Titre: Title:	Wearable Internet of Medical Things (IoMT) 5G Secure Remote Monitoring System Model in Smart Cities
Auteur: Author:	Fatemeh Sattaridaghian
Date:	2022
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Sattaridaghian, F. (2022). Wearable Internet of Medical Things (IoMT) 5G Secure Remote Monitoring System Model in Smart Cities [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/10313/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/10313/
Directeurs de recherche: Advisors:	Samuel Pierre
Programme: Program:	Génie informatique

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

Wearable Internet of Medical Things (IoMT) 5G Secure Remote Monitoring System Model in Smart Cities

FATEMEH SATTARIDAGHIAN

Département de génie informatique et génie logiciel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*

Génie informatique

Avril 2022

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

Ce mémoire intitulé:

Wearable Internet of Medical Things (IoMT) 5G Secure Remote Monitoring System Model in Smart Cities

présenté par **Fatemeh SATTARIDAGHIAN**

en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*

a été dûment accepté par le jury d'examen constitué de :

Alejandro QUINTERO, président

Samuel PIERRE, membre et directeur de recherche

Ranwa AL MALLAH, membre externe

DEDICATION

This thesis is dedicated to my mother, Sedigheh Akbarian. I would not have been able to finish my graduate studies without her unending support and encouragement.

Additionally, I dedicate this to my husband and closest friend, Amir Astaneh, who has provided me with strength, support, patience, and encouragement throughout this process.

I adore you both and am eternally grateful for all you have done for me.

ACKNOWLEDGEMENTS

First, I would like to express my heartfelt gratitude to my research director, Pr. Samuel Pierre, a full professor at Polytechnique Montreal and director of LARIM, for providing me with the opportunity to complete my master's degree in his research laboratory and for his guidance, support, and trust throughout this unforgettable experience. I feel really privileged to have had the chance to be a member of your professional laboratory. I would also want to convey my appreciation to the jury members who agreed to read and evaluate my dissertation and participate in the presenting session.

Following that, I would like to extend my profound gratitude to Dr. Franjieh El Khoury, LARIM's research associate and coordinator, as well as the supervisor of the APSEC project. I will be eternally grateful to Dr. Franjieh El Khoury for her continuous encouragement, support, and patience during this journey. Throughout my master's degree, her supervision, advising, and sharing of experience were far beyond my expectations. I am also grateful for your continuing help and for editing my dissertation critically. It was an enormous honour to learn from such an experienced and competent supervisor as you.

My thanks also go to the whole Flex team and the LARIM team for their assistance.

I would also like to thank my family for their support throughout the years and who had given me confidence in myself when I needed it. My father deserves special recognition for what he has done for me. A very special thank you to my mother, the most admirable person in this world for me and my pillar of strength through the difficult times of life, who always believed in me. Finally, I would like to express my sincere gratitude to my closest friend, comrade, and companion through these days of my life, who has followed me on this challenging journey and given me hope for the future and bright days. Without you, this route would not have been conceivable.

RÉSUMÉ

L'importance de la santé est indéniable dans notre vie, et nous sommes témoins d'un développement rapide dans ce domaine chaque jour. En effet, l'un des aspects les plus critiques de cette évolution pourrait être la contribution entre les objets connectés (IoT) et les systèmes de soins de santé. Aujourd'hui, les services de santé tirent un avantage considérable de l'IoT. Ceci permet de fournir de meilleurs services aux patients et d'aider les installations du personnel médical. L'utilisation de cette technologie pour collecter davantage de données sur la santé, les services de santé à distance, la prévision des maladies et la fourniture de systèmes de surveillance avancés est de plus en plus courante. Néanmoins, outre toutes ces améliorations, de nouveaux défis et problèmes sont apparus, tels que la confidentialité, la sécurité, la consommation d'énergie, la capacité de stockage, le traitement des données massives, etc.

Dans ce mémoire, nous nous concentrons sur la santé en considérant la ville intelligente et ses caractéristiques. Nous utiliserons les objets connectés médicaux portables « Internet of Medical Things » (IoMT) pour fournir un système de surveillance de la santé à distance efficace et sécurisé. Dans un premier temps, nous examinerons les cadres d'interopérabilité actuels utilisés pour aider à fusionner les données de santé générées par les patients et les dossiers médicaux électroniques, et nous proposerons une solution pour préparer un système de surveillance de la santé en temps réel et évolutif. Ensuite, nous concevrons un modèle qui prend en charge différents types de données IoMT portables et leur interopérabilité. Il convient de mentionner que les données d'électrocardiogramme (ECG) sont considérées comme un exemple de données de capteur IoMT pour le cas d'utilisation pratique du modèle.

Nous utiliserons la transformée en ondelettes discrètes « Discrete Wavelet Transform » (DWT) pour réduire le bruit dans les données ECG, ainsi que pour augmenter la robustesse, la sécurité et la fiabilité de ces données. Les résultats obtenus prouvent que nous augmentons le « *Signal-to-Noise Ratio* » (SNR) aux niveaux les plus bas et les plus élevés et que nous fournissons des valeurs « *Mean Square Error* » (MSE) acceptables par rapport aux travaux antérieurs. En outre, nous appliquerons une méthode de cryptage et de chiffrement des données ECG avant de les envoyer sur le réseau, ce qui permet d'ajouter une couche de sécurité supplémentaire à notre modèle proposé.

Dans un second temps, nous concevrons un module de transmission rapide des données basé sur les capacités de la technologie de cinquième génération (5G) pour améliorer la vitesse, réduire la latence et augmenter la sécurité dans ces systèmes. Le modèle proposé est basé sur la technologie « Software-Defined Networking » (SDN), une solution de sécurité robuste pour la 5G. Cette technologie sera améliorée en tirant parti de la plateforme OpenDaylight (ODL). En outre, « Support Vector Machine » (SVM), un algorithme de détection des attaques « Distributed Denial of Service » (DDoS) pour SDN, est intégré pour augmenter la sécurité du réseau dans le modèle proposé, atteignant une détection des attaques DDoS de haute précision. Les résultats de notre solution de détection des attaques DDoS par SVM, évalués par le score F1, prouvent que notre modèle a une grande précision puisque notre score F1 est égal à 1 pour le test des données. Pour conclure, les résultats de la mise en œuvre du modèle proposé montrent l'amélioration de la sécurité du système de surveillance de la santé à distance et valident son efficacité.

ABSTRACT

The importance of healthcare is undeniable in our lives, and we are witnessing rapid development in this area everyday. Indeed, one of the most critical aspects of this evolution could be the contribution between the Internet of Things (IoT) and the healthcare systems. Today, healthcare services are benefiting significantly from IoT. This is providing better services to patients and helping medical staff facilities. The use of this technology to collect health data, remote health services, disease prediction and providing advanced monitoring systems is becoming more common. Besides all these improvements, new challenges and issues have emerged, such as privacy, security, energy consumption, storage capacity, big data processing, etc.

In this dissertation, we focus on health by considering the smart city and its features. We will use the wearable Internet of Medical Things (IoMT) to provide an efficient and secure remote health monitoring system. At first step, we will examine the current interoperability frameworks used to help merge patient-generated health data and electronic medical records, as well as provide a solution to prepare a real-time and scalable health monitoring system. Then, we will design a model that supports different types of wearable IoMT data and interoperability between them. It is worth mentioning that electrocardiogram (ECG) data is considered as an example of IoMT sensor data for the practical use case in the model.

We will propose a Discrete Wavelet Transform (DWT) denoising approach to reduce noise in ECG data, as well as to increase the robustness, security and reliability of this data. Our results for denoising ECG data prove that we increase the *Signal-to-Noise Ratio (*SNR) at both the lowest and the highest levels and we provide acceptable *Mean Square Error (*MSE) values in comparison with the earlier previous works. Moreover, we will apply an encryption and encoding method for ECG data before sending it over the network, which adds an extra security layer to our proposed model.

At second step, we will propose a fast data transmission module based on the capabilities of the fifth generation (5G) technology to improve speed, reduce latency and increase security in these systems. The proposed model is based on the Software-Defined Networking (SDN) technology, a robust security solution for 5G. This technology will be enhanced by taking leverage of the OpenDaylight (ODL) platform. Furthermore, Support Vector Machine (SVM) algorithm, a Distributed Denial of Service (DDoS) attacks detection algorithm for SDN, is integrated to increase

the security of the proposed model, achieving a high accuracy DDoS attacks detection. The results for our SVM DDoS attacks detection solution, evaluated by F1 score, prove that our model has a high accuracy as our F1 score is equal to 1 for the test of the data. To conclude, the results of the implementation of the proposed model show the enhancement of the security of the remote health monitoring system and validate its efficiency.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

3G	Third Generation
4G	Fourth Generation
5G	Fifth Generation
6LoWPAN	Low-Power Wireless Personal Area Networks
ANNs	Artificial Neural Networks
ASVM	Advanced Support Vector Machine
BPNN	Back Propagation Neural Network
BS	Base Station
BSD	Berkeley Source Distribution
CIA	Confidentiality, Integrity, Availability
CoAP	Constrained Application Protocol
DCT	Discrete Cosine Transform
DDoS	Distributed Denial of Service
DICOM	Digital Imaging and Communications in Medicine
DoS	Denial of Service
DWT	Discrete Wavelet Transform
eACR	electronic Ambulance Call Recording
ECA	Event Condition-Action

ECDSA	Elliptic Curve Digital Signature Algorithm
ECG	Electrocardiogram
e-health	Electronic Health
eHIPF	enhanced History-based IP Filtering
EHR	Electronic Health Records
EM	Expectation-Maximization
EMD	Empirical Mode Decomposition
EMG	Electromyogram
EMR	Electronic Medical Records
EMS	Emergency Medical Service
ENR	Electronic Medical Records
ePCR	Electronic Patient Care Reporting
FE	Flash Event
FIR	Finite Impulse Response
FTP	File Transfer Protocol
FUs	Functional Units
Gbps	Gigabytes per second
GE	Generalized Entropy
GID	Generalized Information Distance

GMM	Gaussian Mixture Models
GTP	GPRS Tunneling Protocol
GUI	Graphical User Interface
HL7	Health Level 7
HMAC	Hash-Based Message Authentication Code
HMM	Hidden Markov Model
HTTP	Hypertext Transfer Protocol
IC	Integrated Circuit
ICA	Independent Component Analysis
ICT	Information and Communication Technology
ID	Information Distance
IDE	Integrated Development Environment
IDS	Intrusion Detection System
IHE	Integrated Health Enterprise
IIR	Infinite Impulse Response
IoMT	Internet of Medical Things
IoT	Internet of Things
IT	Information Technology
IV	Intra Venous

JAR	Java ARchive
JESS	Joint Entropy-based Security Scheme
JWT	JSON Web Token
KB	knowledge Base
KL	Kullberg-Leibler
KNN	K-Nearest Neighbor
LCD	Liquid-Crystal Display
LP	Linear Prediction
LSTM	Long Short-Term Memory
LTE	Long Term Evolution
Mbps	Megabits per second
mHealth	Mobile Health
MI	Myocardial Infarction
MISO	Multiple-Input-Single-Output
MIT	Massachusetts Institute of Technology
MITM	Man-in-the-Middle Attack
ML	Machine Learning
MQTT	Message Queuing Telemetry Transport
MTC	Machine Type Communication

NGMN	Next Generation Mobile Networks
NIDS	Network Intrusion Detection System
Node MCU	Node Micro-Controller Unit
ODL	OpenDaylight
OHM	Occupational Health Management
ONF	Open Networking Foundation
PLI	Power Line Interference
RFID	Radio Frequency Identification
RHMS	Remote Health Monitoring System
RNN	Recurrent Neural Network
RS	Recommended Standard
RSA	Rivest–Shamir–Adleman
SDN	Software-Defined Networking
SEMS	Smart Emergency Medical Service
SHA256	Secure Hash Algorithm
SAL	Service Abstraction Layer
SNNs	Simulated Neural Networks
SOM	Self-Organizing Map

SQA	Signal Quality Assessment
SQI	Signal Quality Index
SVD	Singular Value Decomposition
SVM	Support Vector Machine
TCP/IP	Transmission Control Protocol/Internet Protocol
UDP	User Datagram Protocol
UEs	User Equipment
WBSN	Wireless Body Sensor Network
WFDB	WaveForm Databases
Wi-Fi	Wireless Fidelity
WiMAX	Worldwide Interoperability for Microwave Access
WSN	Wireless Sensor Network
WT	Wavelet Transform
XML	Extensible Markup Language

LIST OF APPENDICES

CHAPTER 1 INTRODUCTION

A smart city is an urban area using different types of electronic methods and sensors to collect data from citizens, devices and buildings, in order to efficiently manage resources and services. Smart cities include several systems, such as transit, traffic, health services, safety alerts, etc. The main objective in smart cities is to improve the quality of people's lives and services by enhancing the city's operations. As an application for health services in smart cities, we can consider the *Remote Health Monitoring System (RHMS)*, which is a system that captures, and collects medical and health data, then transmits them for assessment and recommendations with a secure approach to health care providers. These systems use health sensors with the *Internet of Things (IoT)* technologies. Regarding sensitive information in the healthcare domain, security would be a must in health monitoring systems and this domain, like other information systems. In this dissertation, we will propose a model for the remote health monitoring system based on the wearable Internet of Medical Things devices and the *fifth generation (5G)* network capabilities to provide security of health data in smart cities. In this chapter, we will first present the basic definitions and concepts of *e-health security* in the context of *smart city*, followed by the problems statement that we will seek to address, then the objectives of this dissertation, to end with the plan of the dissertation.

1.1 Basic definitions and concepts

In this section, we will review the basic definitions and the main concepts regarding the concept of IoT in smart cities, the e-health systems, the 5G technology, the selected health data that we will use in our simulations and the proposed security solutions. Then, we will present the problem statement, the research objectives, and we will end by the dissertation plan.

1.1.1 IoT for smart cities

In recent years, significant advancements in research and technology have been led to the development of the *Internet of Things (IoT)* with the objective of smart city concepts. One key factor in this enhancement is the exponential growth of smart devices or objects assisting the infrastructure of the Internet of Things. According to recent research, we have a dramatic increase in the number of connected devices in the next ten years (around 80 billion devices), and it causes the growth of connected devices per person as well [1]. Moreover, the same approach is expected

in the number of wearable devices and smart objects, and the estimation would be the trillions of things connected to the Internet in the next 50 years [2].

The *smart city* is a framework created essentially by information and communication technology (ICT) for developing, expanding, and promoting sustainable development practices to address the growing challenges of urbanization [3]. A large part of this framework is essentially an intelligent network of connected objects and machines that transmit information using wireless technology to help municipalities, companies, and citizens make better decisions to improve their quality of life. Therefore, smart transport, smart tourism and recreation, smart health, ambient-assisted living, crime prevention and community safety, governance, monitoring and infrastructure, disaster management, environment management, refuse collection and sewer management, smart homes and smart energy are some of smart city potential applications [3]. People use a variety of ways to connect with the ecosystems of a smart city, such as smartphones, portable smart devices, cars and smart homes. Indeed, sensors and actuators from an *IoT* ecosystem are being used in smart cities for collecting huge amounts of data that are then used for a variety of services, where smart health care is an essential one [4-7].

1.1.2 E-health systems

As we mentioned previously, one of the smart city applications is the *smart health* or *electronic health (e-health)*. Briefly, *e-health systems* are *smart health care systems* that use the information and the communications technologies, where advanced sensor devices are attached to patients to collect medical data and vital signs. *E-health* improves health services and information for both patients and paramedics. The *e-health system* includes domains like Electronic Health Records (EHR), Electronic Medical Records (EMR), Telehealth & Telemedicine, Health IT Systems, Consumer Health IT Data, Virtual Healthcare, Mobile Health (mHealth), Big data systems and so on. One of the main applications in this domain is a *remote health monitoring system*. This latter is a system that captures and collects medical and health data, then transmits them securely to the health care providers for assessment and recommendations. These systems use health sensors with *IoT* technologies. Moreover, they use *wearable devices* as *IoT-based health sensors technologies*, and Wireless communication and networking technologies like cellular 3G (Third Generation)/4G (Fourth Generation)/5G, Wi-Fi (Wireless Fidelity), WiMAX (Worldwide Interoperability for

Microwave Access), Bluetooth (BR/EDR & LE), RFID (Radio Frequency Identification), and Wireless Sensor Networks (WSN). It could be considered a solution for automated monitoring of patients, reducing hospitalization and decreasing common healthcare costs. Furthermore, by information provided in these systems, we are able to predict disease and improve the health situation [8] [9].

In addition, the increasing ubiquity of wearable sensors leads to new scenarios in the continuous monitoring of health parameters. In particular, wearables are becoming a critical part of the *Internet of Medical Things* (*IoMT)* or the *IoT in healthcare*. *Wearable* is anything that someone can wear without restrictions on daily activities or movement restrictions, including glasses, clothing, watches, patches, smartphones [10]. In other words, the implementation of the Internet of Things in the health care domain is called the *Internet of Medical Things (IoMT)* [11]. It comes with huge advances in patients' lives, physicians' work and the economics of the health system [12]. Remote care systems for patients and the elderly, emergency alert systems, fitness programs, chronic disease monitoring and remote management of medical equipment are some *IoMT* applications. In fact, the main objective of *IoMT* systems is monitoring health status endlessly, without interruption in people's routine activities and by using low-cost sensors in familiar objects (i.e., watches, bracelets, shirts, patches). We can consider the *IoMT* as a complicated combination between medical devices and software applications for integrating *healthcare IT systems* over a wireless connection. *IoMT*, as an important part of health monitoring systems, is to sense, store, interpret, and communicate information about the wearer's body or environment [13]. *IoMT* has different Types, including On-Body, In-Home, Community, In-Clinic, In-Hospital. It is worth mentioning that *Wearable IoMT* is the most visible and commonly consumer *IoT technology* and the second-most-owned *IoT device* after smart home appliances [11] [14]. Figure 1 presents some examples of *IoMT applications* [14], such as oximeter built into a ring for heart rate, sensor embedded into clothing like wristband with electrodermal sensor for stress, a smartwatch with an accelerometer for physical activity or sleep patterns, headband with EEG for diagnosing brain conditions or sociometric badges for levels of social interaction.

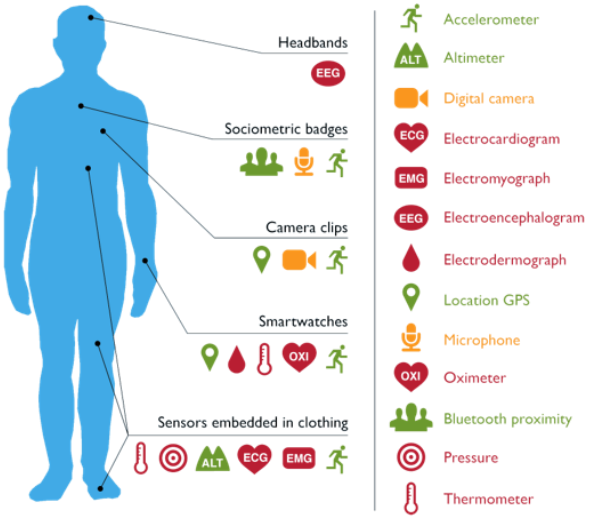

Figure 1.1 Wearable IoMT application examples

1.1.3 Overview of 5G and related concepts

Above all these technologies, we are going to see another key player in our world soon [15]. Considering the Internet infrastructure, the *5G technology* and its features will play an important role in the growing expansion of the *IoT*. The *5G technology* is an industry-standard that will replace the current standard of *4G LTE* (Long Term Evolution) [16]. This technology is the *fifth generation* of this standard, and this standard is designed to be much faster than *4G LTE technology*. *5G standard* will provide high-speed wireless Internet everywhere and for everything, including connected vehicles, smart homes, and *IoT* devices. In theory, *4G* is ultimately 100 Mbps (Megabits per second); *5G* will increase that to 10 Gbps (Gigabytes per second).

Therefore, *5G* will be 100 times faster than current *4G technology*, and it would be the highest speed in theory. *Reduction of latency* is a remarkable prospect of *5G* which leads to high loading speed as well as advancement responsiveness of internet activities. In *5G*, the maximum delay will be four milliseconds, which is much less than 20 milliseconds of *4G technology* [15] [17].

Regarding all the concepts mentioned earlier, we can conclude that *remote health monitoring systems* are one of the key systems in a *smart city* and it is our main focus in this dissertation. For this aim, we are going to prepare our testbed using *Mininet* testbed environment, a free network emulator to support and simulate *Software-Defined Networking (SDN) systems*, which facilitate the

research and the training of *SDN* through a simulated network environment [18]. It is one of the most widely used simulators that provides network equipment and enables software implementation for *SDN* topologies. Moreover, it supports different *SDN controllers*, such as *OpenDaylight (ODL)*, Floodlight and so on [18] [19]. *Software-Defined Networking (SDN)* is a network management approach that efficiently enables programmable network configuration [19]. *SDN* has the capability of improving and monitoring network performance, promises to change the limitations of traditional network infrastructures and is far from the conventional network management approaches. The static architecture of traditional networks is decentralized and complex; *SDN* with an emerging architecture is ideal for dynamic, high-bandwidth applications with more flexibility and easy troubleshooting. The main difference between current network management systems and *SDN* is that *SDN's control* and *data planes* are physically separated [20].

There are several controllers for *SDN*, and one of them is *OpenDyalight (ODL)* [19]. *ODL* is an open-source platform organized by the Linux Foundation for providing centralized, programmable *SDN* control, as well as monitoring network devices. *ODL* includes the controller platform and the service abstraction layers as its two primary layers [21]. In this dissertation, *SDN* and *ODL* are two leading network technologies that are considered as a part of our proposed model.

1.1.4 Our selected health data

The *electrocardiogram (ECG)* signal is considered as *IoT* health data in this model. In health systems, electrocardiograms or *ECG* signals are used for monitoring heart health conditions. More precisely, electrocardiography provides a graph of voltage versus time, based on *ECG* signals and using electrodes that are placed on the skin and measuring the heart's electrical activities [22].

For data preprocessing in the proposed model, *Discrete Wavelet Transform (DWT)* method for denoising *ECG* signals would be our selected approach. *DWT* is one of the popular *ECG* denoising-based strategies for *ECG* signal analysis which is a powerful signal processing approach for non-stationary or dynamic signals like *ECG* signals. It has a high resolution in both the frequency and time domains when applied to signals [23].

1.1.5 Concepts of proposed security solutions

We will provide security on our data by applying *JSON Web Token (JWT)*. *JWT* is an open standard that can provide the transmission of information securely between parties by encoding the data and sending it as *JSON* objects. This will give the ability to encrypt and sign our data digitally [24].

The primary security concern in this dissertation would be *Distributed Denial of Service (DDoS)* attacks. A simple definition of a *DDoS* attack is using all servers' resources by sending massive data and providing unwanted traffic, which leads to interruption of service to real users. When working with *SDN*, a centralized controller still would be an easy attacking point for these kinds of security threats, and we have *IoT-based DDoS* attacks as one of the primary sources of this security threat [25] [26]. Therefore, it is an essential security vulnerability that we are going to mitigate in our proposed model. Consequently, we are going to propose a machine learning approach for *DDoS* attacks detection, and we will implement it using a *Support Vector Machine (SVM)* [27]. *SVM* is a supervised learning algorithm used for classification problems. It is one of the most prominent algorithms and powerful methods with plenty of applications, such as pattern recognition, intrusion detection, spam filtering, and so on [27] [28]. We will discuss the proposed model and more details about these technologies and concepts in chapter 3. The next section will explain our problem statement in this domain.

1.2 Problem Statement

By reviewing the literature in the health monitoring systems, *e-health*, *IoMT* and *smart cities*, we figure out some weaknesses and lacks in existing systems, such as *security*, *scalability*, *latency*, etc. We will detail them in the next chapter. However, we can take into account these critical issues:

- *Lack of security* at different levels: Obviously, for the IoT environment, the most important feature is the security and privacy of the system, as well as given the confidentiality and sensitivity of healthcare data, data security and security requirements in health care systems are a must [29]. Although, in our digital era, the security and privacy of patient health care systems are challenging due to the limited resources of mobile devices [30]. Furthermore, the transmitted data and transmission process both are vulnerable to various sources of

interference, injection and alteration by malicious attackers, sensor faults, etc. Therefore, one of the significant challenge would be security [31].

- *Challenges of dealing with heterogeneous data* provided by various devices: Current e-health trends bring a variety of changes in quantity and quality of information and services. Therefore, it requires new ways for addressing the challenge of dealing with huge amounts of structured and unstructured data in different sources, such as sensors, Web and social networks. As this data includes heterogeneous data, thus, definitely, traditional approaches are not enough, and this is another challenge regarding the emerging of e-health and IoMT [32] [33].
- *Lack of data interoperability* due to information processing: Semantic and functional interoperability between distributed systems is a key function when we work with health data. Considering IoT and e-health as a part of smart city, it is even more important in our domain [34-36].
- *Lack of scalability* considering hardware, network and communication, and data: Only if e-health services are provided being scalable, then a variety of healthcare can be provided through e-health services. Adequate healthcare services must easily increase the capacity and number of resources, as well as scalability is a significant feature for these kinds of services [36] [37].
- *Latency due to data transmission*: Healthcare systems, especially health monitoring systems, are time-critical and include time-sensitive data that can save people's lives. Communication, computation, and network latency are different types of latency in healthcare IoT devices [38] that need to be addressed to provide a reliable health monitoring system. In such scenarios, providing the requirement of minimum latency leads to more values and a suitable healthcare system [39] [40].

Taking into account these problems, our main research question is the following: How can we enhance the level of security for the remote monitoring health system to keep vital and sensitive health information safe from security vulnerability and unauthorized access? From this primary question, derive the following two secondary questions:

1. How can we achieve a real-time and scalable monitoring health system to ensure the availability of this system and make it compatible working with various IoT devices?
2. What are the current interoperability frameworks being used to help merge patient-generated health data and electronic health records?

1.3 Research objectives

The main objective of this dissertation is to propose a model for the remote monitoring health system, which uses wearable Internet of Medical Things devices and 5G network capabilities to provide security in the context of smart cities.

More specifically, this research aims to:

1. Design a model that supports various types of wearable IoMT data and interoperability between them, in order to deal with heterogeneous data and increase the scalability level of this model;
2. Conceive a fast data transmission module based on 5G capabilities to improve the speed and decrease the latencies, in order to have a real-time access to important health data;
3. Design a scalable cloud-based infrastructure to have Scalability in the model;
4. Implement and evaluate the performance of the proposed model.

1.4 Dissertation plan

The remaining of this dissertation is organized as follows. Chapter 2 elaborates the background literature review and the related work of e-health, IoT and health services especially health monitoring and information management systems by focusing on their limitations, the security requirements and the solutions in IoT and 5G, as well as their importance. Chapter 3 presents our proposed model for security in the remote health monitoring system and IoMT devices with details about the development and an explanation of the adopted technologies including the technology SDN and the platform Open Daylight (ODL) in 5G, the network emulator Mininet, the DDoS attacks detection, the Support Vector Machine (SVM) algorithm and other machine learning solutions for DDoS attacks detection, the electrocardiography and the Discrete Wavelet Denoising (DWD) ECG method. In addition, in this chapter will discuss how we set our testbed and explain

procedures of our work with precise information. The Results and evaluation will be presented in Chapter 4. Finally, the Chapter 5 summarizes the work, outlines its limitations and offers some indications for future direction.

CHAPTER 2 LITERATURE REVIEW

In this chapter, we will present a literature review about e-health systems, security requirements and solutions, DDoS attacks detection and ECG signal processing. Then, we will discuss the approaches regarding denoising ECG signals. In reviewing relevant existing techniques and methods in the DDoS attacks and ECG signals fields, we can identify their limitations and propose comprehensive solutions relevant to our goal in this dissertation.

2.1 E-health systems

There are considerable related works in the e-health area, as well as there have been many research papers in the field of Internet of Things (IoT) health services. This section will review the related works in two subdomains: health monitoring systems and health system integration/information management. At the end of this section, we summarize our findings in these two groups of works.

The importance of health is undeniable in our life, and we are witness to fast development in this area every day. Indeed, one of the most critical aspects of this improvement could be the contribution between the Internet of Things and health care systems. Nowadays, health services make extensive use of IoT to provide better services for patients and assist medical staff. In recent years, IoT has been used to collect more health data, provide remote health services, predict disease, and provide advanced monitoring systems. Aside from all these improvements, some new challenges and issues have arisen. We can take into account new problems, such as privacy, security, power usage, storage capacity, processing big data, etc. Otherwise, using these new health systems provides unique opportunities for improvement, such as using accurate algorithms for prediction, improving speed and quality of algorithms for prediction, providing smart health services, etc.

2.1.1 Health monitoring systems

As mentioned before, health monitoring is one of the main domains in IoT health services, and there are plenty of articles on this subject. In this section, we are going to review some of these articles in this domain.

Integrating wearables into corporate Occupational Health Management (OHM) is a fortunate opportunity to create a healthy work culture. Ramani *et al.* [41] proposed an IoT-based employee

health monitoring system based on Web and mobile applications that assists with routine health check-ups of employees at work, including temperature, energy levels, heartbeat rate, and blood pressure. Their proposed system uses Raspberry Pi programmed in Python and IoT as control units and sensors to monitor patient data. This system is not scalable, security issues are not addressed, and they use traditional approaches for storing enormous amounts of data. Additionally, there is a lack of interoperability with other e-health systems, and their model is incapable of working with heterogeneous data.

Ali *et al.* [42] developed an automatic health monitoring and voice disorder detection system for patients suffering from voice complications in smart cities. They used a Linear Prediction (LP) method and Levinson-Durbin recursive algorithm to compute the coefficients-based spectrum and remove formant's effects from speech signals by inverse filtering. In addition, K-means algorithm, Gaussian Mixture Models (GMM) and Expectation-Maximization (EM) algorithm are used in their system to predilect the disease in voice data. The limitations of their work include the effect of filter order on estimating the source signal, an insufficient number of coefficients to handle higher sampling rates, an uncertain security solution, and a lack of interoperability with other healthcare systems.

Irmansyah *et al.* [43] designed a low-cost and real-time heart rate portable device. They used IoT to monitor patients' heart rates and to send SMS notifications as a warning system. They built a PHP and a JavaScript-based website, as well as an employed Transmission Control Protocol/Internet Protocol (TCP/IP) for website communications. Additionally, ESP8266 Wi-Fi modules are used for the webserver gateway and pulse sensor communication [43]. Patients' physical conditions and high physical activity levels, such as sports, are not considered in the design of their system. Furthermore, they did not address security concerns, and there is a lack of interoperability with e-health systems and supporting heterogeneous data.

Shaown *et al.* [44] proposed an IoT-based ECG monitoring system to monitor subjects outside of hospitals. This system uses heart rate and temperature sensors to collect the data. The data generated by wearable ECG sensor AD8232 is processed on the Raspberry Pi model 2 and Arduino uno. A bandpass filter is used to eliminate various noises to produce an ECG graph, and the data is collected by an IoT cloud using a wireless protocol. Moreover, they proposed an alarm system that is only email-based. Furthermore, their results are based on 10 test cases with an 80% accuracy in

diagnosing ECG abnormalities, which is insufficient. Other drawbacks of the driven approach include lack of security solutions, poor interoperability with other systems, and the inability to manage heterogeneous data.

Budida *et al.* [45] proposed an intelligent and resilient health monitoring system that uses IoT to measure patient's body parameters in real-time and deliver an emergency message if abnormal data is detected. Their idea is made up of two functional building blocks: (1) collecting all sensor data and, (2) storing, processing and presenting the resulted information. Sensors (i.e., temperature and pulse rate sensors) capture the patient's bodily characteristics and send them to the ATMEL 89s52 microcontroller, subsequently sending them to the MySQL database server. Moreover, the proposed system includes Recommended Standard (RS)-232, an analog-to-digital converter, and a voltage regulator Integrated Circuit (IC)-7805. Furthermore, the information is instantly supplied to clinicians through an android app on a mobile phone or tablet by using numerous decision-making algorithms. The primary flaw in this study is that it does not address the authentication procedure, patient identification and security concerns. In addition, they did not provide a scalable system, interoperability with other e-health systems and the capacity to cope with heterogeneous data.

Saha *et al.* [46] focused on the communication subdomain, developing a continuous healthcare monitoring system based on IoT and Wireless Body Sensor Network (WBSN). This system monitors health metrics and transmits data via wireless communication, subsequently sent to the network via a Wi-Fi module. They developed a Graphical User Interface (GUI) to store, analyze and present the received data in graphical and text formats and sending SMS, notifying the caregiver, and presenting the current health condition. In this system, an ATmega 328p microcontroller in an Arduino uno board with a Liquid-Crystal Display (LCD) and a Wi-Fi module, Hypertext Transfer Protocol (HTTP), Hypertext Markup Language (HTML) and File Transfer Protocol (FTP) are used. Constant Internet access is required in this system, and they only cover limited sensors. Other limitations include an uncertain strategy for managing security vulnerabilities, handling heterogeneous data and facilitating interoperability with other e-health systems. Furthermore, the 15-second information update rate causes huge delay in cardiac monitoring, and cloud FTP is required for scalability.

