption.

- Algorithm 1: Security measures in Fog node
- Step 1. Before the encryption and signing procedure must generate the private and public keys
  - Step 2. Generate the public-private key pair ( $k_i, k_j$ )
  - Step 3. Feed data ( $D_i$ ) to the hash function and generates the hash value ( $H_i$ )
  - Step 4. Pass ( $H_i, k_j$ ) to the signature algorithm and generates the digital signature ( $DS$ )
  - Step 5. Append the data and signature ( $D_i, DS$ ) using private key ( $p_i$ ) and generates the ciphertext ( $E_d$ )
  - Step 6. ciphertext ( $E_d$ ) will be sent to the verifier
  - Step 7. Verifier decrypts the ciphertext ( $E_d$ ) using private key ( $p_i$ )
  - Step 8. After decryption the verifier generates the hash value ( $H_j$ ) on the supplied data using the hash function
  - Step 9. Compares the output of the verification algorithm with ( $H_j$ ) and if both are the same it is accepted or rejected.

Algorithm 2: Key Generation phase in RSA Digital Signature

- Initialization Phase
- Pick 2 random prime numbers  $m$  and  $n$

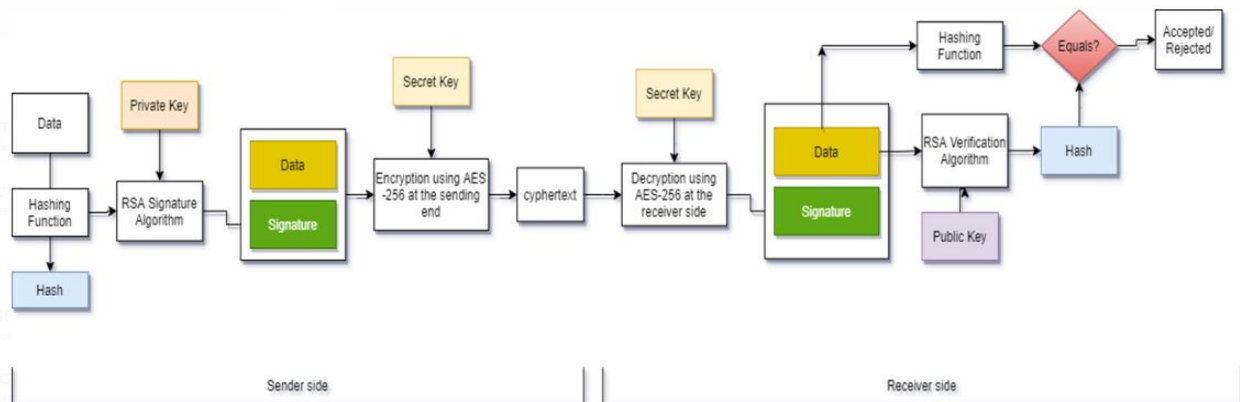


Figure 6. Digital signature and encryption/decryption

$$\text{Calculate } z=mxn \text{ and } \alpha=(m-1)(n-1) \quad (8)$$

Select some random integer  $x$

$$\text{Compute } d \text{ using Euclidean distance algorithm} \quad (9)$$

where  $d$  lies between  $1 < d < \alpha$  such that  $xd=1 \pmod{\alpha}$ .

Finally calculate the public key ( $k_i$ ) and the private key ( $k_j$ )

$$\text{Calculate } y'=Z(y), \text{ which ranges from } [0 \text{ to } n-1] \quad (10)$$

$$\text{Calculate } s=(y')^2 \pmod{n} \quad (11)$$

Computed digital signature of A in Verification Phase  
Get the A's public key  $k_i$  and

$$\text{Compute } s'=sx \pmod{z} \quad (12)$$

If  $s=s'$  then the signature is correct and authentic and it is accepted.

