12-19-2020

Improving a wireless localization system via machine learning techniques and security protocols

Zachary Yorio
James Madison University

Improving a wireless localization system via machine learning techniques and security protocols

Zachary Yorio

A thesis submitted to the Graduate Faculty of

JAMES MADISON UNIVERSITY

In

Partial Fulfillment of the Requirements

for the degree of

Master of Science

Department of Computer Science

December 2020

FACULTY COMMITTEE:

Committee Chair: M. Hossain Heydari

Committee Members/Readers:

Samy El-Tawab

Brett Tjaden

Dedication

This work is dedicated to my parents Jeffrey and Elizabeth. Their constant love and support have allowed me to overcome immense obstacles and continue pursuing excellence, both in my life and work. I am a lucky son.

Acknowledgments

I would first like to thank my advisors, Dr. Samy El-Tawab and Dr. Mohammed Heydari. Their continuous understanding, support, and flexibility through my extenuating circumstances, as well as a global pandemic, were integral in the success of my research. We have proven that through Zoom and Microsoft Outlook, all things are possible. Second, I would also like to thank Dr. Brett Tjaden for serving on my thesis committee. I am grateful to all three of these professors, as well as all other InfoSec faculty members, for the excellent education they have provided me over the past three years.

Table of Contents

List of Tables

List of Figures

Abstract

The recent advancements made in Internet of Things (IoT) devices have brought forth new opportunities for technologies and systems to be integrated into our everyday life. In this work, we investigate how edge nodes can effectively utilize 802.11 wireless beacon frames being broadcast from pre-existing access points in a building to achieve room-level localization. We explain the needed hardware and software for this system and demonstrate a proof of concept with experimental data analysis. Improvements to localization accuracy are shown via machine learning by implementing the random forest algorithm. Using this algorithm, historical data can train the model and make more informed decisions while tracking other nodes in the future. We also include multiple security protocols that can be taken to reduce the threat of both physical and digital attacks on the system. These threats include access point spoofing, side channel analysis, and packet sniffing, all of which are often overlooked in IoT devices that are rushed to market. Our research demonstrates the comprehensive combination of affordability, accuracy, and security possible in an IoT beacon frame-based localization system that has not been fully explored by the localization research community.

Chapter 1

Introduction

The rise of Internet of Things (IoT) and Wireless Sensor Networks (WSNs) has created a new climate for increased implementation of wireless devices to augment day-to-day activities. Advances in technology have also brought the size and price-point of wireless sensors down to a level of eager adoption. Nowadays, IoT wearables such as Fitbits are commonly seen on individuals, or even incentivized by employers, normalizing their presence in all areas of life [1].

This thesis contributes to the range of IoT wearables on the market by proposing a wearable node that can provide room-level localization. This device would only require the Received Signal Strength Indicators (RSSI) of 802.11 WLAN beacon frames already being broadcast in the commercial building. Such technology would provide a cheap and effective solution for companies to implement tracking capabilities without the need for expensive after-market hardware installation. The localization systems currently available need sensors to be installed in nearly every room of the building. The cost of hardware and installation is a significant hurdle that most facilities will never overcome, denying them of this valuable feature.

Room-level localization can be invaluable in places such as hospitals. Administration can track staff patterns, checking for appropriate time spent in patients' rooms, and ensuring tasks are completed efficiently. Patients can also be tracked with a wearable node for their own safety, especially for children or patients with dementia. The other emergent use case for this system is the challenging task of contact tracing for those testing positive for *COVID-19*. Using the historical data recorded in the back-end database, our system allows authorized personnel to query for those who had been in the same room at the same time. In doing so, the self-quarantining measure can be taken to help reduce the spread of *COVID-19* any further [2].

For a system responsible for all of this, accuracy is vital. However, we have discussed the importance of monetary efficiency as well. To find a balance, we have added improvements in the system's software and data analysis functions to improve our localization schema further. Machine learning techniques can be applied to previously collected data to interpret future data sets better. In this way, we can create an accurate model of a building's RSSI features and continuously improve

its ability to correctly predict a node's location in the future at no extra cost.

The functionality of new and exciting IoT technology often overshadows the necessity for accompanying security. In our work, we acknowledge the need to secure data transmission, especially since we have designed our system with implementations in healthcare facilities in mind. We aim to secure data packets, reduce the risk of man-in-the-middle attacks, and mitigate proximal threats like side channel analysis.

In this thesis, we combine the concepts of RSSI-based localization, machine learning, and security protocols for IoT WSNs. In chapter 3, we discuss the hardware and software specifications of our scalable localization system. Chapter 4 introduces machine learning concepts that may improve our localization system and protect WSNs security/privacy from several risks. Our results and implementations are highlighted in chapter 5.

Chapter 2

Related Work

In this chapter, we review contemporary works in the realm of single-entity localization. Outdoor and indoor techniques are compared, as well as the differing methods used for indoor navigation due to the complexities of indoor terrain. The technologies available for indoor tracking techniques are discussed, and examples of current machine learning algorithms are provided. We also discuss security measures implemented in systems such as these to protect user data.

2.1 Outdoor Localization: Historical Background

Localization is the process of determining the position of an object in space [3]. This concept can be achieved using various technologies, in both outdoor and indoor circumstances. Technologically assisted outdoor localization can be traced back to the 1940's with the use of radio-navigation systems such as the Long Range Navigation (LORAN) and Decca Navigator systems. The United States used the LORAN system during World War II in order to assist ships and aircraft in navigating the Pacific ocean [4]. Using low frequency wavelengths and calculating distance via a pulse timing system, LORAN signals could reach distances of up to 1,500 miles with an accuracy within tens of miles [5]. Decca was another hyperbolic radio navigation system developed by scientists in the United Kingdom during World War II. Instead of pulse timing, Decca used the phase comparison of two low-frequency signals to estimate distance [6]. While effective for decades after their implementation, these systems were costly in the requirement that radio stations be constructed and maintained along the coasts in order for ships and aircraft to continue using such radio-navigation methods.

Improvements in radio-navigation continued into the 1980's; however, the next landmark technology was introduced in 1973. The Global Position System (GPS) moved radio navigation from the earth into space with satellites orbiting the planet twice a day. Their consistent pathing around the globe allows for precise signal transmissions to and from individuals' receivers on the ground [7]. By calculating the receiver's latitude, longitude, and altitude via radio waves, location and navigation could be determined. With constant unimpeded signals, satellites provide accurate time and location measurements for all users with GPS receivers. Initially, this was limited to the US

military, but was expanded to the general public for personal use in the 1980s [8].

Recently, in the 21st century, GPS systems have served outdoor navigation and localization systems well but have been adjusted for certain circumstances. A recent example of such ingenuity by Yucer et al. shows how one might use an Unmanned Aerial Vehicle (UAV) as both a receiver and transmitter in a larger tracking system. With this, entities can be tracked via wireless sensors emitting radio frequencies from which the UAV can calculate position via multilateration of the tracking node's received signal strength [9]. In this method, the more energy-consuming task of GPS coordinating is left to the UAV itself, allowing for the wireless sensors to be usable in the field for longer durations. This is ideal in tough terrain where people or vehicles are impeded on the ground, but a small UAV can travel with ease. The test runs show that this method of tracking can track a wireless sensor within seven meters of its true location in a 3.14 square kilometer survey area, a zone within which localization was previously impossible [9].

However, GPS-based systems do have issues to overcome. The main one being the requirement for a direct line of sight with satellites or UAV monitors for accurate localization. Not all outdoor spaces have direct aerial views, with areas like canyons and forests creating dead zones for GPS signals. An option to remedy these circumstances is the implementation of wireless sensor nodes throughout an area for trilateration or multilateration of a tracking node. Ismail et. al. propose a solution to such situations with the installment of Received Signal Strength (RSS-based) sensor nodes within the area needing to be monitored [10]. Nodes would come equipped with a GPS chip to confirm the location of the stationary node, an Arduino Real Time Clock (RTC) module, as well as a Secure Digital (SD) card for data storage. This way, the receiving sensors can record the transmission power, signal strength, and time of reception (via the RTC) from any nearby tracking node. With this data, an accurate estimation of current location of the transmitting node can be calculated by the estimated distance from three or more receiving nodes. In the experimental 10m by 10m grid, location estimations could be within 1.83m of the true location of the node. The accuracy increased with the number of receiving nodes, but at the cost of increased computational complexity.

Another technique used for outdoor localization that has become more prevalent in the past few decades is the use of cellular networks to track mobile devices. Unlike wireless signal nodes that must be installed in each desired area, cellular service companies have already installed and are continuing to expand the infrastructure required for tracking a person by a mobile device. The most basic type of tracking done this way is by Cell-ID. This algorithm simply places the mobile user in the rough area on a map coinciding with the cell tower that the device is currently connected to [11].

In rural areas, this could prove to be highly inaccurate. The second available tracking algorithm is localization via centroid. This takes into account the multiple signal strengths received by all nearby cell towers from a mobile device and uses them all to calculate distances for a triangulation method [12]. Trogh et al. are currently developing an even more detailed algorithm called AMT: antenna, map, and timing information-based tracking. This technique uses information such as antenna orientation, current mapping, time to project direction, and rate of travel for the estimation of where the mobile device will be in the near future [11]. This system also employs metadata for each mobile device ID to infer where one might be going based on historical trends. With this, users can be effectively tracked by foot, bike, or car. Under ideal circumstances, iterations of this AMT algorithm could track devices with an estimated median accuracy of 122m of the true location in urban environments. This is an 88% improvement, proving cellular network tracking to be a viable option for outdoor localization of moving objects [11]. Figure 2.1 visualizes the technologies brought forth for outdoor localization in recent history.

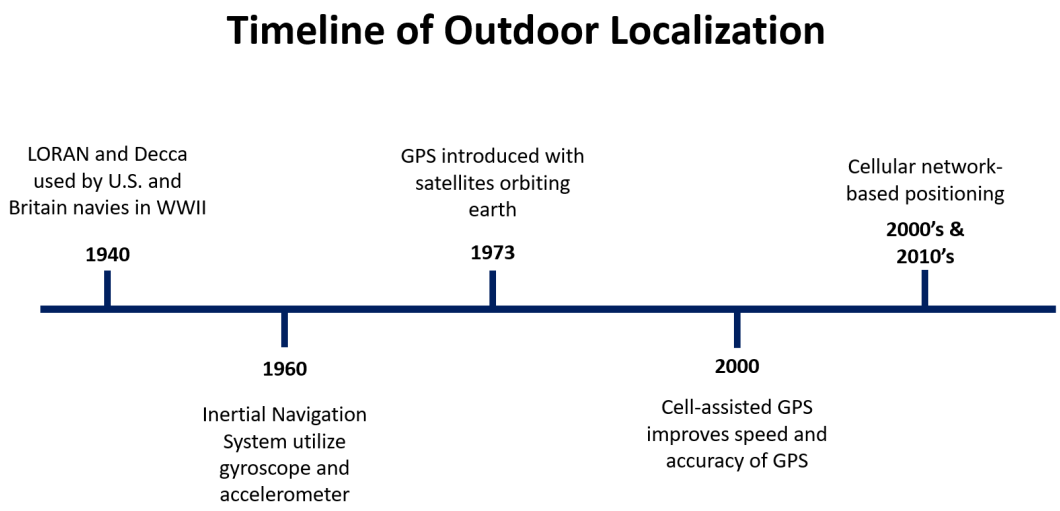

Figure 2.1: Timeline of outdoor localization technologies in recent history

2.2 Indoor Localization

2.2.1 Localization Techniques

Indoor localization has had to become more clever than its outdoor counterpart to reach the levels of accuracy and precision a tracking system would need in order to be viable enough for investment and implementation. Many differing techniques have emerged due to the requirements of such systems. The list includes systems that utilize received signal strength indicator (RSSI), channel

state information (CSI), angle of arrival of a signal (AoA), time of flight of signal packets (ToF), time difference of arrival between sensor nodes (TDoA), return time of flight for a signal (RToF), and phase-of-arrival of a signal (PoA) [13]. While all these techniques incorporate the transmission and reception of a type of wireless signal, the advantages and disadvantages of their implementations dictate their ideal use cases. Table 2.1 provides a brief overview of the techniques discussed.