When it comes to IoT, *Wireless Sensor Network (WSN)* may play a key role here, and it is another subdomain for e-health subject. Sudevan *et al.* [47] presented a vision of incorporating IoT and healthcare monitoring devices by transmitting real-time patient status data to caregivers. Their implementation is real-time and based on cloud computing. In this system, wearable devices transmit the continuous flow of information and sensor status and then retrieve it by the cloud platform. Their applicable protocols are divided into three layers: (1) Extensible Markup Language (XML) and Constrained Application Protocol (CoAP) for uninterrupted mapping to HTTP, (2) User Datagram Protocol (UDP) in transport layer, and (3) IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN) for Web connectivity and specific standards in network layer. Their implementation is limited to Bluetooth and mobile phones supported by Android. Another point is that there are some weaknesses in their transport layer, such as reliability. The CoAP protocol they use is not secure against Distributed Denial of Service (DDoS) amplification attacks. Therefore, we should consider other alternative solutions to solve this weakness, such as applying DDoS attacks detection methods in this system.

Preethi *et al.* [48] created a wireless and real-time IoT-based healthcare monitoring system with a primary emphasis on Intra Venous (IV) fluid flow control, and regulation is the primary purpose of this article. This proposed system is a kind of *cyber-physical health monitoring system*. They used following hardware components in their system: (1) Node Micro-Controller Unit (Node MCU), (2) solenoid valve and Arduino Integrated Development Environment (IDE) as software technologies, (3) firebase real-time database, which is a cloud-hosted NoSQL database, and (4) Massachusetts Institute of Technology (MIT) app inventor to generate software applications for the Android operating system. The suggested system works under the constraints of voltage, temperature, humidity, and pressure voltage, temperature, humidity, and pressure. They did not provide any solutions for unexpected conditions like Wi-Fi outages or power outages. Furthermore, they did not address the critical concerns, such as low latency, system scalability, interoperability with other e-health systems and the authentication procedure, patient diagnosis and security issues.

Jerald *et al.* [35] proposed a secured architecture for smart healthcare based on IoT. Their main objective is to secure the transaction of health care data from the patients to IoT health seva Kendra. Smart gateway, application data server, application programming interface server, security management server, data management server, Web services server, and information alert server

compose all the parts of the proposed architecture. Moreover, the authors used the Elliptic Curve Digital Signature Algorithm (ECDSA) as a cryptographic algorithm and point generation algorithm. In terms of accessibility, users must log in with the phone, and there is no web-based solution. In addition, the certificate generated based on the device, number, username, and password has a limitation, which causes problems when the device is changed. Another considerable limitation in this work is that their dataset for performance analysis contains only 1000 records.

Srilakshmi *et al.* [36] proposed an IoT-based smart health care system primarily based on temperature data. They designed a smart health care system in the application layer of Software-Defined Networking (SDN), considering the confidentiality and security of patient data to prevent security attacks in SDN. They applied cloud-based SDN technology for collecting real-time health reports of patients. Moreover, they provided smart alerts for variation in users' threshold values based on their availability of sources like Twitter accounts, mobile messaging or e-mail accounts. They used Message Queuing Telemetry Transport (MQTT) and CoAP for communication, and Raspberry Pi as a data plane to generate streaming data. Furthermore, they took advantage of ThingSpeak cloud, a platform for MATLAB analysis and visualization, to ensure trustable health reports to the end-users and Machine Learning techniques, in order to analyze data and make predictions. Their limitation is that they proposed an overall structure without precise detail for preparing it, and they did not evaluate their model about the rate of security that their model can provide.

Combining Machine Learning with health monitoring can help to enhance medical services. Gupta *et al.* [49] proposed a model for predicting future diseases based on the integration of Machine Learning with existing hospitals and medical services. In their model, remote healthcare uses four layers: the sensor layer, the Internet layer, the system layer and the administration layer. An Arduino remote healthcare unit with the Raspberry pi is used for data transmission, and an electronic application is created using HTML and CSS. They used ML to predict future diseases and the SK learn data library for the dataset. Then, analysis phase was carried out using a Machine Learning approach (i.e., linear regression). Regarding their work, prediction results are 94% accurate, with a 6% margin of error. By combining two or more of hybrid data analytics approaches, these results could be enhanced. In addition, the proposed model requires a robust

Internet connection, which leads to a serious issue in the term of access to health data. Furthermore, the authors did not address in their work the latency and the interoperability with other e-health systems.

2.1.2 Health system integration/information management

Another critical component of health services is integrating health data created by different sensors and IoT devices in health systems. Using IoT devices leads to generating massive data with various data types. It is more complex to manage these types of data when it comes to the health area. Therefore, we can consider *health information management* as another critical element in e-health systems.

The main purpose of Venkatesh *et al.* [50] is to provide a scalable modular approach based on Machine Learning to reduce complexity for integrating IoT-based smart health applications in the context of a smart city. The authors designed the smart health application as a hierarchical structure of common Multiple-Input-Single-Output (MISO) Functional Units (FUs). They used serialisation to expose intermediate output for reuse by other applications in the smart city infrastructure. Moreover, they predicted user presence, user activity, air quality and other factors. Furthermore, they examined their application using occupancy detection as the context engine technique. The first limitation for this work is that, integrating health systems through Machine Learning means losing accuracy to improve scalability. In addition, the serializing process can increase latency if a highly compact algorithm is expanded. Finally, the security aspects of this work are not apparent.

Park *et al.* [34] investigated the requirements of a Smart Emergency Medical Service (SEMS) system, which offers an event-tracking service as well as bio-signal monitoring to collect up-to-date emergency information. Additionally, they designed a life-log-connected Emergency Medical Service (EMS) for stroke patients with a real-time location service to manage treatment timelines. They provided a personal Electronic Medical Records (EMR) profile by using a tagging technology (i.e., life tag) that serves as emergency medical identification. For reliable EMS activity reporting, they employed electronic Patient Care Reporting (ePCR) and electronic Ambulance Call Recording (eACR). Moreover, they used existing standards, such as Health Level 7 (HL7) and Digital Imaging and Communications in Medicine (DICOM), to build an Integrated Health Enterprise (IHE)

architectural framework. Therefore, there is a possibility of misdiagnosis of stroke, and the procedure for dealing with security requirements is unclear.

Manikanta *et al.* [51] proposed an IoT ambulance with mobile computing automatic traffic light control. Based on their work, the patient's information and emergency condition are collected in the ambulance via IoT and delivered to the hospital before the ambulance arrives. Furthermore, the ambulance's delay at the traffic signals is omitted by controlling traffic lights from the ambulance and automatically making it accessible for its path. There are not enough details of the traffic light controller in their work and their system requires constant connectivity to the Internet network. Moreover, the authentication process, detection of the patient and security issues are not clarified. It is worth mentioning that interoperability is a remarkable factor for connecting the ambulance to hospital systems, and they did not address this aspect in their work.

Cai *et al.* [52] designed and implemented a regional health information system. Their main objective is to develop a regional medical knowledge standard system, a regional medical health big data center and a convergence platform, which are based on medical health service management data at each level and aggregated to a top-level platform. The authors employed a distributed technology architecture for distributing the data collection to form a pyramid-shaped stable regional health service system. Moreover, they used the unified Internet of Things communication devices to receive, forward and upload data. The first limitation is regarding the wearable devices that involve outdoor applications, and they should be capable of automatically storing and uploading data. Furthermore, the impact of distributed architecture on latency must be considered, which we cannot observe in their work. In addition, there are risks of reading data from a multi-source heterogeneous medical database. An alternative would be to standardize all data based on the HL7 standard. Finally, the security concerns are ignored in this work.

A telemedicine system based on ontology-oriented architecture was developed by Peral *et al.* [32] to be applied on heterogeneous data. Their approach is focused on combining knowledge related to natural language processing and artificial intelligence to access hidden data through a classification tree. Therefore, the authors proposed a big data integration approach in a heterogeneous context based on a knowledge Base (KB). Their telemedicine system is provided to assist the physician in delivering diabetes treatment and decision-making. The suggested paradigm

can increase scalability while retaining a high level of interpretability. The biggest flaw in this work is a lack of information on their security strategy.

Sangkla *et al.* [33] proposed a system for exchanging the heterogeneous data from various health information systems and providing an integrated approach based on metadata and Web service techniques. Their proposed system is functional in both modes: online and offline. They take advantage of an operating adapter interface that may operate as a background process. This adapter is an automated program that is used in offline mode to transfer the data when the network is going to connect again. This system attempts to integrate plenty of diagnostic equipment with the electronic patient cards and uses the data acquisition from instantly laboratory analyzers. Moreover, it has the least flexibility to dynamic data and needs its developers to make modifications to the input and output formats, thus we cannot consider it as a scalable system. In addition, the authors did not include any security measures in their system.

2.1.3 Summary

This section will summarize our findings in the e-health literature review, and we categorize them based on their weaknesses and similarities in Table 2.1. According to these findings, the primary limitations would be providing security, dealing with heterogeneous data, enabling interoperability, high scalability and reducing latency.

Providing security: It is evident that security has crucial importance in any health system, and the health monitoring system would not be an exception. Therefore, the security is one of the essential features in health monitoring systems, and we highlighted this point previously (cf. §2.1.1 & §2.1.2) [32-34] [41-46] [48-52].

Dealing with heterogeneous data: We face heterogeneous types of resources in e-health and health care systems, thus remote health monitoring systems are not an exception. However, in most of the previous works, they did not consider this point, and most of the existing works are proposed for specific resources without this capability to handle heterogeneous data resources (cf. §2.1.1 & §2.1.2) [41] [43-46] [52].

Table 2.1 Summary of e-health literature review

Proposed method	Limitations
IoT-based heart rate monitoring system [41] [43] [44] [47].	Lack of security or insufficient information regarding security, dismissed scalability, inefficacious with heterogeneous data, and lack of interoperability.
Communicate system between patient and physicians using the health monitoring system [45] [46].	Lack of security or insufficient information regarding security, dismissed scalability, inefficacious with heterogeneous data, lack of interoperability, and ignoring low latency.
Secured IoT-based smart healthcare system [49] [50] [47].	Weakness in security models.
Voice monitoring and voice disorder detection system to monitor residents' health of voices [42].	Lack of security or insufficient information regarding security, dismissed scalability, and lack of interoperability.
Real-time IoT-based healthcare monitoring system for IV fluid flow control and regulation [48].	Lack of security or insufficient information regarding security, dismissed scalability, lack of interoperability, and ignoring low latency.
IoT ambulance with mobile computing traffic light control [54].	Lack of security or insufficient information regarding security, and lack of interoperability.
Prediction system for diseases or health requirements (using ML, life-log or integrating e-health data) [51-53].	Lack of security or insufficient information regarding security, dismissed scalability, lack of interoperability, and ignoring low latency.
IoT-based e-health systems for integration of heterogeneous data [55-57].	Lack of security or insufficient information regarding security and ignoring low latency.

Enabling interoperability: We can see the same situation about the interoperability to work with standard health care systems in existing works, where authors did not consider that their system should be able to connect and work with other health care systems efficiently (cf. §2.1.1 & §2.1.2) [41-46] [48] [51]. Obviously, we should rely on existing standards for this requirement and follow the approaches that allow us to work with other standard platforms and systems.

High scalability: Another general issue in most existing related works is that they did not prepare their system for high scalability [41-43] [45] [46] [48] [51]. As we discussed previously (cf. §2.1.1 & §2.1.2), these works are mostly customized for specific devices with specific platforms, making it challenging to have scalability in these circumstances.

Reducing latency: For the last point for health systems, it is a significant factor that we access the data without latency, especially in the case of real-time systems. Regarding the literature review, in most previous works [46] [48-50] [52], the authors did not consider this point in their works or did not measure this parameter (cf. §2.1.1 & §2.1.2).

Regarding the importance of security, the overview about security requirements and the existing solution will be detailed in the next section.

2.2 Security requirements and solutions

After reviewing related articles in the previous section, security is recognized as a significant weakness in e-health systems, particularly in health monitoring systems. Therefore, we consider this point as the main objective in our dissertation. We focus on enhancing the security of our proposed model that would be based on IoT and 5G networks. In this section, we will review the security requirements related to IoT and 5G networks, and then we will continue with discussing the security solutions, which will help us to propose our security model for enhancing security.

2.2.1 Security requirements

It is mentioned that new phenomena, including 5G networks and IoT or IoMT, are becoming a crucial part of e-health systems, such as remote health monitoring systems in the concept of smart cities. At the same time, this new era adds new challenges regarding security, and we should find new solutions for handling these challenges. Therefore, it is necessary to learn about these recent changes and their security requirements. This section will review security requirements with respect to 5G networks and IoT or IoMT phenomenon.

Fang *et al.* [53] studied the security of 5G wireless networks by discussing the uniqueness of 5G and its new requirements, services and use cases in terms of security and compared them with the traditional cellular networks. They presented new security features for applied 5G technologies,

such as SDN, heterogeneous networks, device-to-device communications and the Internet of Things. Moreover, they classified security attacks in 5G mobile wireless networks into two main categories, *passive* and *active attacks*. For each category, some famous samples are considered as bellow:

- *Passive attacks* including *eavesdropping* and *traffic analysis;*
- *Active attacks* including *Man-in-the-Middle* attack *(MITM)*, *replay attack*, *Denial of Service (DoS) attack*, and *Distributed Denial of Service (DDoS) attacks.*

Hatzivasilis *et al.* [7] discussed the primary security and privacy controls required in the new-fashioned IoMT for the user and its involving collaborator protection. They proposed a best-practice approach for increasing safety in IoMT deployment. According to this work, although IoMT technologies have numerous advantages and unique services, traditional security approaches are insufficient for new advancements in the mobile wearable IoMT era. Increasing the number of individual devices that should be capable of connectivity to these types of networks and their valuable and sensitive data, makes them a popular target for different kinds of attacks. Furthermore, vendors are unaware of all these threats, and they pay less for its security aspects to make IoMT devices more cost-effective. Therefore, security is considered an extraordinary requirement for IoMT devices and networks. In addition, they claimed that secure IoT development should include three security areas: device security, connectivity security and cloud security.

Regarding the previous two works ([7] [53]), there are four basic security principles for every sort of security services like 5G and IoMT: *authentication*, *confidentiality*, *availability* and *integrity*. *Authentication* is the most crucial security component of 5G networks, and it is classified into two types: *entity authentication* and *message authentication*. The former ensures that the communicating entity is identical to the one claimed, while the latter is used to authenticate a message, indicating that the stated sender sends it. *Confidentiality* refers to the property of not disclosing information to unauthorized people, processes or devices, and it encompasses two concepts: *data confidentiality* and *privacy*. *Availability* is a fundamental precondition for data and applications access, and it means that connected devices and information must be accessible and usable for any legal user anytime, anywhere based on their request. Finally, *integrity* means

unauthorized users cannot alter or corrupt the data throughout the transition process. *Confidentiality*, *integrity* and *availability* are known as *CIA* principles or *CIA* compromise.

Alsubaei *et al.* [54] emphasised the importance of security and privacy as a growing challenge in IoMT. They proposed a security and privacy taxonomy for IoMT. They classified security attacks with respect to several aspects, including IoMT layers, intruder type, attack impact, attack method, CIA compromise, attack origin, attack level and attack difficulty. Their complete taxonomy is shown in Figure 2.1.

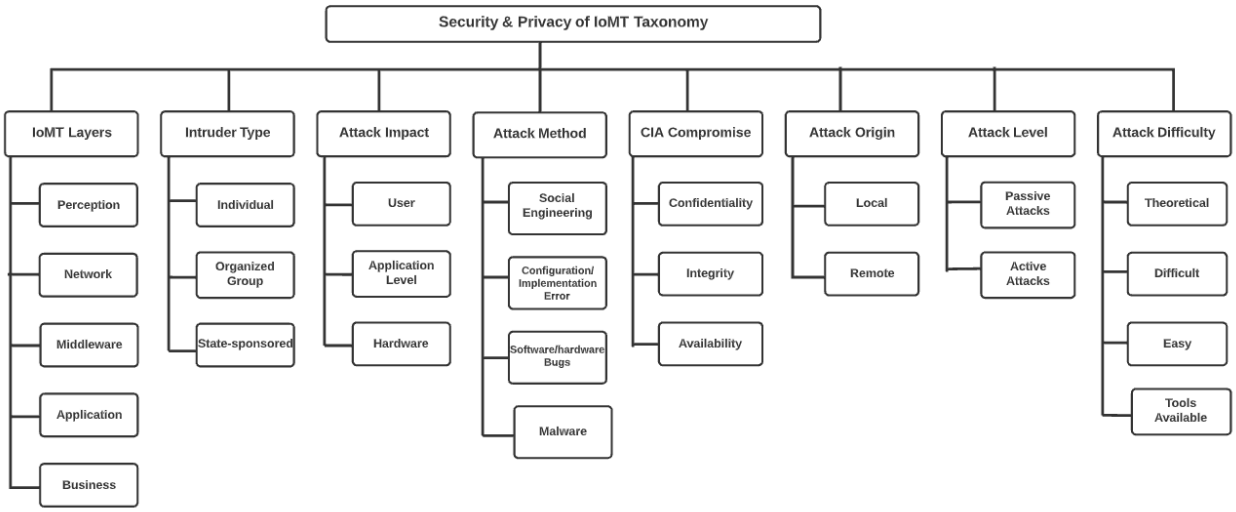

Figure 2.1 Security and privacy of IoMT taxonomy

Based on this classification, a DDoS attacks would be a threat in the network layer considering IoMT layers, which can compromise the *availability* of IoMT services and networks. DDoS attacks are a kind of *active attacks*, and it requires high skilled attacker. Moreover, it is considered as a difficult attack with respect to attack difficulty.

Wireless Sensor Network (WSN) is one of the associated networks with IoT, and there are plenty of applications working based on WSN and IoT [55]. Sharifnejad *et al.* [56] proposed a brief overview of security barriers and security requirements for wireless sensor networks, categorized critical attacks, and listed suggested defences for them. In this survey, primary security requirements for IoT and WSN are listed as below:

- Data authentication;
- Data confidentiality;
- Data integrity;
- Secure localization;
- Scalability;
- Accessibility;
- Flexibility;
- Availability;
- Data freshness;
- Self organization.

The authors considered several ways to perform an attack in WSN, and one of them is the *Denial of Service (DoS) attack*. In addition, regarding the earlier articles [7] [53] [54], we can observe similar requirements in 5G, IoMT and WSN security requirements (i.e., *confidentiality*, *integrity*, a*vailability (CIA) principles* and *authentication*).

Ahmad *et al.* [57-59] investigated security in 5G networks and discussed the fundamental security and privacy challenges in 5G technologies. Moreover, they offered various security methods to address such concerns and difficulties. They highlighted that security is an increasing problem in 5G and in the associated applications and technology. The critical security issues raised by the Next Generation Mobile Networks (NGMN) [57-59] are as follows:

- *Flash network traffic*: it can occur due to the sudden connectivity of a massive number of IoT devices due to known or unknown activities. As a consequence, a flash of signalling traffic will occur, directly harming the core network;
- *Security of radio interfaces*: it is mostly due to the fact that radio interface encryption keys are sent across unsecure channels;
- *User plane integrity*: basically, there is not any cryptographic integrity protection for the user data plane;
- *Mandated security in the network*: it means service-driven constraints on the security architecture leading to the optional use of security measures;

- *Roaming security*: it means user-security parameters are not updated with roaming from one operator network to another, leading to security compromises with roaming;
- *Denial of Service (DoS) attacks* on the infrastructure: it is happened by the visible nature of network control elements and unencrypted control channels;
- *Signaling storms*: it happens because distributed control systems require coordination and overwhelm network device processing capacity;
- *DoS attacks* on *end-user devices*: it happens because there are not any security measures for operating systems, applications and configuration data on user devices.

The authors discussed principles of 5G design, which is expressed by NGMN beyond radio efficiency. They considered the creation of a common combinable core and lightened operation and management via the use of novel computing technologies and networks. They considered technologies, such as mobile cloud, SDN and NFV that conform to the NGMN's design principles. Moreover, they focused on the security of the communication links used by or in conjunction with these technologies. Prior to 5G networks, mobile networks used specialised communication channels based on GPRS Tunneling Protocol (GTP) and Internet Protocol Security (IPsec)tunnels. However, with 5G networks, and particularly with SDN, such separate interfaces and communication channels will be absent.

A summary of security challenges for 5G and targeted points or networks elements for each threat with respect to principles specified by the NGMN and based on these three articles is presented in Table 2.2. In this table, affected technologies for each of these security issues are established [57-59].

According to these works [57-59], DoS and DDoS attacks are critical challenging security threats for 5G networks. They can attack both infrastructure and devices in a 5G network, which threats centralized control elements in networks.

In the literature review in IoT, 5G and the new technologies related to them, security is considered as a significant and growing challenge that traditional security solutions cannot avoid. These security threats are required for new solutions, and in the next section, we will review some related works for the suggested security approach.

Table 2.2 Affected technologies by security threats in 5G

Security threat	Target point/network element	Affected technology
DoS attack	Centralized control elements	SDN, NFV, Cloud
Hijacking attacks	SDN controller, hypervisor	SDN, NFV
Signalling storms	5G core network elements	Channels, Cloud
Resource (slice) theft	Hypervisor, shared cloud resources	NFV, Cloud
Configuration attacks	SDN (virtual) switches, routers	SDN, NFV
Saturation attacks	SDN controller and switches	SDN
Penetration attacks	Virtual resources, clouds	NFV, Cloud
User identity theft	User information databases	Cloud
TCP level attacks	SDN controller-switch communication	SDN, Channels
Man-in-the-middle attack	SDN controller-communication	SDN, Channels
Reset and IP spoofing	Control channels	Channels
Scanning attacks	Open air interfaces	Channels
Security keys exposure	Unencrypted channels	Channels
Semantic information attacks	Subscriber location	Channels
Timing attacks	Subscriber location	Cloud
IMSI catching attacks	Subscriber identity	Channels

2.2.2 Security solution

In this section, we are going to present the existing solutions of security in 5G, IoT and IoMT for threats discussed in the previous section (cf. §2.1).

Several defensive measures techniques are proposed for *Wireless Sensor Network (WSN)* security [56]. Their proposed solutions include *key establishment*, *defending against DoS attacks*, s*ecure broadcasting and multicasting*, *defending against attacks on routing protocols*, d*etecting node replication attacks*, c*ombating traffic analysis attacks*, *defending against attacks on sensor privacy*, *intrusion detection*, s*ecure data aggregation* and *defending against physical attacks*. Some of these solutions are common for other kinds of networks, including 5G networks.

Key establishment [60] is a process or protocol for enabling a shared and secret session key for communication parties whereby they used it for the protection of all exchanges of information. It is considered as a fundamental security solution and service for enhancing the security of WSN communications via the use of cryptography. Conceicao *et al.* [61] proposed a protocol for the *key establishment* between Machine Type Communication (MTC) devices and User Equipment (UE)

for building security in IoT environments in 5G networks. Therefore, *key establishment* is a security solution for 5G and IoT environments as well as WSN [62].

Broadcasting [63] and *multicasting* [64] are two approaches of sending packages to a WSN via its nodes, as well as corresponding receiver methods for forwarding these packages. In *broadcasting*, the package should be received by all nodes in the WSN, whereas, in *multicasting*, the package should be received by a subset of nodes. *Multicasting* and *broadcasting* strategies have been used to reduce the communication and management overhead of sending a single message to multiple receivers.

Furthermore, providing a secure *broadcasting* and *multicasting* approach in 5G and IoT environment is considered as a security solution [65-67]. Therefore, the secure *broadcasting* and multicasting are recommended security approaches in the networks, and they can be done mainly via the use of cryptography [56].

Another critical security solution in WSN is providing *secure routing protocols*, especially when most existing protocols have been designed without security objectives [68]. Providing secure routing protocols would be a security solution for 5G network communication and IoT-based network with the same objectives as WSN [69].

Base Station (BS) are becoming a perfect target for attackers as it has unique role in networks. *Traffic analysis attacks* [56] [70] are based on what the attacker hears in the network by using the BS data supplied by sensors. There are different methods for avoiding these attacks depending on the network types.

Anonymity mechanisms, policy-based approaches and information flooding are different methods for *defending against attacks on sensor privacy* [56]. Anonymity methods may be employed extensively in a 5G network, notably in D2D security protection, for both *direct identity privacy protection* and *indirect location privacy* [71]. Furthermore, the protection of sensor privacy is an important part of security in IoT applications [72].

An *Intrusion Detection System* (IDS) [73] is a network monitoring technology that detects malicious activity or policy violation. It is considered as a WSN security solution. Intrusion detection in the Internet of Things is a new paradigm, and the traditional IDSs are no longer adequate. Moreover, this technique should be considered for security in IoT [74]. Regarding the

5G networks, providing an IDS system will lead to enhanced network security, and it is a part of security solutions in 5G networks [75].

An aggregator collects the raw data from a subset of nodes, then processes and aggregates the raw data from the nodes into more usable data [76]. Another vital security approach in WSN would be applied for *security in data aggregation* techniques. For this purpose, *cryptography* may be useful [76]. Obviously, data aggregation has a significant impact on any system. It is considered in security solutions for both 5G networks [77] and IoT-based systems [78].

In WSN, *defending against physical attacks* [56] is a common occurrence. Therefore, sensor nodes may be equipped with physical hardware to enhance protection against various attacks. Physical attacks are becoming more common and harmful as 5G networks emerge and the IoT environment grows. Therefore, defending against physical attacks should be a part of security solutions in IoT and 5G systems [56] [79] [80].

Cryptography is a valuable method used in security and defensive measures techniques [53]. The *cryptography* is classified into three types: symmetric cryptographic techniques, asymmetric cryptographic techniques and hybrid cryptographic techniques. To encrypt the information, *asymmetric cryptography* uses public key while *symmetric cryptography* employs private keys [53] [56] [58] [81].

Above all of the security solutions mentioned in this section, one of the popular and dangerous attacks for these three types of networks would be DDoS attacks [56]. Therefore, DDoS attacks detection is considered as a powerful solution to add security to the existing networks or systems [56] [82] [83].

In Table 2.3, we summarize the suggested solutions for security based on our findings in this section and in the related works with respect to the network type.

Table 2.3 Summary of existing security solutions

Solution	5G networks	WSN	IoT/IoMT environment
Key establishment	Yes	Yes	Yes
Secure broadcasting and multicasting	Yes	Yes	Yes
Secure routing protocols	Yes	Yes	Yes
Traffic analysis attacks	-	Yes	-
Sensor privacy	Yes	Yes	Yes
Intrusion detection system	Yes	Yes	Yes
Security in data aggregation techniques	Yes	Yes	Yes
Defending against physical attacks	Yes	Yes	Yes
Cryptography	Yes	Yes	Yes
Defending against DoS attacks	Yes	Yes	Yes

2.3 DDoS attacks detection

Regarding our findings in previous sections (cf. §2.1), we can conclude that one of the most common and popular threats in 5G networks and IoT environment would be *Distributed Denial of Service (DDoS) attacks*, which can threaten e-health systems. Therefore, it is essential to mitigate against DDoS attacks in any information system like health monitoring systems, and we will work on this problem as our main security concern in this dissertation. In this section, we will review several existing works regarding DDoS attacks detection solutions for SDN.

The DDoS attacks detection solutions in SDN can be categorised into four groups based on the type of detection metrics and the used methodologies: *information theory-based* DDoS attacks detection solutions, *Machine Learning-based* DDoS attacks detection solutions, *Artificial Neural Network-based* DDoS attacks detection solutions and other methodologies in SDN for DDoS attacks detections. In the below sections, we will review the existing works for each category, then we will summarize our findings for DDoS attacks detection.

2.3.1 Information theory-based DDoS attacks detection solutions

Divergence and entropy are two common metrics that can be extracted from network data and used for DDoS attacks detection. The *information theory-based* approach is basically using network information for calculating the randomness or entropy in the network features or the proportion of

similarity, namely divergence metric [84-86]. Using entropy measurements, we can see how current network behavior deviates from normal network behavior, leading to the detection of DDoS attacks. In this section, we will review related works for DDoS attacks detection based on the *information theory-based* approach.

Safety and availability of information and services in SDN are threatened by DDoS attacks. Kalkan *et al.* [84] proposed a *Joint Entropy-based Security Scheme (JESS)* based on source and destination IP in RYU controller to enhance the SDN security against DDoS attacks. They claimed their statistical solution is the first to use *Joint entropy* for detection and mitigation against DDoS attacks in an SDN. In case of congestion detection, the switch sends just packet information to the controller. The controller will then compute the *Joint entropy* of pair profiles, and any entropy values higher than threshold will be identified as a DDoS attack. Furthermore, a mitigation stage with five functions is proposed: Suspicious pair profile generation, score calculation, threshold determination, rule generation, and differing rules determination to mitigate the attacks. They needed different nominal profiles for exact results, but they used a fixed number of packets in their simulations. The proposed approach is complex to deploy and incompatible with dramatic changes in network status. Furthermore, their model is based on network virtualization that could cause extra security vulnerabilities and is not a scalable model [84].

Sahoo *et al.* [85] proposed a *Generalized Entropy (GE)* based metric using Information Distance (ID) metric to detect DDoS attacks in the control layer. Regarding this work, a centralized controller in SDN would be an excellent target for DDoS attacks. Therefore, the ID metric can quantify network traffic diversions with different possible distributions. Moreover, their proposed model includes a statistical collection module and anomaly detection module for the POX controller [85]. The proposed GE may distinguish *attack traffic* from *normal traffic* with a lower false-negative than other ID metrics. However, their model cannot deal with the network changes and has lots of deployment difficulties.

Sahoo *et al.* [86] proposed an attack detection solution for the POX controller based on the flow table information of the OpenFlow switches. *General Entropy (GE)* and *Generalized Information Distance (GID)* are two *information theory-based* metrics used to distinguish between flash events and high-rate DDoS traffic in this work. The authors evaluated their model by *Shannon entropy* and *Kullberg-Leibler (KL)* divergence and they had fewer false positives in their results [86]. It is

worth mentioning that their proposed model is complex to implement and incompatible with high rate changes in network status.

Xuanyuan *et al.* [87] introduced a lightweight method to enhance SDN security and protection of DDoS attacks by using conditional entropy formerly. They considered source IP, destination IP, destination port and packet length as their parameters, as well as using wildcard filtering policy to mitigate the attack and drop the packets. Their model was applied to the POX controller. Their results indicate that the suggested approach has a high detection rate of 99.372% on average. However, their mechanism is not performing well to discard attack packets reasonably and it is complicated to employ and cannot handle dramatic changes in network status.

In [88], the authors introduced a *φ-entropy* DDoS attacks detection method for SDN, specifically for Floodlight controller. They demonstrated the substantial properties of *φ-entropy* as a strategy for lightweight DDoS attacks early detection. The controller creates a hash table of the destination IP addresses and collects the information, and the entropy value is calculated by using φ. They compared this entropy with predefined thresholds and entropy below the threshold detected as DDoS attacks after five consecutive windows. The authors simulated and evaluated the proposed model by Shannon entropy. Based on their results, in case of high-intensity DDoS attacks, *φ-entropy* performs better than Shannon entropy.

2.3.2 Machine learning-based DDoS attacks detection solutions

Machine Learning (ML) approaches are a widely used methodology for solving complex problems in all disciplines. Therefore, many fellow researchers used Machine Learning algorithms to propose solutions for DDoS attacks detection and classify network behavior as normal or under attack using ML classifiers (e.g., Support Vector Machine (SVM), Hidden Markov Model (HMM), and K-Nearest Neighbor (KNN), etc.) [89-94] . This section will discuss some of these *Machine learning-based* approaches for DDoS attacks detection in the SDN environment.

In [89], the authors proposed a coping DDoS attacks approach in the SDN-based cloud environment concentrating on the application control layer. They used SVM, eHIPF, and SOM classifiers. They leveraged a hybrid *SVM Machine Learning model* and *self-organizing map (SOM)* algorithms for traffic classification. They gathered data using the raw-data processing module,

extracted features, and then sent them to the classifier module. Moreover, they proposed an enhanced History-based IP Filtering (eHIPF) scheme to improve the rate and speed of attack detection. The suggested schema has a high detection rate (99.27%) and 99.30% of accuracy and it could be an efficient approach in the cloud environment.

Cui *et al.* [90] proposed a combined ML-based and *information theory-based* approach for real-time DDoS attacks detection and defense in SDN using cognitive-inspired computing. This mechanism includes three main modules: statistics collection, feature extraction and attack detection. The statistics collecting module collects the switch's flow table characteristics and sends them to the feature extraction module periodically. This module uses entropy to extract features for anomaly detection. The authors evaluated two aspects based on Shannon entropy: source IP and destination IP. Finally, using this information, the attack detection module uses SVM and classifies the traffic as *malicious* or *normal*. They proposed a robust approach, and their results indicate that SVM has a high detection rate and a low false rate.

For two forms of flooding-based DDoS attacks detection in SDN, *Advanced Support Vector Machine (ASVM)* technique [91] is presented. The authors employed an average number of flow packets, an average number of flow bytes, a variation of flow, the packets variation of flow bytes, an average duration as volumetric, and the asymmetric features in their multiclass classification method with three classes: SYN_Flood attack situation, UDP_Flood attack situation and normal situation. They proposed this work for the application layer. The traffic generation and extraction module collects the traffic data. Then classification module uses ASVM to differentiate between normal and attack traffic. Their detection technique has a detection accuracy of over 97% and a fast training and testing model.