Encryption

For every node  $\pi$  in L, pick up a polynomial degree  $p_\pi$  and assign the degree to  $w_\pi = p_\pi - 1$   
For all other nodes  $\pi \in R$  do

$$\text{Assign } p_\pi(0) = \text{parent}(X)(\text{index}(\pi)) \quad (13)$$

Select a random  $w_X$  to describe a polynomial  $p_\pi$   
End for

Let W to be group of leaf nodes in L with the verification key  $K_i$

$$CT=(T, D_i=H_j, D_y) \quad (14)$$

Generate signature DS

$$\text{The ciphertext is } Ed=(CT, DS) \quad (15)$$

Decryption

Do signature verification DS using the private key  $k_j$

For every node  $\pi$  do

If  $\pi$  is a leaf node and  $j \in R$  then

$$\text{Decrypt } (CT, k_j, \pi) \quad (16)$$

End if

End for

Data compression is used to reduce network bandwidth usage and increase efficiency [29, 30]. Either Lossless or lossy algorithms can be used for data compression; however, when compared to lossy techniques, the lossless approach appears to be a better fit for our e-health application. We chose to use lossless data compression and LZO (Lempel–Ziv–Oberhumer) because they are the fastest lossless data compression algorithms compared to other lossless data compression algorithms. The first method compresses data collected from nodes at the fog gateway and sends it directly to the remote server [31, 32].

Compression on data acquired from sensor nodes and save it as files; if a packet is nowhere to be found during transmission, the server can demand the file from the gateway. Because the packet is temporarily kept in the gateway, it is simple to resend it. When the server admits the positive transmission, the packet is destroyed [33, 34].

Table 1 describes the data size, compressed data size, Compression time, and Decompression Time for the varying nodes from 1 to 40.

**Table 1.** LZO algorithm compression results

Number of Nodes	Data Size (Bytes)	Compressed Data Size (Bytes)	Compression Time (ms)	Decompression Time (ms)
1	800	26	16	19
2	1600	32	18	23
4	3200	48	20	27
8	6400	60	22	33
40	32000	180	30	40

#### 4. EXPERIMENTS AND RESULTS

Edge and fog computing paradigms work together to elevate the restrictions of Cloud-centric execution like low latency and low bandwidth in the IoT applications by moving computation resources nearer to the data sources [35]. Raspberry Pi is extensively utilized as computation nodes in both models. It has used the Raspbian Operating system for the Raspberry Pis, and we have implemented SFog-RPi Controller and SFog-RPi services using Python. SFog-RPi framework containerization is done by Docker Platform.

Tables 2 and 3 explain the simulation parameter used in our experiments and time taken by the edge device to transfer the data to fog nodes located in different locations as well as from fog to cloud server. It is noted that the time taken to move data from fog to cloud takes more time. It has used five sensors to collect the blood glucose level, carbohydrates content, physical activity, heart rate and blood pressure values, based on these values the quantity of insulin intake can be suggested

to the patients. Table 4 shows the Power consumption by the monitoring device.

**Table 2.** Simulation parameters

Parameters	Values
Sensors	Near-Infrared Spectroscopy, MPS-2000, Maxim's MAX30100, INA219
Network Protocol	TCP, MQTT
Controller	Arduino
Tools	Visual framework

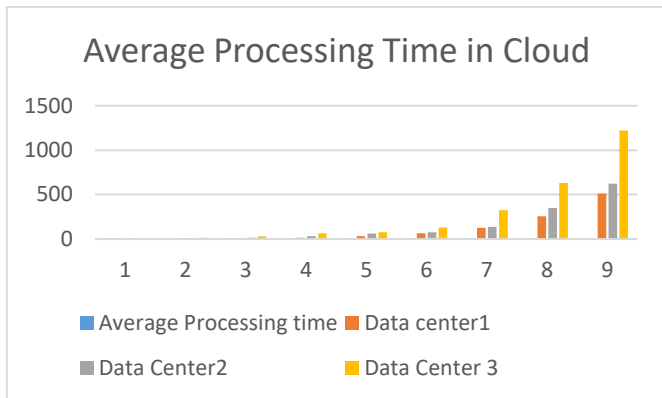
**Table 3.** Time to process the service

Component Type	Component	Time (ms)
Delay	Sensor-Fog1	5
	Sensor-Fog2	11
	Fog-cloud	2200

**Table 4.** Power consumption by the monitoring device

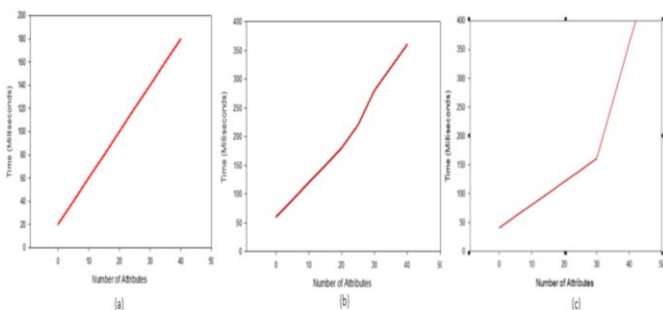
Mode	Voltage Level (v)	Average Power Consumed (mW)
Idle/lazy	4	1
Active mode (AES-256)	4	20
Active mode (without AES-256)	4	18

Figure 7 illustrates the average processing time on different cloud data centers located in different locations. It shows that datacenter 1 located near to the IoT devices took very less processing time compared to the datacenter 3 as it is located very far away from the IoT devices.