Table 2.1: Overview of Indoor Localization Techniques

Technique	Description	Examples
Received Signal Strength Indicator (RSSI)	Distance judged based on the attenuation of a signal sent from a transmitter to a receiver.	[14, 15, 16, 17]
Channel State Information	Distance measured by properties such as the frequency, attenuation, phase, and energy intensity of a wave form.	[13, 18]
Angle of Arrival (AoA)	Time difference of arrival between the communicating nodes allows for an approximate angle to be calculated. The position of the transmitter is then assessed by projecting the intersection point of two nodes' received signal lines.	[13, 18, 19]
Time of Flight (ToF)	Distance calculated by how long it takes for a signal to reach the receivers from the transmitter. Then, distances of two or more nodes are compared and a location can be triangulated.	[13, 18, 20]
Time Distance of Arrival (TDoA)	Calculates location in the same way of ToF, but does not require an absolute time between transmitter and receivers.	[18, 21]
Return Time of Flight (RToF)	Measures the round trip a signal makes between a transmitter and receiver much like ToF. Clock synchronization is required.	[18, 22]
Phase of Arrival	Uses an antenna array to capture the same signal waves at known distances apart. The phase difference between the antennas can be measured and location can be calculated by determining the waves' original point of convergence at the spot of the sender.	[18, 22]

Received Signal Strength Indicator

The first technique, Received Signal Strength Indicator (RSSI), operates on IEEE 802.11 WiFi and can be monitored by most network interface cards (NICs). Distance can be judged based on the attenuation of a signal sent from a transmitter to a receiver. Using these measurements, the location of one device can be calculated given the known placement of other receiver nodes [14, 16]. This is a common technique since almost every mobile and wireless device have NICs that can be monitored. However, the accuracy of RSSI values can be heavily impacted by the surroundings in which they are transmitted. Signals can be absorbed or slowed dramatically by walls, people, or any other obstructions placed between transmitters and receivers. Such impedance can drastically throw off

signal strength values.

Channel State Information

The second technique is the assessment of a signal by its Channel State Information. Distance can be measured by analyzing properties such as the frequency, attenuation, phase, and energy intensity of a wave form. This technique is more robust than RSSI, but requires more hardware. Also, not all NICs support the collection and analysis of all channel state information which makes implementation more expensive since particular hardware must be sought out [13, 18].

Angle of Arrival

The third technique is Angle of Arrival. This uses two or more receiving nodes to collect signals from a transmitter [18]. The time difference of arrival between the communicating nodes allows for an approximate angle to be calculated. The position of the transmitter is then assessed by projecting the intersection point of two nodes' received signal lines [19]. This technique is useful since it only requires two receiving nodes for localization of the transmitter to be possible, but the hardware is complex and expensive for implementation in a system.

Time of Flight

Our fourth technique, Time of Flight (ToF), is a more straightforward technique but uses a similar approach to angle of arrival. The distance of nodes from receivers is calculated by how long it takes for a signal to reach the receivers from the transmitter. Then, the distances of two or more nodes are compared and a location can be triangulated [18]. Time of flight can perform better indoors than RSSI since it can be implemented with wider frequency band signals. However, this precision comes at a higher cost in hardware as well.

Time Distance of Arrival

The fifth technique is Time Difference of Arrival (TDoA). Location is calculated in the same way as ToF, but does not require an absolute time between transmitter and receivers. Since it is only using the time difference between different packets as they are received, it helps reduce the complexity of time synchronization between the two different nodes while maintaining the high accuracy given by the time of flight technique. The major drawback of TDoA is its expensive hardware requirements just as with ToF [21].

Return Time of Flight

The sixth technique, Return Time of Flight, measures the round trip a signal makes between a transmitter and receiver much like ToF [18]. Clock synchronization is required, and it is heavily impacted by sampling rate and the bandwidth of the signal. It is susceptible to high inaccuracy if synchronization is off since the ToF is considered twice, but can be highly accurate when tuned correctly [22].

Phase of Arrival

The seventh and final technique is Phase of Arrival. This technique uses an antenna array to capture the same signal waves at known distances apart. This way, the phase difference between the antennas on the array can be measured and the location of the transmitting node can be calculated by determining the waves' original point of convergence at the spot of the sender [18]. The main drawback of this technique is its dependence on line of sight [22]. This makes it a less desirable option for indoor localization since line of sight cannot always be guaranteed or even possible in some areas.

2.2.2 Localization Technologies

As researchers have proposed and proven new techniques for determining positioning, developers have worked alongside them in creating new technologies that have adopted these methods. This section will cover the primary ways in which the modern techniques have been applied to usable technologies in systems. These technologies include 802.11 WLAN RSSI, Radio Freqency Identification Devices, Bluetooth, Ultra-wideband, ZigBee, LED sensors, and Acoustic sensors. Table 2.2 below provides a brief overview of the technologies covered.

802.11 WLAN

WLAN localization is a radio frequency-based approach that operates on the standard IEEE 802.11 WiFi band signal. RSSI values can be easily collected and parsed while working on this signal band with wireless devices possessing network interface cards. With the recent boom in personal smart devices and pervasive internet-of-things systems in our daily lives, WLAN devices can be found in most buildings [18]. This technique allows for improved tracking, localization, and area fingerprinting. The main drawback of WLAN localization technology is that it was not initially designed for tracking, but rather, communication. The signal properties are not ideal for localization,

Table 2.2: Overview of Indoor Localization Technologies

Technology	Description	Examples
802.11 WLAN RSSI	RSSI values can be easily collected and parsed while working on this signal band with wireless devices possessing network interface cards.	[13, 18, 23]
Radio Frequency Identification Device (RFID)	Cheap devices emit low energy data for readers to collect via high frequency waves.	[18, 24]
Bluetooth	Operates on the IEEE 802.15.1 band signal, transmitting data over short distances via ultra high frequency radio waves. It offers low energy consumption at a cheap price per unit.	[18, 25, 26]
Ultra-wideband (UWB)	A low energy and low range radio signal operating on a larger bandwidth of the radio spectrum. Transmits with no interference from other narrowband radio waves which gives it high accuracy.	[18, 27]
ZigBee	Radio wave signal operating on 802.15.4 that broadcasts at a low rate, consumes less power, and is very affordable to implement.	[18, 28]
LED Sensors	Sensors measure the angle of arrival of an LED emitter to calculate distance and positioning in relation to the sensor.	[18, 29]
Acoustic Sensors	Microphones in digital sensors can measure the ToF of sound waves to create profiles or fingerprints of rooms.	[18, 30]

requiring algorithms to do the heavy lifting for improving accuracy.

Wang et al. propose an indoor localization technique that conducts curve-fitting on RSSI data [23]. A fingerprint of each area of the environment is created during an offline phase, then an RSS distance function for each transmitter is created based on the data. During the online phase, real-time data is computed through algorithms determining the lowest sum of distance errors compared to the theoretical values calculated during the offline phase. Gradient search algorithms such as this can improve indoor localization by up to 20% [23]. This notable improvement in accuracy makes curve-fitting worthy of consideration, but other researchers have tried moving away from the concept in order to reduce upfront time investment. Hernandez et al. instead use vector regression during the offline phase of data collection to improve the location estimation via RSSI [31]. Then, during the online phase of the system, a space estimator function works to fill out each room's fingerprint over time. This method helps to reduce setup time while maintaining localization accuracy. The mean distance error was reduced by 60% using this method to track moving devices in comparison to a traditional fingerprint methodology. A third approach to RSSI-based tracking has been taken by Cheng et al. that also looks to reduce setup time and offline data collection. In their system, piece-wise linear functions are tuned for each type of device used in the setup. At first, signal strength to

distance values are approximated, then fine tuned with their Expectation Maximum algorithm [32]. This system stands out due to its ability to improve over time by adjusting the linear functions used, and the fact that each device's possibly differing signal strength is accounted for in the algorithm's distance calculations.

Researchers have also explored the possibility of using RSSI in a device free manner. Youssef et. al. have proposed a passive system in which the fluctuations in RSSI could be observed in order to detect an intruder in an area, as well as track their general movement [33]. A radio map was constructed for the test area in order to capture the relation between broadcast signal strength and physical distance. Success was shown in trials with limited false positives in areas with nominal WiFi. This technique shows promise as a security measure, but is not robust enough to track many people simultaneously.

Radio Frequency Identification Device (RFID)

Radio Frequency Identification Device (RFID) technology emits low energy data for readers to collect via high frequency waves. Their low cost and low power requirements make them appealing to system engineers, however, their accuracy and range are considerable setbacks. For this reason, RFID tags are commonly used in addition to other systems such as WLAN tracking nodes. Silva et al. have created a system that incoporates both WiFi and RFID fingerprinting for improved accuracy. Having the ability to place multiple RFID readers within a single room at low cost can take a localization system from room-based tracking to a more refined and specific location-based tracking [24].

Bluetooth

Bluetooth operates on the IEEE 802.15.1 band signal, giving wireless devices the ability to transmit data over fairly short distances via ultra high frequency radio waves. This technology also offers cheaper low energy consumption device options and long lasting transmitters that can operate up to 14 years on a single battery life [25]. Bluetooth can be used in systems that calculate location based off RSSI, AoA, and ToF techniques. Accuracy is a common issue with this technology, since like WiFi, it was built primarily for data transmission and not localization utility. Regardless of this issue, Bluetooth's affordability and pervasiveness in wireless technologies still make it a desirable option. Danis and Cemgil demonstrate a model in which Bluetooth low energy beacons can successfully create RSSI fingerprints of rooms by streaming them from multiple points in a room [26]. The affordability of Bluetooth beacons allows for the placement of many more nodes than in

other tracking technologies' designs. Therefore, the less reliable signal of one Bluetooth beacon is bolstered by the signals of others in the same room.

Ultra-wideband Radio

Ultra-wideband (UWB) is a low energy and low range radio signal technology that operates on a large bandwidth of the radio spectrum. The nature of this signal allows it to transmit data with no interference with other narrowband radio waves which allows it to maintain high accuracy. Ultra-wideband has a shorter signal range like Bluetooth, but has much more costly hardware to produce its highly accurate signals. Zwirello et al. put forward a system utilizing UWB technology that gauges the TDoA of wideband radio signals. Inside a 10m by 10m room, location accuracy could be measured within 2.5 cm [27].

ZigBee Radio

ZigBee is the final radio wave technology discussed in this work. Operating on IEEE 802.15.4 low-rate wireless personal area network, ZigBee maintains a low data rate, energy consumption, and operation cost. Achieving high accuracy indoors can be a challenge due to its operating on RSSI for localization. Aykac et al. have put forth a system that dynamically calculates the position based on a ZigBee model. A modified particle filtering algorithm reduces computational overhead and accounts for variances in signal strengths [28]. By doing so, a target can be tracked indoors with an accuracy of approximately 1.5-2m while operating with conservative power consumption.

LED Sensors

One alternative to localization through the analysis of radio wave signals is a technology that transmits information via visible light. Through this method, sensors measure the angle of arrival of an LED emitter to calculate distance and positioning in relation to the sensor. As a positive, LED lights are very cheap and highly accurate since light waves aren't as susceptible to distortion as radio waves. However, light does require a direct line of sight at all times for data to transmit, requiring creative solutions to maintain a constant signal. Shao et al. have designed a real-time indoor localization technique based on common light sources found within a building. By characterizing each light source's signal collected by retroreflectors, as well as the backward channel signals sent to the diodes on the lights from the nodes' reflectors during test trials, high location and orientation accuracy can be achieved. The retroreflector nodes can then communicate a specific node's location

via WiFi [29]. This setup makes the system more robust, allowing multiple signal options in a given area to protect against dynamic line of sight issues.

Acoustic Sensors

Acoustic Signal is the final localization technology considered. While not the easiest to achieve precise accuracy, it can be useful in room-level tracking of people or devices. Microphones in digital sensors can measure the ToF of sound waves to create profiles or fingerprints of rooms. Jia et al. put forth a room by room classification method based on acoustic properties called Room Impulse Responses (RIR). The initial setup of collecting each RIR can be costly in both time and money, but with enough training data, an accuracy of over 95% can be achieved with this system [30].

2.3 Indoor/Outdoor Hybrid Localization

Researchers have also been investigating ways in which a system can handle both indoor and outdoor tracking through one robust system. The main challenges to overcome for systems like this is the drastic reduction of GPS indoors, as well as the cost and accuracy of indoor infrastructure. In order to leap these hurdles, most systems presented have proposed a combination of techniques and technologies in order to maintain accuracy.

Pereira et al. deliver a promising prototype that can be run on any smartphone. Using GPS, cell tower scanning, and WiFi access point fingerprinting, a device should always be trackable when moving from outdoors into a building with wireless internet. The device is connected to a database that has records of georeferenced access points and mobile towers so that locations and RSSI fingerprints are remembered so that future location estimations are easy to compute and always increasing in accuracy [34]. This database also self-improves by consistently scanning and checking for new nearby access points while indoors to add more georeferenced RSSI fingerprint data to repository. Another smartphone-based solution is proposed by Kulshrestha et al. that tries to seamlessly track devices using portable sensor units (PSU) [35]. They have created self-contained tracking units that scan for GPS, WiFi, and Bluetooth signals from a smartphone or provided Bluetooth sensor prototype. Once a tracker is identified by a PSU, its MAC ID is stored in the cloud database for long term reference. This system is similar to Pereira's in that multiple technologies are incorporated, but differs in the use of self-provided infrastructure via the PSUs. While they are able to improve tracking accuracy by up to 44.49% in comparison to just simple GPS, the cost, setup, and maintenance of the PSU's in this system must be taken into consideration for implementation.

