Secure Bluetooth Low Energy Communication in Healthcare Monitoring

Osman Salem, Quentin Guardia, and Ahmed Mehaoua
Borelli Research Center, CNRS UMR 9010 - University of Paris, France
{firstname.lastname}@u-paris.fr

***Abstract*—Healthcare services are increasingly using the Internet of Things (IoT) to acquire physiological data in real time. Medical data has stringent security requirements for the exchanged data between connected objects. With the absence of keyboard/display interfaces in several IoT, the use of default pin code in pairing phase using Bluetooth Low Energy (BLE) is highly insecure, and exposes the data exchange to several security attacks (eavesdropping, spoofing, etc.). The goal of this paper is to propose a new approach to prevent eavesdropping and spoofing using the distance between devices, which is derived from the Received Signal Strength Indicator (RSSI). Thus, it can be used to detect an abnormal distance resulting from a hacker connection using statistical method. Our experimental results show that our approach achieves a good accuracy with a low false alarm rate.**

***Index Terms*—BLE; MITM; RSSI; IoT; Healthcare; Wireless Security; Outliers; Anomaly detection**

I. Introduction

BLE is a Bluetooth-based technology used to connect devices to each other with low energy consumption. It exchanges less data than normal Bluetooth to reduce energy, and devices can stay in "sleep mode" until the next interaction. These advantages make this wireless technology widely deployed in Internet of Medical Things (IoMT) for remote monitoring of patients during long periods of time (months and even years) without charging or changing the battery. Bluetooth standards are managed by the Bluetooth Special Interest Group (Bluetooth SIG).

Devices in BLE are classified into two types: central and peripheral. The central device (e.g., smartphone) has higher computational power and storage than peripherals. The central device uses specific software (GATT protocol) to send commands and to collect data from peripherals. Conversely, the peripheral or the salve cannot initiate a connection and can only connect to a single master. It only executes received orders and gets noticed thanks to the advertising packages it sends. The peripherals stop sending advertising packets when they receive a specific packet, indicating that they are connected to a central device. Peripherals are sensors that collect and send data to central for processing. In Figure 1, the central device is represented by smartphone, tablet or computer, and peripherals are sensors used to capture blood pressure, SpO2 and body temperature, and other physiological parameters from the monitored patient.

BLE operates using ultra high frequency radio waves on a 2.4 GHz to 2.8 GHz band within a distance of 10 meters. This with 40 physical channels, against 80 for legacy Bluetooth, for

Fig. 1. Communication between central and peripherals

frequency and time multiplexing thanks to the L2CAP layer. By simple calculation, the difference between two channels is found to be 2 MHz. The devices in advertising mode send packets of 31 bytes at regular intervals. This task is conducted only on 3 of the channels: 37, 38 and 39. The other channels are reserved for data exchange between devices [1].

To establish a connection, the central alternates between scanning for pairing requests and sending advertising packets. It scans to check if it can find a peripheral to begin the exchange with it. The scanning process is expensive, so the scan usually does not run indefinitely. The BLE devices exchange their services, their capabilities, their input (such as the presence of keyboard or not) and output resources, their names and their manufacturers' information, authentication method, etc. during the first phase of pairing, which is not encrypted. However, the second phase is for key exchange and need to be secured.

In the second phase of pairing, one of the devices generates a Temporary Key (TK) which will be known from both devices. Confirmation of the key is made through the exchange of random values, encrypted and then decrypted. With the TK and random values, a Short Term Key (STK) is derived by devices without traveling in the network. The connection will be encrypted with this key at the link layer level. Eventually, a Long Term Key (LTK) can be exchanged for bonding.

Four pairing models are supported by BLE: "Just Works", Out of Band, Numerical Comparison and Passkey. When the device does not have I/O capabilities, the "Just Works" default pairing is used to derive the STK on both sides, after exchanging the random numbers and setting the TK to 0. This pairing method does not provide any protection against MitM and must not be used in healthcare monitoring services. In the

real world, sensors does not have I/O interfaces and this mode is currently deployed in healthcare products available in the market.

Also, it's important to note that BLE devices can play on MAC addresses to improve security. First, a whitelist can be set up to allow connection only with trusted addresses. Then, MAC addresses can be randomized to avoid Bluetracking.