SUN *et al.* [92] proposed a KNN classifier for DDoS attacks and Flash Event (FE) detection in SDN, and their selected features included *Generalized Entropy* of source IP, destination IP, average byte and duration, as well as a *φ-entropy* improved on the basis of Shannon entropy. The collection module provided the flow table and switch’s data. Then the flow feature extraction module extracts the selected features from the switch’s data. The innovation of this work is reducing the false alarm rate and performing an effective classification for multi-type DDoS attacks detection, and Flash Event (FE) method.

Hu *et al.* [93] proposed a DDoS flooding attack detection and mitigation system based entropy and SVM classifier in SDN. As entropy metrics, the authors employed source IP, destination IP, source port, destination port and protocol. Their implementation is adopted for POX controller and sFlow agents [93]. Information collection, feature extraction and attack detection are their main modules for DDoS attacks detection. Moreover, they proposed an attack mitigation mechanism based on white-list and traffic migration. In high-rate attacks, they achieved 100% detection accuracy and a negligible false alarm rate, while proposing an effective mitigation mechanism.

Hurley *et al.* [94] proposed an adaptive machine-learning Network Intrusion Detection System (NIDS) that monitors and learns network activity using a Hidden Markov Model (HMM). They improved SDN security by making defensive decisions based on network data and their learning system. Moreover, the Baum–Welch algorithm parameters adopted for their proposed HMM system are: the length of the packet, the source IP, the destination IP, and the source and destination ports. Considering their results, the proposed system is inefficient and requires a large training set.

2.3.3 Artificial Neural Network-based DDoS attacks detection solutions

Another technique for DDoS attacks detection would be Neural networks, also known as *Artificial Neural Networks (ANNs)* or *Simulated Neural Networks (SNNs) [95]*. There are many advantages, including self-learning, self-organization, parallelism in Artificial Neural Networks (ANN) techniques, that make it a favorite approach for solving many problems including DDoS attacks detection [95-98]. In this section, we will review recent articles using this approach for DDoS attacks detection in SDN.

Novaes *et al.* [95] proposed a deep learning algorithm called Long Short-Term Memory (LSTM) and a fuzzy inference system in SDN to predict the normal network traffic behavior, as well as todetect and mitigate DDoS and port scan attacks. In fact, LSTM is an architecture of an artificial Recurrent Neural Network (RNN) provided for the Floodlight controller. Their proposed system is divided into three phases: *characterization*, *anomaly detection* and *mitigation*. They used entropy metrics including source and destination IP, and source and destination to quantify the network flow features. These calculated flow attributes are then used by LSTM to forecast standard signatures for each parameter. The network anomalies will be detected using fuzzy logic.

Moreover, an Event Condition-Action (ECA) model is proposed to create dynamic policies and mitigate DDoS. Their automatic self-learning method is efficient to detect DDoS attacks in SDN.

Nam *et al.* [96] introduced two approaches (i.e., SOM combined with k-NN and SOM distributed-center)using *Self-Organizing Map (SOM)* for DDoS attacks detection in SDN. Their system consists of three main modules: *monitor*, *algorithm*, and *mitigation*. The *monitor* module collects and processes traffic information from switches, such as the entropy of source IP and port, the entropy of destination port, the entropy of packet protocol and the total number of packets. This information is then given to the *algorithm* module. The *algorithm* module uses this data to determine if the network is normal or under attack. In the event of an attack, it sends an alarm to the *mitigation* module, and new policies are established and delivered to switches and servers. The suggested model's shortcoming is that it enhanced processing time at the expense of a lower detection rate. |In addition, choosing of features are manually in their work and they need to enhance their system and to have an automate feature selection process.

Cui *et al.* [97] proposed DDoS attacks detection scheme based on the hit rate gradient of the flow table or the time feature by using the number of packets per flow, the number of flows per port and the duration parameters. They extracted attack patterns from the temporal behavior of an attack using a *Back Propagation Neural Network (BPNN)*[97]. Moreover, they proposed an attack detection module for defending and recovery action by updating flow entry in the Open Flow switch. They used BPNN, a classical Machine Learning method with a simple structure that causes some limitations. Therefore, it causes the inaccessibility of legitimate services, and the time required to recover a victim port is difficult to foresee and control.

Cui *et al.* [98] proposed a DDoS attacks detection and mitigation mechanism based on BPNN. Their proposed mechanism combines four modules: *attack detection trigger*, *attack detection*, *attack traceback* and *attack mitigation*. Exact-storm is used by DDoS attacks *detection trigger* module to enhance early response against DDoS attacks. *Attack detection* works based on BPNN after triggering by the first module. *Attack traceback* and *attack mitigation* modules block the traffic and clean flow tables in switches. In this work, the parameters are the number of packets per flow, the number of bytes per flow, the duration, packet rate per flow and the byte rate per flow. The use of BPNN with its limitations and their mechanism require extra calculation time for

multiple parameters, however, they detected the attack quickly (less than 1s) and with high accuracy.

2.3.4 Other approaches in SDN for DDoS attacks detection solutions

When we review the DDoS attacks detection mechanism for SDN, there are other works apart from the above-discussed techniques which we cannot categorize in the previous three categories (i.e., *information theory-based, Machine Learning-based, and Artificial Neural Network-based approach*). The methods, such as *queuing theory*, *graph theory* and others, are some of these approaches [99-101]. In this section, we will review some techniques based on these methods for DDoS attacks detection in SDN.

Bhushan *et al.* [99] proposed a *Queuing Theory* model for DDoS attacks detection and mitigation in SDN. They demonstrated the flow tablespace of switches using the queuing theory-based on a theoretical mathematical model. Furthermore, a novel flow-table sharing approach is proposed, which leverages the idle flow-table of OpenFlow switches for protecting networks in the SDN-based cloud. In case of attack, their approach analyzes the flow tables of the other switches to identify a suitable switch. This mathematical model focuses on the data plane, requires minimum interaction with SDN controllers, and has a low communication overhead. The authors did not perform a simulation to validate their proposed model, only they used the theoretical mathematical for evaluation, which is not accurate and does not reflect the real situation.

AlEroud *et al.* [100] proposed a method using *graph theory* to identify the attack in the SDN network. Their *graph prediction model Pearson correlation* uses existing attacks resemblance in traditional networks. Their method predicts DDoS attacks by providing attack signatures and using a packet aggregation technique. They used source and destination IP, port number, number of packets and protocol type in their proposed approach to detect DDoS attacks in SDN. There are several limitations here, including the need for data on zero-day attacks and overhead on the controller to implement their security approach.

Conti *et al.* [101] proposed a *cumulative sum* method for DDoS attacks detection in SDN. This proposed method is a change point detection technique and depends on sequential analysis. The authors adopted a threshold to find the change pattern in network traffic. They employed a POX

controller, and the cumulative sum value is computed periodically. When this value goes beyond the predefined threshold, then the attack is identified. Their results indicate an average false alarm rate of 11.64% and 4.15 seconds for the detection process.

Xiao *et al.* [102] proposed a real-time DDoS attacks detection system in SDN for dealing with link flooding attacks. Their system consists of two modules detection framework: *Bloom filter* and *Jpcap API*. This system is proposed for the Floodlight controller and works based on packet count, byte count and duration parameters. The limitation of this work is the length of time required to implement their bloom filter map.

2.3.5 Summary of DDoS attacks detection solutions for SDN

In the Section 2.3, we reviewed several DDoS attacks detection and mitigation models that are proposed for enhancing SDN security. According to these literature reviews, the DDoS attacks in SDN can be classified depending on the target plane in SDN. Furthermore, we grouped existing solutions according to their primary method into four categories: *information theory-based* solutions, *Machine Learning-based* solutions, *Artificial Neural Network-based* solutions, and other DDoS attacks detection solutions. Table 2.4 highlights these methods for DDoS attacks detection in SDN depending on the adopted algorithm and detection metrics.

In conclusion, there are plenty of *non-Machine Learning-based* solutions for effectively preventing DDoS attacks, specifically for SDN environments. Nevertheless, the complexity of deployment and implementation for these solutions and the incompatibility of them with dramatic changes in network status, regarding the 5G and SDN-based networks leads us to consider new approaches, such as ML and ANN, for these security requirements [84-88] [99-102].

Although ANN is a powerful tool for identifying DDoS attacks, however, regarding related works, it is evident that they need to use multiple parameters for calculation of current state in the network and detecting anomaly like DDoS attacks. This extra calculation for multiple parameters could add overhead load to the network and compromise network performance [95-98].

Thus, among all available options, we have chosen to work *Machine Learning-based* approaches that are intelligent and adaptive for DDoS attacks detection in SDN and 5G network. Furthermore, among all the existing ML algorithms, we will choose SVM, since it is a popular and powerful

approach for classification requirements, and it provides high accuracy results in related works [89-94].

Table 2.4 Summary of existing DDoS attacks detection solutions in SDN

Approach class	Main using method
Information theory-based [84-88]	Shannon entropy, joint entropy, generalized entropy, generalized information distance, KL-divergence, conditional entropy, and φ-entropy.
Machine Learning-based [89-94]	Stacked autoencoder, HMM, SVM, decision tree (J48), naive bayes, Shannon entropy, bayes net, logistic regression, random tree, binary bat algorithm, random forest, KNN, φ-entropy, HIPF, SOM, and ASVM.
Artificial Neural Network-based [95-98]	SOM, exact-storm, BPNN, CNN, RNN, LSTM, fuzzy logic, and Shannon entropy.
Other [99-102]	SYN cookie algorithm, TRW-CB, rate limiting, graph theory, queuing theory, bloom filters, and cumulative sums.

2.4 ECG signal processing

ECG data is considered as our health dataset, and in this section, we want to review relevant existing methods about *ECG* anomaly detection. *Signal Quality Assessment (SQA)* in *Electrocardiogram (ECG)* is a crucial aspect to enhance the accuracy and reliability of *ECG* analysis systems [103]. Considering this approach in our preprocessing and applying this reliability enhancement to *ECG* data will decrease false alarms in monitoring systems like a health monitoring system. In addition, it leads to overall increasing the accuracy and integrity of our system.

One of the main requirements for *Signal Quality Assessment (SQA)* in an *Electrocardiogram (ECG)* would be detection and removing noise in *ECG* signals. *ECG* noise causes important issues, including false alarms, inaccurate interpretation, and unreliable measurement. This misinformation could cause real health threats or even death, and it is necessary to prevent them. There are two main strategies tackling *ECG* noise, *ECG denoising based strategy* and *Signal Quality Index (SQI) based strategy* [103] [104]. The former approach is simply removing or reducing the noise to

increase *ECG* data quality, and the latter one is to mark and classify the *ECG* data as acceptable or inaccurate data.

In this dissertation, we choose an *ECG denoising based strategy* to improve the quality of our data and enhance the accuracy and integrity of our proposed health monitoring system. There are plenty of methods used for *ECG* denoising in existing works, i.e., *frequency-selective filters, wiener filters, adaptive filters, Singular Value Decomposition (SVD), polynomial filters, Discrete Cosine Transform (DCT), Empirical Mode Decomposition (EMD), Discrete Wavelet Transform (DWT), nonlinear Bayesian filter, Independent Component Analysis (ICA), mathematical morphological operators, nonlocal means method, Empirical Mode Decomposition (EMD)-wavelet method* and *variational mode decomposition* [103].

Among all of these methods, *Discrete Wavelet Transform (DWT)* is our adopted approach for denoising *ECG* signals. *DWT* is a kind of *Wavelet Transform (WT)*, and a powerful application for time-frequency analysis in image processing and digital signal processing [105-107]. Therefore, an important application of *DWT* is removing the noise efficiently, and it has become a popular method for *ECG* denoising [105-107].

One of the effective parts for providing effective signal denoising by *DWT* methods is *mother wavelet selection*. Amri *et al.* [105] proposed a human heart monitoring/*Electrocardiograph (ECG)* using Android smartphones to provide a wireless monitoring system. They used DWT, and implemented various *WT* methods, such as *Meyer (i.e., dmey), Conflic*, and *Symlet wavelet* for the offline *ECG* denoising process. The best results for filtered *ECG* are provided by *Symlet* wavelet in their work.

Singh *et al.* [108] proposed a two-level *DWT ECG* denoising method. They implement various wavelet basis functions. Their result shows that the best performance for removing *ECG* noise is achieved by *Symlet7*.

Eminaga *et al.* [109] proposed a hybrid Infinite Impulse Response (IIR) and Finite Impulse Response (FIR) wavelet filter banks for *ECG* signal denoising. Based on this work, most popular wavelet families would be the *Daubechies (i.e., db4), Symlet (i.e., sym4) and Coiflets (i.e., coif4)*. Their results show that *DWT* provided better denoising performance by applying *Symlet*.

Gon *et al.* [110] proposed a real-time *ECG* denoising method using the *DWT* algorithm. their approach is based on mother wavelets (i.e., *Daubechies*, *Coiflet*, and *Symlet* wavelets), which provide a robust and efficient *ECG* denoising signal. They used the S-median threshold technique for set their threshold and their proposed model reached acceptable ISNR and MSE values.

Regarding this related work, we are going to work with *Symlet* as our selected mother wavelet for proposed *DWT ECG* denoising. We will see more details about *DWT* and *Symlet* in the next chapter.

CHAPTER 3 PROPOSED MODEL FOR IOMT 5G SECURE REMOTE HEALTH MONITORING SYSTEM

In this chapter, we present the architecture of our proposed security model for enhancing security in the *remote health monitoring system* and IoMT devices. Then, we detail each considered layer in the model (i.e., *IoMT Health Data Layer (IHDL), 5G Network Layer (5GNL) and Application Layer (AL)*, their duty, and how they help us reach our security objectives. Furthermore, we examine how current advancements and technologies contribute to each layer of our proposed model.

3.1 Architecture of the proposed model

This section presents the proposed security architecture based on 5G technologies and with the primary focus on *DDoS attacks detection* for *Electrocardiogram (ECG)* data. This architecture is divided into three primary layers, as shown in Figure 3.1: *IoMT Health Data Layer (IHDL), 5G Network Layer (5GNL) and Application Layer (AL)*.

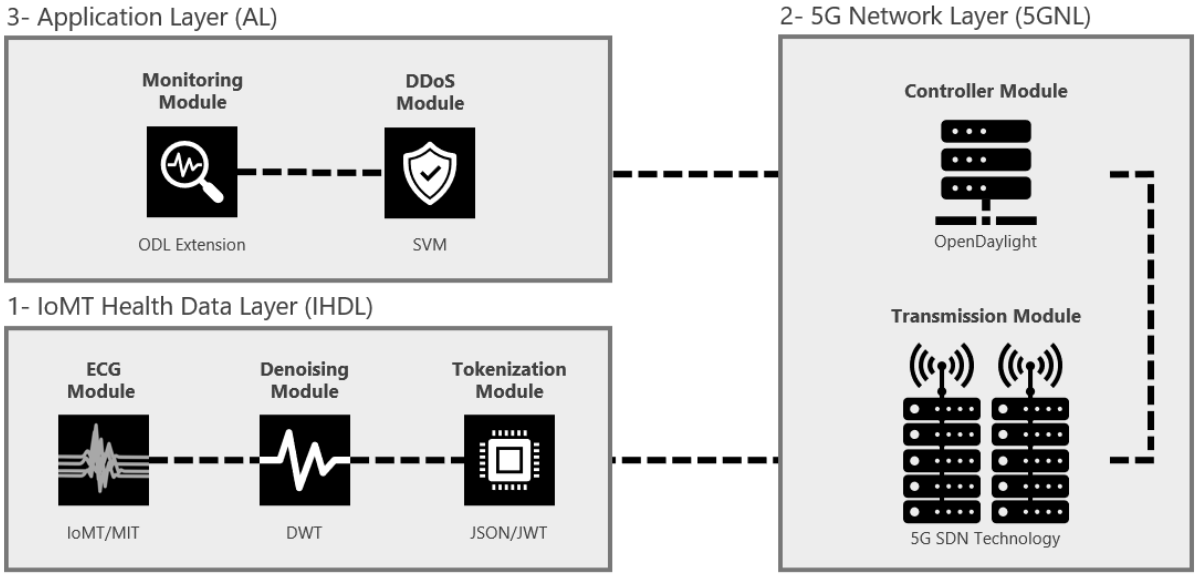

Figure 3.1 Block diagram of the proposed model

The *IoMT Health Data Layer (IHDL)* is the initial layer, and it consists of three primary modules: The *ECG* module, the *Denoising* module and the *Tokenization* module. The *IHDL* is responsible

for collecting our *health data* using the *IoT/IoMT sensors* and *devices*, then preparing this health data for the next layers. Therefore, we consider *ECG* sensors and we work with *ECG* data in our *ECG* module. The second module is the *Denoising* module, which offers preprocessing operations for our data to make it robust, secure and reliable in our model. Furthermore, we propose a *Tokenization* module that converts pure data to Jason format to add data interoperability due to information processing. Dealing with massive volumes of structured and unstructured data would be possible and appropriate with these data formats, allowing us to deal with heterogeneous data. Then this data will be sent to the next layer, which would be the *5G Network Layer (5GNL)*. This *5GNL* is responsible to provide our 5G network with a secure approach and helps us to have ease of network management and real-time monitoring. The *5GNL* leverages the 5G networks and the advanced technologies related to 5G. It includes two main modules: The *Transmission* module and the *Controller* module. In the *Transmission* module, we chose to work with *Software-Defined Network (SDN)* [18] to enhance the security level and decrease the latency in the data transmission process. As we work with *SDN*, the concept of controller plays a crucial role in our proposed architecture. Therefore, we consider a *Controller* module in our model, and for this module, we will work with the *OpenDaylight (ODL)* controller [19]. In the *Controller* module, we use the *OpenDaylight* Rest API to collect our network data and prepare our needed data for the next layer, in order to have real-time monitoring. The *Application Layer (AL)* is responsible for *DDoS* attacks detection in our model. It consists of two primary modules: The *DDoS* module and the *Monitoring* module. The *DDoS* module is responsible for two key functions: The *DDoS* attacks simulation and the *DDoS* attacks detection. When the *ECG* health data is transmitted over our network, by the *DDoS* module, we simulate the *DDoS* attacks on *SDN*. The data obtained from the *ODL* Rest API is then used to build a *Machine Learning (ML)* model for *DDoS* attacks detection. The *Monitoring* module is responsible to provide both offline and real-time *DDoS attacks detection*.

In the remaining of this chapter, we will go over each layer and its modules in further depth, beginning with the *IoMT Health Data Layer (IHDL)* in the next section.

3.2 IoMT Health Data Layer (IHDL)

This section explains the first layer of our proposed architecture. This *IoMT Health Data Layer (IHDL)* consists of three primary modules: *ECG* module, *Denoising* module and *Tokenization*

module. The primary functionalities of *IHDL* would be collecting IoMT health data (*ECG* signals), then denoising ECG signals and lastly transforming them into a standard and interpretable format. Moreover, *IHDL* prepares our required data for other layers and adds security enhancements to them. Figure 3.2 shows our block diagram for the proposed *IoMT Health Data Layer (IHDL)*.

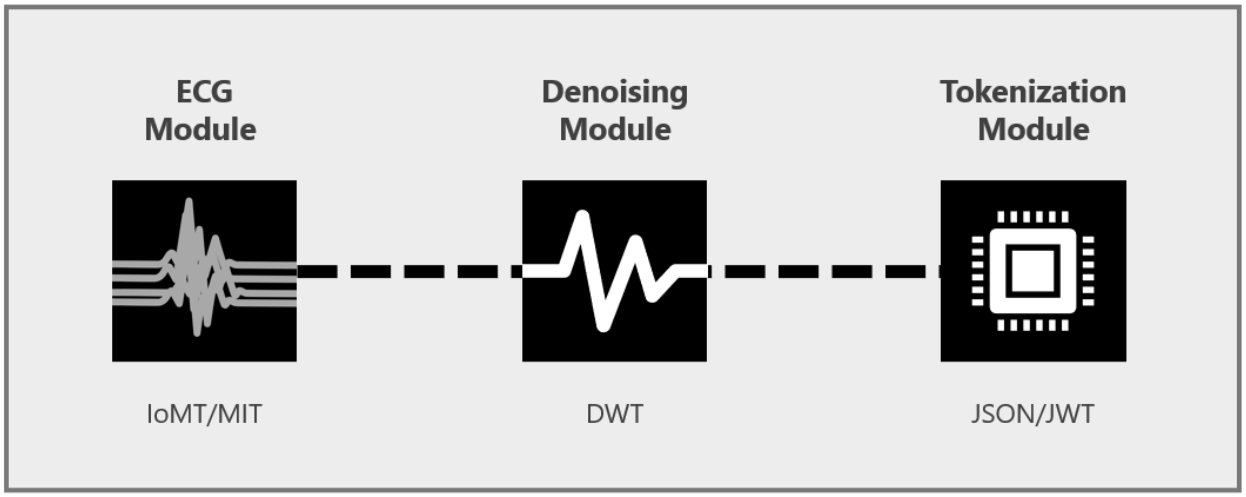

Figure 3.2 IoMT Health Data Layer (IHDL) block diagram

3.2.1 ECG Module

The first module in *IHDL* is the *ECG* module, which aims to prepare the data in our model. In the second chapter, we noticed that monitoring heart health conditions is one of the fundamental and required parts of any *health monitoring system*. Therefore, we chose *ECG* signals collected by the electrocardiogram as our health data for our proposed remote *health monitoring model*. These *ECG signals* are used by *Electrocardiography* for monitoring heart health conditions [44]. They are presented by several local waves (i.e., P-wave, Q-wave, R-wave, S-wave, T-wave and U-wave) as illustrated in Figure 3.3. The Distance between these waves is called an interval or segment, and each of these waves and intervals represents a specific metric of a heart condition [103] [111] [112]. Therefore, precise and trustworthy measurements of the intervals and local waves are extremely required for the accurate determination of the health situation. More details about *ECG signals*, local waves, intervals and segments are provided in Appendix A.

For proposing a reliable health monitoring system, we need to identify the issues or anomalies that may jeopardize the accuracy and reliability of *ECG* signals. Noise in ECG signals is a kind of

anomaly that causes inaccurate health interpretation. In the health domain, the detection of anomaly detection enhances the data's reliability and helps to decrease false alarms in monitoring systems like *health monitoring systems* [113]. Furthermore, it aims to increase the accuracy and the integrity of the *ECG data* [54] [114]. In addition, several anomaly sources in *ECG data* could add *noise* (Appendix A). Therefore, in the *Denoising* module, we want to remove these noises from our data.

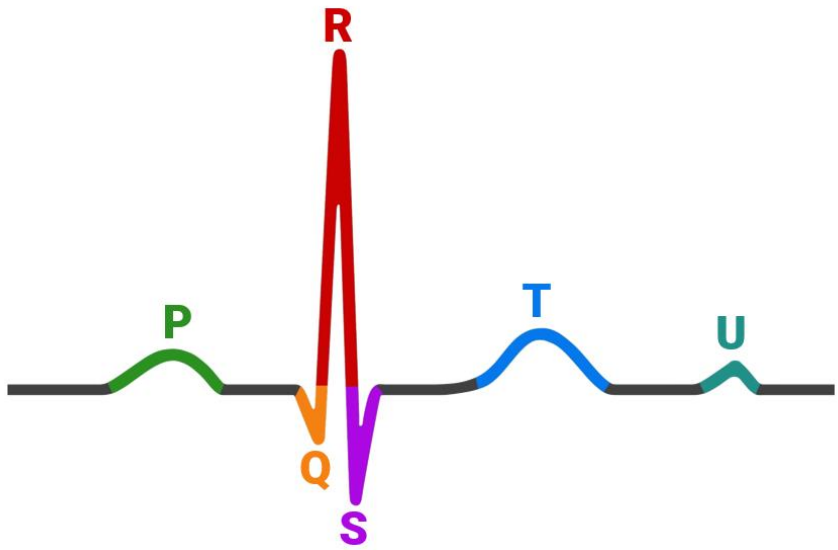

Figure 3.3 ECG signal local waves

To sum up, in the proposed *ECG* module, ECG signals with discussed characteristics are provided in the *MIT* format standard [115]. The next module in the *IHDL*, is responsible for proposing a way to eliminate *ECG* noises from this data. In the next section, our proposed *Denoising* module will increase the quality of the *ECG* health data for our model.

3.2.2 Denoising Module

As mentioned previously (cf. Chap. 2, §2.4), the noise in *ECG* signals causes important issues, including false alarms, inaccurate interpretation and unreliable measurement. Therefore, we consider the *Denoising* module that will apply the *Discrete Wavelet Transform (DWT)* [23] method in the data provided by the *ECG* module. In this section, we will discuss *DWT* and how it can help our model for denoising *ECG* signals and make it robust, secure and reliable.

We provide our ECG denoising signal by using *DWT*, which is one of the popular *ECG* denoising-based strategies for *ECG* signal analysis [23]. It is a powerful signal processing approach for non-

stationary or dynamic signals like *ECG* signals. Therefore, we choose this method, because it has high resolution in both the frequency and time domains when applied to signals [105-107].

The *Wavelet Transform* is designed to solve the shortcomings of the Fourier transform [116] and it is classified into two types: *discrete* and *continuous*. The *Discrete Wavelet Transform (DWT)* uses discrete values for the scale and translation factors, and it is designated by [116]:

$$\boldsymbol{X}(\boldsymbol{a}, \boldsymbol{b}) = \int_{-\infty}^{\infty} x(t)\psi_{a,b}(t)\, dt\,, \qquad (3.1)$$

where *t* represents the time, *a* represents the scale, *b* represents the location and *X (a, b)* represents the wavelet coefficient at scale *a* and location *b*, *x(t)* represents the noisy signal *and* $\psi_{a,b}(t)$ represents the *mother wavelet*.

Considering the *DWT*, we calculate the *DWT* of signal x by passing it through a series of filters [108]. First, a low pass filter is applied on a noisy *ECG* signal, then we use a high pass filter. By combining these results, we are able to eliminate noise from signals [108]. This process is shown in Figure 3.4.

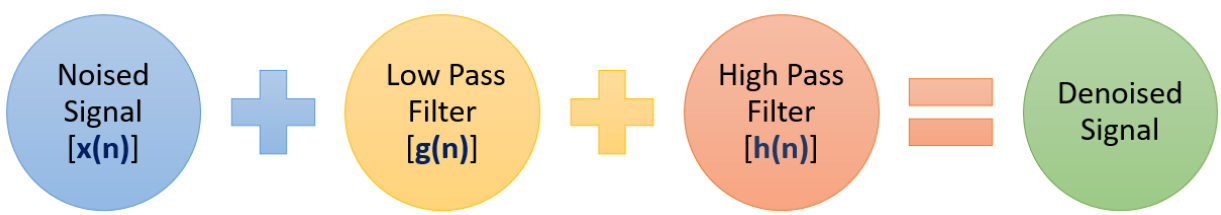

Figure 3.4 DWT approach

The *DWT* is typically implemented as a filter-bank. To apply our proposed *DWT* to a signal, the first step is evaluating its high-frequency behavior, since small scales correlate to high frequencies. Therefore, this evaluation process starts with small scales. Each step increases the scale by a factor of two (i.e., the frequency lowers by a factor of two), and this cycle is continued until the maximum level of decomposition is attained. Finally, high-frequency noise will be filtered out of the signal by excluding it [117]. Figure 3.5 illustrates these cascading phases.

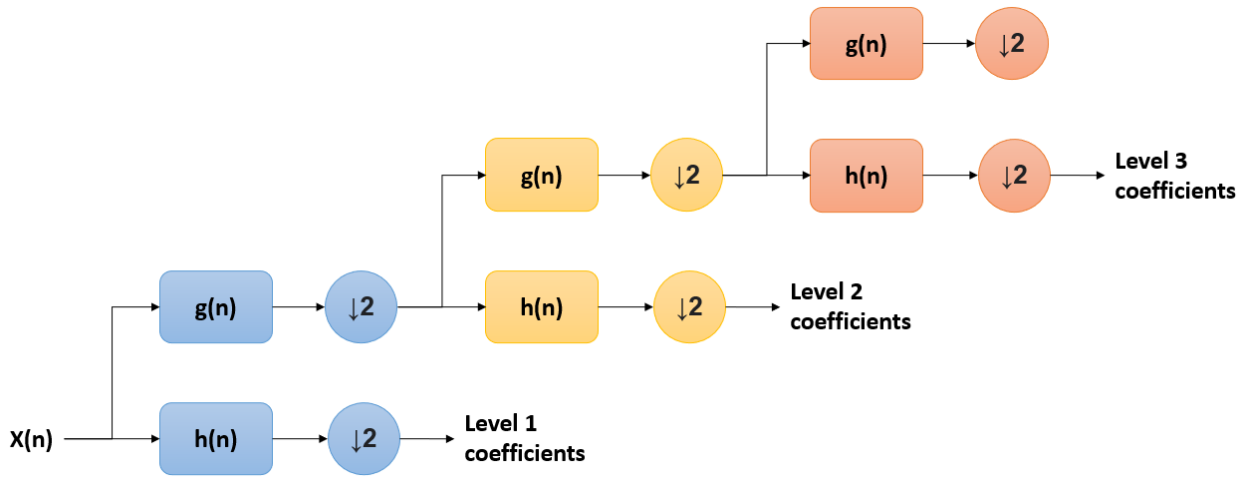

Figure 3.5 Cascading filter banks in DWT

As illustrated in this section, we use a *Discrete Wavelet Transform* technique for our *Denoising* module that removes noises of *ECG* signals provided by the first module (i.e., *ECG* module) of the *IoMT Health Data Layer (IHDL)*.

3.2.3 *Tokenization* Module

In this section, we will present the *Tokenization* module on the *IoMT Health Data Layer*. The *Tokenization* module receives data from the *Denoising* module and converts it to standard formats for health monitoring systems. We select the Jason format as it is compatible with the *HL7* standard and it is easy to do the conversion to the *FHIR* standard [118]. Jason can help us deal with heterogeneous data and it provides data interoperability enhancement due to the information processing. Therefore, the proposed *Tokenization* module receives *ECG* signals in *MIT* format, which we remove their noises with our *Denoising* module and then convert them to Jason files.

Another mission for the proposed *Tokenization* module is to enhance the security of the data to make it reliable and more secure. For this aim, the proposed *Tokenization* module uses *JSON Web Token (JWT)* [24]. In general, *JWT* is an open standard that allows the safe transfer of data between parties by encrypting the data and transmitting it as JSON objects. This will allow us to digitally encrypt and sign our data before sending it to our network.

The *JSON Web Token (JWT)* is a Web security standard (RFC 7519) that offers a self-contained and compression way for securely transmitting data between various destinations using a JSON object [24]. *JWT* can be signed by a private key using the *Hash-Based Message Authentication Code (HMAC)* algorithm [24] or a pair of private and public keys using the *Rivest–Shamir–Adleman (RSA)* algorithm [24]. Therefore, the results of JWT are verifiable and trustworthy due to its digital signature.

In addition, by applying *JWT* in our model, we can reduce the size of our *ECG* Jason objects that facilitates the *ECG* signal's transmission in the network, where the payload of the created token comprises all of the information required for validation [24].

To sum up, the proposed *Tokenization* module encodes our data using the Base64 algorithm and encrypts it with HMAC for header, Secure Hash Algorithm (SHA256) and private key for payload. Then, it provides digitally signed data and since users do not access the private key, they are unable to modify tokens on their own [24] [119]. Consequently, this module, which is the last module of our IHDL, will boost *ECG* data security before it is sent to the next layer and convert it to a secure Jason token.

3.3 5G Network Layer (5GNL)

In this section, we will discuss the second layer *5G Network Layer (5GNL)* of the proposed model. The *5GNL* is responsible for network management and data transmission. It is composed of two modules: The *Transmission* module and the *Controller* module. As indicated previously (cf. Chap. 1, §1.1.3), the advent of 5G technology enables much faster network access and reduces communication and data transmission latency. Therefore, it may contribute to the development of a robust high-speed model for our remote *health monitoring system*. Considering this enhancement, in *5G Network Layer (5GNL)*, we consider 5G technologies and applications for proposing its two modules. Figure 3.6 shows our block diagram for the proposed *5G Network Layer (5GNL)*. The next section discusses the structure of each module and its objectives.

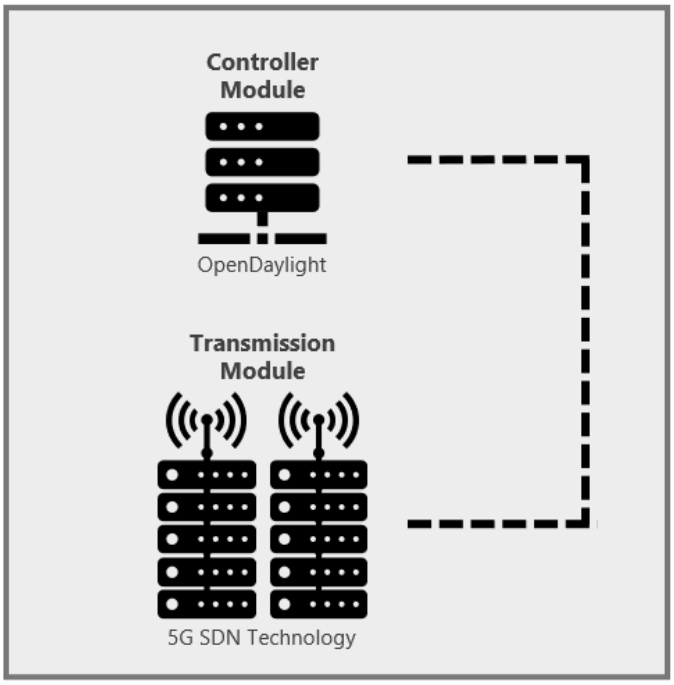

Figure 3.6 5G Network Layer (5GNL) block diagram

3.3.1 Transmission Module

The first module for the proposed *5GNL* is the *Transmission* module. The *Transmission* module is responsible for managing our network and provides an infrastructure to transfer information between nodes securely. Regarding the 5G networks advancements and their applications, we propose our *Transmission* module based on an *SDN* network management system.