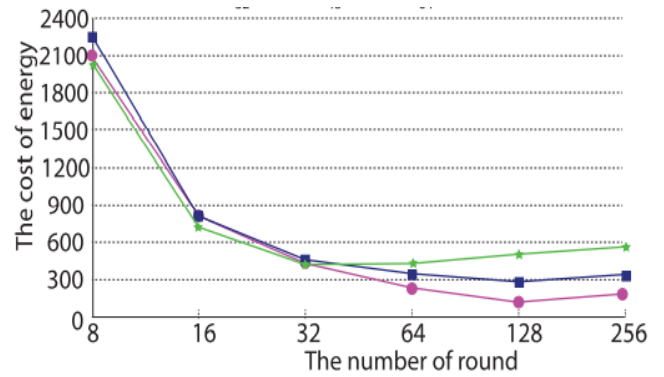


**Figure 7.** Average input processing time on different cloud datacenters

The processing time varies on different Fog platforms available currently, which also compares the performance of our proposed system with the current system. It depicts that the average time taken by SFog-RPi is comparatively very less from all other Fog platforms. The run time of the key generation process based on the attribute's defined range. The execution time is linearly relational to the number of attributes, with the number of attributes increasing the execution time. Also look into the patient's performance during the encrypting portion of the PHR. Observing the safety primitives inside the proposed protocol, we analyze the performance of the decryption phased for an overall of 25 attributes. The time taken for decryption is less than 100 ms, which is adequate for healthcare-based applications. Figure 8 provides Key Generation Phase Performance, Encryption Phase Performance, and Decryption Phase Performance values.

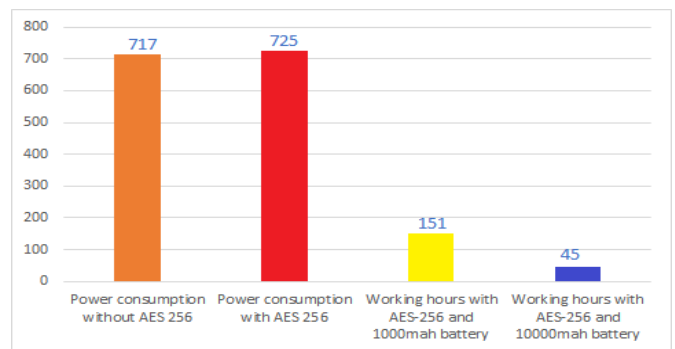


**Figure 8.** (a) Key generation phase performance, (b) Encryption phase performance, (c) Decryption phase performance



**Figure 9.** Relationship between the cost and energy spent per bit

Figure 9 illustrates the relationship between the cost and energy spent per bit. It is observed from the figure the cost of energy utilized per bit decreases when the block size is increased.



**Figure 10.** Power consumption of sensor nodes with and without AES-256 and working hours of sensor nodes with AES-256 on 1000mah battery and 10000mah battery

Figure 10 illustrates the comparison between the power consumption without AES-256 and with AES-256, with AES-256 consuming little more power compared to without AES-256, there are no more variations but security is achieved. And also working hours of 1000 and 10000 mAh battery is compared, it is proved that 1000 mAh achieved greater performance. In our proposed system, the sensor data are collected and sent to the FoG node for processing and analysis. Let x be the time delay in sending data from the sensor node to the mobile device, and y be the time delay in transmitting data from the mobile device to the FoG node, z be the time delay in sending data from FoG to smartphone. So, the latency to complete one task is calculated by:

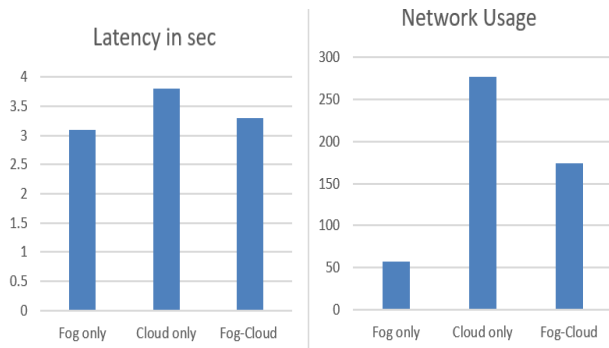
$$\text{Latency } L = x + y + z \tag{17}$$

Because all actions must be completed by the cloud, latency increases in cloud-only implementations. This is due to increased load and, as a result, latency. Because it allows fog nodes to process all incoming tasks and includes fewer difficult calculations for fog node selection, our proposed technique greatly lessens latency when related to cloud-only implementation.