BLE standard permits data ciphering with AES-128 encryption algorithm, MAC address randomization and MAC whitelist. However, when the device does not have input/output interfaces, the "Just Works" pairing does not prevent MiTM from decrypting, eavesdropping or conducting other attacks. Furthermore, an attack with specific tools (adapters, amplifiers, etc.) is able to intercept BLE signal up to 1000 *meters* [2] where the theoretical max distance is 100 *meters*. In this paper, we aim to demonstrate the feasibility of MitM and to propose a solution in order to protect medical data transferred by BLE from MitM attacks.

The rest of this paper is organized as follows. In Section II, we review the recent related work. In Section III, we present our proposed approach to secure the exchanged data. In section IV, we conduct experiments by presenting the hardware and software used to get access to the medical data, and the implementation of our approach to prevent such attacks. In Section V, we conclude the paper and present our future work.

II. Related Work

Despite the security measures adopted in BLE, some attacks are still feasible up to day. They range from simple passive data interception to identity theft and Denial of Service (DoS). Pallavi *et al.* in [3] review feasible security attacks on IoT devices with BLE transmission technology. Sevier *et al.* in [4] highlighted BLE vulnerabilities, prove that TK is vulnerable and showed how to sniff and decrypt acquired BLE data. They used Ubertooth dongle to capture BLE packets and to get the signal strength of the different channel frequencies. As this dongle is able to capture exchanged packets in the handshake, the TK could be cracked using the Crackle software on the Ubertooth data capture. Therefore, the LTK can be derived from the TK [5], and Wireshark can be used to decrypt the BLE packets when the LTK is provided. As Ubertooth output file is a PCAP file, Wireshark can read it and decrypt the packets in automatic manner.

Lounis *et al.* in [6] corroborated the results in [4]. As the most used pairing method to generate the TK is the "Just Works", they demonstrated its weakness by showing how to generate keys. Moreover, simple technologies have been used for conducting the sniffing attack. Data from smart deadbolt, bike lock and a lightbulb have been captured and decrypted in their experiments. However, the "Just Works" pairing method is not secure enough to generate a TK.

Cominelli *et al.* in [7] presented an open-source sniffer based on a Software-Defined Radio framework to capture BLE data packets in a very simple manner. They used the Graphic Processor Unit (GPU) to process the traffic. Even thought sniffing can be dangerous for sensitive medical data, the attacker can induce a Denial of Service (DoS) or even spoof a device.

As stated in [4], BLE devices are likely to have less battery, and sending many spoofed pairing requests can drain the energy of the device and induce DoS. The packet manipulation can be achieved using Scapy and as the replay attack is feasible, the injection with spoofed MAC address of the slave using the program bdaddr was demonstrated in [4]. In remote healthcare monitoring, such security holes may threat the life of monitored patient by preventing the system from raising alarms or by injecting false measurements to raise false alarms and make the system unreliable.

The work of Lahmadi *et al.* in [2] demonstrated a MitM attack against BLE and showed the low security features and inherent vulnerabilities. Afterward, they compared two unsupervised learning techniques to detect suspicious data, followed by classification method to tag packets as normal or attack from suspicious measurements. Their work is very near in his spirit to our work, where they combined supervised and unsupervised techniques to detect anomaly. However, the supervised classification requires labelled training data, which is not easy to find or to build. It is interesting to propose a lightweight and reliable sequential and non-parametric approach to prevent passive and active attacks conducted by MitM.

III. Proposed approach

In this section, we propose a novel approach to prevent the MitM from accessing or injecting data in the IoMT. The deployment scenario is presented in Figure 2, where MitM is able to intercept data transmitted by the BLE device to the central device. These data are sent using BLE from the monitored patient to his smartphone, which processes the received data to detect anomaly and raises an alarm to the healthcare professionals.

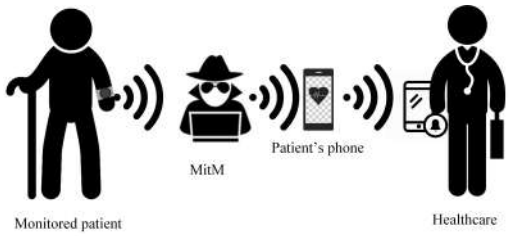

Fig. 2. MitM attack

To prevent the sniffing and injection attacks, a key exchange is initially conducted using the Elliptic Curve Diffie-Hellman (ECDH) to derive AES-CCM encryption key. It provides confidentiality and integrity [8]. In our proposed approach, only public keys from devices within a distance of 1.5 meter (distance derived from RSSI [9]) are accepted in the exchange of the shared key shown in Figure 3. The public key in ECC is derived in Equation 1:

$$P_i(x, y) = [d_i * G(x, y)]modp \quad (1)$$

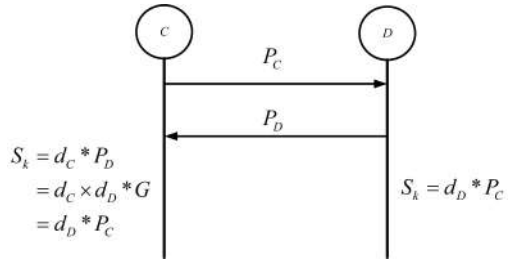

Fig. 3. Shared key derivation using ECDH

Where the generator $G(x, y)$ is a base point in the elliptic curve, $P_i(x, y)$ is the public key of the device i, which is obtained by adding the point $G(x, y)$ to itself d_i times. The central transmits its public key P_C to the device and the device transmits its key P_D to the central, where public keys are the only information to exchange. We used P-256 (also known as prime256v1) elliptic curve over prime fields.

The derived Shared Key (S_K) by ECDH is given in Equation 2:

$$S_K = d_C * P_D = d_C * d_D * G = d_D * P_C \tag{2}$$

After the ECDH key derivation, both devices use the S_k as seed for Pseudo Random Function (PRF) to derive a matrix of size 8×8 for key rotation, where the length of each key is 128 bit. The resulted matrix is rotated by 90 degrees to provide more randomness in the key rotation:

$$\nearrow \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix} \hookleftarrow \Rightarrow \begin{pmatrix} 13 & 9 & 5 & 1 \\ 14 & 10 & 6 & 2 \\ 15 & 11 & 7 & 3 \\ 16 & 12 & 8 & 4 \end{pmatrix}$$

After a period of T second, the next value in the matrix is used as a new key for AES-128 until the last value. To prevent data suppression by MitM, the transmission of each measurement is acknowledged by the central device. During the usage of last key, we implement the bloom file to hash the measurements in the last period and to derive a new seed based on the hash of measurements as shown in Figure 4. The new seed for PRF is used to generate the key matrix for securing the data exchange in the next epoch.

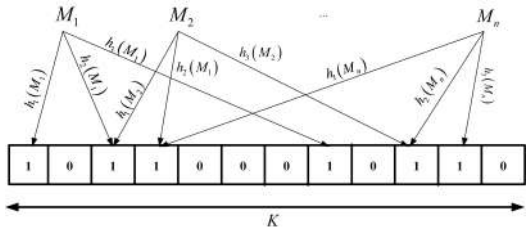

Fig. 4. Derivation of new seed using Bloom filter

The Bloom Filter (BF) is an array A of k elements initialized to 0 ($A_i = 0$ for $i = 0, \ldots, k-1$). Each measurement is hashed using m hash functions. We used the universal hash functions given in Equation 3 to uniformly distribute measurements M_i over hash table, where a_m and b_m are chosen randomly:

$$h_m(M_i) = ((a_m M_i + b_m) mod\ P) mod\ k \tag{3}$$

The result of $H_m(M_i)$ is the index of element in the hash table that will be updated if its value is zero according to Equation 4:

$$A_i(h_m(M_i)) = 1 \qquad \textit{iff} \qquad A_i(H_m(M_i)) == 0 \tag{4}$$

Despite the key rotation mechanism, we use an additional security layer by checking the Received Signal Strength Indication (RSSI) to detect sudden abnormal change. The RSSI has been widely used in BLE for indoor geolocalization and its value depends on the distance between communicating devices. The MitM will not be located at the same distance as sensors, hence the corresponding RSSI value will significantly differ from the RSSI value of sensor when injecting a new measurement.