Even more recently still, hybrid localization systems have come to incorporate more technologies. In 2019, Li et al. revealed a crowd sensing positioning system that creates a radio map based off of smartphones' GPS, WiFi, as well as pedestrian dead reckoning (PDR) sensor information [36]. PDR is the tracking of a smart device based on built in inertial sensors like pedometers, accelerometers, and magnetometers [37]. With the help of PDR on top of the usual GPS and WiFi technologies, accurate radio maps of areas can be created without prior site surveillance by the system's engineers. The system's database also conducts particle filtering in order to reduce the noise of the data and create seamless maps of floorplans for future tracking. Experiments carried out by the team at large venues such as shopping malls have confirmed the system's viability, delivering a location accuracy of under 2.8 meters when using GPS, WiFi, and PDR while tracking a moving target inside a previously unmapped venue [36].

2.4 Machine Learning

Localization will always be at the mercy of balancing technique and technology limitations, as well as the cost and efficiency of the hardware involved in a system. The newest frontier for innovation has moved in the direction of advanced algorithms, looking towards machine learning and Big Data to help increase accuracy by analyzing trends and patterns in user data. Machine learning can be defined as the ability of a system to improve upon itself by assessing data, rather than being explicitly programmed to do so [38]. In localization, this improvement would be the accuracy of tracking a node only given the data from previous trials. While there are many machine learning algorithms, we will look at the ones most common to localization problem solving. These algorithms include: artificial neural networks, online sequential learning, link distance-support vector, cooperative machine learning, as well as random forest. Table 2.3 provides a brief overview of the algorithms covered.

Artificial Neural Networks (ANN) model the decision making scheme of the human brain. Input values are assigned weights that simulate the strengthening or weakening of neural pathways due to increase or decrease in task efficiency. A neural network typically learns from training data since error can be calculated between its calculation and the known or expected value available [47]. Adjustments can then be made to further reduce the error value. Hashem et. al. utlize an ANN in their DeepNar system, performing localization based on Round Trip Time measurements. These signal values are collected to map out a fingerprint of each room. A neural network is created with the access points as the input layer and the probability distribution of grid locations as the output

Table 2.3: Overview of Machine Learning Techniques

Technology	Description	Examples
Artificial Neural Networks (ANN)	Modeled after neurons in the human brain, making decisions based off of input data with different weights based on their class's importance. Increase accuracy from training data. Back propagation can be used to learn from live data.	[13, 20, 39]
Extreme Learning Machine (ELM)	A type of neural network usually with a single hidden layer of neurons and does not use back propagation. Can learn very fast in comparison because of these modifications and provides good generalizations about input data.	[40, 41, 42]
Support Vector Machines (SVM)	Supervised algorithm able to perform both regression and classification calculations. Train by solving a constrained quadratic equation, theoretically finding a correct answer based on data, not just making a prediction.	[43, 44]
Random Forest Algorithm	Supervised algorithm able to perform both regression and classification calculations. Uses many decision trees to make a majority vote decision based off of provided classifiers. Resistant to overfitting by using many decision trees.	[45, 46]

layer. The hidden layers in the network learn via training data to properly classify fingerprints with specified grid locations with the highest percent certainty as possible [20].

Neural networks may also be used in other aspects of localization, such as to normalize the differing signal strengths from varying devices. In their OmniCells model, Rizk et. al. implement autoencoders to boost their system's accuracy by extracting transformed features of the original RSSI values before applying them to calculate physical distance [48]. This allows varying devices with different broadcast strengths to be tracked in the same trained ANN system. This technique has built-in safeguards from overfitting training data, allowing test trials to reduce the median location error without their algorithm from approximately 4 meters, down to under 2 with their algorithm implemented.

Farid et al. have created a hybrid localization system using both WiFi and wireless sensor nodes manually placed in the researchers' testing area. WiFi scanner software collects RSSI data, and ZigBee nodes also produce RSS data collected by a receiver. The ANN (Artificial neural network) analyzes training data from these devices. The accuracy and precision for predicted values can increase by running this data against the known or predicted values. The researchers found that the average distance error could be reduced to 1.05m or better using the trained ANN. [49]. This is much more accurate than a system based solely on WiFi or wireless sensor networks (WSNs).

Li et al. present another neural network that uses a hybrid approach. Instead of wireless sensors, the WiFi technology is accompanied by PDR inertial sensors to help with the continuity of movement tracking [39]. In practice, this system tracks smart devices via WiFi receivers and PDR around a large indoor area such as a shopping mall. The system is enhanced on the back end by utilizing Gaussian distribution in its machine learning analysis of data when calculating wireless fingerprinting uncertainty values. The Gaussian training of the database helps normalize the data and estimate how normal or likely a data point is, or if it is an outlier that needs to be thrown out [50].

Another popular model for machine learning is an Extreme Learning Machine (ELM), which performs well and learns much faster than machines that use backpropagation, such as ANNs. For localization practices, many researchers have investigated the implementation of Online Sequential ELMs (OS-ELM) due to their ability to handle data inputs one at a time, or chunk by chunk [40].This constant learning during online data collecting makes OS-ELM a desirable option for a dynamic problem such as real-time tracking.

Zou et al. implement an OS-ELM system that significantly reduces the time needed to set up a tracking system compared to ANN-based counterparts since no offline site survey is needed [41]. This localization-oriented ELM has two phases. First, it conducts an initialization phase where a small amount of training data is given to the machine. After that data is parsed, the system will constantly update and improve itself based upon online data while constantly tracking online wireless nodes. One main purpose of this machine learning model is to adapt to environmental changes, making for a more robust localization system. Using Android smartphones, offline training data, and offline calibration points to help orient the system, the algorithm far exceeded other algorithms such as K nearest neighbor (KNN), fuzzy KNN, and batch ELM. It provided localization accuracy up to 1.973m, compared to providing 3.098m, 2.728m, and 2.581m, respectively, by the aforementioned competing algorithms.

One downfall of OS-ELMs is that they constantly try to improve based on the most current data given to them. The number of features within the machine will increase with new data and incorporate them into future estimations, creating the danger of overfitting. Al-Kahleefa et al. present a Knowledge Preserving OS-ELM (KP-OSELM) that is designed to recall knowledge from any previous state it has been in to provide the best estimation for the current data in question. This system is designed to identify cyclic behavior for an individual in an indoor environment. With this functionality, the KP-OSELM reached up to 92.74% accuracy in test trials as compared to only the standard OS-ELM's 7.99% for the same data set cycle [42].

Support Vector Machines (SVM) have also become a viable machine learning option for localiza-

tion systems. SVMs are a supervised type of algorithm, able to assess and classify data for regression or classification. Unlike ANNs, which are set up to use backpropagation, SVMs train by solving a constrained quadratic equation, which theoretically has a correct answer instead of an educated prediction [43]. Anusha et al. have applied a support vector regression (SVR) model to their device-free indoor localization technique, which uses distance-support vector measurements instead of the classic RSSI model [44]. In this system, sensor nodes are set up in a room to constantly communicate with one another. When a person or people enter the room, fluctuations in the link signals between nodes are collected then analyzed via SVR. A 3m x 3m x 3m room was tested with 16, 32, 48, and 64 nodes in different trials. The researchers found that while increasing the nodes improved the number of links and data points, it also increased the chance for error. The system could optimize its learning algorithm by assigning more weight to the nodes with link signals being directly affected by the target being tracked and all but forgetting the unused nodes that could end up providing too many false positives [44].

the Random forest algorithm is another supervised machine learning algorithm that is able to perform both classification and regression-oriented calculations. Though not as fast as an optimized neural network, random forest models train quickly and are flexible in handling features and their dynamic weights . It also self-mitigates against the common machine learning problem of overfitting data by having many decision trees in the system's "forest" to help randomize its calculations' starting points [45]. Guo et al. have utilized a random forest algorithm to train and test their localization framework. By doing so, the performance of the original system based solely off a single fingerprint RSS signal or signal strength difference is drastically increased [46]. Groups of RSS fingerprints collected from multiple antennas are collected and used to calculate distances using multiple different algorithms. These calculations are then treated as the trees of the algorithm's forest in order to arrive at an agreed upon answer via regression by means of analyzing the different trees' most commonly agreed upon answer. It is worth noting that this localization solution can also operate in an unknown indoor scenario[46]. The robust accuracy and speed with which Guo et al.'s system operates makes random forest a promising machine learning algorithm to investigate in future works.

2.5 Security

The range of research being conducted and products coming to market in the realm of IoT and localization has shown the dramatic strides in technology we have made in the past few years.

Unfortunately, many are designed solely with function in mind and without any consideration for their users' privacy or security. It is important to bring this to attention because once built; it is often difficult to re-engineer a system to have the necessary precautions installed.

Destiarti et al. have begun to consider this problem with their encryption implementations in a WSN localization system based on RSSI, path loss exponent (PLE), and anchor node coordinates. Before an anchor node in the WSN communicates with a mobile node, the RSS, PLE, and coordinates of the anchor node are encrypted. The mobile node then simultaneously receives and decrypts this information from three different anchor to perform trilateration [51, 52].

When the nodes initially connect, RSA public keys are shared between them. All recipients can decrypt the initial packets encrypted with secret keys in the network through asymmetrical encryption. These initial packets contain each node's AES and MD5 symmetric keys that must be shared with all other legitimate nodes for symmetric encryption and decryption [51]. The RSA asymmetric encryption is necessary to communicate over an insecure line such as WiFi safely. However, asymmetric encryption and decryption are computationally expensive[53]. That is why once the symmetric AES and MD5 keys are shared safely via RSA, the real-time tracking messages are encrypted and decrypted much more efficiently via AES. The MD5 hashing that accompanies the AES encryption ensures that the data being sent maintains integrity between sender and receiver [54].

Experimental results show that the Raspberry Pi units acting as sending nodes achieved a processing time of 4.19 ms to encrypt and send tracking data. As receiving nodes, 82.64 ms was required. Data size and distance between receivers also impacted transmission performance [51]. No other similar experiments could be found for this survey, implying that data transmission security in localization systems has not taken priority in current works, nor been optimized beyond the scope of a single team's investigations.

Chapter 3

Localization System

This chapter discusses multiple parts of our localization system. We begin with an overview of edge nodes and edge computing. Then, our system architecture is explained. Next, the physical device is inspected and broken down into its multiple components. The software is also analyzed to further understand how a singular edge node operates. We also cover the back end of the system, which is the Cloud server and storage. Finally, we end the chapter by explaining our localization system's importance and viability in the healthcare setting.

3.1 IoT Edge Nodes and Edge Computing

With the developments and innovations seen in IoT systems and technologies, systems have also seen an increase of data transmission and computation complexity as well. Centralized cloud computing cannot always remain viable in systems monitoring and communicating with hundreds, if not thousands of devices. To allow system scaling to remain unhindered, researchers are constantly devising new ways to handle workloads within a system's architecture [55]. A current popular technique is the implementation of edge computing. Edge computing differs from the previously popular cloud computing, as responsibilities are distributed in the system. Rather than simply allowing the cloud servers to perform all complex tasks, the processing power of the nodes are also employed, moving computation closer to the targets [56].

Initially, edge nodes only sensed and recorded data to be reported back through the IoT framework for assessment. In an edge computing design, the node will also perform appropriate measurements and interpretations to be sent to the server for storage and analysis via a wired or wireless connection [57]. An edge node can be constructed and equipped with various sensors to collect data in relation to metrics such as current, motion, and temperature, with which conclusions can be drawn before data is submitted through the framework. This can be necessary for systems that require fast decision-making capabilities such as visual guidance devices [55]. The difference between 25ms local ping and upwards of 175ms ping when communicating with a cloud server can significantly impact the performance and viability of a device meant to guide a person in motion quickly.

With this decentralized design, multiple advantages can emerge. For one, hardware may become more specialized and capable of stronger computations with a specific task in mind during its design. It can also be built for robust data collection, able to record data under instances of lost signal until communications are operable again [56]. As mentioned before, latency can be drastically reduced when all decisions are not required to be sent through the cloud and back to the local device. The amount of data needing to be transmitted over the network can also be reduced, lowering network loads, since the node will filter and submit only necessary data. Implementing edge nodes can also help with the scalability and adaptability of a system. Instead of a single purpose server, these specialized devices can help carry out tasks independently and allow the central server to focus on critical services [56].