Therefore, *Software-Defined Network (SDN)* is a kind of network management technology that can help our module to overcome the constraints of conventional networking [91]. By using the *SDN* in our *Transmission* module, network control and forwarding devices are physically separated, and network control is dynamic, controllable and flexible. Therefore, the *Transmission* module can address several security concerns associated with traditional networks [91].

SDN enables rapid threat detection via a cycle of collecting information from network resources, states and flows owing to its logically centralized control layer with global network visibility and programmability. Therefore, using the *SDN* in our *Transmission* module provides extremely reactive and proactive security monitoring, traffic analysis and response systems, which enable network forensics, policy modification and service insertion [57] [101].

As depicted in Figure 3.7, an *SDN* architecture consists of three layers: The *SDN infrastructure layer* (data plane), the *SDN control layer* (control plane) and the *SDN application layer* (management plane) [99].

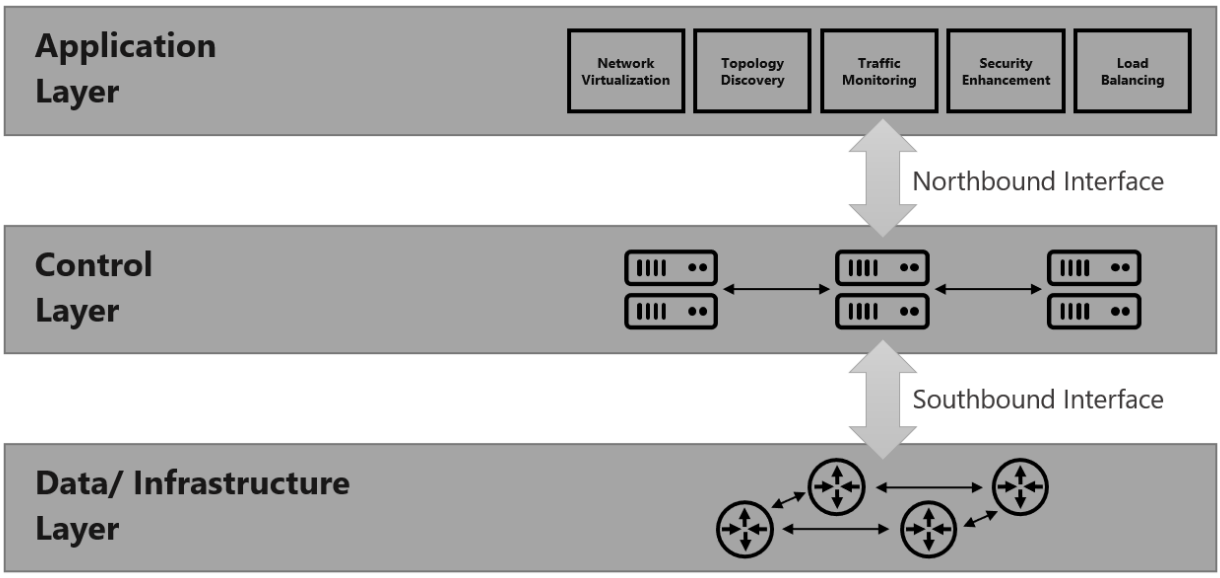

Figure 3.7 SDN architecture

The *SDN application layer* in our *Transmission* module incorporates standard network applications and functionalities, such as intrusion detection, load balancing and firewalls. The *SDN control layer* is referred to as an *SDN's* brain. This layer's intelligence is given by a centralized software for the *SDN* controller. This latter is installed on a server and handles network regulations and traffic flow in our *Transmission* module. The *infrastructure layer* of our *Transmission* module includes the physical switches. In our module, the *SDN Application Layer* interacts with the *SDN control layer* through *Northbound APIs*, whereas the *control layer* interacts with the *SDN infrastructure layer* via the *Southbound APIs* [57] [91] [99] [101].

As mentioned above, the *Transmission* module needs to work with a controller, thus, the next module in the *5GNL* provides the controller for SDN. Then, the proposed *Transmission* module will assist with the next module *Controller module* and provides our network infrastructure with security enhancements that come from these 5G applications.

3.3.2 Controller Module

Our *Controller* module is the second module in the *5G Network Layer (5GNL)*. The primary function of the *Controller* module is to assist the *Transmission* module in achieving its objectives and to act as the controller for *SDN* in our *Transmission* module in the *5GNL*. Furthermore, the *Controller* module is in charge of supplying our network status data for the *Application Layer*. Moreover, it is responsible to add the capability of live network monitoring into our proposed model. This section outlines the controller we chose for SDN and its functionality in our proposed model.

The *SDN* controller, which must interface with all network nodes, is responsible for centralized control and rapid data transmission. The controller and the switch interact using the *OpenFlow* protocol [21]. The *OpenFlow* protocol is a freely available open standard created by the Open Networking Foundation (ONF). This protocol provides the fundamental primitives for programming the *Control* layer on network devices. The controller server decides how to manage data streams flowing via network devices by forwarding them to a certain port, flooding them, or discarding data packets [21].

Indeed, the *Control* layer determines the network's behavior based on its unique requirements and helps the application level through APIs. Additionally, as mentioned previously, communication between the *Control* layer and the other levels occurs through the *Northbound* and *Southbound APIs* [91].

OpenDaylight (*ODL*) is a modular and open-source *SDN* platform that enables centralized, programmable control and monitoring of network devices. This controller is a collaborative effort led by the Linux Foundation with the goal of accelerating the global adoption of *SDN* [8]. As depicted in Figure 3.8, the *OpenDaylight* controller [120] is composed of two layers: The *Controller Platform* layer and the *Service Abstraction Layer (SAL)*. We use the *Lithium OpenDaylight* structure [120] (Appendix C).

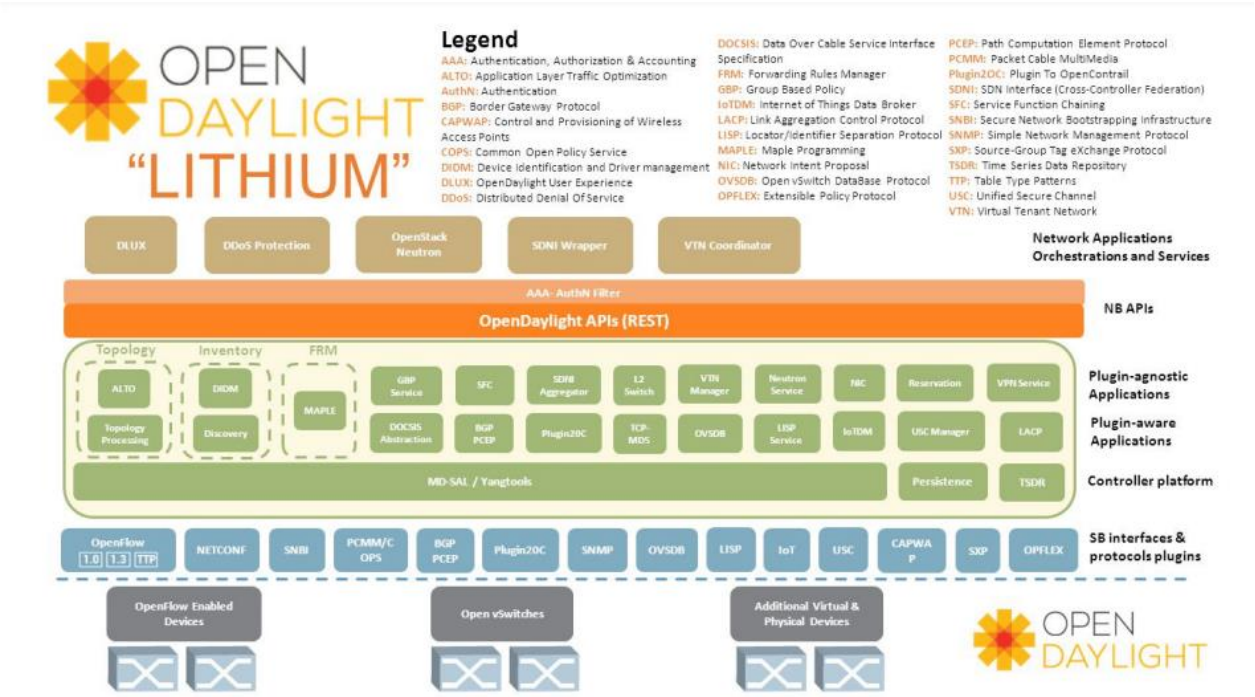

Figure 3.8 Lithium OpenDaylight structure

Therefore, by choosing *OpenDaylight* in our *Controller* module, we will provide a scalable and multiprotocol controller for our *Transmission* module, that supports many other *Southbound communications protocols*, as shown in Figure 3.8. Moreover, the proposed *Controller* module can accommodate more devices in our network and improves interoperability. Additionally, ODL provides us the capability to support *Open vSwitch*, physical and virtual switches, and any switch. As a result, the *Controller* module can support the *OpenFlow* protocol in the *infrastructure* layer.

We employ these technologies in *Lithium OpenDaylight* and in *OpenDaylight* APIs, in order to connect the controller to our SDN in the *Transmission* module, to get access to a Web user interface for monitoring our network and to gather network data online. We use this gathered data for our *DDoS attacks detection* solutions in the *Application Layer (AL)*.

3.3.3 Choice of the Simulators

As mentioned in the previous section, we will use *SDN* in the *Transmission* module in our *5G Network Layer*. Therefore, we need to provide an environment for working with *SDN*. Several

simulators and emulators exist that aim to simulate the *SDN*. This section will briefly compare them and explain our selected options in more detail.

- *Mininet:* It is an open-source project under the Berkeley Source Distribution (BSD) license provided by lightweight virtualization-based Linux [20]. *Mininet* is an emulator widely used for *SDN*. It creates a realistic virtual network and runs a real kernel, switch and application code on a single machine. *Mininet* can support around 4,096 hosts and switches, and it is an inexpensive network test platform with a quick and straightforward configuration. Furthermore, it supports the *OpenFlow* protocol and works with a kind of software *OpenFlow* switch called *Open vSwitch* for virtual switches in *Mininet* [121-123].
- *Ns3: Ns* simulator family is a general network simulation platform, and it was not created for *SDN* networks specifically. For *SDN*, it has two essential weaknesses [124]: 1) These simulations make it challenging to connect the *SDN* network to the controller and 2) *Ns3* does not support all versions of *OpenFlow*.
- *EstiNet: I*s a commercial and powerful simulator and it is not open source. Therefore, it is not flexible to modify and extend the source code level [121-124].
- *OfNet:* Is another emulator environment for *SDN*. However, it is a new tool and it is not as famous as *Mininet* [125].
- *MaxiNet: Mininet* is not a suitable simulator for large-scale networks. Therefore, *MaxiNet* extended the famous *Mininet* emulator environment to span the emulation across several physical machines. Indeed, *MaxiNet* enables emulating large-scale S*oftware-Defined Network* [126].
- *Mininet-Wifi:* Is a newly developed platform based on a clean extension of the *Mininet* emulator. In this new emulator for *SDN*, some new classes and abstractions are added to support wireless devices, nodes and links by adding virtualized Wi-Fi stations and access points. Moreover, other classes were added to support the addition of these wireless devices in a *Mininet* network scenario and to emulate a mobile station's attributes, such as position and movement relative to the access points.

Regarding all the simulators mentioned above, we decided to use *Mininet-Wifi* to simulate our network nodes in the *Transmission* module. We chose this simulator/emulator because it is open source based on *Mininet*, which is one of the most popular platforms for the virtualization of *SDN* networks. In addition, *Mininet-Wifi* allows us to work with virtualized Wi-Fi stations and access points.

3.4 Application Layer (AL)

The last layer of our proposed architecture is the *Application Layer (AL).* Increasing security in two previous layers in the proposed model is the main duty of this layer. We divide the *Application Layer* into two modules: the *DDoS* module and the *Monitoring* module. The objective of *AL* is to propose a *DDoS* attacks detection solution based on *ECG* data and a 5G network. Considering this objective, the proposed *DDoS* module has two main responsibilities: *Simulate DDoS attacks* in the *5G Network Layer (5GNL)* and *detection* of *DDoS attacks* by using *SVM* classification. The *Monitoring* module aims to present the results of the *DDoS* module and our *DDoS* attacks detection solution, as well as to provide the capability of live monitoring in the *5GNL*. Figure 3.9 shows our block diagram for the proposed *Application Layer (AL)*. The rest of this chapter will discuss these two modules with more detail.

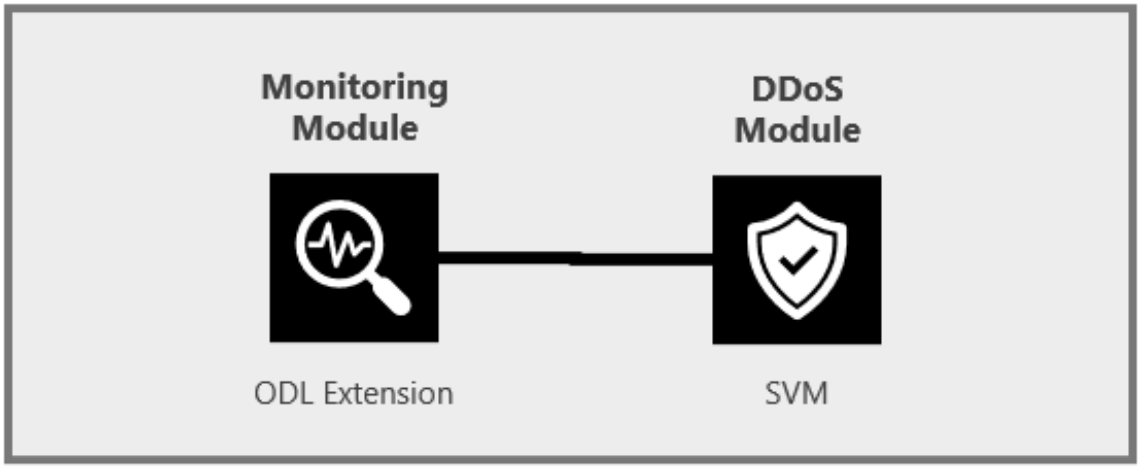

Figure 3.9 Application Layer (AL) block diagram

3.4.1 DDoS Module

Despite all the benefits of *SDN* in network administration and the enhancement of the security over conventional networks, *SDN* still contains some security problems. A *Distributed Denial of Service (DDoS)* attack is the most severe vulnerability related to *SDN* [91]. *DDoS* attacks on *SDN* have become a crucial problem, and many solutions have been developed to detect and mitigate them [91].

Therefore, we propose a *DDoS* module in our *Application* Layer to provide a solution for *DDoS* attacks detection in *SDN*, and to boost the security of our *Transmission* and *Controller* modules in the second layer of our architecture. The *DDoS* module serves two main purposes: *DDoS* attacks simulation and *DDoS* attacks detection. In the next chapter, we will discuss how we will simulate DDoS attacks in this module, and in this section, we will describe the second capability of the proposed *DDoS* module which is *DDoS* attacks detection.

As discussed in the second chapter (cf. Chap. 2, §2.3), the *Support Vector Machine (SVM)* algorithm is one of the supervised *Machine learning (ML)* algorithms that is used to solve classification problems and is one of the most popular options for detecting *DDoS* attacks. Therefore, we use the *SVM* algorithm in the *DDoS* module for *DDoS attacks detection*.

For providing our *DDoS attacks detection solution* by *SVM*, we need to determine the optimal classifier between two groups of the normal situation and the situation of the *DDoS attacks* situation, as shown in Figure 3.10. Therefore, we use two support vectors in the vicinity of the closest data point to these two groups and then pick the optimal hyperplane with the maximum distance between them [28] [91].

Equation (7) depicts the *SVM* formulation, where w is the weight vector or maximum margin, n is the number of samples and b denotes the bias. Therefore, *SVM* attempts to optimize this equation for each classification problem [28] [91] [127].

$$\left[\frac{1}{n}\sum_{i=1}^{n} max\big(0,1 - y_i(w^T x_i - b)\big)\right] + \lambda||w||^2 \qquad (3.2)$$

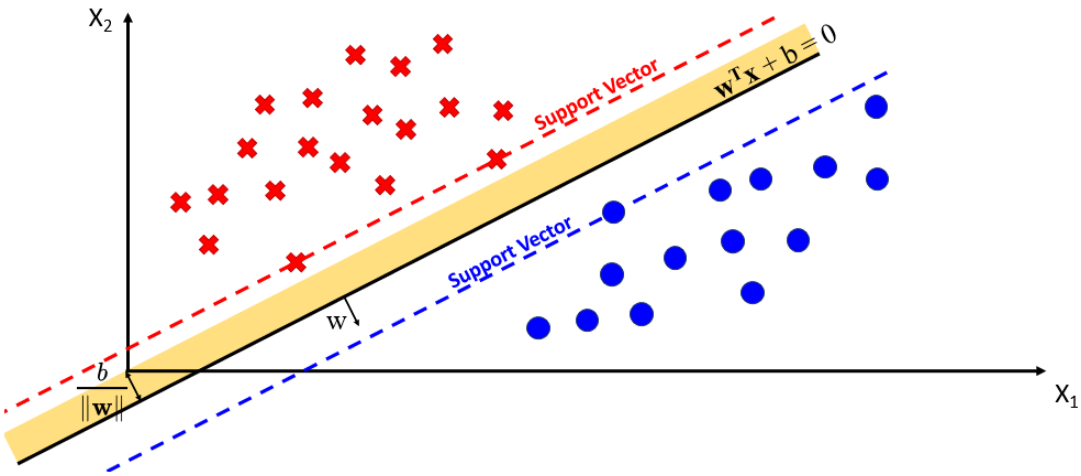

Figure 3.10 Support Vector Machine illustration

SVM has numerous advantages and is not only a well-known solution for *DDoS attacks detection* problems; it also performs well with small datasets like our work. It has accuracy and a low false positive rate, which is critical *for remote health monitoring systems*. In addition, we can use it in both linear and non-linear situations. It is robust to noise and effective with high-dimensional data (Appendix B). Therefore, it would be an excellent candidate for *DDoS* attacks detection functionality in our *DDoS* Module.

DDoS attacks detection, which is the second functionality of our proposed *DDoS module*, is going to use *SVM* and provide the detection of a *DDoS attack* based on *Machine Learning.* Our approach for *DDoS attacks detection* is divided into four main phases: *data collection*, *feature extraction*, *training phase* and *detection of the DDoS attacks*.

Data collection phase: For collecting the network data, we tack advantage of the *OpenDaylight* Rest API provided by the *Controller* module in the second layer. In this phase, the number of packets, the number of bytes, the durations and the destination IPs are collected from ODL using the Rest API, and then the provided data will use in the next phase.

Feature extraction phase: An entropy approach is proposed for the feature extraction phase to make the collected data in the data collection phase more effective for the training phase. Considering the literature review (cf. Chap. 2, §2.3), we know that entropy is one of the common

metrics that can be extracted from network data and used for *DDoS attacks detection*. For defining an entropy, first, we need to define a window, and then calculate our data based on the window size. Therefore, we use time-based entropy for measuring our collected information. The output of our feature selection phase would be a dataset with several entropy features, which will be used by the next phase.

Training phase: In this phase, we use the provided feature selection data provided by the previous phase, and then train our SVM model based on this data. When dealing with the *Machine Learning* algorithm, one of the most typical issues is imbalanced data, which leads to incorrect training model outcomes. Thus, before the training phase, we attempted to address this problem. Handling imbalanced data has two approaches: *upsampling* and *downsampling*. The first technique consists of creating data points (matching minority classes) and injecting them into the dataset. There are two primary solutions [128] [129]: *SMOTE* and the *DataDuplication* function.

The second technique is *downsampling*, which decreases the number of training samples that fall into the majority categories. It assists in balancing the number of target categories via *Tomek (T-Links)* or *Centroid-based functions* [130-132]. After this phase, we have sufficient data to run the *SVM* algorithm and train our model.

Detecting the DDoS attacks phase: In this phase, the trained model can be applied to any additional datasets supplied by the proposed model to identify the attacks in that dataset. We can consider this phase as an offline DDoS attacks detection solution that works for our proposed model. These four phases serve as the *DDoS* attacks detection section in our *DDoS* module in our *Application* Layer (LA).

3.4.2 Monitoring Module

In addition to the offline *DDoS* attacks detection solution proposed by the *DDoS* module, we consider the *Monitoring* module for offering al-time *DDoS* attacks detection in our proposed model. As we mentioned, the *Northbound* and *Southbound APIs* in *ODL* are powerful assets for working with data, extracting and updating them. Therefore, the *Controller* module in the second layer can provide us with real monitoring capability, which helps us to propose a real-time *DDoS attacks detection* solution [133-135].

The proposed *Monitoring* module is intended to be a multiplatform approach, which is independent of the device type or the operating system. Therefore, it assists network administrators in monitoring network status in real-time mode. Moreover, the network administrators will be warned if *DDoS attacks* occur in our proposed model. We will ensure health data security against these *DDoS attacks* with our approach.

3.5 Summary

This chapter presented our proposed model for a *remote monitoring health system* that provides security via the *Wearable Internet of Medical Things (IoMT)* devices and *5G network* capabilities. The *IoMT Health Data Layer (IHDL)*, the *5G Network Layer (5GNL)* and the *Application Layer (AL)* are the three fundamental layers in our proposed model.

The interoperability of the information processing is due to the conversion of the data to Jason in the first *IoMT Health Data Layer*, which is an acceptable format by HL7 and most standard health systems. Furthermore, by using this standard data format, we can deal with structured and unstructured data more effectively and boost our capacity to work with heterogeneous data.

We aimed to boost speed and minimize latencies by using these *5G technologies* in the *5GNL* at the level of the *Transmission* module. In addition, it can enhance security compared to traditional networking approaches and offers flexibility and dynamic structure to our network model. The *Controller* module in *5GNL* provides a real-time network monitoring functionality and access to flow tables and network status information for training our model in the *Application Layer*.

In the *Application Layer*, we addressed one of the *SDN*'s fundamental weakness (i.e., vulnerability to *DDoS* attacks) and proposed a *Machine Learning-based DDoS attacks detection solution*. We employed the *SVM*, a powerful and well-known supervised *ML* algorithm, to identify *DDoS* attacks and provided our solution in both offline and real-time modes. It is worth mentioning that our model is scalable since we're gathering data via the *ODL* controller.

CHAPTER 4 IMPLIMENTATION AND RESULTS

In Chapter 3, we presented the architecture of the proposed security model for enhancing security in the *remote health monitoring system* and IoMT devices with its three primary layers. In this chapter, we present the implementation of the proposed model to evaluate its performance. We describe our testbed and how we set up our testing environment. In addition, we present our findings and results for each layer of the proposed architecture.

4.1 Testbed

We provide our testbed on two separate machines by using *VMware Workstation Pro* version 15.5.5 [136] and installing desktop ubuntu version 20.04.2.0 [137], as follows:

- *Main machine:* This machine is served to our *Transmission* module in the *5GNL* and to our *SDN* network. We set up *Mininet-Wifi* network simulator on Linux virtual machine to simulate the *SDN*. The characteristics of this Linux machine are set for 2GB memory and 2 processors with around 100 GB ofhard disk space, as shown in Figure 4.1(a).
- *Controller machine:* This machine is prepared for installing and simulating our *OpenDaylight* controller and it is providing our *Controller* module for the *5GNL*. The characteristics of this Linux machine are set for 2GB memory and 2 processors with around 100 GB of hard disk space, as shown in Figure 4.1(b).

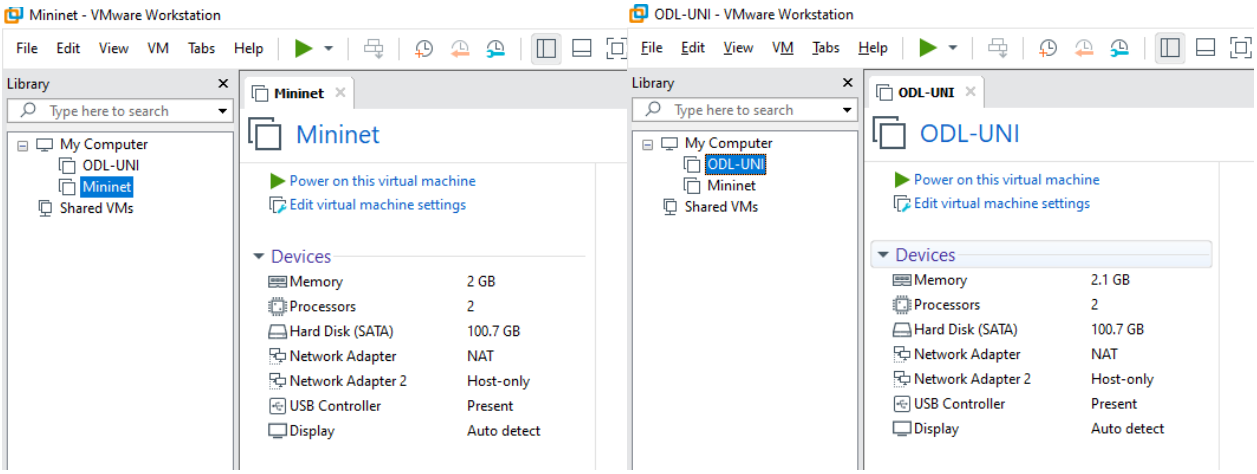

Figure 4.1 VMware Workstation characteristics (a) Main machine (b) Controller machine

By using Python codes on the *main machine*, we simulated our network and provided a *tree-topology* network with a depth of d = 4 and a fanout of f = 3. The number of hosts in the *tree-topology* network is calculated by equation (1), where the number of levels of switches is indicated by depth. In addition, fanout specifies the number of output ports that switches or hosts use for connection in topology [138]. Considering the formulation of the number of nodes in *tree-topology* in *Mininet* and our values for depth and fanouts, we will have a network with 81 host nodes. The "createTopology" function (Appendix A) details our method for providing our *tree-topology SDN* in *Mininet*. A preview of our network provided by *OpenDaylight* Web UI is presented in Figure 4.2.

$$No.hosts = f^d \qquad where\ f = number\ of\ \text{Fanouts}, \qquad (4.1)$$

$$\text{d} = \text{Depth of tree}$$

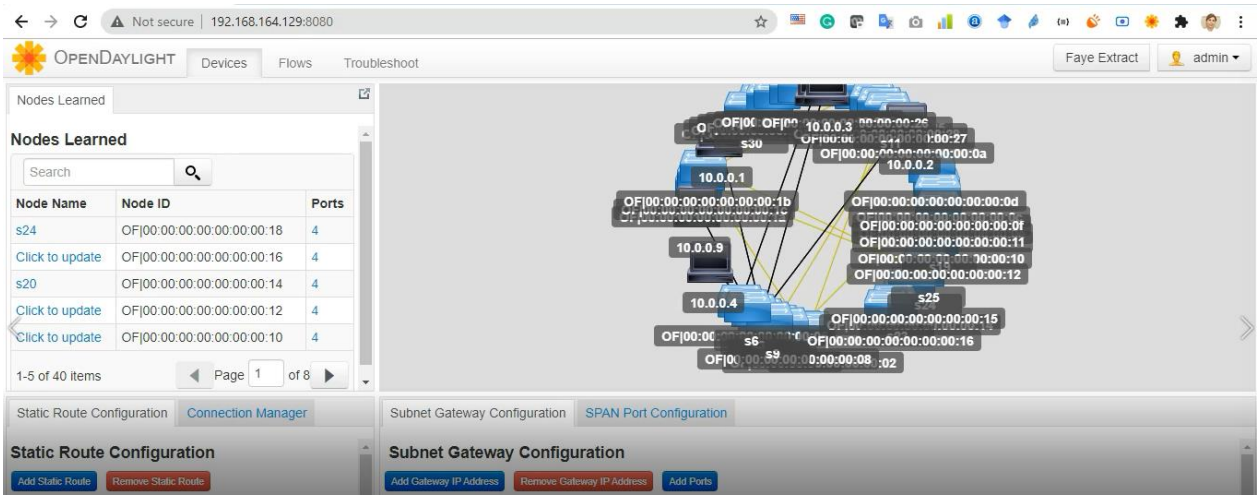

Figure 4.2 View of tree-topology network (depth= 4 and fanout= 3)

In the *controller machine*, we set up our *OpenDaylight* controller, and for this purpose, we install and configure the *Lithium* version [139]. In addition, the *SDN* in the *main machine* and the *ODL* controller in the *controller machine* are used to manage the flow controls in our simulated network.

4.2 Implementation of IoMT Health Data Layer (IHDL)

In this section, we describe our developments for each of the three modules of *IHDL*: *ECG* module, *Denoising* module and *Tokenization* module (cf. Chap. 3, §3.2). In addition, we present our results for the *Denoising* module evaluation. We use Python to implement these three modules.

4.2.1 ECG module Implementation

In Chapter 3 (cf. Chap. 3, §3.2.1), we explained that the first module in *IHDL* is the *ECG* module, which is used to prepare the data in our model. This section presents our selected *ECG* dataset for the *ECG* module in the *Denoising* module. This data will be denoised using our suggested *DWT* denoising algorithm. There are plenty of popular public datasets for *ECG* data that are used for *ECG* denoising regarding the literature review in chapter 2 (cf. Chap. 2, §2.4). Among all of them, we will work on a combination of two public datasets from the PhysioBank database [115], which is a large and growing archive of physiological data. Both selected datasets are WaveForm Databases (WFDB) based on *MIT* format standards. Other information for these two *ECG* datasets is:

- PhysioNet/Computing in Cardiology Challenge dataset [115]: It has 1000 records, and its data is based on standard 12-lead *ECG* recordings for 10 seconds with full diagnostic bandwidth;
- PTB Diagnostic *ECG* Database [140]: It has 549 records, and its data is based on standard 12-lead *ECG* recordings with full diagnostic bandwidth.

The *MIT* Format is one of the WaveForm Database format standard categories, and it includes three types of files [115]:

- *Signal*: They are binary files containing samples of digitized signals. They store the WaveForm, but they cannot be adequately interpreted without their corresponding *header* files. The extension for these files is in the form of .dat;
- *Header*: They are short text files that describe the contents of associated *signal* files. The extension for these files is .hea;

- *Annotation*: They are binary files containing labels or annotations for referring to samples in associated *signal* files. These annotation files should be read with their associated *header* files, and they are named with the .atr extension.

We use the *MIT* format and after reading a signal based on these three files, we convert it to a Jason file. This step is accomplished by reading *ECG MIT* data using the "wfdb library", and then converting them to Jason using the "ecgInfo library" in Python. In addition, we prepare a code to plot the *ECG* signals using the same libraries in our code (Appendix D).

4.2.2 Denoising Module Implementation

The second module of our first layer is the *Denoising* module (cf. Chap. 3, §3.2.2), where we used the *Discrete Wavelet Transform (DWT)*. In Python, we can use the "PyWavelets library" to implement the *Wavelet Transform (WT)* and the "pywt.dwt()" or "pywt.wavedec()" commands in this library to apply the *DWT*. This command returns two sets of coefficients: *approximation coefficients* and *detail coefficients*. In the *DWT*, the *approximation coefficients* represent the output of the low-pass filter, and the *detail coefficients* represent the output of the high-pass filter.

Regarding the "PyWavelets library", we have seven types of *wavelet families* in *DWT*: *Haar*, *Daubechies*, *Symlets*, *Coiflets*, *Biorthogonal*, *Reverse biorthogonal* and *Discrete Meyer (FIR Approximation)*. These *mother wavelets* or *wavelet families* vary in the degree of compression and smoothness of the wavelet, and we can choose the one that best matches the standard shape of our signal. We plot *mother wavelets* for *DWT* (Appendix E), and they are presented in Figure 4.3.

In this step, we need to select the *mother wavelet* that would be most similar to the standard *ECG* signal shape. In the studied articles (cf. Chap. 2, §2.4), among the existing *wavelet families*, *Dabuchies* and *Symlets* family wavelets have been introduced as appropriate wavelets for *ECG* signals [105] [110]. Given that the criterion for selecting a wavelet is its degree of similarity to the signal under study, and by examining the plotted shape of these mother wavelets, it can be intuitively seen that these two *wavelet families* are similar in shape to the pulses in the *ECG* signals. In this dissertation, we select the *Symlets* family, and Figure 4.4 displays its resemblance to a standard *ECG* signal.

After decomposing the noise signal into *approximation coefficients* (low-pass filter) and *detail coefficients* (high-pass filter) using *DWT*, a portion of the frequency spectrum can be filtered by removing some *detail coefficients*. If the signal contains a significant amount of high frequency noise, this approach may assist in denoising that signal. The undesired portion of the *detail coefficients* may be deleted using the pywt.threshold() function. Indeed, this command eliminates values of specific coefficients that exceed a specified threshold. After this step, we can reconstruct the signal using the remaining coefficients and combine them step by step, and finally get the signal without noise [116] [117].