The basic idea is to accept measurements from sensors within nearest locations using the RSSI, where sensors are near as they are deployed on the body of the monitored patient, and to refuse data from other far away sensors as their RSSI deviates from the majority. The RSSI is continuously decreasing with the distance from the other device, and the intensity attenuation model is described as:

$$RSSI(d) = RSSI(d_0) - 10\gamma \log_{10}\left(\frac{d}{d_0}\right) + x_\sigma \tag{5}$$

Where $RSSI(d)$ is the strength of signal measured at distance d, $RSSI(d_0)$ is the signal strength at reference distance d_0 (usually $d_0 = 1$ meter) and x_σ is the path loss which is a Gaussian random variable with zero mean induced by shadowing. At distance 1 m, the measured $RSSI_{d_0}$ is -66.50 dBm, and for simplification x_σ is set to 0 in our experiments, where the distance is derived using Equation 6. The path loss exponent γ is 1.25 and determined by fitting the acquired data.

$$d = 10^{(RSSI_{d_0} - RSSI_d)/10\gamma} \tag{6}$$

The central device receives health measurements with the associated RSSI and we used the underlying time series values to detect outliers. To do so, we use Isolation Forest (IF [10]), which is a light and powerful algorithm to detect anomalies. In such case, the system raises a network error exception for the user and ignores the abnormal associated values.

The IF algorithm consists in randomly splitting the data set into subsets to isolate the instance. The faster an instance is isolated, the higher probability to be an outlier. Concretely, the algorithm randomly selects an attribute and then randomly selects a split value within an interval. The IF is made of trees called the Isolation Trees (iTrees) as shown in Figure 5. It results from the partitioning where a node is a child or have two children. External nodes are points from the data set. Internal nodes are new partition, containing a "test", which is composed of the attribute q and the split value p, given

$q < p$. The Path Length to reach an external node x from the root represents the number of required partitions to isolate the point x. The anomaly score for each node in each iTrees is

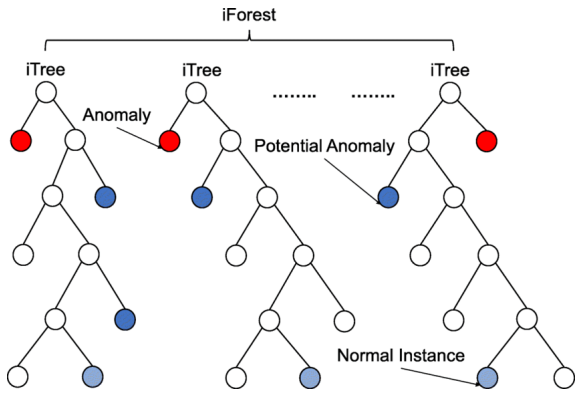

Fig. 5. iTrees making an iForest

derived using the Binary Search Trees (BST) as the average of path length $PL(x)$. The score is a value between 0 and 1. An outlier measurement has a score close to 1, and a normal one has a score lower than 0.5. We used the sklearn library in Python for the implementation of IF function, which uses the contamination parameter to set the threshold for outlier, making the detection less or more sensitive. The function returns -1 for outlier RSSI and 1 for normal RSSI. The pseudo code of IF to detect abnormal RSSI is presented in Algorithm 1, where the central device starts by scanning the surrounding BLE sensors to check if there is a missing or duplicate MAC address before applying the IF algorithm. If it is the case, an alert is raised for user, indicating there is a problem with spoofed or duplicated sensor.

Algorithm 1: Prevention from data injection

```
services[]=RSSI[]=NULL; idx=0;
while Received measurements do
    counter=0;
    services[idx]=measurements;
    RSSI[idx]=rssi;
    if Scan has duplicate or missing MAC address then
        Raise alarm: abnormal MAC list;
    end
    for Each record do
        if iForest(record)[idx]=-1 then
            counter++;
        end
        if counter > 0 AND iForest(RSSI)[idx]=-1 then
            Raise alarm: abnormal data received;
            services[idx]=RSSI[idx]=NULL;
        end
    end
    idx++;
end
```

IV. Experimental results

In this section, we present our experimental results conducted to prove the feasibility of MitM attack, followed by the implementation of our proposed approach to secure the communication. To sniff health data, we used a general scenario shown in Figure 2, where the smartphone is collecting and transmitting physiological data to central device (Raspberry Pi 4) using BLE. To conduct MitM, we used a PC with an Adafruit Bluefruit nrf51822 V2 presented in Figure 6. In the first set of experiments, the smartphone sends only the Heart Rate (HR) and the Wireshark sniffer running on the PC of MitM displays the value of HR as shown in Figure 7. The capture and replay of medical data is simple in "Just works" mode.