3.2 System Architecture

Figure 3.1 shows a diagram of our proposed system architecture. This localization system is comprised of edge nodes, WLAN access points, a router provided by the internet service provider, and the cloud server and database hosted outside the local network. The arrows demonstrate how the edge nodes communicate freely with the access points, and vice versa. Through these received data packets, RSSI data from the nearby access points is collected by the edge node and computed into distance values.

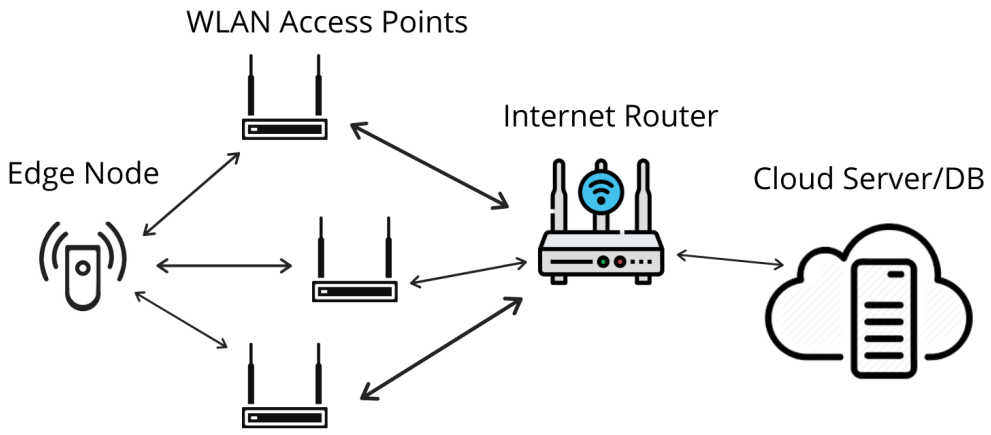

Figure 3.1: Localization system architecture diagram

The access points also have access to the network's router, which allows communication with the internet. Through this chain of connections, the edge node's collected RSSI data can be submitted

to the cloud database for storage and analysis.

3.3 Device Hardware

Our wireless edge node's primary component is an IoT device (e.g., a Raspberry Pi microcomputer). In early testing, a Raspberry Pi 3 Model B was used. This microcomputer comes with a quad-core 1.2 GHz Broadcom 64 CPU, 1 GB of RAM, Ethernet port, WiFi and Bluetooth connectivity, 4 USB ports, full-sized HDMI port, and a micro SD port for running an operating system [58]. The IoT device is powered by a micro-USB cable, either through a wall adapter plug or a portable battery pack as shown in figure 3.2. The Raspberry Pi was chosen as a prototype for this system due to its affordability, only costing approximately $35, its portability, and its reliability. These single-board computers have all the necessary ports of a regular computer, are easily carried by hand, and consistently boot up properly and quickly during testing trials.

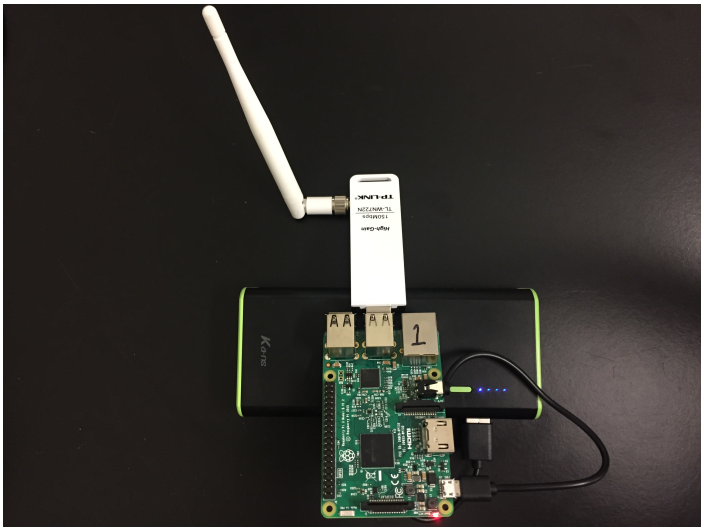

Figure 3.2: Raspberry Pi 3 micro computer

The communication between the node and nearby access points is made possible by the TP-Link TL-WN722N USB dongle attached to the Raspberry Pi. This wireless adapter provides data rates of up to 150 Mbps, allowing for signal reception even if separated by two floors [59]. This strength helps ensure that many access points are reached while the node is scanning. While the Pi computer has on-board wireless connectivity, the firmware does not allow the wireless card to be set to monitor mode. This setting allows for wireless data packets, or beacon frames, to be collected for analysis. Enabling the monitor mode on the TP-Link wireless adapter allows for beacon frames to be collected while the Raspberry Pi can still stay connected to the WLAN with the on-board NIC.

The edge node stores data and runs its operating system on a removable micro SD card. The

Rasberry Pi hardware is compatible with multiple operating systems such as Ubuntu, Mozilla Web Things, and PiNet. For our purposes, the official Raspberry Pi OS, previously called Raspbian, was chosen [60]. This operating system is a 32-bit Debian-based Linux system optimized for the single-board microcomputer. It uses an ARM architecture, which decreases the number of instructions produced by the operating system so that a smaller microcomputer such as a Raspberry Pi can run effectively with a lower cost, low power consumption, and head production [61].

The final necessary piece of hardware needed for the node is the power supply. The Raspberry Pi can run using a wall outlet, but our device must be mobile to collect beacon frames from access points to calculate positioning and direction. Figure 3.2 shows a rechargeable lithium battery pack. This battery connects to the microcomputer via micro USB and can power the node for up to eight hours.

Once a proof of concept node was completed with the Raspberry Pi 3 Model B, the next step was to reduce the node's size, since its final design is a wearable accessory for patients or employees. The first measure taken to do so was replacing the Raspberry Pi 3 with the smaller Raspberry Pi Zero W model. This $15 single-board computer is only 6.6 cm long by 3.05 cm wide with a thickness of 0.5 cm, reducing the size, weight, and cost of the node drastically [62]. The Raspberry Pi Zero W comes equipped with a 1GHz single core processor, 512MB of RAM, a mini HMDI port, one mini USB port, a micro USB power port, and provides 802.11 wireless LAN, Bluetooth 4.1, and Bluetooth Low energy connectivity capabilities [63].The Raspberry Pi Zero W can perform the functions in our design easily.

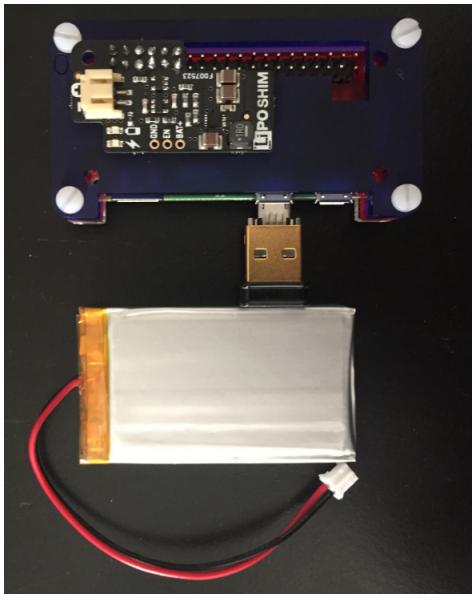

Figure 3.3: Raspberry Pi Zero micro computer

Figure 3.3 shows a prototype model of the edge node that runs on a Raspberry Pi Zero W. Instead of the sizeable TP-Link TL-WN722N USB dongle used to enable monitoring mode, the relatively small TP-Link Nano Wireless N150 Adapter was employed. For a similar price, this much smaller wireless adapter can still transmit data at a rate of 150 Mbps [64]. The other advantage is the reduced battery size that powers the node. Due to the Zero W's reduced power consumption, the battery size can also be reduced and maintain a similar operating time per charge. In our model, a LiPo SHIM adapter is soldered onto the GPIO pins of the microcomputer. This allows the LiPo battery pack to be easily connected for a lower profile mobile power supply, powering the Raspberry Pi [65].

3.4 2.4 GHz Channels and Channel Hopping

When creating a node that communicates on and monitors the packet traffic of 802.11 WiFi, it is essential to understand how devices coordinate to reduce network interference. Most devices operate on the 2.4 GHz frequency, where there are eleven channels on which a device can broadcast. The spectrum spans from around 2.4 GHz up to 2.5 GHz. Each channel is about 20 MHz wide, with an offset of 5 MHz for each channel. Channel one begins at 2.412, two at 2.417, etc. This means that all channels overlap a considerable amount. Only channels one, six, and eleven can be operating simultaneously without adjacent-channel interference, shown here in figure 3.4 [66].

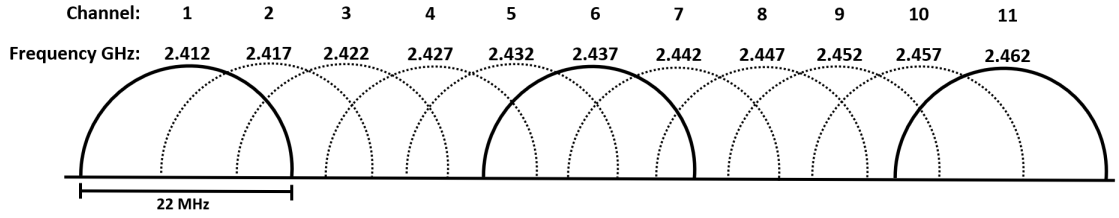

Figure 3.4: WiFi 2.4 GHz channel frequencies

A wireless device can only be set to send and receive information on one channel at a time, even if its frequency range overlaps with multiple other channels. The situation where many nearby devices are communicating on the same or close channels greatly reduces network efficiency. For this reason, most network administrators try to remain on channels one, six, and eleven whenever possible. Broadcasting on the channels in between will only cause interference for your device and any others within the same 22 MHz range [67].

Regardless of overlapping frequencies, wireless devices can only listen to one channel at a time. While the wireless network interface card is limited in this way, the software can allow a workaround

to monitor all channels. We call this technique Channel Hopping and will be discussed in the next section.

3.5 Software Implementations

Working within the Raspberry Pi OS, the edge node runs multiple scripts and uses different programs to perform its localization technique. The first to mention is the cron job scheduler, which allows all of the programs and scripts to execute automatically. Next, shell scripts are executed to set up the proper wireless connection settings and clear out temporary data files. The Tshark program is then run to collect beacon frames from surrounding access points, and then a Python program is run to parse and submit the data to the cloud database.

Since the Raspberry Pi operating system is a Debian-based Linux kernel, it comes with the Cron utility. This is an automation tool that can schedule tasks based on a time, date, or interval. It works by referencing a file-based configuration table called a *cron tab* that the user creates. Figure 3.5 shows the list of instructions present on our device's *cron tab*.

```
1  @reboot sleep 5; /home/pi/wap1-mon.sh &
2  @reboot sleep 5; sudo /usr/local/bin/chanhop.sh -i wap1 -d .50
3  @reboot sleep 1; /home/pi/counterReset.sh
4  @reboot sleep 1; /home/pi/resetBootTest.sh
5  * * * * * sleep 20; /home/pi/tsharkworking_2018.sh
6  * * * * * sleep 10; python2.7 /home/pi/Test5.py
7
```

Figure 3.5: Edge node cron tab instructions

The @reboot command denotes that this specific command will immediately occur once the device has booted up its operating system. The sleep command allows for more fine-tuning, waiting a certain number of seconds after the device's booting up is finished to begin the subsequent command. In figure 3.5, we see that the device sleeps for five seconds once up and running, then executes the wap1-mon.sh script. Then, the device sleeps five more seconds before running the chanhop.sh script. One more second is given between the counterReset.sh and resetBootTest.sh script to ensure every task is completed methodically and uninterrupted. The final two commands with five asterisks before them in lines five and six denote that they are to be completed every second. The asterisks from left to right stand for the minute, hour, day of month, month, and day of week that the command should execute [68]. By leaving them all as asterisks, these commands will run every minute, only staggered by the sleep times of twenty and ten seconds.

The first script run by the *cron tab* on startup is the wap1-mon.sh file. This script changes the configuration of the WLAN1 wireless card from managed, to monitor mode. To do so, a user must

first take the card down from its operational status, change the mode, and bring it back online. This task is illustrated in figure 3.6 and can be automated by a script such as the one shown in figure 3.7. By doing so, the node can still connect to the internet regularly with the on-board wireless card, denoted as wlan0, while the TP-Link Nano Wireless N150 Adapter that is identified as wlan1 is set to monitor mode to just listen and be able to capture all data being transmitted around the device.