The sum of task execution time and network propagation delay is used to model service delivery latency. It is well recognized that the Fog infrastructure's computing capability is not enhanced, but it is located nearer to the data source. As



as a result, the network propagation delay for Fog infrastructure stands significantly reduced. The network utilization in different SFog-RPi settings is shown in Figure 11. The Fog only setting performs better than the Integrated Fog-Cloud and Cloud only cases because it only uses local networking resources.



**Figure 11.** (a) Latency of various platforms (b) Network Usage of various platform

## 5. CONCLUSION

The communication technologies used at all tiers of the design are abstracted from our model, which focuses solely on their QoS. Foresee medium-term operative maintenance, and long-term business intelligence duties of an application spreading throughout the Cloud-Fog-IoT system or collapsing into a single layer depending on the circumstances and the condition of the network. To excuse interoperability and confederation at every layer, our model incorporates inter-Cloud and inter-Fog communication. The SFog-RPi framework, which we propose, can connect various IoT-enabled equipment to both Cloud and Fog infrastructures. For IoT application deployment, management and monitoring, the framework is lightweight and could be used together with the edge and remote resources. SFog-RPi is written in cross-platform programming languages, which aids in overcoming infrastructure heterogeneity during application execution. A new Load Balancing System is also presented to poise the load among fog nodes. Next, we proposed to use the lightweight cryptographic algorithm, efficient key exchange protocol, and digital signature to achieve confidentiality, authentication, and user privacy. The results of the proposed framework and added state-of-the-art frameworks are compared, it is shown that the proposed system beats the other system in terms of improving response time, management of energy consumption system, and computation resources through distributed unloading and reduces latency and network usage.

## REFERENCES

[1] Azimi, I., Anzanpour, A., Rahmani, A.M., Pahikkala, T., Levorato, M., Liljeberg, P., Dutt, N. (2017). HiCH: Hierarchical fog-assisted computing architecture for healthcare IoT. *ACM Transactions on Embedded Computing Systems (TECS)*, 16(5s): 1-20. <https://doi.org/10.1145/3126501>

[2] Chang, C., Srirama, S.N., Buyya, R. (2017). Indie fog: An efficient fog-computing infrastructure for the Internet of Things. *Computer*, 50(9): 92-98.

<https://doi.org/10.1109/MC.2017.3571049>

[3] Mahmud, R., Ramamohanarao, K., Buyya, R. (2019). Edge affinity-based management of applications in fog computing environments. In *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing*, pp. 61-70. <https://doi.org/10.1145/3344341.3368795>

[4] Chiu, T.C., Pang, A.C., Chung, W.H., Zhang, J. (2018). Latency-driven fog cooperation approach in fog radio access networks. *IEEE Transactions on Services Computing*, 12(5): 698-711. <https://doi.org/10.1109/TSC.2018.2858253>

[5] Asghar, A., Abbas, A., Khattak, H.A., Khan, S.U. (2021). Fog based architecture and load balancing methodology for health monitoring systems. *IEEE Access*, 9: 96189-96200. <https://doi.org/10.1109/ACCESS.2021.3094033>

[6] Surendran, R., Varthini, B.P. (2013). Inject an elastic grid computing techniques to optimal resource management technique operations. *Journal of Computer Science*, 9(8): 1051-1060. <https://doi.org/10.3844/jcssp.2013.1051.1060>

[7] Yigitoglu, E., Mohamed, M., Liu, L., Ludwig, H. (2017). Foggy: A framework for continuous automated iot application deployment in fog computing. In *2017 IEEE international conference on AI & Mobile Services (AIMS)*, Honolulu, HI, USA, pp. 38-45. <https://doi.org/10.1109/AIMS.2017.14>

[8] Borthakur, D., Dubey, H., Constant, N., Mahler, L., Mankodiya, K. (2017). Smart fog: Fog computing framework for unsupervised clustering analytics in wearable internet of things. In *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, Montreal, QC, Canada, pp. 472-476. <https://doi.org/10.1109/GlobalSIP.2017.8308687>