Fig. 6. Sniffer FlueFruit

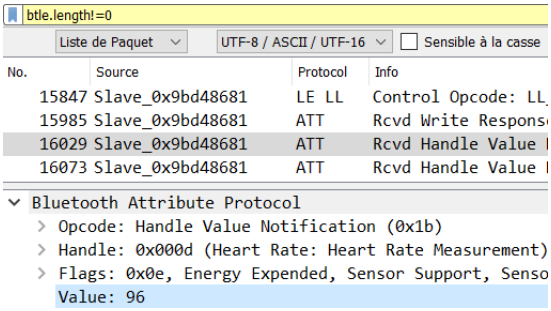

Fig. 7. MitM: Wireshark with the value of HR

To prevent access, replay and injection attacks, we implement our proposed approach for key rotation and anomaly detection in the RSSI of transmitted measurements. In this set of experiments, we used real physiological data from the MIMIC (Multiple Intelligent Monitoring in Intensive Care) database from the Physionet [11] web site. The data are saved in a file on Raspberry pi, which reads and transmits records to the central on the capture timestamp of each record using BLE.

The used data set in the Raspberry pi contains 6 attributes: Blood Pressure (BP), Heart Rate (HR), Pulse, Respiration Rate (RR), Oxygenation Ratio (SpO2) and body temperature. The variations of the BP are shown in Figure 8. The HR and PULSE exhibit similar variability and they superpose when drawing in the same figure. Figures 9 and 10 present the variations of the HR and Pulse respectively. The unit of measurement is Beat Per Minute (BPM). The variations of the Respiration and the SpO2 are shown in Figure 11 and 12. The respiration is measured in term of Respiration Per Minute and

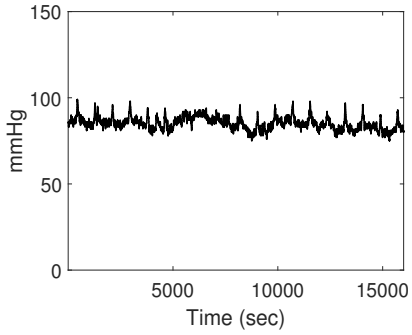

Fig. 8. Blood Pressure

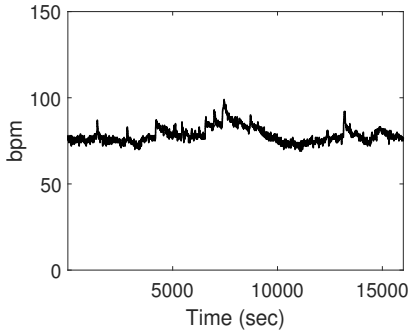

Fig. 9. Heart Rate

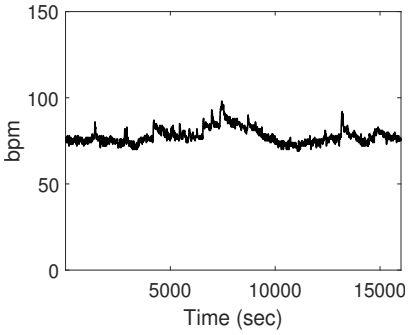

Fig. 10. PULSE

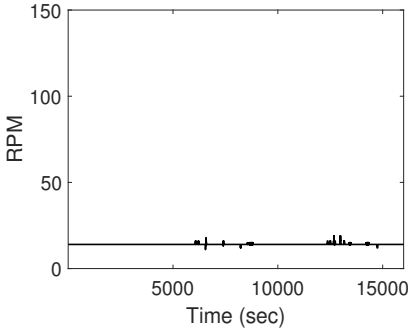

Fig. 11. Respiration rate

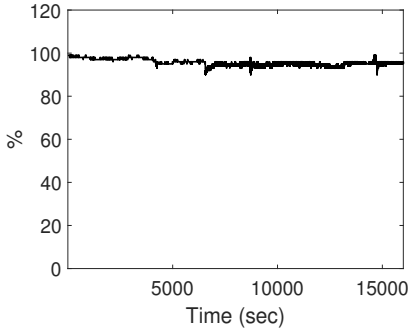

Fig. 12. SpO2

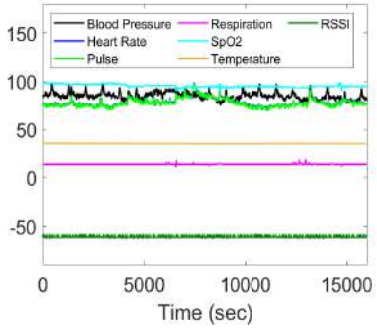

Fig. 13. All parameters

the SpO2 is a percentage that must be greater than 95%. A value of SpO2 lower than 95% is synonym to asphexia and requires ventilator and medical assistant.