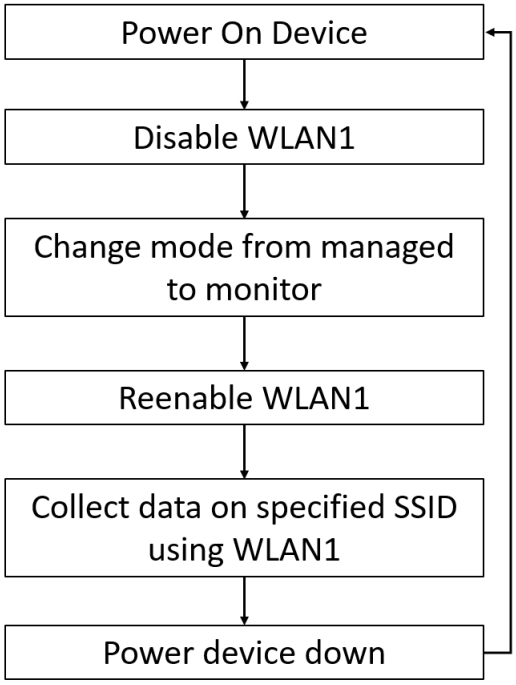

Figure 3.6: Flow chart for changing a wireless card from managed to monitor mode.

```
#!/bin/bash

sudo ifconfig wlan1 down
sudo iwconfig wlan1 mode monitor
#echo "Interface now in monitor mode"
sudo ifconfig wlan1 up
```

Figure 3.7: Shell script for enabling monitor mode

To reduce interference and congestion of a wireless network at any given frequency, access points are set to different frequency ranges or bands [69]. However, if a wireless adapter is set to operate at a default channel, monitoring for beacon frames may only yield a fraction of the total packets being transmitted in the area. To fix this, the second command on the *cron tab* initiates the WiFi channel hopping script that can be found in Appendix A. Yoshio Hanawa provides this code on GitHub to allow a wireless adapter to continuously cycle through different possible channels that access points

may be broadcasting on in an area [70]. In doing so, the WLAN1 adapter on our node that has been set to monitor mode can now observe and record more beacon frames in the vicinity to further improve localization calculations.

The *counterReset.sh* and *resetBootTest.sh* commands are simple shell scripts that reset the global counters back to null values every time the node is restarted. These counters increment with every scan the node conducts approximately once every minute. These counter values help keep the data submitted to the cloud database organized for querying. Counter values can be used as a value to sort or search by in the database, not having to use dates or timestamps.

The first recurring command in the *cron tab* is the call to use the *Tshark* network analysis program. It lets a user capture and filter data packets from the command line, with various options to include for more precise collections. In figure 3.8, we see the type of commands used to capture useful network data for localization.

```
sudo tshark -i wlan1 -a duration:15 -Y '(wlan.fc.type_subtype==0x08)
and (wlan_mgt.ssid == SSID-Name-Here)' -T fields -e frame.time
-e wlan.bssid -e radiotap.dbm_antsignal > boottest4.txt
```

Figure 3.8: Terminal command for Tshark network scan

With the -i flag we set the capture interface to WAP1, which is the network adapter set to monitor mode. The -a allows for an autostop, which we have set to fifteen seconds [71]. We can filter the packets better with -Y, only recording packets with 0x08 packet types, which are beacon frames, as well as specify for only those beacon frames of a certain SSID [72]. In this example, we were testing our system on campus, thus filtering for the specified SSID's beacon frames. We can further reduce the amount of unneeded information by using the -T flag to select specific data fields from the packets using the -e flag for each piece of data we want [71]. The -e flags used here are retrieving the time stamp, SSID, and signal strength of each packet. Finally, these specified pieces of data are stored in the text file boottest4.txt. All flags are explained below in table 3.1.

Table 3.1: Useful Tshark Flags

Wirshark Flag	Functionality
- i	Sets the interface for Tshark to capture data on.
-a	Enables autostop functionality, allowing for timed scans.
-Y	Filters for only packets of a specified type.
-T	Sets the format of the packet output for specified fields.
-e	Adds a field to the output, used in conjunction with -T.
-u	Specifies duration type. s for seconds and hms for hours, minuts, seconds.

The collection of network data sitting in the boottest4 text file is then handed off to the final piece

of code in the node's cron tab, testTrial5.py. In this Python script, each collected beacon frame is iterated through one at a time, having its timestamp, MAC ID, and signal strength values identified and pulled from the plain text file. The signal values are converted into distance measurements in units of feet, and then if they are within an acceptable and expected range, they are submitted to the MySQL database from a connection established using the WLAN0 network adapter that has remained in managed mode. This process's code is provided in Appendix B. The boottest4 file is replenished within the next minute or two with new beacon frames due to the cron tab calling the Tshark script again, and the data collection and submission continues repeatedly while the node is powered on.

Figure 3.9 visualizes the cron tab instructions in flow diagram form. Every time on startup, the wlan1 is set to monitor mode, channel hopping commences, and all temporary local files are zeroed out. The device then waits twenty seconds to ensure these changes have taken place and that the channel hopping has commenced. Tshark is then run to collect packets on all 802.11 channels, looking for beacon frames on the specified SSID. In our case, this would be the SSID of the hospital's main WiFi network. The length of time spent collecting packets can also be easily altered for either shorter, more rapid sampling, or longer and more accurate location measurements. Finally, our Python script parses through the output file created by Tshark and submits all data to our database on the cloud. The node then loops back to begin waiting again before running its next Tshark scan. The scanning and submission of data will continue indefinitely to provide localization of the node.

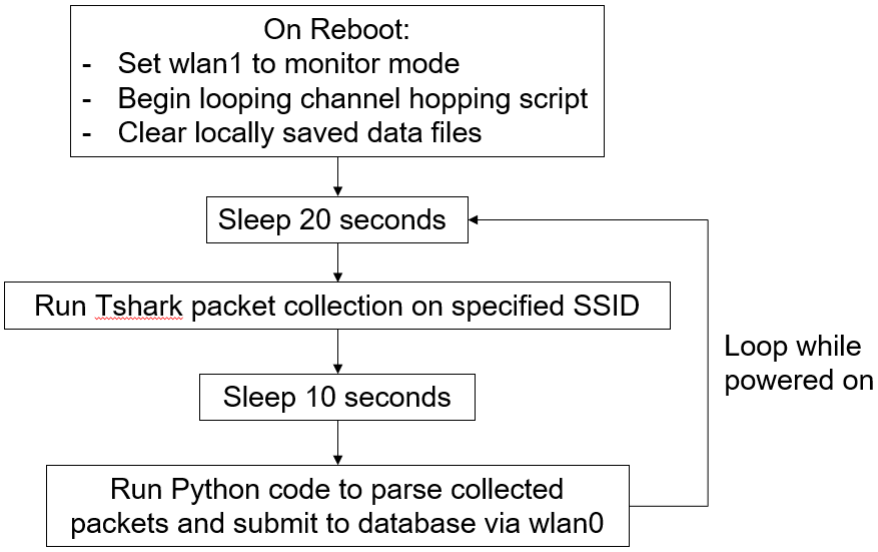

Figure 3.9: Diagram of edge node's *cron tab* flow of commands

3.6 Cloud Server and Storage

The back end of the localization system must be flexible and highly available. Using local servers for this task would increase the number of systems to maintain and security services required of the administrators at the server's installation place. We can avoid this by using cloud storage and computing. In our system, we use Amazon Web Services (AWS).

AWS has data centers worldwide that ensures practically constant up time for their services. Additionally, their customers can scale their servers up or down with demand quickly, rather than needing to upgrade or downgrade hardware at a much higher expense. Their business model allows for pay-as-you-go pricing, cutting down costs to the bare minimum needed for quality server hosting. Most importantly, the AWS platform is extremely secure. Along with other certifications and audits, the AWS platform is HIPAA (Health Insurance Portability and Accountability Act) certified for the security of stored information [73].

The primary service our system uses is Amazons Relational Database Service. It is from here that we can host a MySQL database to store and query our data. Table 3.2 gives an example of the raw data submitted from the Raspberry Pi node, formatted in .CSV style, from which we can calculate positioning. You can see that the primary information submitted is the date and time, MAC address of the discovered access points, and the distance measured between the node and access points.

Table 3.2: Example of data stored in one .CSV file

ID	Date	MAC	distance	lastUpdated
1.1	7/2/18	34:fc:b9:79:81:60	63.614	7/2/18 17:25
1.1	7/2/18	34:fc:b9:78:9c:c0	14.856	7/2/18 17:25
1.1	7/2/18	34:fc:b9:79:81:60	68.968	7/2/18 17:25
1.1	7/2/18	34:fc:b9:78:9c:c0	13.703	7/2/18 17:25
1.1	7/2/18	34:fc:b9:79:81:60	63.614	7/2/18 17:25
1.1	7/2/18	34:fc:b9:78:9c:c0	13.703	7/2/18 17:25
1.1	7/2/18	34:fc:b9:79:81:60	63.614	7/2/18 17:25
1.1	7/2/18	34:fc:b9:78:9c:c0	16.107	7/2/18 17:25
1.1	7/2/18	34:fc:b9:79:81:60	68.968	7/2/18 17:25
1.1	7/2/18	34:fc:b9:78:9c:c0	16.106	7/2/18 17:25
1.1	7/2/18	34:fc:b9:79:81:60	63.615	7/2/18 17:25
1.1	7/2/18	34:fc:b9:78:9c:c0	10.754	7/2/18 17:25
1.1	7/2/18	34:fc:b9:79:81:60	63.615	7/2/18 17:25

Figure 3.10 shows how to query the necessary information from the raw data located in the MySQL database to calculate distance from one specific access point. One can easily pull the MAC ID and distance values within a range of dates and times. An average of the packets' distance

values is taken from all collected in the specified time window to improve accuracy. This example used a time window of two minutes to allow for a larger sample size of packets for average distance calculation.

```
SELECT MAC, AVG(distance)
FROM data
WHERE lastUpdated > "2018-7-10 18:54:00"
and lastUpdated < "2018-7-10 18:56:00"
GROUP BY MAC;
```

Figure 3.10: Example query for calculating average distance of node from specific access point

When this process is conducted for three or more MAC IDs, trilateration can be used to calculate the node's position in relation to the corresponding access points based on the averaged distance measurements. Initial trials with a functional node were conducted in the ISAT/CS building on the James Madison University campus. The trials were conducted on the second floor of the building, with four different stops made along the way during each trial, enumerated in red in figure 3.11. The green circles are the nearby access points installed in the ceiling.

Figure 3.11: Predetermined locations in JMU's ISAT/CS building used to test node

Multiple trials were conducted that had the node stay at each designated spot for one, two, and four-minute intervals to collect different amounts of beacon frames. Each time interval was repeated to help reduce anomalous behavior in data collection. Table 3.3 shows the node's average distances in feet from the nearby access points, labeled by the last four values of the MAC ID's [16]. Blank cells indicate that no packets were collected for that MAC ID during that specific trial. The most

notable trend in the table is that the longer the node stayed in place, namely the four minute trials, the more access points were recorded. This is likely due to the channel hopping script's time required to cycle through all frequencies to record all possible access points in range.

Table 3.3: Distances Observed in Feet from Access Points via Monitoring With Channel Hopping Enabled.

MACs	**1 Min Intervals**		**2 Min Intervals**		**4 Min Intervals**	
	Trial 1	**Trial 2**	**Trial 1**	**Trial 2**	**Trial 1**	**Trial 2**
Spot 1						
9a:60			35.96	37.5	45.2	49.92
9c:c0	22.73	12.55	25.46	10.3	21.8	15.37
81:60	46.73	49.92		49.1	48.3	46.5
42:60				31.8	47.7	32.25
Spot 2						
42:60			15.11	19.6	16	19.39
44:a0			27.51	24.33	22.5	32.34
49:00	26.78	31.87	24.07	20.5	23.3	30.28
4b:e0			16.64	16.8	17.2	13.8
81:60	49.38		39.85	49.9	34.8	46.96
Spot 3						
aa:e0			49.4	41.7	35.9	45.3
43:60	24.7	31.72	30.6	31.9	26.2	16.88
44:60	42.11	40.96	40.8	20.5	25.8	31.49
49:00	31.19	24.21	10.1	23.7		20.46
5a:60			41.9	46	40.1	44.66
5e:60			13.8 2	18	18.9	14.75
Spot 4						
43:60	23.2	16.73	10.5	11.6	11.9	7.53
44:60	49.7	48.36	44.6	34.7	41.8	40.73
49:00			42.9		49.9	49.54
5e:60			45.9	47.7	49.9	37.72

The higher consistency is obtained by the four minute collection intervals made it the best candidate for further mathematical analysis. We started with trimmed mean calculation to investigate whether or not erroneous packets recorded with extreme outlying RSSI values were having a profound effect on our calculations. Figure 3.12 illustrates the trimmed mean method. The blue lines indicate subtracting the top and bottom 10% from the overall data set, removing major outliers to try and bringing the mean closer to most packets' values. The red lines denote a 20% trim on both sides of the curve, leaving only 60% of the data representing the middle pack of the most expected values.