[9] Ferrer, A.J., Marques, J.M., Jorba, J. (2019). Ad-hoc edge cloud: A framework for dynamic creation of edge computing infrastructures. In *2019 28th International Conference on Computer Communication and Networks (ICCCN)*, Valencia, Spain, pp. 1-7. <https://doi.org/10.1109/ICCCN.2019.8847142>

[10] Yousefpour, A., Patil, A., Ishigaki, G., Kim, I., Wang, X., Cankaya, H.C., Zhang, Q., Xie, W., Jue, J.P. (2019). FOGPLAN: A lightweight QoS-aware dynamic fog service provisioning framework. *IEEE Internet of Things Journal*, 6(3): 5080-5096. <https://doi.org/10.1109/JIOT.2019.2896311>

[11] Nkenyereye, L., Islam, S.M., Hossain, M., Abdullah-Al-Wadud, M., Alamri, A. (2020). Fog based secure framework for personal health records systems. *arXiv preprint arXiv:2011.06211*. <https://doi.org/10.48550/arXiv.2011.06211>

[12] Surendran, R., Karthika, R., Jayalakshmi, B. (2021). Implementation of dynamic scanner to protect the documents from ransomware using machine learning algorithms. In *2021 International Conference on Computing, Electronics & Communications Engineering (iCCECE)*, Southend, United Kingdom, pp. 65-70. <https://doi.org/10.1109/iCCECE52344.2021.9534855>

[13] Mahmud, R., Toosi, A.N. (2021). Con-Pi: A distributed container-based edge and fog computing framework for raspberry pis. *arXiv preprint arXiv:2101.03533*.

[14] Rahmani, A.M., Gia, T.N., Negash, B., Anzanpour, A., Azimi, I., Jiang, M., Liljeberg, P. (2018). Exploiting smart e-Health gateways at the edge of healthcare