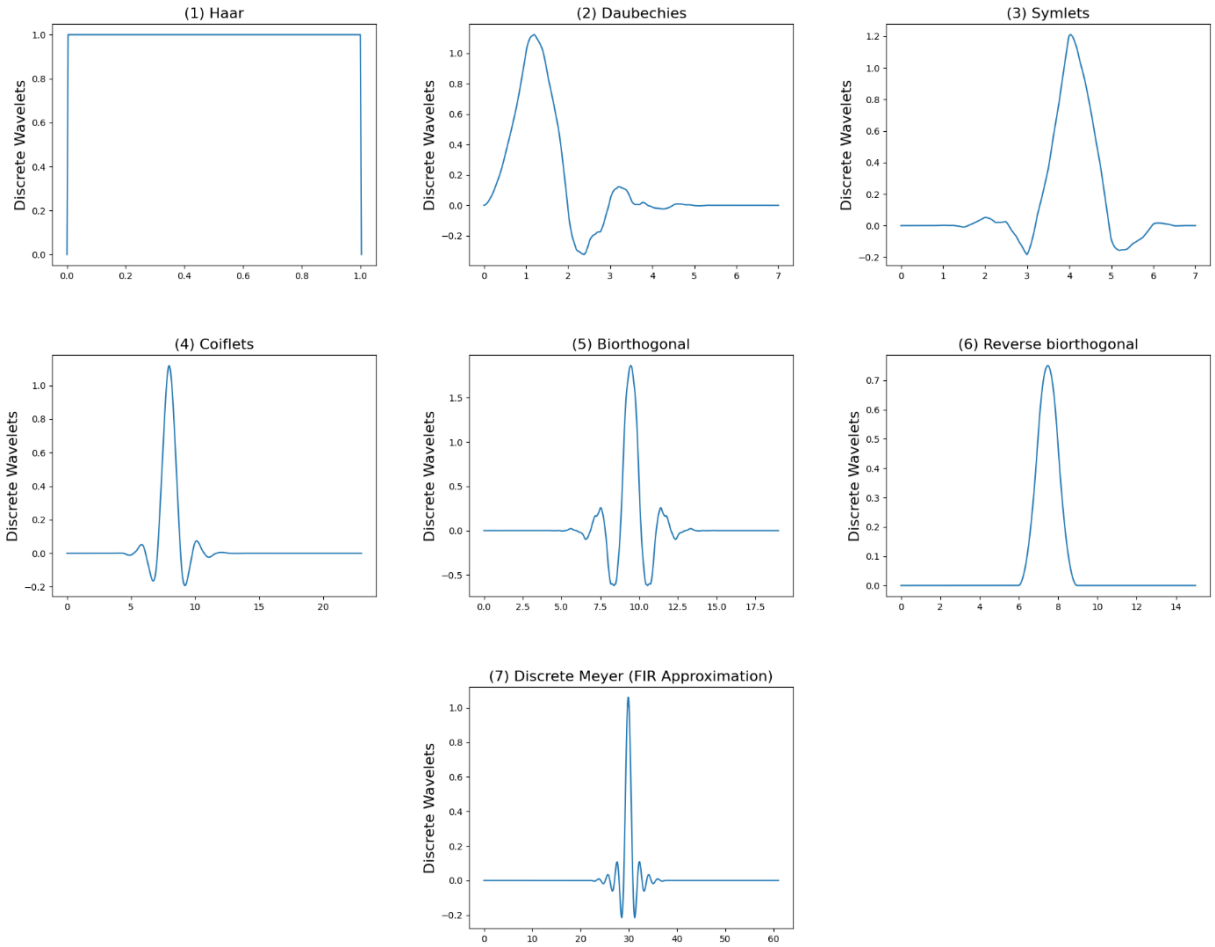

Figure 4.3 Mother wavelets for Discrete Wavelet Transform.

(1) Haar, (2) Daubechies, (3) Symlets, (4) Coiflets, (5) Biorthogonal, (6) Reverse biorthogonal, (7) Discrete Meyer (FIR Approximation)

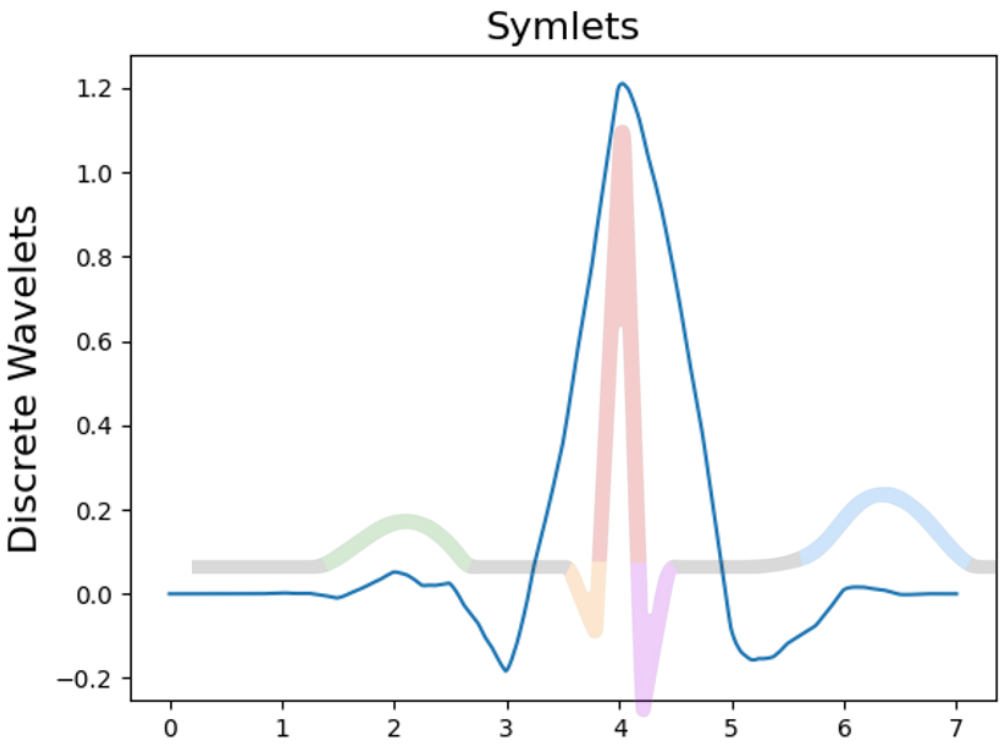

Figure 4.4 Similarity of Symlets mother wavelets and standard ECG signal

In Wavelet, there are four basic approaches to thresholding, which will be discussed as follows [106] [141] [142]:

- *Minimax*: This strategy, which is used in statistics to develop estimators, employs a fixed threshold determined for its calculation. It is designated by equation (2).
- *Rigorous Sure*: It can be applied to the soft threshold estimator based on the quadratic loss function. Equation (3) depicts this thresholding approach.
- *Universal*: It is another strategy that can be employed instead of the minimax threshold. It is bigger in magnitude than the minimax threshold, and its value is derived using equation (4).

- *Heuristic Sure:* It is a heuristic variant of the rigorous sure approach. We can consider it as a mixture of the two previous methods by equation (5).

$$\boldsymbol{\lambda_M} = \sigma\lambda_n^* \tag{4.2}$$

$$\boldsymbol{\lambda_S} = argmin_{0<\lambda<\lambda_u}\, Sure\left(\lambda,\ \frac{s(a,b)}{\sigma}\right) \tag{4.3}$$

$$\boldsymbol{\lambda_{UNIV}} = \sigma\sqrt{2\,log(N)} \tag{4.4}$$

$$\boldsymbol{\lambda_H} = \lambda_S + \lambda_{UNIV} \tag{4.5}$$

The noisy signal is represented by λ, while the coefficients are represented by σ in these equations. The σ can be a fixed value in the first formula, or it can be calculated by using the median value of the signal and probable error for normal distribution in other equations. In our proposed method, we calculate the *DWT* threshold based on the *universal* approach [106] [142].

To summarise what has been done in this section, we proposed a *Discrete Wavelet Transform* technique for denoising *ECG* signals, implemented in Python. In our proposed strategy, the *Symlets wavelet family* has been considered as the *mother wavelet*, while the coefficient threshold was calculated using the *universal thresholding* approach. To get the best results, we used eight decomposition cycles in our approach to signal analysis (Appendix F). We plotted a sample of our results. Figure 4.5 displays an example of a noisy signal and the result for our suggested technique.

For evaluating our *Denoising* module results, we have four metrics as follows:

- *Signal-to-Noise Ratio (SNR)*;
- *Percent Root Distortion (PRD);*
- *Mean Square Error (MSE);*

- *Root MSE (RMSE).*

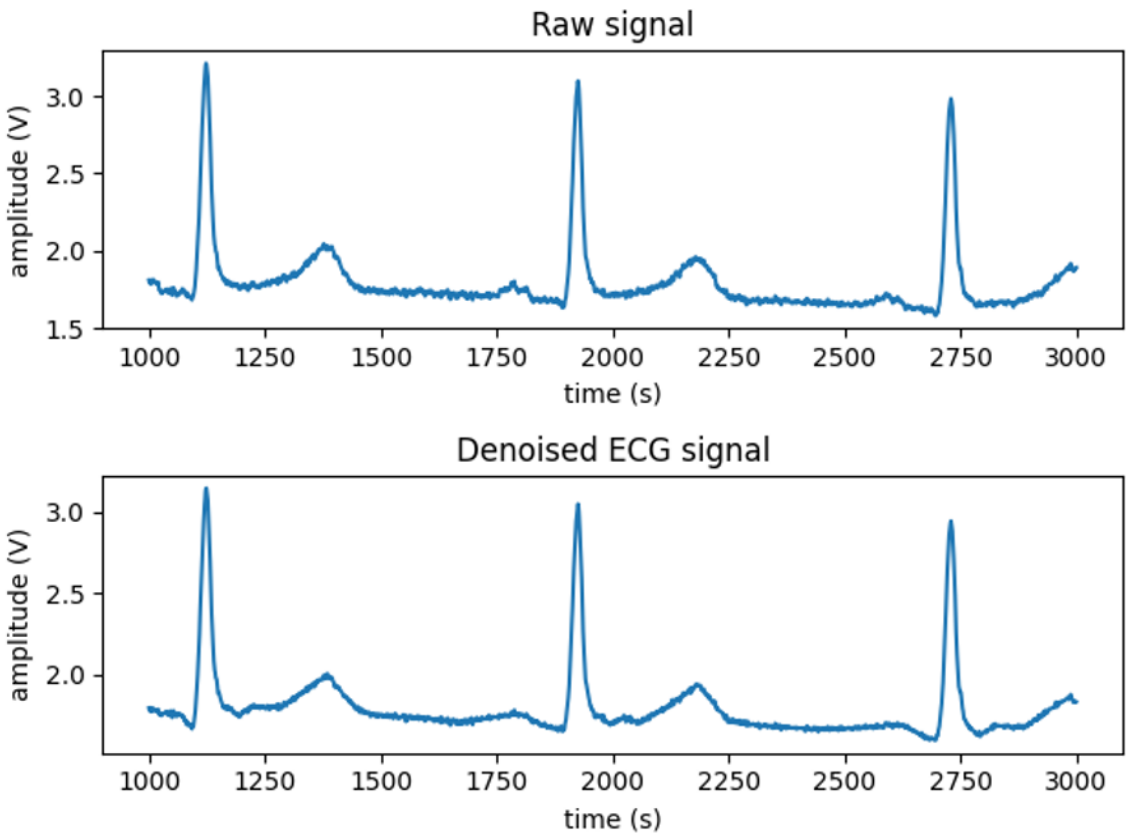

Figure 4.5 A sample of raw ECG signal and denoising signal

To evaluate our *DWT* results, we took two random samples from each dataset, then calculated *SNR* and *MSE* for each record. Higher values for *SNR* indicate better performance of a denoising *ECG* approach and it means the result signals are more accurate. *MSE* should be kept to a minimum since it highlights mean square error in the results. Table 4.1 displays our findings and Table 4.2 shows the results of three earlier previous works [6] [7] [14] for denoising ECG signal. By comparing our results to earlier works, we improved *SNR* values at both the minimum and maximum values. Figure 4.6 compares our SNR values with these three earlier previous works. In addition, the proposed denoising solution provides acceptable *MSE* values compared to these three earlier previous works.

Table 4.1 MSE and SNR results of our denoising approaches

Dataset	Signal Name	MSE	SNR	Min MSE	Max MSE	Min SNR	Max SNR
PhysioNet/Computing in Cardiology Challenge (CICC)	2981187	0.003	16.92	0.003	0.006	10.61	16.92
	1003574	0.006	10.61				
PTB Diagnostic ECG Database (PTB)	s0205_re	0.031	15.527	0.031	0.031	12.36	15.527
	s0341lre	0.031	12.36				

Table 4.2 MSE and SNR results of earlier previous works for denoising ECG signal

Dataset	Min MSE	Max MSE	Min SNR	Max SNR
Significance of non-local means estimation in DWT based ECG signal denoising [108]	0	0.001	8.44	9.68
An improved multivariate wavelet denoising method using subspace projection [143]	-	-	11.5	12
Hybrid IIR/FIR Wavelet Filter Banks for ECG Signal Denoising[109]	0.02	0.06	11.19	13.33

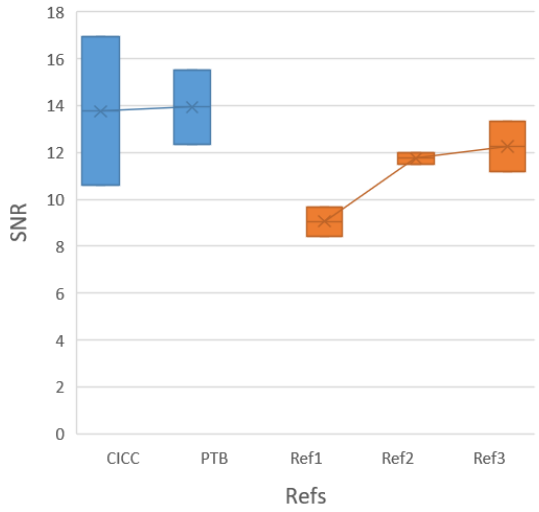

Figure 4.6 Comparing our denoising approaches and the three earlier previous works on SNR

4.2.3 Tokenization Module Implementation

The third module in *IHDL* layer is the *Tokenization* module (cf. Chap. 3, §3.2.3) and we are implementing this module using Python. Before that, we will explain the structure and procedure of tokenization with *JWT*. An example of a *JWT* token for a sample *Jason* data and using "LARIM" for the private key is presented in Table 4.3 [119].

Table 4.3 Sample of JWT and how it works

Header (Algorithm & Token type)	{
	"alg": "HS256",
	"typ": "JWT"
	}

Payload (data)	{
	"subject": "dissertation",
	"name": "faye"
	}

Verify signature	HMACSHA256(base64UrlEncode(header) + "." + base64UrlEncode(payload), LARIM) secret base64 encoded
Result	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWJqZWN0IjoiZGlzc2VydGF0aW9uIiwibmFtZSI6ImZheWUifQ.1-bJv7c-Qqu7dK_hWcDj4Hz-tSLfWNuJWpSGFKMcDKU

Typically, the header part has two sections that define the kind of token and the algorithm that was used to create it. *JWT* tokens are used here, and the methods are often HMAC SHA256 [24] [119] or RSA [24] [119]. The payload part of the token contains our data and some metadata. Both parts are encoded with the base 64 algorithm, as we can see in the result with red and green color. Then the signature is created based on the header and payload.

As we saw, all data is encoded using the base 64 algorithm, which is simply decoded in any system, and anybody may produce such a token and transmit it to us, but not with the third part (i.e.,

signature). In the signature part, the header part is appended to the second part (i.e., payload) and then encrypted with the private key (private key only accessible on the server) and the result is used as the third party (i.e., our token signature). If the payload data is tampered with, the server's signature is invalid. Additionally, since users do not have a private key, they are unable to produce tokens on their own [24] [119].

We implemented this approach to encode and encrypt *ECG* data and to increase the security of our data before it is sent to our network. At first, we will read *ECG* data stored in the *MIT* format and convert it to Jason file, which is compatible with HL7 format. Then, we will encrypt this data using *JWT* and a private key. Therefore, we will transform each *ECG* file into a Jason file that is encrypted and encoded using *JWT*. The code source for this part will be developed using the "jwt library" (Appendix D).

4.2.4 Summary of IHDL

We first constructed our dataset by integrating two publicly available *ECG* datasets. Then, we implemented the *Discrete Wavelet Transform (DWT)* technique to apply our *ECG* denoising strategy for the *Denoising* module. At this stage, we used the *Symlets* family as our *mother wavelet* and the universal thresholding technique to establish the coefficient threshold and filter the *detail coefficients* (high-pass filter) in eight cycles to get the level eight coefficient. Finally, in the *Tokenization* module, we converted our *ECG* data to a Jason file and then used the *JWT* approach to encode and encrypt it to be more secure and trustworthy. Furthermore, since we converted the format to an HL7-approved format, the suggested model would be interoperable with other e-health systems. Now that we have this data, we can use it in our simulation environment in *Mininet* and using *SDN*.

4.3 Implementation of Application Layer (AL)

In chapter 3 (cf. Chap. 3, §3.4), we saw that in the *Application Layer (LA)*, we have two main modules: The *DDoS* module and the *Monitoring module*. The proposed *DDoS* module has two main functionalities (cf. Chap. 3, §3.4.1): *DDoS* attacks simulation and *DDoS* attacks detection. In this section, we will explain our implementation and then we will present our results for each functionality of the *DDoS* module and the *Monitoring module*, respectively.

4.3.1 DDoS Module implementation for *DDoS* attacks simulation

The first duty of our proposed *DDoS* module is *DDoS* attacks simulation (cf. Chap. 3, §3.4.1), and this section will outline how our *DDoS* module simulates our network's *DDoS* attack scenario. It is indicated that the network layer in our model prepared our network in a simulated *SDN* using a *tree-topology* (cf. §4.1). In addition, the proposed *ECG* module of the first layer extracted data from two available public *ECG* datasets. Then, the *Denoising* module in *IHDL* applied our *DWT* denoising technique to this data, and then the *Tokenization module* applied *JWT* and encrypted and encoded the denoised *ECG* data into separate files (cf. §4.2). Now, we are going to establish two distinct circumstances in our network using our *DDoS* module: *normal* and *attack*.

In a *normal* situation, we send our data to the network as files in two distinct threads. Senders and receivers are chosen randomly. Our source code has two classes that handle our normal simulated situations (Appendix G): "sendNormalData" and "normalflow".

In the *DDoS* attack scenario, we use hping3 to simulate *DDoS* attacks and connect with random senders and receivers using the "ddosFlow" and "sendDDoSData" classes in our code (Appendix H). We run our code for an attack situation for 85 minutes and for a normal situation for 160 minutes to prepare our dataset and train our model. In both situations, we execute our code on the *Mininet-Wifi* machine. Figure 4.7 is a schematic block diagram of a *DDoS* attack scenario provided by the *DDoS* module on our *5G Network Layer (5GNL)*. The data plane in this Figure refers to our *Mininet-Wifi* machine for our *Transmission* module, while the control plane refers to our *controller machine* for our *Controller* module.

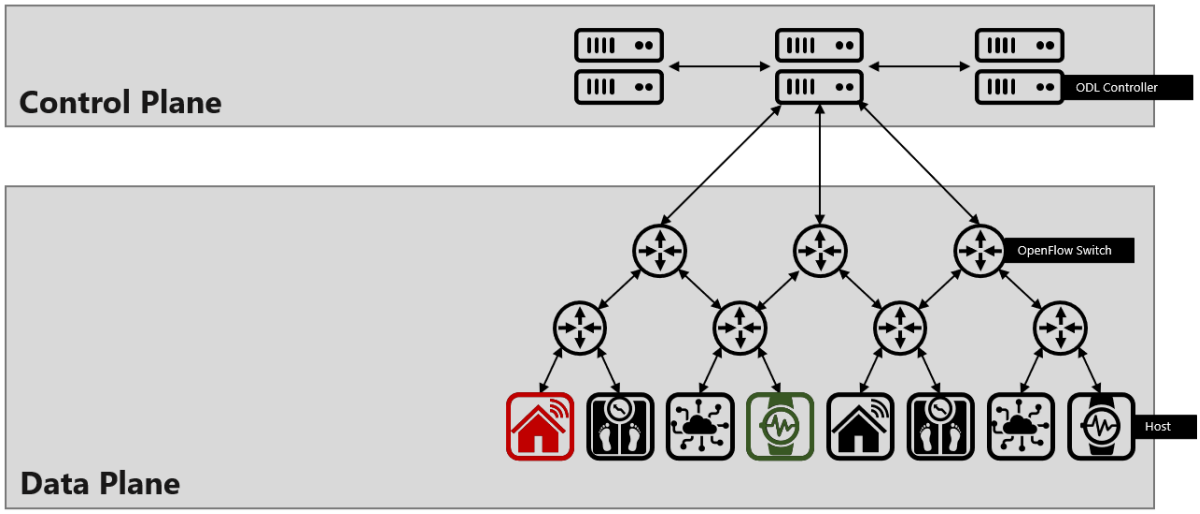

Figure 4.7 Schematic block diagram of a DDoS attacks scenario

4.3.2 Proposed DDoS attacks detection solution

The second duty of our proposed *DDoS* module is proposing a *DDoS* attacks detection solution (cf. Chap. 3, §3.4.1). The implementation of our *SVM* and *Machine Learning-based* solution is presented in this section. The proposed solution is divided into four main phases: *data collection*, *feature extraction*, *training phase* and *detection of the DDoS attacks*.

Data collection phase: For implementing this phase, we developed our solution with the *OpenDaylight* Rest API [120] and prepared our code in Python for using this API. As we mentioned, the northbound and southbound APIs in *ODL* are powerful assets for working with data, extracting them and updating them. This solution provides us with real monitoring capability, which helps us to propose our next module (i.e., *Monitoring module*). We have several APIs [133-135] in *OpenDaylight* and we are working with the statistical API available on:

http://[ControllerURL]:8080/controller/nb/v2/statistics/default/flow

We monitor the *ODL* controller and retrieve the traffic data every ten seconds using this API in both *normal* and *attack* situations. We chose the following information for our work from the plenty of data offered by the API in the result dataset: number of packets, number of bytes, durations and destination IPs. Figure 4.8 illustrates the plot of each data.

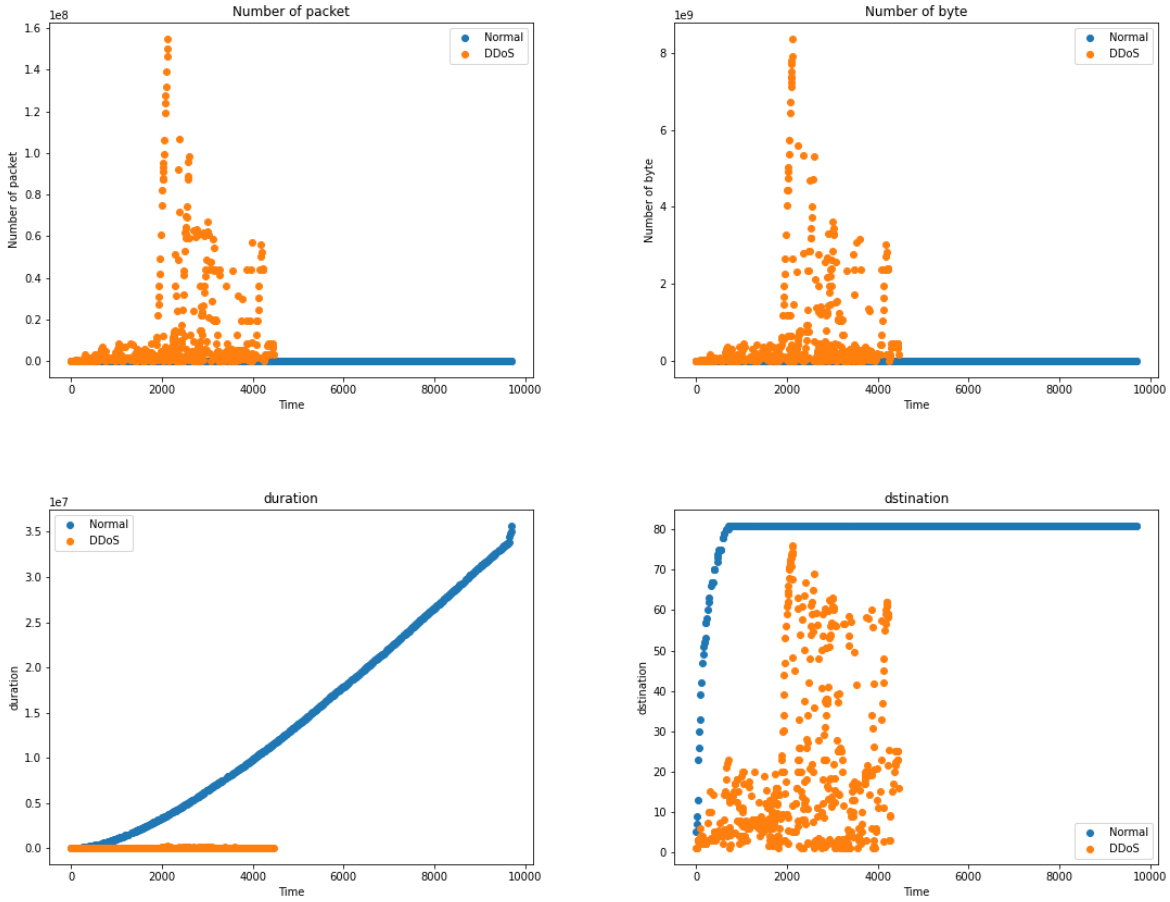

Figure 4.8 The data collected by ODL Rest API

Feature extraction phase: After completing the data collecting step for feature extraction, we used the entropy approach. In this phase, we used time-based entropy approach and for this aim, we defined the 10 seconds for the size of our window as we already collected the data every 10 seconds.

Therefore, the total number of each feature per 10 second period represents our chosen features based on our dataset and time-based entropy. In addition, we added two calculated features using our raw data, including the total number of packets per total duration times and the total bytes per total duration times. This phase is implemented in Python code, and the file "datacollection.py" contains both the data collection and feature extraction code. The output of our feature selection process would be a dataset with six distinct features, as shown in Figure 4.9. The results of this

phase are two Excel files that we named “normalFeaturesECG.csv” and “ddosFeaturesECG.csv” (Appendix I).

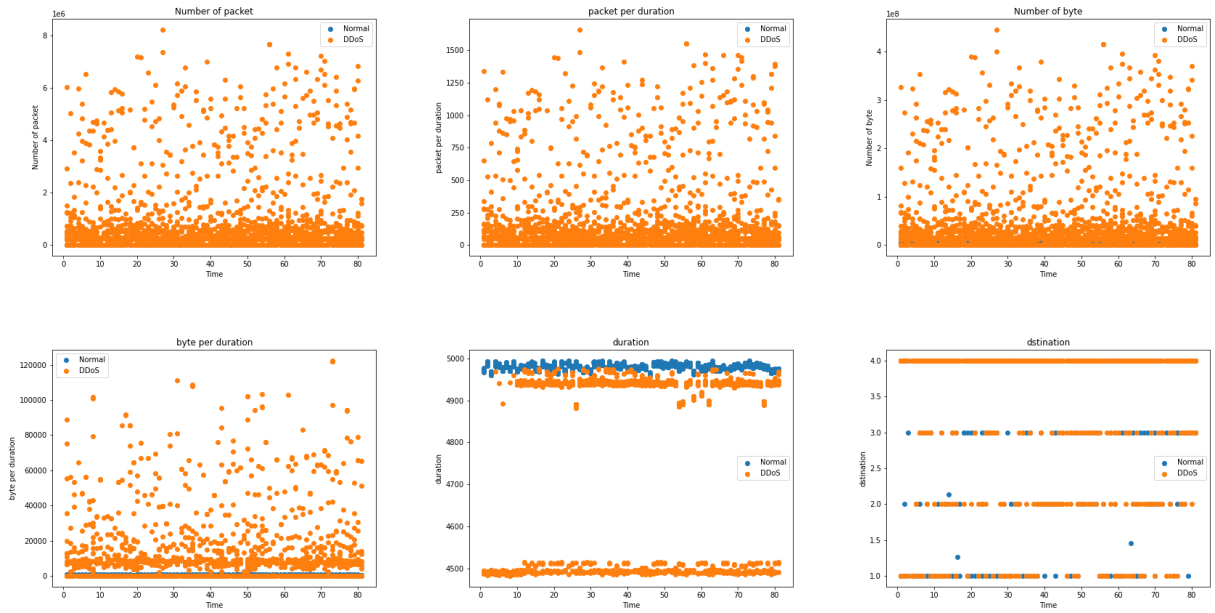

Figure 4.9 Entropy of the selected features

Training phase: In the training phase, we employed the *upsampling* approach and, more precisely, the *SMOTE* function to handle imbalanced data in our model. It is done by *SMOTE* function in “imblearn.over_sampling” Python library (Appendix J). Figure 4.10 shows the results of this function for one of our selected features.

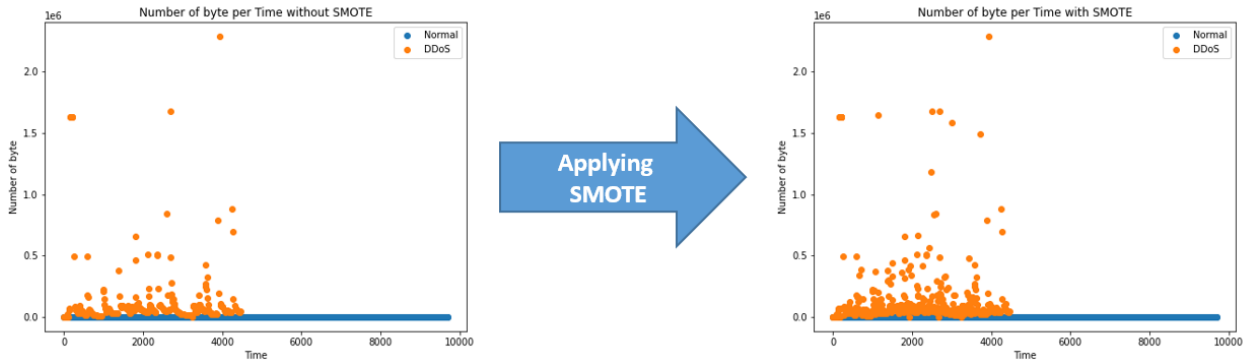

Figure 4.10 Sample result of our upsampling SMOTH method

Detection of the DDoS attacks phase: We now have sufficient data to run the *SVM* algorithm and train our model. The provided dataset in the previous phase is divided into a training set, a validation set, and a test set with 60%, 20%, and 20%, respectively. "GridSearchCV" was used to conduct cross-validation on our model. This implies that "GridSearchCV" uses distinct data portions to test and train a model across several iterations, in order to get the optimum results (Appendix J). Finally, we determine the optimal values for the following *SVM* algorithm parameters:

- C: 1.0;
- coef0: 0.0;
- Degree: 1;
- Gamma: scale;
- Kernel: linear.

For evaluating our results by *SVM*, we provide our performance results for the train, validation and test sets by applying *F1 score*. Table 4.4 presents our results for the *SVM algorithm* and the *DDoS attacks detection solutions*.

Table 4.4 F1 score for the proposed DDoS attacks detection solution

Dataset	F1 Score
Train Data	0.997
Validation Data	0.995
Test Data	1

In *ML evaluations Precision* and *Recall* provide an *F1 score*. The *Precision* is the ratio of True Positives to False Positives, and it shows the *accuracy*. As a result, the poor precision indicates a high rate of False Positives. The *Recall* is the ratio of True Positives to False Negatives and a metric of completeness. Therefore, a high rate of False Negatives provides a low recall value. The *F1 score* is calculated as the harmonic mean of *Precision* and *Recall*, which provides a considerably more accurate representation of the model. Thus, a low rate of false positives and negatives are equal when a good F1 score [144].

This model can be applied to any other datasets that fulfill our requirements and detects the *DDoS* attacks in offline mode using the given datasets. Moreover, the resulting model of this phase is recorded in the file “SVCModel.sav”, and we will use this model in the next module and propose our real-time *DDoS attacks detection* approach.

4.3.3 Monitoring Module implementation

The last module in our *Application* layer is the *Monitoring module* (cf. Chap. 3, §3.4.2), which provides a real-time monitoring capability inside the *ODL* User Interface (UI). In our source code, we included the “detectDDoS.py” and “mlResults.py” classes for real-time *DDoS attacks detection* (Appendix K). The first class, established in the previous section (cf. §4.3.2), gets real-time network data from our *OpenDaylight* controller via the *ODL* rest API. The raw data is manipulated to give the features specified in the feature extraction phase. Then, it runs the second class, “mlResults.py”, to classify this data as a *normal* or an *attack* situation. The findings are instantly shown in the console.

In addition, we provided a chrome extension for *ODL*'s web-based User Interface. The data for this chrome extension is also provided via the “mlResults.py” class. Therefore, we do not need to run any additional program or application to identify *DDoS* attacks on our network, and we can monitor the network state in real-time and only by using the *OpenDaylight* Web User Interface. In other words, we may use any device or operating system to discover our results as long as chrome is available on them. The chrome extension's source code is provided by HTML, CSS and JavaScript (Appendix L). Figure 4.11 depicts the *ODL* Web UI and our proposed chrome extension. The blue color indicates a *normal* situation, whereas the red color denotes an *attack* situation. The extension's data will be refreshed every 3 seconds.