Table 3.4 shows the calculated averages for a ten and twenty percent trim from the top and bottom of the data sets. Blank cells in the table denote that not enough data was present for those MAC addresses during the specified trials for the trimming function to successfully calculate the new mean. We expected to see the averages come closer to the expected and approximated distances

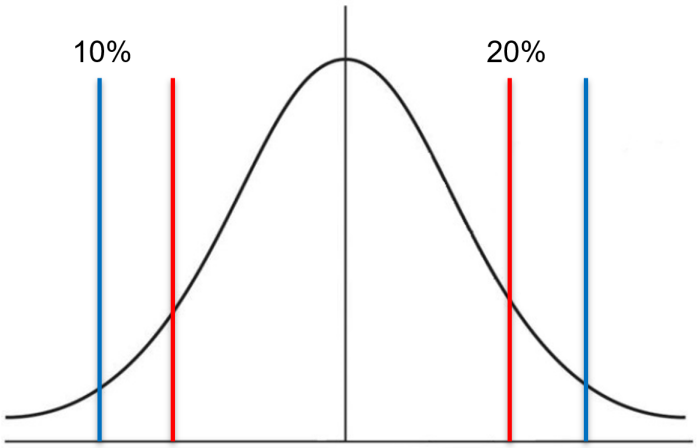

Figure 3.12: Trimmed mean analysis illustrated on a normal curve of data.

from the spots to the access points, but instead the means wavered back and forth with no real trend as the trimming increased. This leads us to believe that packets with extremely high or low values do not significantly impact our calculated distances, and the implementation of a trimmed mean function in our code may not be necessary. Since our final goal for these nodes is to track what room a person is currently in, deviation within approximately one-foot difference is negligible [16].

Another form of analysis we performed was examining the percent error for the different trials. To do so, the experimental distance values were subtracted from the expected distance values, and then divided by the expected values. The resulting numbers were then multiplied by 100 to translate them into percentages. Table 3.5 shows the measured expected values for each access point's distance in feet when the node is located at the designated spots on the map [16].

The calculated percent errors are better visualized in graphical form to show the trends of accuracy and precision. Figure 3.13 shows the percent errors of both trials while the node was at spot 1 on the map. We see a consistent trend of positive error in both trials, ranging from about twenty to fifty percent. The primary explanation for this is structural impediments in the surrounding floor plan such as walls or doors, encumbering packet transmission from access point to edge node.

We see a slightly different trend at spot 2, shown in figure 3.14. This spot was more central in the building, in the middle of a long hallway. Because of this, the error range is more tightly grouped around the expected value, or zero in terms of percent error. The error values fluctuate between negative and positive which shows the fluid nature of signal strength that is expected with RSSI-based localization [16].

Table 3.4: Trimmed Mean Calculations for Distances in Feet from 4 Minute Monitoring Trials.

MACs	**4 Min Avg**		**10% Trim Mean**		**20% Trim Mean**	
	Trial 1	**Trial 2**	**Trial 1**	**Trial 2**	**Trial 1**	**Trial 2**
Spot 1						
9a:60	45.2	49.92	45.15	49.92	44.78	49.92
9c:c0	21.8	15.37	21.92	15.61	21.91	15.91
81:60	48.3	46.5		46.58	49.15	46.72
42:60	47.7	32.25	48.45	32.04	48.95	32.04
Spot 2						
42:60	16	19.39	15.99	19.87	16.11	19.87
44:a0	22.5	32.34	22.16	32.51	21.53	32.51
49:00	23.3	30.28	23.22	30.09	23.15	30.09
4b:e0	17.2	13.8	17.26	13.92	17.41	13.92
81:60	34.8	46.96	35.15	47.06	35.29	47.07
Spot 3						
aa:e0	35.9	45.3	36.13	45.35	36.13	45.7
43:60	26.2	16.88	26.17	16.88	26.16	16.94
44:60	25.8	31.49		31.63	26.16	31.69
49:00		20.46		19.81		18.95
5a:60	40.1	44.66	40.27	44.67	40.09	44.85
5e:60	18.9	14.75	18.86	14.93	18.82	14.94
Spot 4						
43:60	11.9	7.53	11.23	7.51	11.27	7.52
44:60	41.8	40.73	42.06	40.78	42.36	40.77
49:00	49.9	49.54		49.92		49.92
5e:60	49.9	37.72		37.55		37.12

Table 3.5: Distances from spots to access points in feet.

	MACs in Range	Distance
Spot 1		
	9c:c0	16
	81:60	50
	42:60	56
Spot 2		
	42:60	25
	44:a0	30
	49:00	25
	4b:e0	15
	81:60	35
Spot 3		
	43:60	35
	44:60	25
	49:00	20
Spot 4		
	43:60	18
	44:60	30
	49:00	35

The fluctuation in error values at spot 3 are even more exaggerated, depicted in figure 3.15. The nearby access points are behind the walls and around the corners from the spot at which beacon

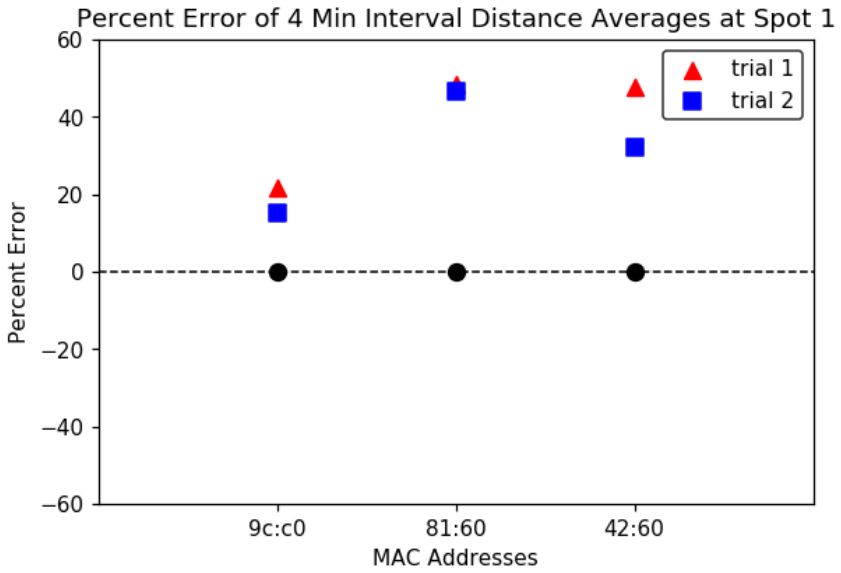

Figure 3.13: Percent error calculated for all access point MAC addresses in range at spot 1 for 4 minute interval scan trials.

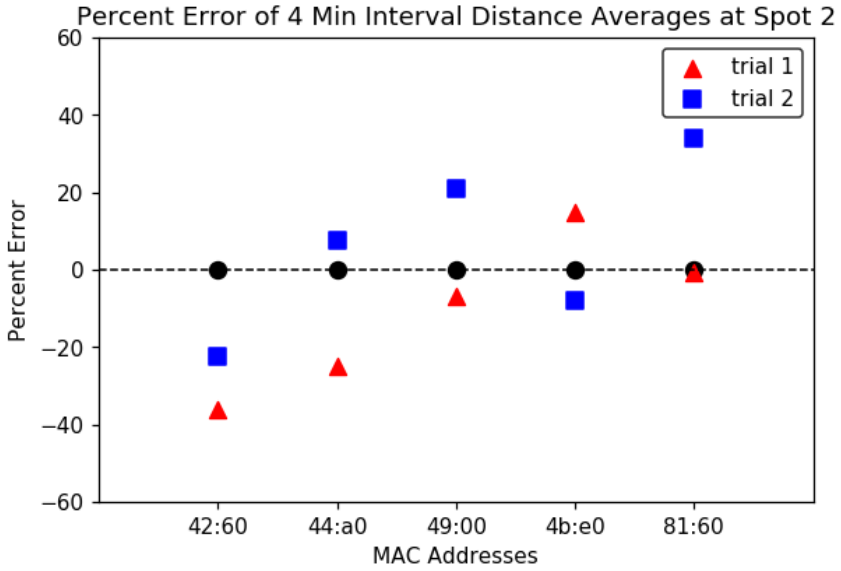

Figure 3.14: Percent error calculated for all access point MAC addresses in range at spot 2 for 4 minute interval scan trials.

frames were recorded. It is also worth pointing out that the MAC ID ending in ”49:00” was not picked up by the node during trial 1, explaining the absence of the red triangle in the graph [16].

Spot 4 showed some of the highest consistent and precise data collecting of all. Figure 3.16 shows how both trials have practically the same amount of error compared to the expected distance values. This shows that the node is capable of consistency and reliability. It is just a matter of

improving how calculations of distance are performed, converting dBm to feet and accounting for physical barriers more elegantly [16].

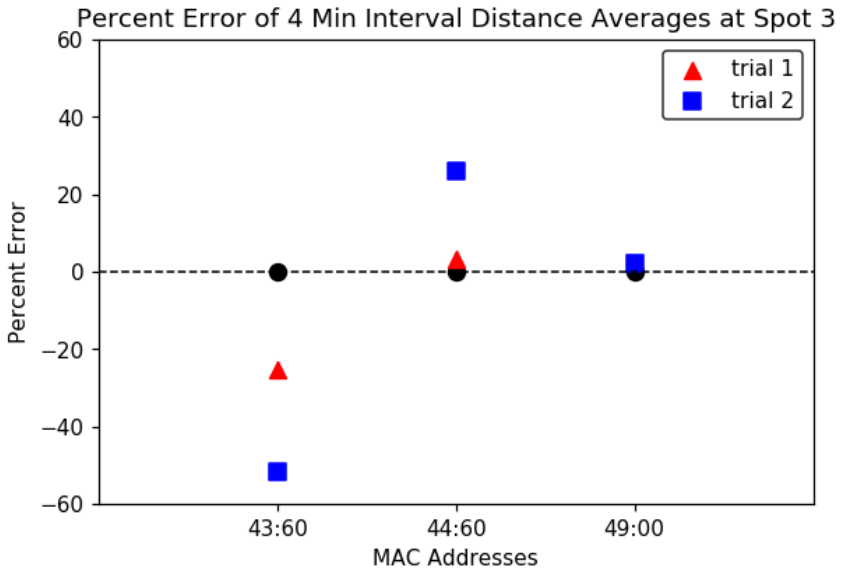

Figure 3.15: Percent error calculated for all access point MAC addresses in range at spot 3 for 4 minute interval scan trials.

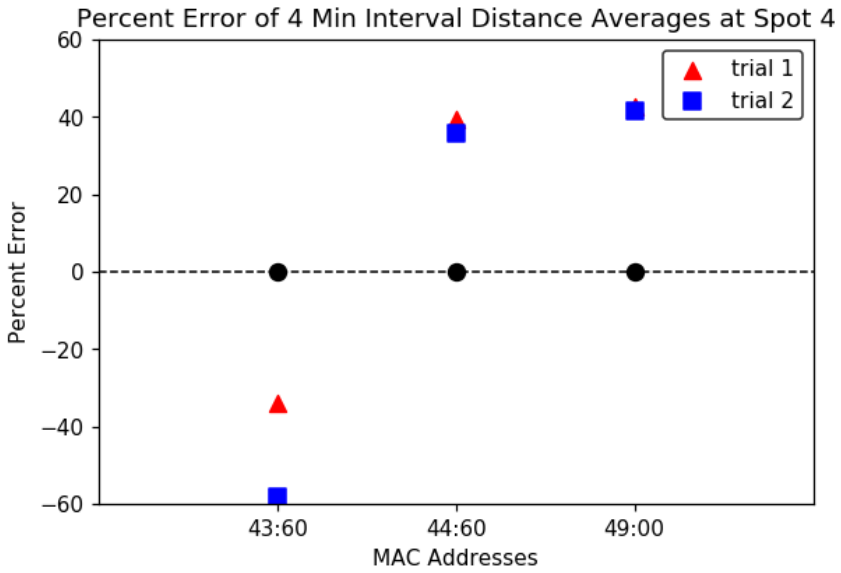

Figure 3.16: Percent error calculated for all access point MAC addresses in range at spot 4 for 4 minute interval scan trials.

3.7 Viability in Healthcare

Our proposed localization system is designed in a manner very complementary to a hospital setting. A well-planned and supplied hospital will have ample access points and other IoT devices such as printers or medical machinery that will be projecting beacon frames to help shape a signal fingerprint in each hospital area. Offline data collection of all permanent devices during system setup can provide the necessary functional room-level localization measurements.