The variations of whole parameters (BP, HR, Pulse, Respiration, SpO2, Temperature) and the associated RSSI are presented in Figure 13. The RSSI is the first curve with an average of -65 dBm. The depicted physiological parameters in Figure 13 have some normal variations without anomaly, as we extract normal records from the labelled data set. The objective of using normal data is to ensure the functioning of the system without false alarm.

We start by conducting a MitM attack to verify the encryption of data. A screenshot of the captured data with wireshark is presented in Figure 14, where we can notice that encrypted data can not be decoded by the sniffer. Furthermore, we assumed the extreme case, where the MitM succeeds to get the key and to break in the system to inject abnormal measurements to trigger false alarms and make the monitoring system unreliable.

The MitM modifies and injects records with modified HR as shown in Figure 15, where the value of injected HR increases and can be distinguished from the Pulse. We can also notice that the attacker is not located in the same position as the patient, and there is a change point in the underlying timeseries of RSSI around time instant 6000. The IF algorithm detects such change and raises a network alarm to preinitialise the system by putting the sensor and the smartphone near to each other within a distance of one meter. In the other hand,

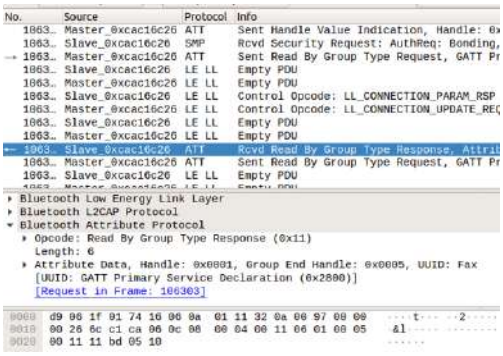

Fig. 14. MitM: encrypted data

abnormal measurements of SpO2 (inside the second ellipse) reflect health emergency. These values raise a medical alarm as shown in Figure 16, while the injected data is removed by the central device and does not trigger an alarm .

In the third set of experiments, we conduct performance analysis to analyse the accuracy of our proposed system using the labelled dataset. The Receiver Operating Characteristic (ROC) is used to study the impact of the score on the accuracy of the system in terms of True Positive Rate (TPR) and False Alarm Rate (FAR).

The ROC represents the variation of TPR with respect to FAR when changing the value of the score. A value of TPR closer to 100% indicates a high detection accuracy, while a lower value of FAR is desirable to achieve to enhance the

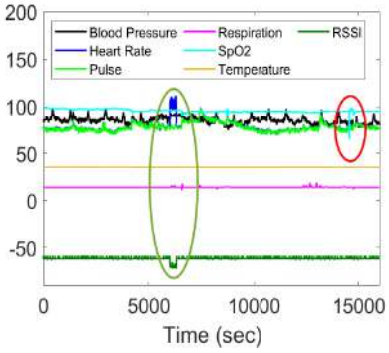

Fig. 15. Injected values

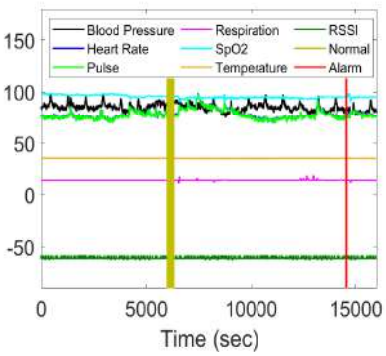

Fig. 16. Normal & Alarm

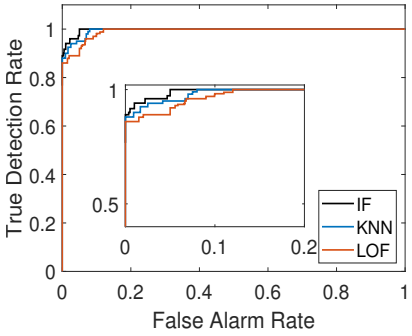

Fig. 17. ROC

reliability of the system. However, increasing the value of TPR induces an increase of FAR, and decreasing the FAR induces a reduction in TPR. Therefore, a tradeoff between TPR and FAR is required by changing the value of decision threshold. The obtained ROC is presented in Figure 17, where our approach achieves a TPR of 100% with 5% of FAR.