Figure 4.11 ODL chrome extension

4.4 Summary

This chapter presented the implementation of our proposed model. First, we established our testbed in two distinct virtual machines by using *Mininet-Wifi* for *SDN* and *OpenDayLight*. Then, we discussed the implementation for the three modules in the *IoMT Health Data Layer (IHDL)*: The *ECG* module, the *Denoising* module and the *Tokenization* module. For *ECG* module, we prepared a code for reading *MIT* format data, plotted and converted them to Jason. The implementation of *JWT* for denoising *ECG* signals is proposed for the *Denoising* module. Then, we implemented our *JWT* method for encoding and encrypting our data.

In the *Application* Layer, we presented our approach for simulating *DDoS* attacks by the *DDoS module*. After, we provided our development for *DDoS* attacks detection in the same module. Finally, we discussed our implementation to provide real-time monitoring in the *Monitoring module*, which is done by CSS, HTML, JavaScript, *ODL* rest API and Python.

Table 4.5 provides an overview of the proposed model, the technologies and the enhancements depending on each layer in our model based on both chapters 3 and 4.

Table 4.5 Summary of our proposed model

Layers	Module	Technologies/ dataset	Duty	Achievements
IoMT Health Data Layer (IHDL) (cf. Chap. 3, §3.2)	ECG module	Public datasets, Python, wfdb library	Read, plot, ECG MIT format.	Enhance interoperability in data transmission, ability to work with heterogeneous data.
	Denoising module	Python, PyWavelets library	Denoise the ECG signals.	Enhance reliability of data.
	Tokenization module	Python, JWT library	Convert ECG MIT format to Jason, Encode and encrypt the ECG data.	Enhance security of data, increase scalability.
5G Network Layer (5GNL) (cf. Chap. 3, §3.3)	Transmission module	SDN, Mininet-Wifi	Managing network and transmission process	Add security by SDN, increase flexibility and network scalability
	Controller module	ODL, ODL Rest API, Python	Extract network data, present the live DDoS attacks detection results.	Provide easy access to data in both real-time and offline mode, increase network scalability
Application Layer (AL) (cf. Chap. 3, §3.4)	DDoS module	ML, SVM, Python, ODL Rest API	Simulate normal and DDoS attacks situation, provide time-based entropy features, train SVM model for detecting DDoS, propose offline DDoS attacks detection	Increase network security, detect DDoS attacks on 5G networks using SDN
	Monitoring module	Python, HTML, JavaScript, CSS, ODL Rest API	Propose real-time DDoS attacks detection, provide multiplatform monitoring tool	Increase network security, detect DDoS attacks on 5G networks using SDN

CHAPTER 5 CONCLUSION

This dissertation presented our three-layer model for securing the *IoMT health monitoring system* through 5G. In addition, we addressed the suggested model's outcomes and the implementation phases of this model. This chapter concludes the dissertation. The first section provides a summary of the work. The second section details the limitations of our work. Finally, the third section discusses the future works.

5.1 Summary of works

Healthcare's significance in our lives is evident, and we are experiencing significant advancement in this sector every day. Therefore, healthcare is a system that is addressed in smart cities. Healthcare services gain tremendously from IoT in a smart city. These systems collect health data using health sensors and Internet of Things (IoT) technology. Moreover, 5G technology and its features will be critical in accelerating the growth of the IoT and smart cities. The *Remote Health Monitoring System* is an application for health services in smart cities. Due to sensitive data in the healthcare domain and security vulnerabilities on the Internet of Things, security can be considered a critical component in a *Remote Health Monitoring System*.

By conducting the literature review on the health monitoring systems, *e-health*, *IoMT* and *smart cities*, we identify several weaknesses and shortcomings in existing systems, including lack of security at various levels, difficulties in dealing with heterogeneous data, lack of data interoperability, lack of scalability and data transmission latency. Therefore, several questions come to mind. How can we enhance the level of security for the remote monitoring health system to protect vital and sensitive health information from security vulnerabilities and unauthorized access? How can we achieve a real-time and scalable monitoring health system that ensures the system's availability and compatibility with a variety of IoT devices? Which interoperability frameworks are now being used to merge patient-generated health data with electronic health records?

Regarding all weaknesses mentioned above, this dissertation proposes a model for the remote monitoring health system using wearable *IoMT* devices and 5G network capabilities to ensure security in smart cities. In addition, the proposed model should support heterogeneous data and

interoperability between them to boost model scalability. In this model, we aim to develop a 5G-capable fast data *Transmission* module to boost speed and reduce latencies.

To accomplish our objectives, we developed a three-layer model consisting of the *IoMT Health Data Layer (IHDL)*, the *5G Network Layer (5GNL)* and the *Application Layer (AL)*. In this model, we considered 5G applications and technologies, and we provided two security enhancements at the data level and network level. The outcomes of the proposed security approaches are acceptable, and in the following paragraphs, we will discuss each layer, its results, and how it contributes to achieving our objectives.

The *IoMT Health Data Layer (IHDL)* is initially composed of three fundamental components: an *ECG* module, a *Denoising* module and a *Tokenization* module. *IHDL* is responsible for obtaining our health data from *IoT*/*IoMT* sensors and devices, and securely preparing it for the other layers of our model. The *ECG* module is the first of these three modules, and it provides data on *ECG* signals by integrating two public datasets (i.e., PhysioNet/Computing in Cardiology Challenge (CICC) and PTB Diagnostic ECG Database (PTB)) [115] [140]with MIT format and data from 12 leads. This data is used to monitor the heart's health state, and it is a necessary component of any remote health monitoring system. In addition, precise and trustworthy *ECG* data are extremely required for the accurate determination of the patient's health status.

The second layer of the *IoMT Health Data Layer (IHDL)* is the *Denoising* module, which uses the *Discrete Wavelet Transform (DWT)* to reduce noise in *ECG* data, and thus increase their robustness, security and reliability. The proposed *DWT* calculates the *DWT* of the *ECG* signal x by passing it through the low pass filter and the high pass filter, and decomposing it into approximation and detail coefficients, respectively. Then, we filtered these approximation and detail coefficients using Symlets mother wavelet [105] and a universal thresholding technique. We performed this procedure eight times and then combined the results to get our denoised *ECG* signal.

We evaluated our results using *Signal-to-Noise Ratio (SNR)* and *Mean Square Error (MSE)* for two random samples from each dataset. Then, we compared them with the three earlier previous works [108] [109] [143] for denoising ECG signal. Therefore, our *SNR* value ranges from 10.61 to 16.92 for the PhysioNet/Computing in Cardiology Challenge (CICC) dataset, and between 12.36

and 15.527 for the PTB Diagnostic ECG Database (PTB) dataset. By comparing our results to earlier works, we improved *SNR* values at both the minimum and maximum values.

Tokenization is the last module in the first layer (*IHDL*). It converts the denoised signal given by the *Denoising* module to Jason, a standard format for health monitoring systems. Jason enables us to manage heterogeneous data and enhances data interoperability via information processing. Jason is compliant with the *HL7* standard and supports the *FHIR* standard. In addition, the *Tokenization* module, used *JWT* to enhance the security of the data to make it reliable and more secure. By encrypting and encoding data and delivering it as JSON objects, *JWT* enables the secure movement of data between parties. Therefore, the *Tokenization* module encodes our denoised *ECG* data using the Base64 technique and encrypts it using the HMAC algorithm for the header, the Secure Hash Algorithm (SHA256), and the private key for the contents in the *JWT* token. The data is then digitally signed, and since users do not have access to the private key, they cannot edit tokens and it is secure now.

Up to this point, we have added data security and interoperability due to the information processing by converting the data to Jason in the *IoMT Health Data Layer*, which is a valid format for *HL7* and the majority of standard health systems. Moreover, by using this standardized data format, we can manage structured and unstructured data efficiently and increase our capacity for working with heterogeneous data. The second layer in the proposed model is the *5G Network Layer (5GNL)*, which is responsible for network administration and data transmission. This layer has two modules: The *Transmission* module and the *Controller* module. The *Transmission* module is responsible for network management and provides the required infrastructure to safely transfer data between nodes through *SDN*. By establishing physically distinct layers for infrastructure, control and application, *SDN* makes our network more secure than conventional networks. Therefore, network control is dynamic, programmable and adaptable. *Mininet-wifi* is used to simulate our *Transmission* module.

In *5GNL*, the *Controller* module is responsible to assist the *Transmission* module in accomplishing its goals and acting as the controller for *SDN* in our *Transmission* module. Moreover, it provides data about our network's state for the *Application Layer*. In addition, we used *Lithium OpenDaylight* to create a scalable and multiprotocol controller for our *Transmission* module, which allows us to connect more devices and enhances interoperability.

The *Application Layer (AL)* is the last layer in our model. The primary function of this layer is to increase the security of the two preceding layers (i.e., *IHDL* and *5GNL*) in the proposed model. Despite the advantages of *SDN* in network management and the enhancement of the security compared to traditional networks, *SDN* still contains some security problems. One of the most serious vulnerabilities associated with *SDN* is a *Distributed Denial of Service (DDoS)* attack. Therefore, in this layer, we proposed our solution for *DDoS* attacks detection based on *SDN* and *ODL*.

The Application Layer is divided into two modules: the *DDoS* module and the *Monitoring* module. The *DDoS* module is responsible for two primary functions: simulating *DDoS* attacks at the *5G Network Layer (5GNL)* and detecting *DDoS* attacks through *SVM* classification. We used *hping3* [100] to simulate *DDoS* attacks using random senders and recipients in our network. The dataset includes an *attack* scenario lasting 85 minutes and a *normal* scenario lasting 160 minutes.

Then, in the *DDoS* module, we applied the *SVM* algorithm to identify *DDoS* attacks. Our approach to detecting *DDoS* attacks consists of four main phases: *data collection*, *feature extraction*, *training phase* and *detection of the DDoS attacks*. During the data collection phase, we used the *ODL* Rest API to gather information about our network, such as the number of packets, the number of bytes, the durations, and the destination IP addresses. Then, for the feature extraction step, we used a time-based entropy with 10 seconds for the size of our window to determine the entropy of our network data. Moreover, two computed characteristics are added: total packets per total duration time and total bytes per total duration time. Following that, in the training phase, we used the *SMOTE* function in conjunction with an *upsampling* approach to deal with unbalanced data in our model.

Finally, in the *detection of the DDoS attacks* phase, the provided dataset is divided into a training set, a validation set and a test set with 60%, 20% and 20%, respectively. Then, using these datasets and the cross-validation in our implementation, we trained our *SVM* model with the optimal parameters being *C: 1.0*, *coef0: 0.0*, *Degree: 1*, *Gamma: scale*, and *Kernel: linear*. Therefore, in this phase, we provided our offline *DDoS* attacks detection solution by preserving the model and applying it to other datasets. For evaluating our results by *SVM*, we provide our performance results

for the train, validation and test sets by applying *F1 score*. The results are 0.99 for the train and validation sets and 1 for the test set, indicating that our performance for detecting *DDoS* attacks in *SDN*, while we are sending *ECG* data, is almost 100% accurate.

Our last module in *Application Layer (LA)* is the *Monitoring* module, which aims to present the results of the *DDoS* module and our *DDoS* attacks detection solution, as well as provide live monitoring functionality in the *5GNL*. Regarding our training model, we gathered data in real-time through the *ODL* Rest API and enabled real-time *DDoS* attacks detection. The results are presented inside the *OpenDaylight* UI by providing an *ODL* extension and a real-time monitoring tool. It is worth mentioning that our model is scalable since data is collected through the *ODL* controller.

5.2 Limitations

Our proposed model has several limitations regarding our objectives at each layer of the proposed architecture. In the first layer, *IoMT Health Data Layer (IHDL)*, we just considered *ECG* data. Therefore, there is a limitation in covering additional *IoMT* devices and working with other health data, such as temperature, blood pressure and so on. In addition, we did not include heterogeneous data in our testing and evaluation of our model and its capability to operate with heterogeneous data. Another weakness in our model is that we had a limitation of access to standard health systems for verifying interoperability in our model that we presented by converting data to Jason format.

In the *5G network layer*, the primary weakness is that we used a simulator to construct our testbed, and the result might be different in a real situation in a 5G network. Moreover, we noticed that there are multiple *SDN* controllers, and we just examined one of them, *ODL*. Therefore, our model is incompatible with other *SDN* controllers, particularly with commercial controllers that are used in the real world.

Moreover, we were constrained by the lack of a real-world test environment for our application layer and had to rely on simulated data to validate our *SVM* model. Another weakness of our *DDoS* attacks detection model is that we only consider hping3 when simulating *DDoS* attacks in our network, ignoring other tools and solutions for providing *DDoS* attacks in the network.

5.3 Future Work

The first improvement in our proposed model would be expanding the number of *IoMT* variations and providing real data from wearable *IoMT* devices. With this enhancement, we can collect real-world data from wearable *IoMT* devices, particularly commercial devices, to improve the quality of our health data. Another possibility for improving our *Denoising* module is to implement more denoising algorithms, evaluate the results, and then choose the optimal strategy for this purpose.

Instead of employing simulators for the *Transmission* module, we can offer a real testbed for 5G networks and *SDN*, in our *5G Network Layer*. Additionally, we may include additional *SDN* controllers into our design or use several *OpenDaylight* controllers to provide a more complex network topology in our model. For future work, there are plenty of other *DDoS* attacks detection approaches and we can propose other approaches, then compare the results and select the most efficient approach for our work. Regarding the security issues, we only addressed the *DDoS* attacks in our model; however, more security attacks might become the focus of our security solution in the future.

Another possibility is to design a cloud-based health monitoring system that includes all works for security in our model, and to implement a real application for patients and healthcare professionals to monitor their health conditions. In our work, we considered this point and converted our health data to Jason format to provide optimum interoperability with current health care systems. However, the obtained data in Jason format could be integrated with current open source and cloud-based infrastructures. One of these options is the novel tool *Fast Healthcare Interoperability Resources (FHIR) FHIR* [118] [145], which runs on *Amazon Web Services (AWS)*. Another tool for this objective could *be Google's Cloud Healthcare API [146],* which enables rapid development of healthcare solutions that are managed, enterprise-scale, and compliant with the *HL7* and *FHIR* standards.

REFERENCES

[1] E. Theodoridis, G. Mylonas, and I. Chatzigiannakis, "Developing an IoT Smart City framework," in *IISA 2013*, 2013, pp. 1-6.

[2] L. Chettri and R. Bera, "A Comprehensive Survey on Internet of Things (IoT) Toward 5G Wireless Systems," *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 16-32, 2020.

[3] N. Dlodlo, O. Gcaba, and A. Smith, "Internet of things technologies in smart cities," in *2016 IST-Africa Week Conference*, 2016, pp. 1-7.

[4] J. M. d. Fuentes, L. Gonzalez-Manzano, A. Solanas, and F. Veseli, "Attribute-Based Credentials for Privacy-Aware Smart Health Services in IoT-Based Smart Cities," *Computer*, vol. 51, no. 7, pp. 44-53, 2018.

[5] M. Sidorov, P. V. Nhut, Y. Matsumoto, and R. Ohmura, "LoRa-Based Precision Wireless Structural Health Monitoring System for Bolted Joints in a Smart City Environment," *IEEE Access*, vol. 7, pp. 179235-179251, 2019.

[6] M. S. Hossain, C. Xu, S. Göbel, and A. E. Saddik, "IEEE Access Special Section Editorial: Advances of Multisensory Services and Technologies for Healthcare in Smart Cities," *IEEE Access*, vol. 6, pp. 62335-62338, 2018.

[7] G. Hatzivasilis, O. Soultatos, S. Ioannidis, C. Verikoukis, G. Demetriou, and C. Tsatsoulis, "Review of Security and Privacy for the Internet of Medical Things (IoMT)," in *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2019, pp. 457-464.

[8] H. Ahmadi, G. Arji, L. Shahmoradi, R. Safdari, M. Nilashi, and M. Alizadeh, "The application of internet of things in healthcare: a systematic literature review and classification," *Universal Access in the Information Society*, vol. 18, no. 4, pp. 837-869, 2019/11/01 2019.

[9] A. Papa, M. Mital, P. Pisano, and M. Del Giudice, "E-health and wellbeing monitoring using smart healthcare devices: An empirical investigation," *Technological Forecasting and Social Change*, vol. 153, p. 119226, 2020/04/01/ 2020.

[10] F. Lamonaca *et al.*, "An Overview on Internet of Medical Things in Blood Pressure Monitoring," in *2019 IEEE International Symposium on Medical Measurements and Applications (MeMeA)*, 2019, pp. 1-6.

[11] G. J. Joyia, R. M. Liaqat, A. Farooq, and S. Rehman, "Internet of medical things (IoMT): Applications, benefits and future challenges in healthcare domain," *J. Commun.*, vol. 12, no. 4, pp. 240-247, 2017.

[12] L. D. Vito, F. Lamonaca, G. Mazzilli, M. Riccio, D. L. Carnì, and P. F. Sciammarella, "An IoT-enabled multi-sensor multi-user system for human motion measurements," in *2017 IEEE International Symposium on Medical Measurements and Applications (MeMeA)*, 2017, pp. 210-215.

[13] M. A. Khan and F. Algarni, "A Healthcare Monitoring System for the Diagnosis of Heart Disease in the IoMT Cloud Environment Using MSSO-ANFIS," *IEEE Access,* vol. 8, pp. 122259-122269, 2020.

[14] L. Piwek, D. A. Ellis, S. Andrews, and A. Joinson, "The rise of consumer health wearables: promises and barriers," *PLoS medicine,* vol. 13, no. 2, p. e1001953, 2016.

[15] E. Ezhilarasan and M. Dinakaran, "A Review on Mobile Technologies: 3G, 4G and 5G," in *2017 Second International Conference on Recent Trends and Challenges in Computational Models (ICRTCCM)*, 2017, pp. 369-373.

[16] B. Bangerter, S. Talwar, R. Arefi, and K. Stewart, "Networks and devices for the 5G era," *IEEE Communications Magazine,* vol. 52, no. 2, pp. 90-96, 2014.

[17] J. Jusak, H. Pratikno, and V. H. Putra, "Internet of Medical Things for cardiac monitoring: Paving the way to 5G mobile networks," in *2016 IEEE International Conference on Communication, Networks and Satellite (COMNETSAT)*, 2016, pp. 75-79.

[18] A. Khalid, J. J. Quinlan, and C. J. Sreenan, "MiniNAM: A network animator for visualizing real-time packet flows in Mininet," in *2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN)*, 2017, pp. 229-231.

[19] M. Erel, E. Teoman, Y. Özçevik, G. Seçinti, and B. Canberk, "Scalability analysis and flow admission control in mininet-based SDN environment," in *2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*, 2015, pp. 18-19.

[20] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," *Proceedings of the IEEE,* vol. 103, no. 1, pp. 14-76, 2015.

[21] C. D. Cajas and D. O. Budanov, "SDN Applications and Plugins in the OpenDaylight Controller," in *2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, 2020, pp. 9-13.

[22] W. Li, "Wavelets for Electrocardiogram: Overview and Taxonomy," *IEEE Access,* vol. 7, pp. 25627-25649, 2019.

[23] M. Golgowski and S. Osowski, "Anomaly detection in ECG using wavelet transformation," in *2020 IEEE 21st International Conference on Computational Problems of Electrical Engineering (CPEE)*, 2020, pp. 1-4.

[24] M. Haekal and Eliyani, "Token-based authentication using JSON Web Token on SIKASIR RESTful Web Service," in *2016 International Conference on Informatics and Computing (ICIC)*, 2016, pp. 175-179.

[25] B. Mladenov, "Studying the DDoS Attack Effect over SDN Controller Southbound Channel," in *2019 X National Conference with International Participation (ELECTRONICA)*, 2019, pp. 1-4.

[26] K. Huang, L. X. Yang, X. Yang, Y. Xiang, and Y. Y. Tang, "A Low-Cost Distributed Denial-of-Service Attack Architecture," *IEEE Access,* vol. 8, pp. 42111-42119, 2020.

[27] K. Kato and V. Klyuev, "An intelligent ddos attack detection system using packet analysis and support vector machine," *IJICR*, vol. 14, no. 5, p. 3, 2014.

[28] J. Ye, X. Cheng, J. Zhu, L. Feng, and L. Song, "A DDoS Attack Detection Method Based on SVM in Software Defined Network," *Security and Communication Networks*, vol. 2018, p. 9804061, 2018/04/24 2018.

[29] M. M. Dhanvijay and S. C. Patil, "Internet of Things: A survey of enabling technologies in healthcare and its applications," *Computer Networks*, vol. 153, pp. 113-131, 2019.

[30] J. J. Hathaliya, S. Tanwar, S. Tyagi, and N. Kumar, "Securing electronics healthcare records in healthcare 4.0: a biometric-based approach," *Computers & Electrical Engineering*, vol. 76, pp. 398-410, 2019.

[31] O. Salem, K. Alsubhi, A. Mehaoua, and R. Boutaba, "Markov Models for Anomaly Detection in Wireless Body Area Networks for Secure Health Monitoring," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 2, pp. 526-540, 2021.

[32] J. Peral, A. Ferrández, D. Gil, R. Muñoz-Terol, and H. Mora, "An Ontology-Oriented Architecture for Dealing With Heterogeneous Data Applied to Telemedicine Systems," *IEEE Access*, vol. 6, pp. 41118-41138, 2018.

[33] K. Sangkla and P. Seresangtakul, "Information Integration of Heterogeneous Medical Database Systems Using Metadata," in *2017 21st International Computer Science and Engineering Conference (ICSEC)*, 2017, pp. 1-5.

[34] E. Park, J. H. Kim, H. S. Nam, and H. J. Chang, "Requirement Analysis and Implementation of Smart Emergency Medical Services," *IEEE Access*, vol. 6, pp. 42022-42029, 2018.

[35] A. V. Jerald and S. A. Rabara, "Secured Architecture for Internet of Things (IoT) Based Smart Healthcare," in *2020 International Conference on Inventive Computation Technologies (ICICT)*, 2020, pp. 828-833.

[36] A. Srilakshmi, P. Mohanapriya, D. Harini, and K. Geetha, "IoT based Smart Health Care System to Prevent Security Attacks in SDN," in *2019 Fifth International Conference on Electrical Energy Systems (ICEES)*, 2019, pp. 1-7.

[37] S. E. Kafhali, K. Salah, and S. B. Alla, "Performance Evaluation of IoT-Fog-Cloud Deployment for Healthcare Services," in *2018 4th International Conference on Cloud Computing Technologies and Applications (Cloudtech)*, 2018, pp. 1-6.

[38] S. Sengupta and S. S. Bhunia, "Secure Data Management in Cloudlet Assisted IoT Enabled e-Health Framework in Smart City," *IEEE Sensors Journal*, vol. 20, no. 16, pp. 9581-9588, 2020.

[39] S. Shukla, M. F. Hassan, M. K. Khan, L. T. Jung, and A. Awang, "An analytical model to minimize the latency in healthcare internet-of-things in fog computing environment," *PLoS One*, vol. 14, no. 11, p. e0224934, 2019.

[40] S. Shukla, M. F. Hassan, L. T. Jung, A. Awang, and M. K. Khan, "A 3-tier architecture for network latency reduction in healthcare internet-of-things using fog computing and machine learning," in *Proceedings of the 2019 8th International Conference on Software and Computer Applications*, 2019, pp. 522-528.

[41] J. G. Ramani, S. Madhusudan, A. L. Nila, A. Pradeep, and S. Manibharathi, "IOT Based Employee Health Monitoring System," in *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, 2020, pp. 298-301.

[42] Z. Ali, G. Muhammad, and M. F. Alhamid, "An Automatic Health Monitoring System for Patients Suffering From Voice Complications in Smart Cities," *IEEE Access*, vol. 5, pp. 3900-3908, 2017.

[43] M. Irmansyah, E. Madona, A. Nasution, and R. Putra, "Low Cost Heart Rate Portable Device for Risk Patients with IoT and Warning System," in *2018 International Conference on Applied Information Technology and Innovation (ICAITI)*, 2018, pp. 46-49.

[44] T. Shaown, I. Hasan, M. M. R. Mim, and M. S. Hossain, "IoT-based Portable ECG Monitoring System for Smart Healthcare," in *2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*, 2019, pp. 1-5.

[45] D. A. M. Budida and R. S. Mangrulkar, "Design and implementation of smart HealthCare system using IoT," in *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, 2017, pp. 1-7.

[46] S. Chaudhury, D. Paul, R. Mukherjee, and S. Haldar, "Internet of Thing based healthcare monitoring system," in *2017 8th Annual Industrial Automation and Electromechanical Engineering Conference (IEMECON)*, 2017, pp. 346-349.

[47] S. Sudevan and M. Joseph, "Internet of Things: Incorporation into Healthcare Monitoring," in *2019 4th MEC International Conference on Big Data and Smart City (ICBDSC)*, 2019, pp. 1-4.

[48] S. P *et al.*, "IoT based Healthcare Monitoring and Intravenous Flow Control," in *2020 International Conference on Computer Communication and Informatics (ICCCI)*, 2020, pp. 1-6.

[49] S. Gupta, D. Dahiya, and G. Raj, "Remote Health Monitoring System Using IoT," in *2018 International Conference on Advances in Computing and Communication Engineering (ICACCE)*, 2018, pp. 300-305.

[50] J. Venkatesh, B. Aksanli, C. S. Chan, A. S. Akyurek, and T. S. Rosing, "Modular and Personalized Smart Health Application Design in a Smart City Environment," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 614-623, 2018.

[51] P. Manikanta, S. S. K. Hussian, and R. T. Kodi, "Iot Ambulance With Automatic Traffic Light Control," in *2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN)*, 2019, pp. 1-3.

[52] Y. Cai, F. Ouyang, Y. Zhang, Y. Huang, J. Chen, and H. Liu, "Design and Implementation of Regional Health Information Collection Transmission and Integration System," in *2019 International Conference on Virtual Reality and Intelligent Systems (ICVRIS)*, 2019, pp. 382-385.

[53] D. Fang, Y. Qian, and R. Q. Hu, "Security for 5G Mobile Wireless Networks," *IEEE Access*, vol. 6, pp. 4850-4874, 2018.

[54] F. Alsubaei, A. Abuhussein, and S. Shiva, "Security and Privacy in the Internet of Medical Things: Taxonomy and Risk Assessment," in *2017 IEEE 42nd Conference on Local Computer Networks Workshops (LCN Workshops)*, 2017, pp. 112-120.

[55] M. A. Mahdi and S. T. Hasson, "A Contribution to the Role of the Wireless Sensors in the IoT Era," *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, vol. 9, no. 2-11, pp. 1-6, 2017.

[56] M. Sharifnejad, M. Shari, M. Ghiasabadi, and S. Beheshti, "A survey on wireless sensor networks security," in *4th International Conference: Sciences of Electronic, Technologies of Information and Telecommunications (SEIT), TUNISIA, March 2007*, 2007.

[57] I. Ahmad, T. Kumar, M. Liyanage, J. Okwuibe, M. Ylianttila, and A. Gurtov, "5G security: Analysis of threats and solutions," in *2017 IEEE Conference on Standards for Communications and Networking (CSCN)*, 2017, pp. 193-199.

[58] I. Ahmad, S. Shahabuddin, T. Kumar, J. Okwuibe, A. Gurtov, and M. Ylianttila, "Security for 5G and Beyond," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3682-3722, 2019.

[59] I. Ahmad, T. Kumar, M. Liyanage, J. Okwuibe, M. Ylianttila, and A. Gurtov, "Overview of 5G Security Challenges and Solutions," *IEEE Communications Standards Magazine*, vol. 2, no. 1, pp. 36-43, 2018.

[60] Q. Cheng, C. Hsu, Z. Xia, and L. Harn, "Fast Multivariate-Polynomial-Based Membership Authentication and Key Establishment for Secure Group Communications in WSN," *IEEE Access*, vol. 8, pp. 71833-71839, 2020.

[61] F. Conceicao, N. Oualha, and D. Zeghlache, "Security establishment for IoT environments in 5G: Direct MTC-UE communications," in *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, 2017, pp. 1-5.

[62] A. A. BUTT, G. R. CHUGHTA, A. KABIR, Z. MAHMOOD, and J. OLÁH, "Analysis of Key Establishment Techniques for Secure D2D Communication in Emerging 5G Cellular Networks," *Acta Montanistica Slovaca*, vol. 26, no. 3, 2021.

[63] D. Yubo, N. Jianwei, and L. Lian, "Modeling of Broadcasting Based on Distance Scheme for WSN," in *2009 Fifth International Joint Conference on INC, IMS and IDC*, 2009, pp. 1176-1179.

[64] J. S. Silva *et al.*, "Multicast and IP Multicast Support in Wireless Sensor Networks," *J. Networks*, vol. 3, no. 3, pp. 19-26, 2008.

[65] K. Xiao, L. Gong, and M. Kadoch, "Opportunistic Multicast NOMA with Security Concerns in a 5G Massive MIMO System," *IEEE Communications Magazine*, vol. 56, no. 3, pp. 91-95, 2018.

[66] H. Gao, Y. Zhang, T. Wan, J. Zhang, and H. Duan, "On Evaluating Delegated Digital Signing of Broadcasting Messages in 5G," in *2021 IEEE Global Communications Conference (GLOBECOM)*, 2021, pp. 1-7.

[67] G. Araniti, M. Condoluci, P. Scopelliti, A. Molinaro, and A. Iera, "Multicasting over Emerging 5G Networks: Challenges and Perspectives," *IEEE Network,* vol. 31, no. 2, pp. 80-89, 2017.

[68] D. Wu, G. Hu, and G. Ni, "Research and Improve on Secure Routing Protocols in Wireless Sensor Networks," in *2008 4th IEEE International Conference on Circuits and Systems for Communications*, 2008, pp. 853-856.

[69] S. Rajasoundaran, A. Prabu, S. Routray, P. P. Malla, G. S. Kumar, and Y. Qi, "Secure routing with multi-watchdog construction using deep particle convolutional model for IoT based 5G wireless sensor networks," *Computer Communications,* 2022.

[70] J. R. Ward and M. Younis, "A cross-layer defense scheme for countering traffic analysis attacks in Wireless Sensor Networks," in *MILCOM 2015 - 2015 IEEE Military Communications Conference*, 2015, pp. 972-977.

[71] S. Zhang, Y. Wang, and W. Zhou, "Towards secure 5G networks: A Survey," *Computer Networks,* vol. 162, p. 106871, 2019.

[72] N. Park, H. Hu, and Q. Jin, "Security and privacy mechanisms for sensor middleware and application in internet of things (IoT)," vol. 12, ed: SAGE Publications Sage UK: London, England, 2016, p. 2965438.

[73] C. Ioannou, V. Vassiliou, and C. Sergiou, "An Intrusion Detection System for Wireless Sensor Networks," in *2017 24th International Conference on Telecommunications (ICT)*, 2017, pp. 1-5.

[74] B. B. Zarpelão, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga, "A survey of intrusion detection in Internet of Things," *Journal of Network and Computer Applications,* vol. 84, pp. 25-37, 2017.

[75] A. Gupta, R. K. Jha, P. Gandotra, and S. Jain, "Bandwidth Spoofing and Intrusion Detection System for Multistage 5G Wireless Communication Network," *IEEE Transactions on Vehicular Technology,* vol. 67, no. 1, pp. 618-632, 2018.

[76] Y. Sang, H. Shen, Y. Inoguchi, Y. Tan, and N. Xiong, "Secure Data Aggregation in Wireless Sensor Networks: A Survey," in *2006 Seventh International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'06)*, 2006, pp. 315-320.

[77] Y. Zhang, D. Zheng, Q. Zhao, C. Lai, and F. Ren, "PADA: Privacy-Aware Data Aggregation with Efficient Communication for Power Injection in 5G Smart Grid Slice," in *2017 International Conference on Networking and Network Applications (NaNA)*, 2017, pp. 11-16.

[78] A. Ullah, M. Azeem, H. Ashraf, A. A. Alaboudi, M. Humayun, and N. Jhanjhi, "Secure Healthcare Data Aggregation and Transmission in IoT—A Survey," *IEEE Access,* vol. 9, pp. 16849-16865, 2021.

[79] W. Yan, A. Fu, Y. Mu, X. Zhe, S. Yu, and B. Kuang, "EAPA: Efficient attestation resilient to physical attacks for IoT devices," in *Proceedings of the 2nd International ACM Workshop on Security and Privacy for the Internet-of-Things*, 2019, pp. 2-7.

[80] B. Bordel, R. Alcarria, T. Robles, and M. S. Iglesias, "Data Authentication and Anonymization in IoT Scenarios and Future 5G Networks Using Chaotic Digital Watermarking," *IEEE Access*, vol. 9, pp. 22378-22398, 2021.

[81] M. A. Latif, M. B. Ahmad, and M. K. Khan, "A Review on Key Management and Lightweight Cryptography for IoT," in *2020 Global Conference on Wireless and Optical Technologies (GCWOT)*, 2020, pp. 1-7.

[82] N. Vlajic and D. Zhou, "IoT as a Land of Opportunity for DDoS Hackers," *Computer*, vol. 51, no. 7, pp. 26-34, 2018.

[83] H. Ghorbani, M. S. Mohammadzadeh, and M. H. Ahmadzadegan, "DDoS Attacks on the IoT network with the Emergence of 5G," in *2020 International Conference on Technology and Entrepreneurship - Virtual (ICTE-V)*, 2020, pp. 1-5.