One might think that smartphones could provide more accurate data than our system without the setup and training of an entire system. However, there will always be the issue of not all staff or patients having smartphones and their refusal to install the application giving a third party permission to track their device. It is also often reported in many large hospitals that cell service is unreliable and spotty at best. This is due to the thick concrete walls of the building, with some imaging areas specifically built to block high intensity X-rays from traveling beyond one room [74]. Necessary construction specifications like this make many outdoor or hybrid localization technologies unusable. GPS systems will run into the same problems as cell service, and neither are well equipped to handle tracking on multiple floors. For these many reasons, our cheaper node-based tracking design that the hospital can provide for all staff and patients remains the best candidate for implementation.

Acknowledging tools from the past that are still in use, the communication gadget that has stood the test of time for doctors is the common pager. It only requires a simple radio frequency signal to receive messages and is much more reliable in the hospital setting. It is constantly listening to the transmitter and awaits a signal. While effective, it is considered somewhat crude or low-tech by modern day standards. The device only allows simple messages to be sent and in one direction. Management's general desire is to move away from the pager and towards a two-way communication device with the same level of reliability [74]. We believe our edge node could be one step closer to the realization of that goal with its two-way communication potential.

3.7.1 Staff and Patient Benefits

For now, we will concentrate on the edge node's one-way communication, but in the opposite direction of the pager. We wish to send the positioning data of the user to the database for analysis. One major advantage this will provide is tracking staff for the betterment of staff's provision of care. Historical data collected will help paint a picture of a doctor's time spent with patients in consultation rooms, a nurse's up-time while tending to patients on the floor, or even custodial staff's routes

taken when cleaning the rooms and halls of the hospital. The administration will have the ability to go over days' records and see which staffing methods worked and which didn't. Chief nursing officer Patty Toor of Florida Hospital Celebration Health cites helpful outcomes from tracking nursing staff. Their system was comprised of tracking sensors that counted the number of times nurses entered and left patient rooms, as well as stopped by the nurses' station on their floor [75]. In their hospital, it was found out that storerooms were consistently being insufficiently stocked with medical supplies all around the facility, causing nurses to have to travel unnecessarily far to care for their patients [75]. Identifying and resolving a problem like this can leave staff more time in their shifts to carry out their actual jobs instead of providing short term solutions to unknown chronic problems.

Another example of an issue that tracking systems improve upon is the slow patient turnover rate in hospitals with a limited number of beds. Typically, custodial staff does not know if a room is empty and ready to be cleaned after a patient's discharge until nurses notify them. This could take some time while they are also caring for multiple other patients. With a localization system in place, once a patient's tracking node is turned back in upon discharge, it would notify the system that the patient is no longer occupying the bed or room, and that it may be cleaned and prepared for the next patient. Automating this communication can cut down wasted bed time by over three hours in some hospitals [76].

Assigning tracking nodes to patients improves upon the safety during their stay as well. From newborns to the elderly, having the ability to know patients are safely where they are supposed to be without constant visual surveillance can help reduce the stress of a nurse's job. Li et al. discuss how small Bluetooth tracking monitors can be placed on a newborn's ankle to ensure they stay in the nursery room [77]. Each tracking device broadcasts a unique signal that can identify which baby is being moved. An accompanying smartphone application can then further identify which user, in this case nurse, is transporting the baby. This increases accountability and deters anyone from attempting to steal a child from the neonatal wing.

The benefits of this system can apply in various other situations. Patients who may suffer from dementia or any other brain or memory related illness can become confused or disoriented and wander out of their room. It is rarely the case that patients can receive constant supervision by staff members or a family member or guardian during their hospital stay to prevent this. Unfortunately, patients with dementia are 20% more likely to experience sleep disturbances and wander out of confusion or anxiety at night. This is especially true when in foreign or new environments [78]. A hospital at night is a likely trigger for this activity, with less staff on duty to watch and care for such patients. Having a wearable tracker on their person at all times can help keep them safe by alerting

staff if they were to leave their room or the hall they are on.

3.7.2 Contact Tracing

The COVID-19 pandemic of 2020 has brought forward another important aspect of patient tracking, which is contact tracing. Contact tracing is the act of identifying people who have an infectious disease and everyone who has recently contacted them. This is in an attempt to slow or stop the spread of the disease [2]. Figure 3.17 shows how most contact tracing is done at the moment. A patient exhibiting symptoms is tested and then confirmed a positive case. He is then interviewed and asked where he's been, what precautionary measures were taken, and who he has been in contact with recently. The individuals he lists are ideally notified, tested, and interviewed as well. This process is carried out until all subjects interviewed are interviewed and confirmed as not infected.

Developers from different states and health organizations have also rushed to develop contact tracing mobile applications that can help people self-report cases in specific areas for the benefit of their communities. One issue these applications face is the reluctance of people to participate and self-report in fear of personal inconvenience or privacy concerns. Another is that these applications trace location based on the phone's GPS and Bluetooth signals. In big areas with lots of people, or indoors with less accurate reception, these signals cannot confirm close proximity between two individuals. This makes contact tracing much more complicated and a guessing game of who actually came into contact with who.

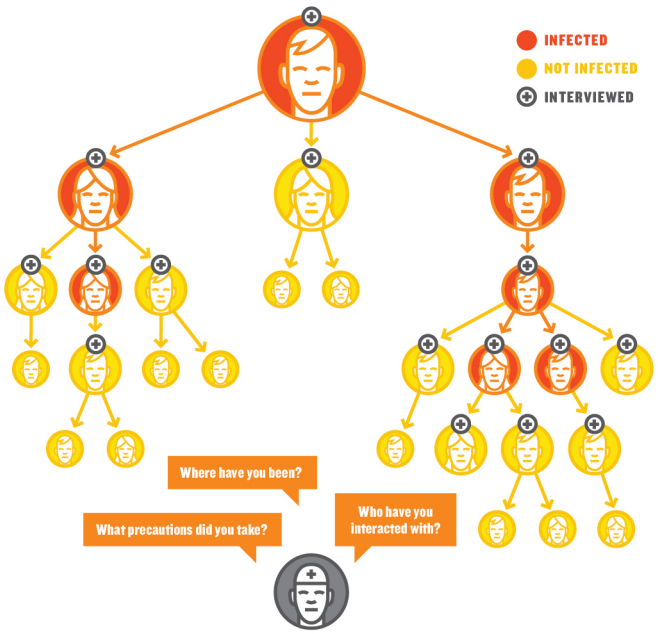

Figure 3.17: Contact tracing through interviews and confirmed cases

Our system would reduce these GPS-based systems' ambiguity with our room-level localization tailored to the specified building. If a patient or staff member of the hospital is tested and confirmed a carrier of COVID-19, historical data can be queried and analyzed for a list of other edge nodes that had been in the same rooms at the same time as that infected person. Due to the virus's high infection rate and ease of transmission, indoor contact is a high risk that must be monitored as closely as possible. Basing our tracing off of logged data instead of inexact or unsure human testimony could significantly improve tracing accuracy.

3.8 Chapter Summary

In this chapter, we reviewed our operational localization system. We discussed the system's architecture, the necessary hardware, as well as the software developed for it to run effectively. Additional improvements to the system like channel hopping for better RSSI coverage and AWS cloud storage for increased data availability and security were included. Finally, the benefits of this technology in a healthcare setting were discussed for both staff and patients alike, as well as this system's potential use for contact tracing in the current times of the COVID-19 pandemic.

Chapter 4

Machine Learning Techniques and Security Protocols

Localization systems deal with data ranges spanning into the millions of distinct records. To draw coherent conclusions and make educated inferences, we need tools to reduce the noise from data collected and calculate accurate results. This chapter includes an overview of such tools, namely machine learning techniques. We also discuss pertinent security protocols to be included in a system with access to so much data about its users, especially in a healthcare setting.

4.1 Machine Learning Overview

Machine learning can be thought of as the optimization of a task through the use of one or more algorithms that base decisions on examples or past experience [79]. It is the driving source of the new applications we interact with regularly such as: recommendation engines, speech and handwriting recognition, automatic captioning, spam filters, and many more. For each of these tasks and many more, a specific type of algorithm must be applied to yield proper results. The two main categories of classic machine learning are supervised learning and unsupervised learning [80].

4.1.1 Supervised Learning

In supervised machine learning, input variables, denoted as "X", are mapped to output variables "Y". They exist within the mapping function f(X) = Y. It is the goal of the algorithm to tune the mapping function f(X) with provided data so that accurate future predictions can be made of output variable Y from hypothetical values of X. It is considered to be a supervised algorithm because it begins with a training set, that knows the correct values, to calibrate its mapping function according to [81]. Supervised learning models can solve two different types of problems. They can either be used for regression algorithms or classification.

In a regression algorithm, a continuous prediction of a value can be made, given the factors provided in the model's design [82]. This can be values like prices or salaries based on various aspects of the economy or people's lives. The goal of the algorithm is to find a line of best fit and map the input value X with the continuous output value Y. Because of the mathematical nature of the regression type calculation, the output value Y will always be a natural or real number value

[82]. Figure 4.1 demonstrates how a linear function can be calculated to fit the given data and, thus, accurately predict future Y values. The data points denoted as red dots are used to calculate the regression function, shown as the black line of best fit.

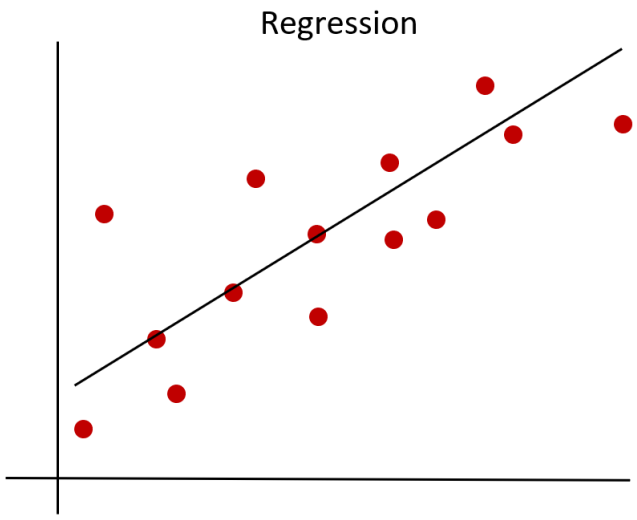

Figure 4.1: Example graph of regression calculation

A classification algorithm works to create a function that divides the given data set into two or more classes based on parameters the user is interested in [82]. A training set of data that has already been properly classified is given to the algorithm. This correct data will create a tuned mapping function and decision boundary with which it can correctly categorize future data input. Instead of a real or natural number value output, a classification algorithm's output is always a discrete value. A common example of this type of algorithm is an application for labeling email as either spam or not spam based on previous examples of both types of emails [82]. In figure 4.2, the example of spam email classification is visualized. Given specific qualities of an email, the algorithm can either sort an incoming email as spam, denoted as a red dot, or as a genuine email, shown in blue. The decision boundary is shown as the black line, calculated from the training data.

4.1.2 Unsupervised Learning

Unsupervised learning uses unlabeled raw data to model data in a way that helps find patterns or groupings in the provided data that were otherwise unknown. In this type of algorithm, the designer has no right answer or direction at the beginning. Metrics of similarity may be tuned, but the data itself presents the answers that will dictate further actions taken by researchers [83].

The main category of unsupervised learning algorithms is those that perform cluster analysis.

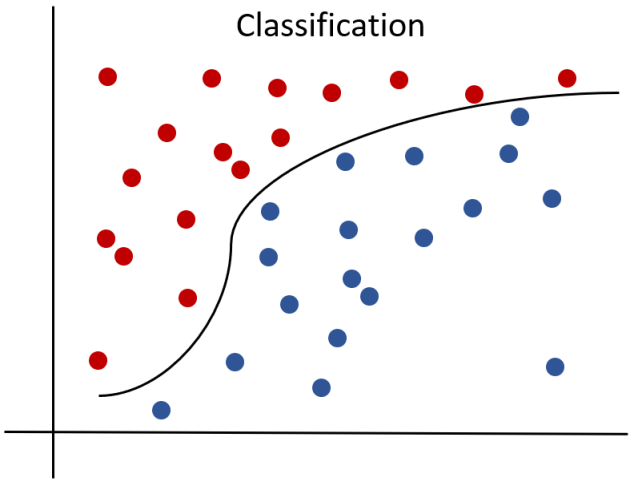

Figure 4.2: Example graph of classification calculation

A common algorithm to use in calculating clusters is the k-means algorithm. In this algorithm, a number of points, called centroids, are specified by the designer as the number of clusters wished to be calculated. The algorithm then finds the best placement for centroids by first assigning the data points to clusters based on the current random position of the centroids, and then reassigning centroid locations based on the data points' location within their clusters. This process is repeated until the mean error calculated cannot be reduced any further [84].