Afterward, we conduct performance comparison with K-Nearest Neigh (KNN [12]) and Local Outlier Factor (LOF [13]). The comparison results are presented in Figure 17 where we can notice that iForest slightly outperforms the KNN ($k = 3$), where this last reaches a TPR of 100% with a FAR of 8%. However, both (IF and KNN) achieve better performance than LOF ($k = 10$), which reaches 100% with a FAR of 12%.

The required computational complexity to derive the model in KNN is $O(ndk)$, where n is the number of records in training phase and d is the dimension of record. The computational complexity to detect anomalies in LOF is $O(n^2)$, while the complexity is linear $O(1)$ for IF. The measured execution time for IF is 0.009 sec, while it is 0.079 sec for LOF and 0.284 sec in KNN. It is important to note that these values are derived by our implementation in Raspberry PI 4 model B (4GB of RAM and 1.5GHz quad-core processor). They depend on the memory and CPU of the device. Our results show that IF requires significantly less resources and less processing power, especially as it is intended for use in devices with low processing power and small memory capacity.

V. Conclusion

Remote healthcare monitoring is reliant on BLE and devices without keyboard/display with the insecure "Just Works" pairing mode and its default pin code. In this paper, we propose a new approach to secure the exchange using ECDH and Bloom Filter. They are used to generate keys from devices within predefined RSSI (or limited distance). Once the connection is secured, the keys should be updated regularly from a matrix of keys to prevent disclosure. Furthermore, the transmitted data are acknowledged to prevent blackhole attack, and to prevent the injection attack, the central device detects outliers records using their RSSI. Our experimental results showed that our proposed approach enhances the security and the reliability of the system by reducing false alarms triggered by maliciously injected data. In our future work, we are looking for reducing the computational complexity and enhancing the authentication between the central and the peripheral device.

References

[1] Bluetooth SIG. Bluetooth Radio Versions, Last visited: September 2021.

[2] Abdelkader Lahmadi, Alexis Duque, Nathan Heraief, and Julien Francq. MitM Attack Detection in BLE Networks using Reconstruction and Classification Machine Learning Techniques. In *2nd Workshop on Machine Learning for Cybersecurity (MLCS'20)*, pages 1–16, Ghent, Belgium, 2020.

[3] Sode Pallavi and V Anantha Narayanan. An Overview of Practical Attacks on BLE Based IOT Devices and Their Security. In *5th International Conference on Advanced Computing Communication Systems (ICACCS'19)*, pages 694–698, 2019.

[4] Seth Sevier and Ali Tekeoglu. Analyzing the Security of Bluetooth Low Energy. In *International Conference on Electronics, Information, and Communication (ICEIC'19)*, pages 1–5, 2019.

[5] Kai Ren. Bluetooth Pairing Part 3 – Low Energy Legacy Pairing Passkey Entry, 2016.

[6] Karim Lounis and Mohammad Zulkernine. Bluetooth Low Energy Makes "Just Works" Not Work. In *3rd Cyber Security in Networking Conference (CSNet'19)*, pages 99–106, 2019.

[7] Marco Cominelli, Paul Patras, and Francesco Gringoli. One GPU to Snoop Them All: a Full-Band Bluetooth Low Energy Sniffer. In *Mediterranean Communication and Computer Networking Conference (MedComNet'20)*, pages 1–4, 2020.

[8] IEEE Std. IEEE Standard for Authenticated Encryption with Length Expansion for Storage Devices. *IEEE Std 1619.1-2018 (Revision of IEEE Std 1619.1-2007)*, pages 1–55, 2019.

[9] Faheem Zafari, Ioannis Papapanagiotou, Michael Devetsikiotis, and Thomas J. Hacker. Enhancing the accuracy of iBeacons for indoor proximity-based services. In *IEEE International Conference on Communications (ICC'17)*, pages 1–7, 2017.

[10] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation Forest. In *Eighth IEEE International Conference on Data Mining*, pages 413–422, 2008.

[11] Physionet. http://www.physionet.org/physiobank/database/mimicdb, Available online: Last visited September 2021.

[12] Thai-Mai Thi Dinh, Ngoc-Son Duong, and Quoc-Tuan Nguyen. Developing a novel real-time indoor positioning system based on BLE beacons and smartphone sensors. *IEEE Sensors Journal*, 2021.

[13] Omar Alghushairy, Raed Alsini, Terence Soule, and Xiaogang Ma. A Review of Local Outlier Factor Algorithms for Outlier Detection in Big Data Streams. *Big Data and Cognitive Computing*, 5(1):1, 2021.