[84] K. Kalkan, L. Altay, G. Gür, and F. Alagöz, "JESS: Joint Entropy-Based DDoS Defense Scheme in SDN," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2358-2372, 2018.

[85] K. S. Sahoo, D. Puthal, M. Tiwary, J. J. Rodrigues, B. Sahoo, and R. Dash, "An early detection of low rate DDoS attack to SDN based data center networks using information distance metrics," *Future Generation Computer Systems*, vol. 89, pp. 685-697, 2018.

[86] K. S. Sahoo, M. Tiwary, and B. Sahoo, "Detection of high rate DDoS attack from flash events using information metrics in software defined networks," in *2018 10th International Conference on Communication Systems & Networks (COMSNETS)*, 2018, pp. 421-424.

[87] M. Xuanyuan, V. Ramsurrun, and A. Seeam, "Detection and Mitigation of DDoS Attacks Using Conditional Entropy in Software-defined Networking," in *2019 11th International Conference on Advanced Computing (ICoAC)*, 2019, pp. 66-71.

[88] R. Li and B. Wu, "Early detection of DDoS based on <tex>φ</tex>-entropy in SDN networks," in *2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, 2020, vol. 1, pp. 731-735.

[89] T. V. Phan and M. Park, "Efficient Distributed Denial-of-Service Attack Defense in SDN-Based Cloud," *IEEE Access*, vol. 7, pp. 18701-18714, 2019.

[90] J. Cui, M. Wang, Y. Luo, and H. Zhong, "DDoS detection and defense mechanism based on cognitive-inspired computing in SDN," *Future generation computer systems*, vol. 97, pp. 275-283, 2019.

[91] M. Myint Oo, S. Kamolphiwong, T. Kamolphiwong, and S. Vasupongayya, "Advanced support vector machine-(ASVM-) based detection for distributed denial of service (DDoS) attack on software defined networking (SDN)," *Journal of Computer Networks and Communications*, vol. 2019, 2019.

[92] S. U. N. G, J. W, G. U. Y, R. E. N. D, and L. I. H, "DDoS Attacks and Flash Event Detection Based on Flow Characteristics in SDN," in *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2018, pp. 1-6.

[93] D. Hu, P. Hong, and Y. Chen, "FADM: DDoS Flooding Attack Detection and Mitigation System in Software-Defined Networking," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, 2017, pp. 1-7.

[94] T. Hurley, J. E. Perdomo, and A. Perez-Pons, "HMM-Based Intrusion Detection System for Software Defined Networking," in *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2016, pp. 617-621.

[95] M. P. Novaes, L. F. Carvalho, J. Lloret, and M. L. Proença, "Long Short-Term Memory and Fuzzy Logic for Anomaly Detection and Mitigation in Software-Defined Network Environment," *IEEE Access*, vol. 8, pp. 83765-83781, 2020.

[96] T. M. Nam *et al.*, "Self-organizing map-based approaches in DDoS flooding detection using SDN," in *2018 International Conference on Information Networking (ICOIN)*, 2018, pp. 249-254.

[97] J. Cui, J. He, Y. Xu, and H. Zhong, "TDDAD: Time-based detection and defense scheme against DDoS attack on SDN controller," in *Australasian Conference on Information Security and Privacy*, 2018, pp. 649-665: Springer.

[98] Y. Cui *et al.*, "SD-Anti-DDoS: Fast and efficient DDoS defense in software-defined networks," *Journal of Network and Computer Applications*, vol. 68, pp. 65-79, 2016/06/01/ 2016.

[99] K. Bhushan and B. B. Gupta, "Distributed denial of service (DDoS) attack mitigation in software defined network (SDN)-based cloud computing environment," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 5, pp. 1985-1997, 2019.

[100] A. AlEroud and I. Alsmadi, "Identifying cyber-attacks on software defined networks: An inference-based intrusion detection approach," *Journal of Network and Computer Applications*, vol. 80, pp. 152-164, 2017/02/15/ 2017.

[101] M. Conti, A. Gangwal, and M. S. Gaur, "A comprehensive and effective mechanism for DDoS detection in SDN," in *2017 IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, 2017, pp. 1-8.

[102] P. Xiao, Z. Li, H. Qi, W. Qu, and H. Yu, "An Efficient DDoS Detection with Bloom Filter in SDN," in *2016 IEEE Trustcom/BigDataSE/ISPA*, 2016, pp. 1-6.

[103] U. Satija, B. Ramkumar, and M. S. Manikandan, "A Review of Signal Processing Techniques for Electrocardiogram Signal Quality Assessment," *IEEE Reviews in Biomedical Engineering*, vol. 11, pp. 36-52, 2018.

[104] U. Satija, B. Ramkumar, and M. S. Manikandan, "Automated ECG Noise Detection and Classification System for Unsupervised Healthcare Monitoring," *IEEE Journal of Biomedical and Health Informatics*, vol. 22, no. 3, pp. 722-732, 2018.

[105] M. F. Amri, M. I. Rizqyawan, and A. Turnip, "ECG signal processing using offline-wavelet transform method based on ECG-IoT device," in *2016 3rd International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE)*, 2016, pp. 1-6.

[106] R. Chauhan, R. Dahiya, and P. Bansal, "Optimal choice of thresholding rule for denoising ECG using DWT," in *2017 4th International Conference on Signal Processing, Computing and Control (ISPCC)*, 2017, pp. 288-292.

[107] Y. Qian, "A wavelet denoising method based on improved threshold and autocorrelation," in *2018 Chinese Control And Decision Conference (CCDC)*, 2018, pp. 4058-4063.

[108] P. Singh and G. Pradhan, "Significance of non-local means estimation in DWT based ECG signal denoising," in *2018 5th International Conference on Signal Processing and Integrated Networks (SPIN)*, 2018, pp. 18-22.

[109] Y. Eminaga, A. Coskun, and I. Kale, "Hybrid IIR/FIR Wavelet Filter Banks for ECG Signal Denoising," in *2018 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, 2018, pp. 1-4.

[110] A. Gon and A. Mukherjee, "Removal of Noises from an ECG Signal Using an Adaptive S-Median Thresholding Technique," in *2020 IEEE Applied Signal Processing Conference (ASPCON)*, 2020, pp. 89-93.

[111] S. Shahbudin, S. N. Shamsudin, and H. Mohamad, "Discriminating ECG signals using Support Vector Machines," in *2015 IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE)*, 2015, pp. 175-180.

[112] M. İ. Hayıroğlu *et al.*, "A simple independent prognostic electrocardiography parameter in first acute anterior myocardial infarction; precordial total Q wave/precordial total R wave," *Journal of Electrocardiology*, vol. 51, no. 1, pp. 38-45, 2018.

[113] M. M. Nezhad and M. Eshghi, "Sensor Single and Multiple Anomaly Detection in Wireless Sensor Networks for Healthcare," in *2019 27th Iranian Conference on Electrical Engineering (ICEE)*, 2019, pp. 1751-1755.

[114] A. Ukil, S. Bandyoapdhyay, C. Puri, and A. Pal, "IoT Healthcare Analytics: The Importance of Anomaly Detection," in *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*, 2016, pp. 994-997.

[115] (09-Nov-2021). *PhysioNet/Computing in Cardiology Challenges*. Available: https://archive.physionet.org/challenge/2011/

[116] P. S. Addison, "Wavelet transforms and the ECG: a review," *Physiological measurement*, vol. 26, no. 5, p. R155, 2005.

[117] (2022, 18 March 2022). *A guide for using the Wavelet Transform in Machine Learning*. Available: https://ataspinar.com/2018/12/21/a-guide-for-using-the-wavelet-transform-in-machine-learning/

[118] A. Benhamida, A. Kanas, M. Vincze, K. T. Papp, M. Abbassi, and M. Kozlovszky, "SaECG: a new FHIR Data format revision to enable continuous ECG storage and monitoring," in *2020 IEEE 20th International Symposium on Computational Intelligence and Informatics (CINTI)*, 2020, pp. 000115-000120.

[119] (September 27, 2021). *JSON Web Tokens*. Available: https://jwt.io/

[120] B. Ribes Garcia, "OpenDaylight SDN controller platform," Universitat Politècnica de Catalunya, 2015.

[121] Y. Li, X. Guo, X. Pang, B. Peng, X. Li, and P. Zhang, "Performance Analysis of Floodlight and Ryu SDN Controllers under Mininet Simulator," in *2020 IEEE/CIC International Conference on Communications in China (ICCC Workshops)*, 2020, pp. 85-90.

[122] S. Wang, "Comparison of SDN OpenFlow network simulator and emulators: EstiNet vs. Mininet," in *2014 IEEE Symposium on Computers and Communications (ISCC)*, 2014, pp. 1-6.

[123] M. Chan, C. Chen, J. Huang, T. Kuo, L. Yen, and C. Tseng, "OpenNet: A simulator for software-defined wireless local area network," in *2014 IEEE Wireless Communications and Networking Conference (WCNC)*, 2014, pp. 3332-3336.

[124] J. Ivey, H. Yang, C. Zhang, and G. Riley, "Comparing a scalable SDN simulation framework built on ns-3 and DCE with existing SDN simulators and emulators," in *Proceedings of the 2016 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, 2016, pp. 153-164.

[125] A. H. M. Hassan, A. M. Alhassan, and F. Izzeldean, "Performance Evaluation of SDN Controllers in Ofnet Emulation Environment," in *2019 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE)*, 2019, pp. 1-6.

[126] P. Wette, M. Dräxler, A. Schwabe, F. Wallaschek, M. H. Zahraee, and H. Karl, "MaxiNet: Distributed emulation of software-defined networks," in *2014 IFIP Networking Conference*, 2014, pp. 1-9.

[127] M. A. M. Hasan, M. Nasser, B. Pal, and S. Ahmad, "Support vector machine and random forest modeling for intrusion detection system (IDS)," *Journal of Intelligent Learning Systems and Applications*, vol. 2014, 2014.

[128] R. Rustogi and A. Prasad, "Swift Imbalance Data Classification using SMOTE and Extreme Learning Machine," in *2019 International Conference on Computational Intelligence in Data Science (ICCIDS)*, 2019, pp. 1-6.

[129] (2020, March 27, 2022). *Handling Imbalanced Data- Machine Learning, Computer Vision, NLP*. Available: https://www.analyticsvidhya.com/blog/2020/11/handling-imbalanced-data-machine-learning-computer-vision-and-nlp/

[130] M. Duan, D. Liu, X. Chen, R. Liu, Y. Tan, and L. Liang, "Self-Balancing Federated Learning With Global Imbalanced Data in Mobile Systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 1, pp. 59-71, 2021.

[131] W. Lei, R. Zhang, Y. Yang, R. Wang, and W. S. Zheng, "Class-Center Involved Triplet Loss for Skin Disease Classification on Imbalanced Data," in *2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI)*, 2020, pp. 1-5.

[132] S. Bae and K. Yoon, "Polyp Detection via Imbalanced Learning and Discriminative Feature Learning," *IEEE Transactions on Medical Imaging*, vol. 34, no. 11, pp. 2379-2393, 2015.

[133] (2016, March 24, 2021). *simplecodepieces*. Available: https://simplecodepieces.wordpress.com/

[134] (2013, March 24, 2022). *openflow*. Available: https://fredhsu.wordpress.com/tag/openflow-2/

[135] (2019, March 24, 2022). *OpenFlow Plugin Operation*. Available: https://docs.opendaylight.org/projects/openflowplugin/en/latest/users/operation.html#openflow-statistics

[136] (April 2, 2021). *VMware Workstation 15.5.5 Pro Release Notes*. Available: https://docs.vmware.com/en/VMware-Workstation-Pro/15.5/rn/VMware-Workstation-1555-Pro-Release-Notes.html

[137] (April 2, 2021). *Ubuntu*. Available: https://ubuntu.com/download/desktop

[138] I. Z. Bholebawa and U. D. Dalal, "Design and performance analysis of OpenFlow-enabled network topologies using Mininet," *International Journal of Computer and Communication Engineering*, vol. 5, no. 6, p. 419, 2016.

[139] (20-Sep-2021, 10-Mar-2022). *OpenDaylight* [Online]. Available: https://www.opendaylight.org/

[140] (09-Nov-2021). *PTB Diagnostic ECG Database*. Available: https://physionet.org/content/ptbdb/1.0.0/

[141] Abba, "A Study on Wavelet Threshold Functions in Denosing," *International Journal of Applied Engineering Research*, vol. 14, no. 13, pp. 3081-3096, 2019.

[142] G. Garg, V. Singh, J. R. P. Gupta, and A. P. Mittal, "Optimal algorithm for ECG denoising using Discrete Wavelet Transforms," in *2010 IEEE International Conference on Computational Intelligence and Computing Research*, 2010, pp. 1-4.

[143] H. Hao, H. Wang, N. ur REHMAN, L. Chen, and H. Tian, "An improved multivariate wavelet denoising method using subspace projection," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 100, no. 3, pp. 769-775, 2017.

[144] (2022). *How to measure an AI models performance - F1 score explained*. Available: https://abeyon.com/ai-performance-measurement-f1score/

[145] (April 3, 2021). *Taproot: a story about a gardener and a ghost*. Available: https://aws.amazon.com/about-aws/whats-new/2020/12/introducing-fhir-works-on-aws/

[146] (April 3, 2021). *Cloud Healthcare API | Google Cloud*. Available: https://cloud.google.com/healthcare-api

[147] (2022). *The ultimate ECG book & course: learn ECG interpretation, videos, test/quiz*. Available: https://ecgwaves.com/

[148] L. D. Sharma and R. K. Sunkaria, "Detection and delineation of the enigmatic U-wave in an electrocardiogram," *International Journal of Information Technology*, pp. 1-8, 2019.

[149] M. Fahim and A. Sillitti, "Anomaly Detection, Analysis and Prediction Techniques in IoT Environment: A Systematic Literature Review," *IEEE Access*, vol. 7, pp. 81664-81681, 2019.

APPENDIX A ECG SIGNALS

Electrocardiography uses *ECG signals* for monitoring heart health conditions [44]. *ECG* provides a graph of voltage versus time based on *ECG* signals and it uses electrodes placed on the skin to measure the heart's electrical activities. For *ECG* systems, precise and trustworthy measurements of the morphological attributes of intervals and local waves are extremely required for the accurate determination of the characteristic points, also called fiducial points, of the *ECG* signal [103]. In *ECG* signal, amplitude, duration, polarity and shape are called morphological features [103] [111] [112]. Local waves of the *ECG* signal are listed below, and their schematic presentation is illustrated in Figure 3.3:

- P-wave: It reflects atrial depolarization or activation with a duration of 80 milliseconds (ms) on average [103] [111].
- Q-wave: It is the first negative wave in the *ECG* signal that are connected to myocardial contractile function [112].
- R-wave: Myocardial Infarction (MI) could be identified by ECG signals. One of the prognostic parameters in acute MI is R-wave. It is considered the first positive wave in a standard *ECG* signal [112].
- S-wave: This means the first negative wave occurring after R-wave [147].
- T-wave: It is another typical *ECG* wave for repolarization of the heart's ventricles [112].
- U-wave: It is not an ordinary wave, and it occasionally happens after T-wave. It could be used for identifying some certain cardiac disorders [148].

Considering *ECG* waves, there are several intervals and segments, such as the PR interval (the distance between the start of the P-wave and the start of the R-wave), the QT interval (starts from the Q-wave to the end of the T-wave), the PR segment (from the end of the p-wave to the start of the QRS complex) and the ST-segment [111]. These intervals and segments are shown in Figure 3.11.

As mentioned above that each of these waves and intervals represents a specific metric of a heart condition, and they usually have a standard range of duration considering the gender and the age

of the patient. Therefore, in e-health systems, these waves and intervals help health professionals to identify patient health conditions.

IoMT devices provide *ECG* signals and anomalies could affect the quality of this vital data. Anomaly is the events and the observations that deviate from a dataset's normal behavior. The anomaly detection identifies rare items, occurrences or observations that raise suspicions by differing significantly from the majority of the data [149].

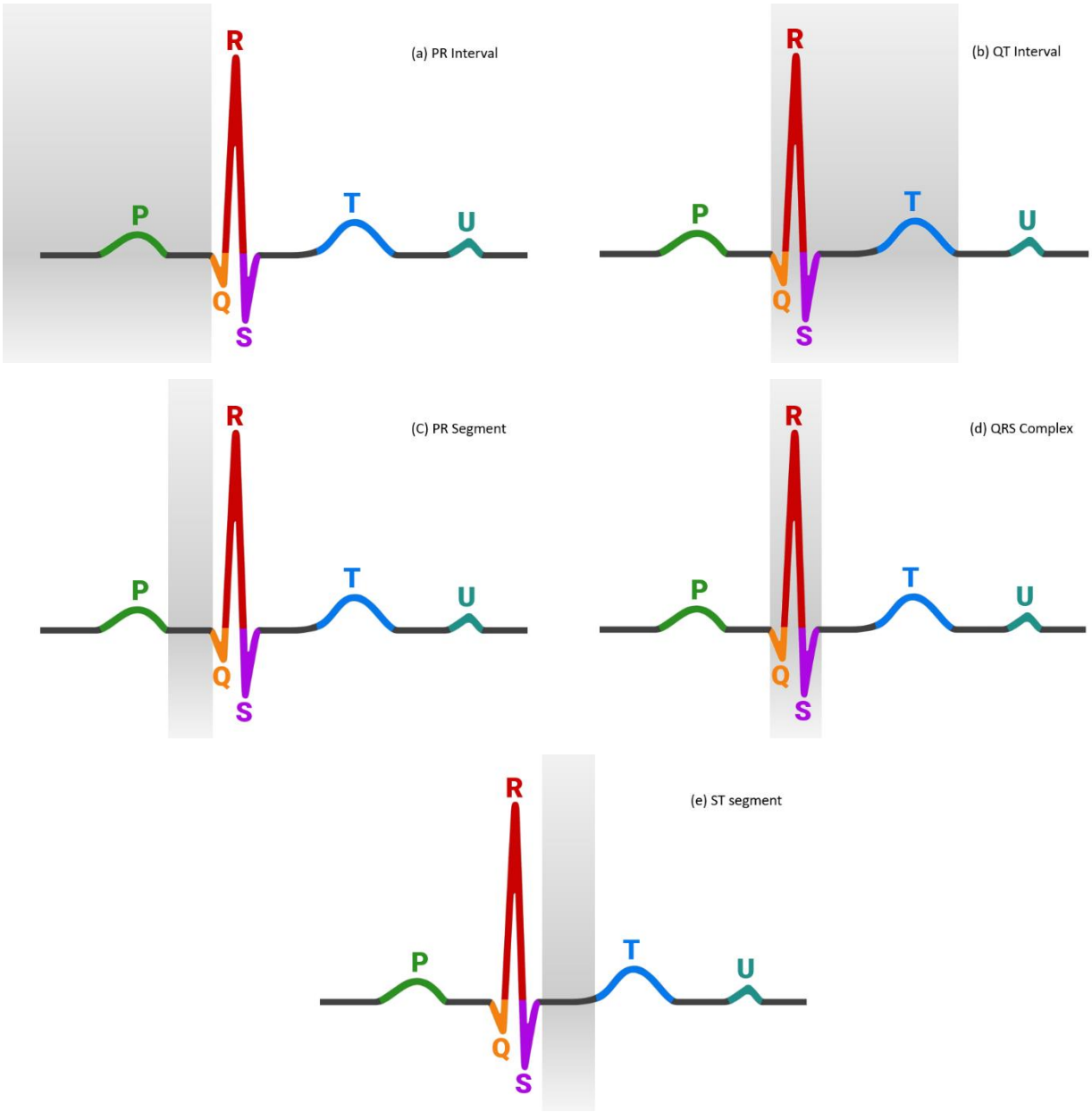

Figure 5.1 ECG signal intervals and segments.

(a) PR interval, (b) QT interval, (c) PR segment, (d) QRS complex and (e) ST-segment

These anomaly sources in *ECG* data could add noise to the actual data and cause inaccurate health interpretation. A summary of *ECG* noise sources and their temporal and spectral characteristics is *baseline wander, Power Line Interference (PLI), Electromyogram (EMG)* and *instrumentation*. *Baseline wander* is a low-frequency artifact in the *ECG* that arises from breathing, electrically charged electrodes or subject movement. It affects the ST-segment detection. *Power Line Interference (PLI)* is a significant noise source in the *ECG* signal, severely affecting its interpretation. In other words, *PLI* is a high-frequency additive noise that occurs because of the coupling of power line frequency with the signal carrying cables. The electrical activity of muscles causes *Electromyogram* or muscle artifacts during periods of contraction or due to a sudden body movement. Finally, *instrumentation* noise comes from the accuracy of the device and its configuration [103].

APPENDIX B SUPPORT VECTOR MACHINE (SVM)

SVM is a linear classifier, and there is a trade-off between maximum margin and accuracy when selecting the hyperplane. In this case, *SVM* chooses the hyperplane that correctly classifies the classes before maximizing the margin. This approach is illustrated in Figure 3.12. The support vectors indicated by the dot and the black hyperplane in the center are based on the maximum margin, but they are not accurate. Moreover, the optimal hyperplane indicated by the green color is the selected hyperplane by *SVM* that accurately classifies the sample data. It is worth mentioning that a common concern in linear classification problems is the presence of outliers in the data, and *SVM* classification is robust to outliers. Another significant strength of *SVM* is its capacity to tackle non-linear problems [28] [91] [127].

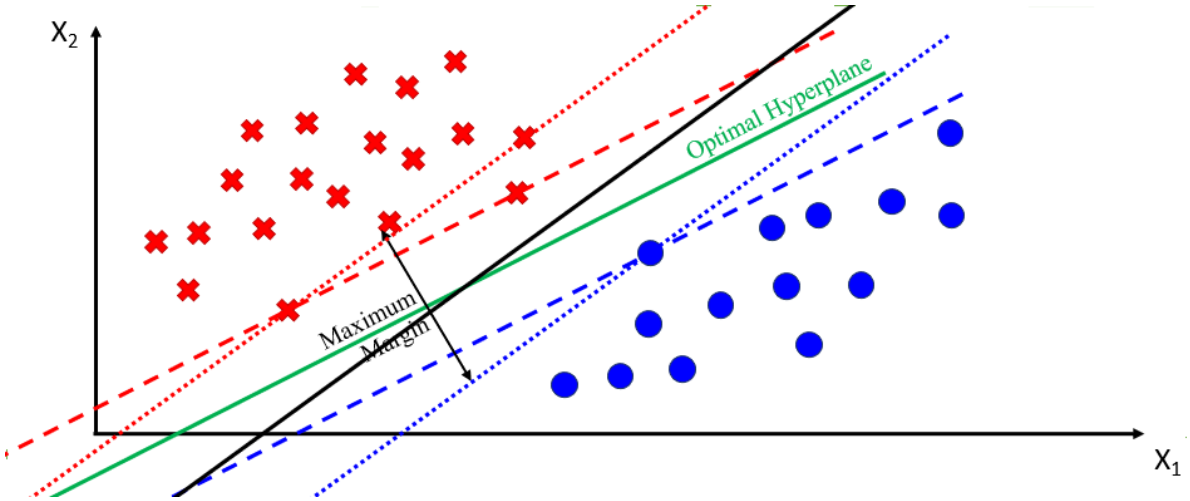

Figure 5.2 SVM hyperplane selection approach

In non-linear classifiers, the *SVM* approach makes use of a technique known as the kernel trick [111]. The *SVM* kernel is a function that converts an input space with a high dimension to a space with a lower dimension. Then, *SVM* consider this converted input as a linear problem, but on a wider scale, and solve it simply by re-optimizing *SVM* equations [27] [28] [111]. This approach and kernel function are illustrated in Figure 3.13.

SVM has numerous advantages and is not only a well-known solution for *DDoS attacks detection* problems; it also performs well with small datasets like our work. It has accuracy and a low false positive rate, which is critical *for remote health monitoring systems*. In addition, we can use it in both linear and non-linear situations. It is robust to noise and effective with high-dimensional data.

Therefore, it would be an excellent candidate for *DDoS* attacks detection functionality in our *DDoS* Module.

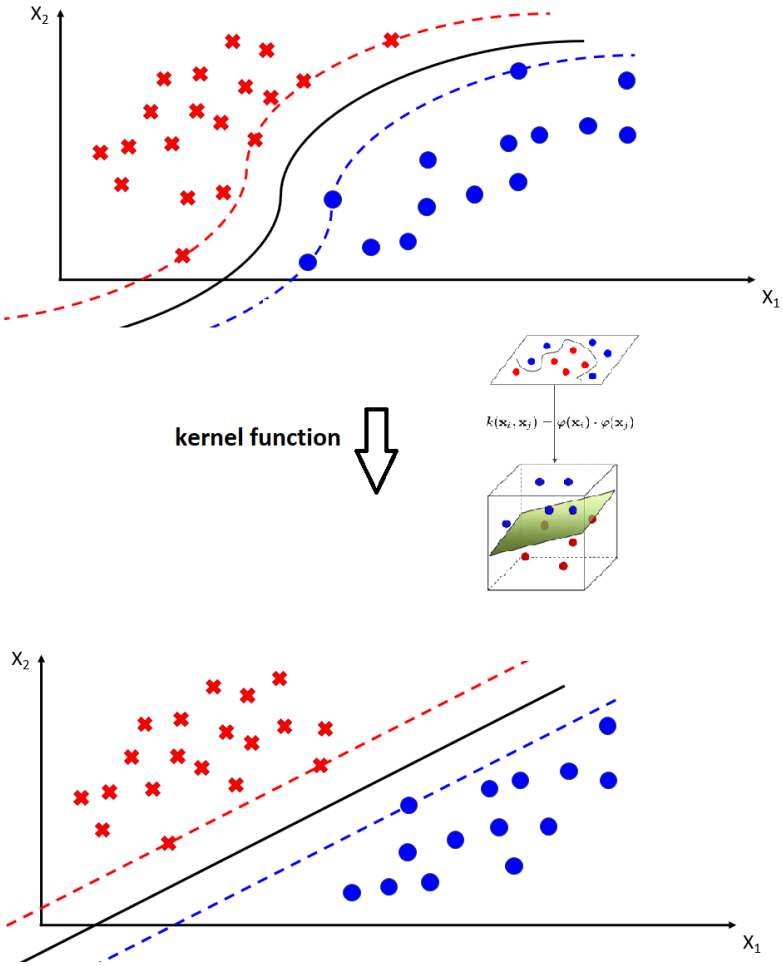

Figure 5.3 Non-linear classification with SVM

APPENDIX C PREPARING SDN NETWORK

[Class] createTopology.py:

```
from mininet.net import Mininet
from mininet.topolib import TreeTopo
from mininet.node import Controller, RemoteController,OVSSwitch
import random
import threading

# Create and start Mininet.
def createTopology(odlIP, treeDepth, treeFanout):
    odlControllerIP=odlIP
    # Using tree topology with selected depth and fanout.
    myTopo = TreeTopo(depth=treeDepth, fanout=treeFanout)
    net = Mininet(topo=myTopo, controller=None ,switch=OVSSwitch)
    odl = net.addController( 'c0', controller=RemoteController, ip=odlIP,
port=6633)
    net.start()
    return net

                ##################################################
```

[Class] generateIP.py:

```
from random import randrange
# This function randomly generates the IP addresses of the hosts based on the
entered start and end values. Host IPs start with 10.0.0., the last value
entered by the user.
def gendest(start, end):

 first = 10
 second = 0
 third = 0
 # It generates a number between start and end to provide a random IP.
 ip = ".".join([str(first),str(second),
 str(third),str(randrange(start,end))])
 return ip
```

```
##################################################
```

[Class] getRandNum.py:

```
import random

# This function creates random numbers in our specified range.
def getRandNum(start , end , exeptList):
     number = random.choice([i for i in range(end) if i not in exeptList])
     return number
```

```
##################################################
```

[Class] stopNet.py:

```
# Using this function to clear the network after finishing our simulation.
def stopNet():
    os.system('sudo mn -c')
```

APPENDIX D CONVERT ECG TO JASON/ APPLY JWT

[Class] ecgInfo.py:

```
# ECG Class for ECG Data with our required fields.
class ecgInfo:
    Signals=[]
    RecordName=''
    SignalsName=[]
    NumofSignals=0
    fs=0

    ######################################

    # Constructor of our class to provide our new customized ECG structure.
    def __init__(self ,Signals , RecordName, SignalsName , NumofSignals,  fs):

        self.Signals = Signals
        self.RecordName = RecordName
        self.SignalsName = SignalsName
        self.NumofSignals = NumofSignals
        self.fs=fs

              ##################################################
```

[Class] readEcgData.py:

```
from numpy.core.fromnumeric import shape
import wfdb as wfdb
import matplotlib.pyplot as plt
import numpy as np
import glob
import jwt
import ecgInfo
import json
import pandas as pd
import ecgDenoising as ed

# This function draws ECG signals.
def draw_ecg(x ,y ):
```

```
    # Read the ECG signal information.
    arr= np.array(x)
    signame = np.array(y)
    print("y" , y)
    print("signame " , signame )
    z=arr.shape[0]
    # Plot 12 leads of the signal.
    for i in range(arr.shape[1]):
        j=i*5
        plt.plot(arr[0:z,i]+j , label=signame[i])

    # Configure the chart style.
    ax = plt.gca()
    ax.spines['bottom'].set_color('red')
    ax.spines['top'].set_color('red')
    ax.spines['right'].set_color('red')
    ax.spines['left'].set_color('red')

    # Configure the axis label and style.
    ax.set_yticklabels([])
    ax.set_xticklabels([])
    xMajorTicks = np.arange(0, 1600, 400)
    xMinorTicks = np.arange(0, 1600, 80)
    yMajorTicks = np.arange(0, 60, 12)
    yMinorTicks = np.arange(0, 60,3 )
    ax.set_xticks(xMajorTicks)
    ax.set_xticks(xMinorTicks, minor=True)
    ax.set_yticks(yMajorTicks)
    ax.set_yticks(yMinorTicks, minor=True)
    ax.grid(which='both',color='red')
    handles, labels = ax.get_legend_handles_labels()

    # Configure the legend style.
    plt.legend(handles[::-1], labels[::-1], title='Signals', loc='center left',
bbox_to_anchor=(1, 0.5) )
    plt.show()

########################################

# This function uses JWT and our private key to encode and encrypt the ECG
signals that have already been converted to Jason format.
def jwtEcoder(item , privateKey):
    encoded = jwt.encode({'data': item}, privateKey, algorithm="HS256")
    return encoded
```

```
########################################

# This function uses JWT and our private key to decode and decrypt the ECG
signals.
def jwtDecoder(item , privateKey):
    encoded = jwt.decode(item, privateKey, algorithms="HS256")
    return encoded

########################################

# This function reads an ECG signal considering the provided name for the
signal, MIT format and provides Jason format.
def readEcgData(name):
    record = wfdb.rdrecord(name, sampfrom=0, sampto = 1500)
    info = ecgInfo.ecgInfo(record.p_signal.tolist() , record.record_name ,
record.sig_name , record.n_sig , record.fs)

    return  json.dumps(info.__dict__)

########################################

# This function saves a text file based on encoded and encrypted data for each
ECG signal.
def encodeECG(item , privateKey , fileName):
    encodeitem = jwtEcoder(item , privateKey )
    file = open(fileName +'.txt', "w")
    n = file.write(encodeitem)
    file.close()
    print("your data is encoded and successfully saved in file
",fileName+'.txt' )
    return encodeitem

########################################

# This function reads the encoded and encrypted files for each ECG and provides
Jason's format with decoded and encrypted information.
def decodeECG(  privateKey , fileName ):
    file = open(fileName +'.txt',"r+")
    encodeitem = file.read()
    file.close()
    decodeitem = jwtDecoder(encodeitem , privateKey )
    print("your data is decoded and the result is:", decodeitem)
    return decodeitem
```

```
########################################

# This function reads all the ECG signals in our dataset and provides encoded
files.
def readAllEcgData():

    encodeRecords = []
    # Defining our private key for encryption
    privateKey= "LARIM"
    i=1
    # Read all the files with the.hea extension in our dataset path.
    datas = glob.glob("ecgData/*.hea")
    for data in datas:
        size= len(data) - 4
        item= readEcgData(data[0:size] )
        # Encode and encrypt the data of each file.
        encodeitem = encodeECG(item , privateKey , 'ecgEncodeData/'+ str(i))

        encodeRecords.append(encodeitem)
        i=i+1
    # Save encoded and encrypted data to an excel file.
    df = pd.DataFrame(encodeRecords)
    df.to_csv('ecg2022.csv')

########################################

# This function reads an ECG signal and draws it.
def readOneEcgData(item):
    record = wfdb.rdrecord(item , sampfrom=0, sampto = 1500)
    print(record)
    draw_ecg(record.p_signal, record.sig_name)
    print("Your record is ploted!")
    return record

# The primary function of our ECG signal procedure.
if __name__ == "__main__":

    # Defining our private key for encryption.
    privateKey= "LARIM "
    # Set our dataset path.
    datas = glob.glob("ecgData/*.hea")
    # Take a random number from the user to select a random ECG file in our
dataset path.
```

```
    recordNo = input ("Please choose one of recordes, enter a number between 1
to 1500:\n")
    index = int(recordNo)
    data = datas[index]
    size = len(data) - 4
    mainRecord=""
    decodeitem=""
    # Take a requested operation from the user.
    while(True):
        userNumber = input ("Choose one of the following options:\n"
        + " 1 for read one data and plot it \n"
        + " 2 for encoding and encrypting the one data \n"
        + " 3 for decoding and decrypting the data \n"
        + " 4 for plot the decoded data \n"
        + " 5 for Denoising data \n"
        + " 6 for read all data, encode and save them in file \n"
        + " 9 for Exit \n"
        + " Please Enter your number: ")
        userChoice = int(userNumber)
        # When the user selects to read and plot an ECG signal.
        if(userChoice==1):
            mainRecord = readOneEcgData(data[0:size])
        # When the user selects encoding and encryption, an ECG signal.
        elif(userChoice==2):
            if(mainRecord is None):
                print("There is no data to Encode!")
            else:
                item= readEcgData(data[0:size] )
                encodeECG(item , privateKey , "myFileECG")

       # When the user selects decoding and decrypting an ECG signal.
        elif(userChoice==3):
            decodeitem= decodeECG( privateKey , "myFileECG")
       # When the user selects to plot the results of the previous step.
        elif(userChoice==4):
            if(len(decodeitem)>0):
                draw_ecg(eval(decodeitem["data"])["Signals"] ,
eval(decodeitem["data"])["SignalsName"])
            else:
                print("There is no decode data to plot!")
       # When the user selects denoising an ECG signal
        elif(userChoice==5):
            b = wfdb.rdsamp(data[0:size])
            input_sig = b[0][:,0]
```

```
            wavelet_denoised_ecg = ed.wavelet_denoising(input_sig,
list(range(650000)), True)

        # When the user selects encoding and encryption, all ECG signals are
saved in text files.
        elif(userChoice==6):
            readAllEcgData()
        elif(userChoice==9):
            break
```