- Internet-of-Things: A fog computing approach. *Future Generation Computer Systems*, 78: 641-658. <https://doi.org/10.1016/j.future.2017.02.014>
- [15] Riya, K.S., Surendran, R., Tavera Romero, C.A., Sendil, M.S. (2023). Encryption with user authentication model for internet of medical things environment. *Intelligent Automation & Soft Computing*, 35(1): 507-520. <http://dx.doi.org/10.32604/iasc.2023.027779>
- [16] Nguyen, D.T., Le, L.B., Bhargava, V.K. (2019). A market-based framework for multi-resource allocation in fog computing. *IEEE/ACM Transactions on Networking*, 27(3): 1151-1164. <https://doi.org/10.1109/TNET.2019.2912077>
- [17] Gia, T.N., Ali, M., Dhaou, I.B., Rahmani, A.M., Westerlund, T., Liljeberg, P., Tenhunen, H. (2017). IoT-based continuous glucose monitoring system: A feasibility study. *Procedia Computer Science*, 109: 327-334. <https://doi.org/10.1016/j.procs.2017.05.359>
- [18] Kurniawan, A. (2019). Programming on Raspbian OS. *Raspbian OS Programming with the Raspberry Pi: IoT Projects with Wolfram, Mathematica, and Scratch*, 79-96. [https://doi.org/10.1007/978-1-4842-4212-4\\_3](https://doi.org/10.1007/978-1-4842-4212-4_3)
- [19] Iorga, M., Feldman, L., Barton, R., Martin, M.J., Goren, N.S., Mahmoudi, C. (2018). Fog computing conceptual model. Special Publication (NIST SP), National Institute of Standards and Technology, Gaithersburg, MD. <https://doi.org/10.6028/NIST.SP.500-325>
- [20] Tuli, S., Mahmud, R., Tuli, S., Buyya, R. (2019). Fogbus: A blockchain-based lightweight framework for edge and fog computing. *Journal of Systems and Software*, 154: 22-36. <https://doi.org/10.1016/j.jss.2019.04.050>
- [21] Chen, L., Li, X., Ji, H., Leung, V.C. (2019). Computation offloading balance in small cell networks with mobile edge computing. *Wireless Networks*, 25: 4133-4145. <https://doi.org/10.1007/s11276-018-1735-y>
- [22] Hahn, C., Kwon, H., Hur, J. (2018). Trustworthy delegation toward securing mobile healthcare cyber-physical systems. *IEEE Internet of Things Journal*, 6(4): 6301-6309. <https://doi.org/10.1109/JIOT.2018.2878216>
- [23] Mahmud, R., Srirama, S.N., Ramamohanarao, K., Buyya, R. (2019). Quality of Experience (QoE)-aware placement of applications in Fog computing environments. *Journal of Parallel and Distributed Computing*, 132: 190-203. <https://doi.org/10.1016/j.jpdc.2018.03.004>
- [24] Rahmani, A.M., Thanigaivelan, N.K., Gia, T.N., Granados, J., Negash, B., Liljeberg, P., Tenhunen, H. (2015). Smart e-health gateway: Bringing intelligence to internet-of-things based ubiquitous healthcare systems. In 2015 12th annual IEEE consumer communications and networking conference (CCNC), Las Vegas, NV, USA, pp. 826-834. <https://doi.org/10.1109/CCNC.2015.7158084>
- [25] Gia, T.N., Tcareenko, I., Sarker, V.K., Rahmani, A.M., Westerlund, T., Liljeberg, P., Tenhunen, H. (2016). IoT-based fall detection system with energy efficient sensor nodes. In 2016 IEEE Nordic Circuits and Systems Conference (NORCAS), Copenhagen, Denmark, pp. 1-6. <https://doi.org/10.1109/NORCHIP.2016.7792890>
- [26] Mahmud, R., Toosi, A.N. (2021). Con-Pi: A distributed container-based edge and fog computing framework. *IEEE Internet of Things Journal*, 9(6): 4125-4138. <https://doi.org/10.1109/JIOT.2021.3103053>
- [27] Aazam, M., Zeadally, S., Harras, K.A. (2020). Health fog for smart healthcare. *IEEE Consumer Electronics Magazine*, 9(2): 96-102. <https://doi.org/10.1109/MCE.2019.2953749>
- [28] Goudarzi, M., Wu, H., Palaniswami, M., Buyya, R. (2020). An application placement technique for concurrent IoT applications in edge and fog computing environments. *IEEE Transactions on Mobile Computing*, 20(4): 1298-1311. <https://doi.org/10.1109/TMC.2020.2967041>
- [29] Nagappan, K., Rajendran, S., Alotaibi, Y. (2022). Trust aware multi-objective metaheuristic optimization based secure route planning technique for cluster based IIoT environment. *IEEE Access*, 10: 112686-112694. <https://doi.org/10.1109/ACCESS.2022.3211971>
- [30] Sun, Y., Liu, J., Yu, K., Alazab, M., Lin, K. (2021). PMRSS: privacy-preserving medical record searching scheme for intelligent diagnosis in IoT healthcare. *IEEE Transactions on Industrial Informatics*, 18(3): 1981-1990. <https://doi.org/10.1109/TII.2021.3070544>
- [31] Raja, G., Kottursamy, K., Theetharappan, A., Cengiz, K., Ganapathisubramaniyan, A., Kharel, R., Yu, K. (2020). Dynamic polygon generation for flexible pattern formation in large-scale UAV swarm networks. In 2020 IEEE Globecom Workshops (GC Wkshps), pp. 1-6. <https://doi.org/10.1109/GCWkshps50303.2020.9367501>
- [32] Li, H., Yu, K., Liu, B., Feng, C., Qin, Z., Srivastava, G. (2021). An efficient ciphertext-policy weighted attribute-based encryption for the internet of health things. *IEEE Journal of Biomedical and Health Informatics*, 26(5): 1949-1960. <https://doi.org/10.1109/JBHI.2021.3075995>
- [33] Tan, L., Yu, K., Shi, N., Yang, C., Wei, W., Lu, H. (2021). Towards secure and privacy-preserving data sharing for COVID-19 medical records: A blockchain-empowered approach. *IEEE Transactions on Network Science and Engineering*, 9(1): 271-281. <https://doi.org/10.1109/TNSE.2021.3101842>
- [34] Palani, V., Thanarajan, T., Krishnamurthy, A., Rajendran, S. (2023). Deep learning based compression with classification model on CMOS image sensors. *Traitement du Signal*, 40(3): 1163-1170. <https://doi.org/10.18280/ts.400332>
- [35] Alharbi, M., Rajagopal, S.K., Rajendran, S., Alshahrani, M. (2023). Plant disease classification based on ConvLSTM U-Net with fully connected convolutional layers. *Traitement du Signal*, 40(1): 157-166. <https://doi.org/10.18280/ts.400114>