APPENDIX E PLOT MOTHER WAVELETS FOR DWT

[Class] plotWDT.py:

```
import pywt
import matplotlib.pyplot as plt

# Print the name of each wavelet and its abbreviations.
print(pywt.families(short=False))
print(pywt.families(short=True))

# Plot 'Haar' mother wavelet
wavelet = pywt.Wavelet('haar')
family_name = wavelet.family_name
biorthogonal = wavelet.biorthogonal
orthogonal = wavelet.orthogonal
symmetry = wavelet.symmetry
a= wavelet.wavefun()
wavelet_function =a[0]
x_values = a[-1]
plt.plot(x_values, wavelet_function)
plt.ylabel("Discrete Wavelets", fontsize=16)
plt.title("(1) " + family_name, fontsize=16)
plt.show()

# Plot 'Daubechies' mother wavelet
wavelet = pywt.Wavelet('db4')
family_name = wavelet.family_name
biorthogonal = wavelet.biorthogonal
orthogonal = wavelet.orthogonal
symmetry = wavelet.symmetry
a= wavelet.wavefun()
wavelet_function =a[0]
x_values = a[-1]
plt.plot(x_values, wavelet_function)
plt.ylabel("Discrete Wavelets", fontsize=16)
plt.title("(2) " +family_name, fontsize=16)
plt.show()

# Plot 'Symlets' mother wavelet
wavelet = pywt.Wavelet('sym4')
family_name = wavelet.family_name
biorthogonal = wavelet.biorthogonal
orthogonal = wavelet.orthogonal
```

```
symmetry = wavelet.symmetry
a= wavelet.wavefun()
wavelet_function =a[0]
x_values = a[-1]
plt.plot(x_values, wavelet_function)
plt.ylabel("Discrete Wavelets", fontsize=16)
plt.title("(3) " +family_name, fontsize=16)
plt.show()

# Plot 'Coiflets' mother wavelet
wavelet = pywt.Wavelet('coif4')
family_name = wavelet.family_name
biorthogonal = wavelet.biorthogonal
orthogonal = wavelet.orthogonal
symmetry = wavelet.symmetry
a= wavelet.wavefun()
wavelet_function =a[0]
x_values = a[-1]
plt.plot(x_values, wavelet_function)
plt.ylabel("Discrete Wavelets", fontsize=16)
plt.title("(4) " +family_name, fontsize=16)
plt.show()

# Plot 'Biorthogonal' mother wavelet
wavelet = pywt.Wavelet('bior3.9')
family_name = wavelet.family_name
biorthogonal = wavelet.biorthogonal
orthogonal = wavelet.orthogonal
symmetry = wavelet.symmetry
a= wavelet.wavefun()
wavelet_function =a[0]
x_values = a[-1]
plt.plot(x_values, wavelet_function)
plt.ylabel("Discrete Wavelets", fontsize=16)
plt.title("(5) " +family_name, fontsize=16)
plt.show()

# Plot 'Reverse biorthogonal' mother wavelet
wavelet = pywt.Wavelet('rbio3.7')
family_name = wavelet.family_name
biorthogonal = wavelet.biorthogonal
orthogonal = wavelet.orthogonal
symmetry = wavelet.symmetry
a= wavelet.wavefun()
```

```
wavelet_function =a[0]
x_values = a[-1]
plt.plot(x_values, wavelet_function)
plt.ylabel("Discrete Wavelets", fontsize=16)
plt.title("(6) " +family_name, fontsize=16)
plt.show()

# Plot 'Discrete Meyer (FIR Approximation)' mother wavelet
wavelet = pywt.Wavelet('dmey')
family_name = wavelet.family_name
biorthogonal = wavelet.biorthogonal
orthogonal = wavelet.orthogonal
symmetry = wavelet.symmetry
a= wavelet.wavefun()
wavelet_function =a[0]
x_values = a[-1]
plt.plot(x_values, wavelet_function)
plt.ylabel("Discrete Wavelets", fontsize=16)
plt.title("(7) " +family_name, fontsize=16)
plt.show()
```

APPENDIX F DWT DENOISING ECG

[Class] ecgDenoisingEvaluation.py:

```
import wfdb
import matplotlib.pyplot as plt
import pywt
import pywt.data
import numpy as np
import math

# This function receives an ECG signal and denoises it.
def wavelet_denoising(ecg, index, doPlot):
   # Create a wavelet object and define the parameters.
   wavelete = pywt.Wavelet('sym4')
   maxlev =pywt.dwt_max_level(len(ecg), 8)

   # Decompose into wavelet components.
   coeffs = pywt.wavedec(ecg, wavelete, level=maxlev)
   for i in range(1, len(coeffs)):
       # Calculate our universal threshold.
       threshold= 0.97 * np.sqrt(2*np.log(len(ecg))/len(ecg))
       # Apply this threshold to the detailed coefficients.
       coeffs[i] = pywt.threshold(coeffs[i], threshold*max(coeffs[i]))

   # Multilevel reconstruction using waverec
   datarec = pywt.waverec(coeffs, wavelete)

   # If the user requests the plot, we plot the denoised result.
   if doPlot:
       mintime = 1000
       maxtime = mintime +2000
       plt.figure()
       plt.subplot(2, 1, 1)
       plt.plot(index[mintime:maxtime], ecg[mintime:maxtime])
       plt.plot(index[mintime:maxtime], ecg[mintime:maxtime] , "-b", label="Raw
signal" )
       plt.plot(index[mintime:maxtime], datarec[mintime:maxtime] , "-r",
label="Denoised signal")
       plt.xlabel('time (s)')
       plt.ylabel('amplitude (V)')
       plt.title("Denoised ECG signal1")
       plt.legend(loc="upper left")
```

```
        plt.tight_layout()
        plt.show()
    return datarec

# Create initial parameters.
time = list(range(650000))
data=[]
# Choose two ECG signals from each dataset to evaluate our denoising approach.
list_of_files =  ['ecgData/2981187' , 'ecgData/s0205_re' , 'ecgData/s0341lre' ,
'ecgData/1003574' , 'ecgData/2863747' , 'ecgData/s0224_re', 'ecgData/2584163' ,
'ecgData/1168042' ]
# For each selected ECG signal, calculate SNR and MSE.
for file_path in list_of_files:
    record_nm = file_path#[:-4]
    #path = 'ecgData/' + str(record_nm)
    try:
        record = wfdb.rdsamp(record_nm)
        print("Record file " + str(record_nm) + " exists" )

        # Select lead 1 as input signal.
        input_sig = record[0][:,0]

        # Add gaussian noise to the raw signal.
        cleanSignal = input_sig
        mu, sigma = 0, 0.14
        noise = np.random.normal(mu, sigma, input_sig.shape)
        noisySignal = input_sig + noise

        # ECG Denoising using Wavelet Transform.
        wavelet_denoised_ecg = wavelet_denoising(noisySignal, time, False)
        # Calculate SNR
        SNR = 10* math.log(np.sum((noisySignal-input_sig) **2) /
np.sum((wavelet_denoised_ecg-input_sig) **2))
        # Calculate MSE
        MSE = (1/len(input_sig))*np.sum((wavelet_denoised_ecg-input_sig) **2)
        data.append([record_nm , SNR ,MSE] )
        print("#####################################")
        print("filename: " , record_nm)
        print("SNR - wavelet_denoised_ecg" , SNR , MSE)
        print("MSE1: ", (1/5000)*np.sum((wavelet_denoised_ecg-input_sig) **2))
        print("#####################################")

    except IOError:
       print("Record file " + str(record_nm) + " does not exist!")
```

APPENDIX G SIMULATE NORMAL SITUATION

[Class] sendNormalData.py:

```
from mininet.net import Mininet
from datetime import datetime
import pandas as pd

# Define our custom structure for normal situations in the network.
class sendNormalData:

    def __init__(self ,net , src , dest,  period_time, size,filename):
        self.net = net
        self.src = src     # Source IP.
        self.dest = dest   # Destination IP.
        self.size=size
        self.filename=filename # Encoded and encrypted ECG file name.
        self.shell = None
        self.period_time = period_time


    ########################################

   # Send an encoded and encrypted ECG file from host to destination.
    def sendNormalData(self):

      host = self.net.hosts[self.src]
      host.cmd('timeout ' + str(self.period_time) + 's hping3 ' +  '10.0.0.' +
str(self.dest) + ' --icmp -d '+ str(self.size)+' -c 1 --file ' + self.filename)

      host.cmd('killall hping3')

                ##################################################
```

[Class] normalFlow.py:

```
from getRandNum import getRandNum
from sendNormalData import sendNormalData
from stopNet import stopNet
from createTopology import createTopology
import time
```

```
import threading
import time
import csv
from datetime import datetime
# Send an encoded and encrypted ECG file to provide a normal situation on the
network.
def normalflow( normalSeconds, nodes , len_size , net , period_time, filename):

    i=0
    j = filename
    # Set time
    normalStartTime = time.time()
    while True:
        current_time = time.time()
        elapsed_time = current_time - normalStartTime

        if elapsed_time > normalSeconds:
            print("Finished iterating in: " + str(int(elapsed_time))  + "
seconds")
            break
        # Select two random numbers for sourceIPs and destination IPs.
        srcId= getRandNum(0 , nodes ,[])
        desctId= getRandNum(0 , nodes ,[srcId])
        size = 2500
        srcId1= getRandNum(0 , nodes ,[])
        desctId1= getRandNum(0 , nodes ,[srcId1])
        size1 =2500
        # Select two encoded and encrypted ECG files.
        filename1 = 'ecgEncodeData/'+str(j) +'.txt'
        filename2 = 'ecgEncodeData/'+str(j+1) +'.txt'

       # Send the first file from the first sourceIP to the first DestinationIP
in the normal situation.
        x= sendNormalData(net,srcId, desctId , period_time,size, filename1)
        t1 = threading.Thread(target=x.sendNormalData, args=())
        print("First thread and round " + str(i) + " - source id is " +
str(srcId) + " destination id is "+ str(desctId)) + " File name is " +
filename1
        # Send the first file from the first sourceIP to the first
DestinationIP in the normal situation.
        y= sendNormalData(net,srcId1, desctId1 , period_time,size1, filename2)
        t2 = threading.Thread(target=y.sendNormalData, args=())
```

```
        print("Second thread and round " + str(i) + " - source id is " +
str(srcId1) + " destination id is "+ str(desctId1)) + " File name is " +
filename2
        t1.start()
        t2.start()
        t1.join()
        t2.join()
        j=j+2
        if(j>=1500):
            j=1
    i=i+1
    return j

                ##################################################
```

[Class] mainNormal.py:

```
from getRandNum import getRandNum
from sendNormalData import sendNormalData
from sendDDoSData import sendDDoSData
from stopNet import stopNet
from createTopology import createTopology
from normalFlow import normalflow
from ddosFlow import ddosFlow
import time
import threading
from scapy.all import *
import pandas as pd


# The main function for simulating normal situations in our network.

# Set the Controller IP.
odlIp='192.168.164.143'

# Create an SDN with the tree topology.
treeDepth=4
treeFanout=3
period_time = 5
len_size = 1600
nodes = treeFanout ** treeDepth
print("Number of Nodes" , nodes)
```

```
net = createTopology(odlIp,treeDepth,treeFanout)

# Set the initial data.
print("Start")
start_time = time.time()
seconds = 10800
innerSeconds = 40
i=1
while True:
    current_time = time.time()
    elapsed_time = current_time - start_time
    # Check the time of the simulation.
    if elapsed_time > seconds:
        print("Finished iterating in: " + str(int(elapsed_time))  + " seconds")
        break
    # Send normal flow in the network.
    i= normalflow(innerSeconds, nodes , len_size , net , period_time, i)


    if(i>1500):
        i=1

# Stop network
stopNet()
print("End")
```

APPENDIX H SIMULATE DDOS SITUATION

[Class] sendDDoSData.py:

```
from mininet.net import Mininet
from datetime import datetime
import pandas as pd

# Defining our custom structure for DDoS situations in our network.
class sendDDoSData:

    def __init__(self ,net , src , dest,  period_time, size):
        self.net = net
        self.src = src
        self.dest = dest
        self.size=size
        self.shell = None
        self.period_time = period_time

########################################

# Send an encoded and encrypted ECG file from host to destination.
    def sendDDoSData(self):

      host = self.net.hosts[self.src]
      host.cmd('timeout ' + str(self.period_time) + 's hping3 --flood '
+  '10.0.0.' + str(self.dest)) + ' -d ' + str(self.size)
      host.cmd('killall hping3')

                ##################################################
```

[Class] ddosFlow.py:

```
from getRandNum import getRandNum
from sendNormalData import sendNormalData
from sendDDoSData import sendDDoSData
from stopNet import stopNet
from createTopology import createTopology
```

```
import time
import threading
# This function sends DDoS attacks into the network.
def ddosFlow(seconds, nodes , len_size , net , period_time):


   # Set time
    ddosStartTime = time.time()
    i=0

    while True:
        current_time = time.time()
        elapsed_time = current_time - ddosStartTime
        # Select two random numbers for sourceIPs.
        AttackerId1= getRandNum(0 , nodes ,[])
        AttackerId2= getRandNum(0 , nodes ,[AttackerId1])


        if elapsed_time > seconds:
            print("Finished iterating in: " + str(int(elapsed_time))  + "
seconds")
            break

       # Select two random numbers for DestinationIPs.
        VictId1 = getRandNum(0 , nodes ,[AttackerId1,AttackerId2])
        size2 = getRandNum(0 , len_size ,[])
        VictId2 = getRandNum(0 , nodes ,[AttackerId1,AttackerId2])
        size3 = getRandNum(0 , len_size ,[])
        # Simulate the first attack.
        y= sendDDoSData(net,AttackerId1, VictId1 , period_time,size2)
        t1 = threading.Thread(target=y.sendDDoSData, args=())
        print("Second thread and round " + str(i) + " - Attacker Id1 is " +
str(AttackerId1) + " Vict Id1 is "+ str(VictId1))
        # Simulate the second attack.
        z= sendDDoSData(net,AttackerId2, VictId2 , period_time,size3)
        t2 = threading.Thread(target=z.sendDDoSData, args=())
        print("Third thread and round " + str(i) + " - Attacker Id2 is " +
str(AttackerId2) + " Vict Id2 is "+ str(VictId2))
        t1.start()
        t2.start()
        t1.join()
        t2.join()
        i=i+1
```

```
##################################################
```

[Class] mainDDoS.py:

```
from getRandNum import getRandNum
from sendNormalData import sendNormalData
from sendDDoSData import sendDDoSData
from stopNet import stopNet
from createTopology import createTopology
from normalFlow import normalflow
from ddosFlow import ddosFlow
import time
import threading
from scapy.all import *
import pandas as pd

# The primary function for simulating normal situations in our network
# Set the Controller IP
odlIp='192.168.164.143'

# Create SDN with the tree topology.
treeDepth=4
treeFanout=3
period_time = 5
len_size = 1600
nodes = treeFanout ** treeDepth
print("Number of Nodes" , nodes)
net = createTopology(odlIp,treeDepth,treeFanout)
print("Start")
# Set the initial data.
start_time = time.time()
seconds = 10800
innerSeconds = 40
i=1
while True:
    current_time = time.time()
    elapsed_time = current_time - start_time
    # Check the time of simulation
    if elapsed_time > seconds:
        print("Finished iterating in: " + str(int(elapsed_time))  + " seconds")
        break
    # Send DDoS flow in the network.
```

```
    ddosFlow(innerSeconds, nodes , len_size , net , period_time)

    if(i>1000):
        i=1

# Stop network
stopNet()
print("End")
```

APPENDIX I FEATURE SELECTION PHASE

[Class] dataCollection.py:

```
import httplib2
import json
import pandas as pd
import time
from datetime import datetime

# For reading the data from the controller when we have normal flow or DDoS
attacks in SDN.
try:
    h = httplib2.Http(".cache")
    h.add_credentials('admin', 'admin')
    start_time = time.time()
    print("start_time",start_time)
    featureData=[]
    while(True):
        # Call the ODL Rest API of our ODL controller.
        resp, content =
h.request('http://192.168.164.143:8080/controller/nb/v2/statistics/default/flow
', "GET")
        allFlowStats = json.loads(content)
        # Select statistical flow information.
        flowStats = allFlowStats['flowStatistics']
        finalData= []
        print("start")
        for fs in flowStats:
            print ("\nSwitch ID : " + fs['node']['id'])
            print ('{0:8} {1:8} {2:5} {3:15}'.format('packetCount', 'Action',
'Port', 'DestIP'))
            # Read our selected information and save it in variables.
            for aFlow in fs['flowStatistic']:

                packetCount = aFlow['packetCount']
                byteCount = aFlow['byteCount']
                durationSeconds = aFlow['durationSeconds']
                actions = aFlow['flow']['actions']
                actionType = ''
                actionPort = ''

                if(type(actions) == type(list())):
```

```
                    if(len(actions)>1):

                        actionType = actions[1]['type']
                        actionPort = actions[1]['port']['id']
                    else:
                        actionType = actions[0]['type']
                        actionPort = actions[0]['port']['id']

                dstination = aFlow['flow']['match']['matchField'][0]['value']
                now = datetime.now()
                timeVal = now.strftime("%H:%M:%S")
                # Provide a list of our selected data from the network.
                finalData.append([packetCount, byteCount, durationSeconds,
actionPort, dstination ,timeVal])


                print ('{0:8} {1:8} {2:5} {3:15} {4:25}'.format(packetCount,
actionType, actionPort, dstination , timeVal))
        # Save the results for DDoS or the normal situation.
        dfresult = pd.DataFrame(finalData ,columns=['packetCount', 'byteCount',
'durationSeconds', 'actionPort' , 'dstination' , 'time'])
        print("Start writing if files")
        dfresult.to_csv('./output-ddos-result2022.csv', mode='a', header=False)
        #dfresult.to_csv('./output-normal-result2022.csv', mode='a',
header=False)
        print("End writing if files")

        # Calculate the entropy of our features.
        totalPacketCount = dfresult['packetCount'].sum()
        totalByteCount = dfresult['byteCount'].sum()
        totalDurationSeconds = dfresult['durationSeconds'].sum()
        destinationLists = dfresult['dstination'].unique()
        destinationQty = len(destinationLists)
        featureData.append([totalPacketCount,totalByteCount,totalDurationSecond
s,destinationQty,timeVal])
        # Set the window size.
        time.sleep(10)
        current_time = time.time()
        elapsed_time = current_time - start_time

        if elapsed_time > 11000:
            print("Finished iterating in: " + str(int(elapsed_time))  + "
seconds")
            break
```

```
    print("End")
   # Save the entropy features for DDoS or normal situations.
    dfFeatureData = pd.DataFrame(featureData ,columns=['packetCount',
'byteCount', 'durationSeconds', 'dstination' , 'time'])
    dfresult.to_csv('./output-DDoSFeatureData2022.csv')
    #dfresult.to_csv('./output-NormalFeatureData2022.csv')
except Exception:
    pass
```

APPENDIX J TRAINING THE MODEL

[Class] MLLearning.py:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
from imblearn.over_sampling import SMOTE
from collections import Counter
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import f1_score
from sklearn.model_selection import PredefinedSplit
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
import operator
import pandas as pd
from sklearn.svm import LinearSVC
from sklearn import svm
import pickle
import joblib

# This function uses cross-validation to train our model and find the best
parameter.
def trainModel( Param , Method , xTrain  , yTrain , xValid  , yValid ):
    xComp = np.concatenate((xTrain, xValid), axis=0)
    yComp= np.concatenate((yTrain, yValid) , axis=0)
    fold = [ -1 for i in range(xTrain.shape[0])] + [0 for i in
range(xValid.shape[0])]
    P = PredefinedSplit( test_fold= fold)
    # Using Grid Search to provide Cross-validation.
    csvPipeline = Pipeline([("scaler", StandardScaler()), ("svm", Method)])
    tempClf= GridSearchCV (csvPipeline , Param  , cv=P , refit = True )
    tempClf.fit(xComp,yComp)
    print("tempClf",tempClf.best_params_)
    return tempClf

########################################

# This function uses the F1 score to calculate the performance of our results.
def calcPerformance ( clfTune , x ,y   ):
```

```
    trainYPred=clfTune.predict(x)
    trainF1 = f1_score(trainYPred, y, average='macro')
    print("f1ScorePerformance:",trainF1)

# Read the entropy features of the normal situation and provide our x and y
matrix.
DataNormal = pd.read_csv("ECGdataresults/normalFeaturesECG.csv" , header=None)
DataNormal[6]=round(DataNormal[1]/DataNormal[3],2)
DataNormal[7]=round(DataNormal[2]/DataNormal[3],2)
xDataNormal = DataNormal.to_numpy()
yDataNormal =  np.zeros(xDataNormal.shape[0])

# Read the entropy features of the DDoS situation and provide our x and y
matrix.
DataDDoS = pd.read_csv("ECGdataresults/ddosFeaturesECG.csv" , header=None)
DataDDoS[6]=round(DataDDoS[1]/DataDDoS[3],2)
DataDDoS[7]=round(DataDDoS[2]/DataDDoS[3],2)
xDataDDoS = DataDDoS.to_numpy()
yDataDDoS =  np.ones(xDataDDoS.shape[0])
xData = np.concatenate((xDataNormal[:,1:8], xDataDDoS[:,1:8]), axis=0)
yData = np.concatenate((yDataNormal, yDataDDoS), axis=0)
print("xData" , xData.shape)
print("yData" , yData.shape)

# Split our data and select 20% for the test.
x, xTest, y, yTest = train_test_split(xData, yData, test_size = 0.20 ,
random_state=42)
# Split our data and select 20% for validation and 60% for the train set.
xTrain, xValid, yTrain, yValid = train_test_split(x, y, test_size = 0.20 ,
random_state=42)

# Handel imbalanced data with upsampling approach and smote on our train data.
oversample = SMOTE()
xTrainNew, yTrainNew = oversample.fit_resample(xTrain, yTrain)
counternew = Counter(yTrainNew)
print(counternew)

# Select SVM parameters and the initial range for them.
params = [{'svm__kernel':['linear', 'poly', 'rbf', 'sigmoid'],
'svm__gamma':['scale', 'auto']
 , 'svm__C' : np.arange(1.0, 5, 100) ,'svm__coef0': np.arange(0.0, 10, 100),
'svm__degree': [1,2,3,4,5,6]
  }]
```

```
# Train our model using train set, validation set, and initial parameters by
cross-validation approach.
Method = svm.SVC()
clfLinearSVC = trainModel( params , Method , xTrainNew , yTrainNew ,  xValid ,
yValid )
print("xTrainNew" , xTrainNew.shape)
print("yTrainNew" , yTrainNew.shape)

# Calculate the performance of our model for test, training, and validation
data by the F1 score.
print("Performance for Train Data")
calcPerformance(clfLinearSVC , xTrainNew , yTrainNew   )
print("Performance for Valid Data")
calcPerformance(clfLinearSVC , xValid , yValid   )
print("Performance for test Data")
calcPerformance(clfLinearSVC , xTest , yTest   )

# Save our model for future use in real-time monitoring.
svclassifier = clfLinearSVC.fit(xData, yData)
filename = 'newModel.sav'
joblib.dump(svclassifier, filename)
```

APPENDIX K REAL-TIME DDOS ATTACKS DETECTION

[Class] detecteDDos.py:

```
import httplib2
import json
import pandas as pd
import time
from datetime import datetime
import sys
from mlResults import mlResults

# Real-time monitoring and DDoS detection.
try:
    h = httplib2.Http(".cache")
    h.add_credentials('admin', 'admin')
    start_time = time.time()
    featureData=[]
    timeCounter=0
    while(True):
        # Call the ODL Rest API of our ODL controller.
        resp, content =
h.request('http://192.168.164.143:8080/controller/nb/v2/statistics/default/flow
', "GET")
        allFlowStats = json.loads(content)
        # Select statistical flow information.
        flowStats = allFlowStats['flowStatistics']
        finalData= []
        now = datetime.now()
        timeVal = now.strftime("%H:%M:%S")
        for fs in flowStats:
            # Read our selected information and save it in variables.
            for aFlow in fs['flowStatistic']:
                packetCount = aFlow['packetCount']
                byteCount = aFlow['byteCount']
                durationSeconds = aFlow['durationSeconds']
                actions = aFlow['flow']['actions']
                actionType = ''
                actionPort = ''
                if(type(actions) == type(list())):
                    if(len(actions)>1):
                        actionType = actions[1]['type']
                        actionPort = actions[1]['port']['id']
```

```
                    else:
                        actionType = actions[0]['type']
                        actionPort = actions[0]['port']['id']
                dstination = aFlow['flow']['match']['matchField'][0]['value']
                dstination_0 = aFlow['flow']['match']['matchField'][1]['value']
                # Provide a list of our selected data from the network.
                finalData.append([packetCount, byteCount, durationSeconds,
actionPort, dstination ,timeVal , timeCounter,dstination_0])

       # Save the results for DDoS attacks or the normal situation.
        dfResult = pd.DataFrame(finalData ,columns=['packetCount', 'byteCount',
'durationSeconds', 'actionPort' , 'dstination'  , 'timeCounter',
'time','dstination_0'])
       # Calculate the entropy of our features.
        dfResultTemp = dfResult[dfResult.packetCount != 0]
        totalPacketCount = dfResultTemp['packetCount'].sum()
        totalByteCount = dfResultTemp['byteCount'].sum()
        totalDurationSeconds = dfResultTemp['durationSeconds'].sum()
        destinationLists = dfResultTemp['dstination'].unique()
        destinationQty = len(destinationLists)
        if(totalPacketCount>0):
            featureData.append([totalPacketCount,totalByteCount,totalDurationSe
conds,destinationQty , timeCounter])
            dfFeatureData = pd.DataFrame(featureData )
            dfFeatureData.drop_duplicates(inplace=True)
            # Call our real-time DDoS attacks detection function with the
calculated entropy feature.
            mlResults(dfFeatureData)
        # Set the window size.
        time.sleep(10)
        current_time = time.time()
        elapsed_time = current_time - start_time
        timeCounter += 10

        if (elapsed_time >15500):
            print("Finished iterating in: " + str(int(elapsed_time))  + "
seconds")
            break
        print("End")

except Exception:
    print("Unexpected error:", sys.exc_info()[0])
    pass
```

```
##################################################
```

[Class] mlResults.py:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm
from collections import Counter
from numpy import where
import pickle
import joblib


# This function detects DDoS attacks using our trained model in real-time mode.
def mlResults(data ):

    # Set our trained model.
    filename = 'newModel.sav'
    # Provide the data as array
    data[6]=round(data[1]/data[3],2)
    data[7]=round(data[2]/data[3],2)

    xData = data.to_numpy()

    xData=xData[:,:]

   # Plot information based on entropy featured and save them in png format for
use by ODL Extension.
    plt.figure(figsize=(6,4))
    plt.ion()
    plt.scatter(xData[:,4],xData[:,6],label="Number of packet",
linewidth=3,linestyle="--" , marker="o" , s= 15 )
    plt.legend()
    plt.title("Number of packet per Time")
    plt.xlabel("Time")
    plt.ylabel("Number of Packet")
    plt.grid()

    plt.ioff()
    plt.savefig("c:/odl-packet.png")
```

```
    plt.figure(figsize=(6,4))
    plt.ion()
    plt.scatter(xData[:,4],xData[:,5],label="Number of byte",
linewidth=3,linestyle="--" , marker="o" , s= 15 )
    plt.legend()
    plt.title("Number of byte per Time")
    plt.xlabel("Time")
    plt.ylabel("Number of byte")
    plt.grid()

    plt.ioff()
    plt.savefig("c:/odl-byte.png")

    # Load our model.
    svclassifier =loaded_model = joblib.load(filename)

   # Predict the network situation considering the trained model and entropy
features.
    yPred = svclassifier.predict(xData)
    print(yPred)
    result = np.where(yPred > 0)
    # Print the results if we detect the DDoS attacks in our network.
    if(len(result)==0):
        print("DDoS attack is detected.")
    print("#####################################################")
    plt.figure(figsize=(6,4))
    plt.ion()
    counter = Counter(yPred)
   # Print out the results of our prediction.
    for label, _ in counter.items():
        fllowLabel="Normal"
        if(label==0):
            fllowLabel="Normal"
        else :
            fllowLabel="DDoS"
        row_ix = where(yPred == label)[0]

        # Plot the results of our prediction for use by ODL Extension.
        plt.scatter(xData[row_ix,4],yPred[row_ix],label=fllowLabel,
linewidth=3,linestyle="--" , marker="o" , s= 15 )
        plt.legend()
        plt.title("Detect DDoS Attack")
        plt.xlabel("Time")
```

```
        plt.ylabel("Normal=0 / DDos = 1")
        plt.grid()
        plt.ioff()
        plt.savefig("c:/odl-fllow.png")
    plt.close('all')
```

APPENDIX L ODL CHROME EXTENTION

[File] popup.html:

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta http-equiv="refresh" content="3">
    <title>OpenDayLight | LARIM/FLEX</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet"
        integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
crossorigin="anonymous">

    <style>
        body {
            width: 720px;
            margin: 10px;
            padding: 10px;
        }
    </style>
</head>

<body class="border border-1">
    <header>
        <h2 class="fw-bolder"></h2>
    </header>
    # Show the png files provided by mlResults.py for providing a real-time
monitoring tool and provid the real-time results of our prediction.
    <main class="container-fluid mb-4">
        <div class="row"></div>
        <div class="row">
            <div class="col-8">
                <img src="file:/C:/odl-fllow.png" alt="Detect DDoS Attack"
width="100%" height="auto">
                <div class="mt-3">
                    <a class="btn btn-sm btn-secondary mb-2" target="_blank"
                    href="http://192.168.164.131:8080/controller/nb/v2/statisti
cs/default/flow">Show RAW Data</a>
                </div>
```

```
            </div>
            <div class="col-4">
                <img src="file://C:/odl-packet.png" alt="Number of packet per
Time" width="100%" height="auto">
                <img src="file://C:/odl-byte.png" alt="Number of byte per Time"
width="100%" height="auto">
            </div>
        </div>
        </div>
    </main>
    <footer class="fixed-bottom m-3 ps-1">
        <small>
            <small><a href="http://www.larim.polymtl.ca/">LARIM/FLEX</a> - <a
                    href="https://www.polymtl.ca/">Polytechnique Montréal</a>
2021</small>
        </small>
    </footer>
    # Call popup.js to set the needed information.
    <script src="./src/popup.js"></script>
</body>

</html>

                ##################################################
```

[File] odl.js:

```
const isHelum = () =>
    document.title === 'OpenDaylight'

const fayeButtonOnClick = () => {

    var URL =
"http://192.168.164.136:8080/controller/nb/v2/statistics/default/flow"
    window.open(URL, '_blank')
    console.log("ODL Extention: Successfully extract!")
}

const fayeButtonAdd = () => {
    // Check/Find toolbar
    const elmToolbar = document.getElementById("toolbar")
```

```
    if (!elmToolbar) {
        console.log("ODL Extention: Can't find the toolbar")
        return;
    }

    // Create Button
    var elmButton = document.createElement("a")
    elmButton.appendChild(document.createTextNode(" Faye Extract "))
    elmButton.setAttribute('class', 'btn btn-faye');
    elmButton.setAttribute('href',
'http://192.168.164.136:8080/controller/nb/v2/statistics/default/flow');
    elmButton.setAttribute('download', 'file.xml');
    elmButton.onclick = () => fayeButtonOnClick()
    elmToolbar.prepend(elmButton)
    console.log("ODL Extention: The extention is enable")
}

(function () {
    if (isHelum()) {
        console.log("ODL Extention: Start working")
        fayeButtonAdd()
    }
})();

                ##################################################
```

[File] odl.css:

```
.btn-faye{
    margin-right: 0.5rem;
}
```