Figure 4.3 [84] shows the steps involved. In sub-figure a, the data is plotted. Next, two centroids are introduced at random points in sub-figure b, denoting that k is equal to 2 in this calculation. Sub-figure c shows how the data is then assigned to a cluster based on the centroids. Next, the centroids are then recalculated based on the data clusters to try and reduce the mean error as shown in sub-figure d. The data points are reevaluated, and some reassign to different centroids if their error values are too high, as demonstrated in sub-figure e. Finally, sub-figure f shows how the centroids find their lowest calculated mean errors when they reach the middle of their data clusters.

These clusters can be formed for many different reasons and by various methods. Centroid clustering has been demonstrated, but other paradigms exist. Density-based clustering can shape areas of high-density data points that share similar characteristics. This differs from clusters by not assigning outliers to any clusters. The boundaries of density-based clustering are also more finely tuned and precise [85].

Distribution-based clustering uses density functions such as Gaussian distribution to calculate how close to a centroid a data point is. The further away from a centroid a data point lays, the lower the probability it belongs to that centroid cluster. This type of clustering algorithm is only

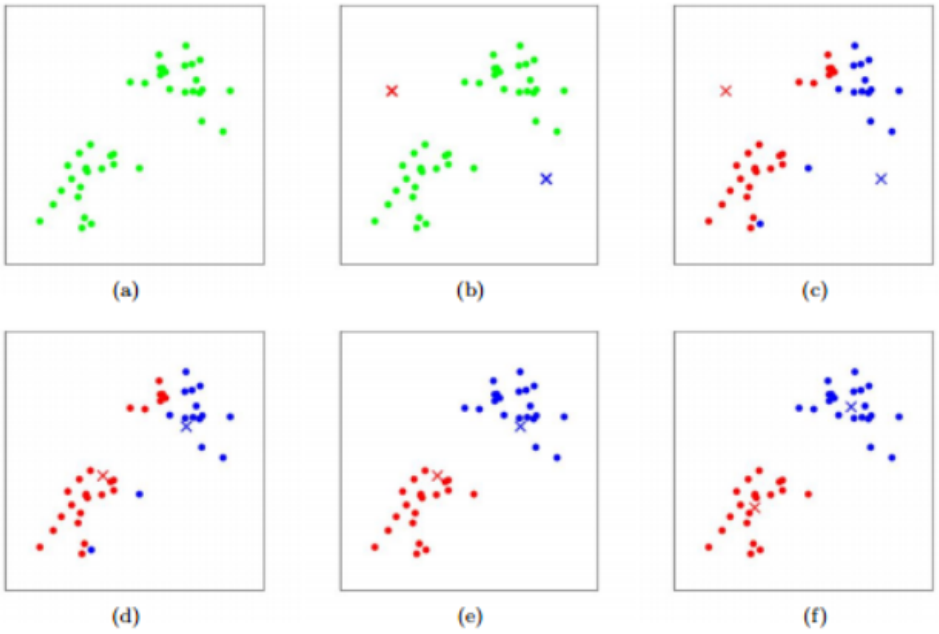

Figure 4.3: K-means sorting algorithm steps

useful when you already know the type of distribution your collected data falls under [85].

The final clustering technique mentioned here is hierarchical clustering. In this method, a tree of clusters is created, which lends itself well to classification problems such as taxonomies [85]. A common example is the classification of a set of animals into smaller clusters such as vertebrates and invertebrates. From there, those two clusters can be broken down further into mammals, reptiles, insects, etc.

Examples of applied unsupervised learning can be found in all types of data-driven sciences. The biomedical field conducts intensive sequence analysis and genetic clustering when studying genomics, leaning on the analysis of large data sets to help steer medical research. Data and pattern mining is another application that helps improve technologies like computer vision and object recognition in boundary-pushing technology [83].

4.1.3 Supervised Machine Learning Algorithms

In our work, we were confident that we would need a supervised learning algorithm. The localization system being built would be tailored to the facility it would be setup for, and the data sampling would coincide with the specific building as well. Here we investigate multiple popular algorithms among researchers creating systems in need of efficient supervised classification algorithms.

The k-Nearest Neighbors (kNN) algorithm is a popular tool for scientists wanting to classify

data. Since it is supervised, the algorithm first needs training data in order to predict the classes of future data. Once data is loaded into the algorithm, the data scientist must also assign a value to k. In this method, k is the number of closest data points the algorithm will consider when evaluating into which class to assign the new datum. The algorithm begins by measuring the distance from the data point in question to all other data points in the data set[86]. A popular way in which to measure the distance from point to point is the Euclidean distance formula, while other formulas like cosine, Chi square, and Minkowsky methods are also viable in different studies [87]. After the measurement, the algorithm then sorts the data by distance in ascending order. The top k values are chosen from the array. The classes of these values are assessed and the most frequent class in this sample is assigned to the new data point[86, 88].

As demonstrated in figure 4.4 [86], the k value chosen by the researcher can impact the algorithm's decision making significantly. If k is set to three in this plot, the algorithm will classify the new green data point as a red triangle with a majority vote of two to one. If the k value is increased up to five, the blue squares now win the majority of three to two. A low k value will be strongly influenced by outliers, while a high value can possibly overfit the data and classify incorrectly [86]. For these reasons, the researcher must be aware of the data's trends and fine-tune the k value over many tests and trials.

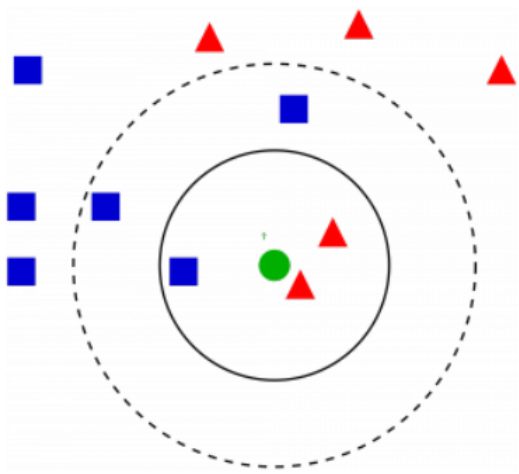

Figure 4.4: K-nearest neighbor classification algorithm

The random forest algorithm is a classification algorithm that reaches conclusions based on the many decision trees it creates when assessing new data. A decision tree is comprised of nodes, branches, and leaf nodes. A node is a single test for a certain attribute or feature that the data will

undergo. A branch is an outcome from the previous node's test that connects to the next node or leaf in the tree. Leaf nodes are the end nodes of a tree, predicting the outcome by stating the class predicted based on the prior tests and their outcomes [89].

A simple and symmetrical decision tree is shown below in figure 4.5 [90]. The tree begins with the first node, determining whether the person in question is over or under the age of thirty. The yes and no answers are the branches. These lead to two more decision nodes, asking yes or no questions about salary and number of children. From these nodes, two more branches are created for each, which lead to the leaves or leaf nodes, which are the decisions made to either yes, get the loan, or no, don't.

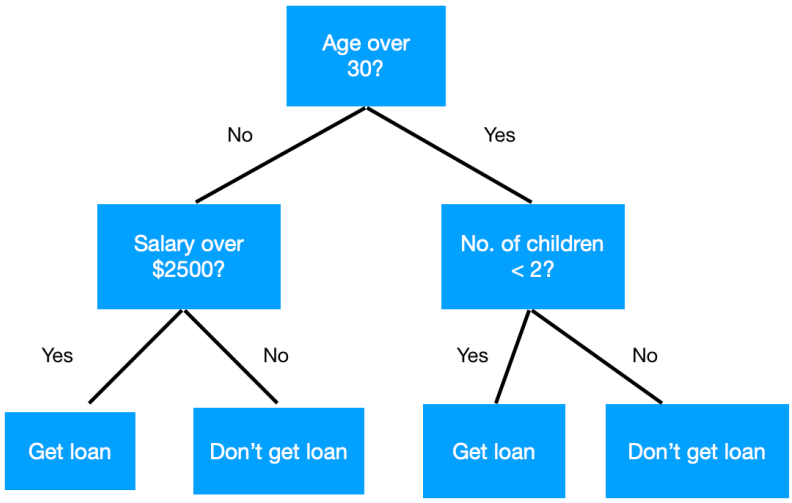

Figure 4.5: Decision tree example

A random forest is comprised of many decision trees like figure 4.5. Each tree is created randomly with a random selection of nodes or attributes that will be assessed. These are selected with replacement, such that the same attributes may be selected again and exist on different trees as well [91]. An ideal tree will be made with enough features that it can make a well-informed decision. The trade-off is having so many features that it correlates with many other trees in the forest, such that the decisions are always similar. There is a middle ground between choosing too many nodes that increases the correlation and strength of each tree, and choosing too little nodes for each tree, which reduces both correlation and strength. This is the primary adjustable and sensitive feature of the random forest algorithm that must be dialed in by the designer [92].

An example of a random forest is shown in figure 4.6. Two trees are randomly generated from the pool of features or attributes available about our data with replacement, meaning attributes can be reused. At each grey node, a decision is made, either to definitively classify the test data as a certain class in a leaf node, or to keep moving on to the next feature node. Ultimately, all branches will end in leaf nodes with classification decisions based on the path of feature nodes that was taken. In this example, both trees in the forest had a majority of leaves vote green as the final class assignment. Therefore, the new data is assigned to the class green and added to the data set.

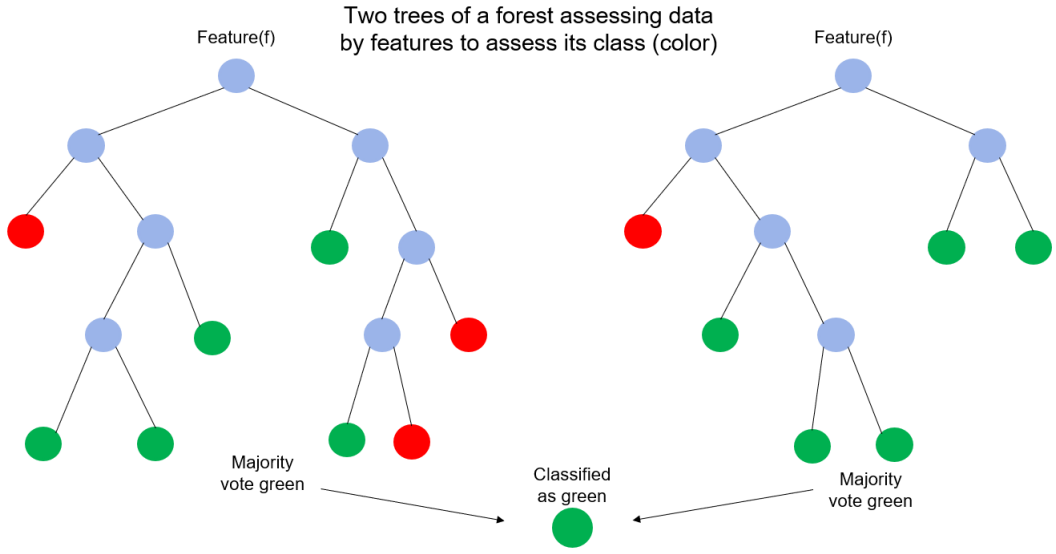

Figure 4.6: Example of two decision trees within a larger forest of many trees deciding on the class of a data point (circle).

Due to the nature in which the decision trees in a forest are created, the random forest algorithm avoids many common pitfalls encountered by classification algorithms. The randomization of trees prevents overfitting of training data because the correlation will remain low and close to that of test data [91]. Each tree also only has access to a small subset of the data, increasing diversity as well [93]. The algorithm also remains insensitive to outliers because of the majority vote, preventing it from impacting many leaf node decisions. Finally, the model can also be further trained to increase the weight or importance of specific features so that they are given more consideration when drawing conclusions [91].

Artificial neural networks (ANNs) are another type of classification algorithm, modeled after the structure of and processes performed by organic neurons in brains. Each neuron in the network is called a threshold unit and functions by receiving input from multiple external sources. It weighs the different data values and totals their collective value. If the value is greater than the set threshold,

it sends a signal onward. If it is not, then the signal stops there. These signals are sent through one or more hidden layers within the designed network and arrive at the output layer where a decision is made [94]. Threshold units are trained by iterating over data with known output values. This assists the neurons in assigning weight values for different characteristics while calculating the overall threshold values for its unit. The most basic neural network design is the feed-forward neural network shown in figure 4.7 [95]. We see that N amount of inputs are provided to all units in the hidden layer. At each neuron in the hidden layer, a value is assigned for all inputs based on the predetermined weight of each input feature. The total of all values at the specific unit is then compared against the predetermined threshold value. Finally, all decisions made by the hidden layer are sent forward to the output unit for assessment and formation of a final decision Y.

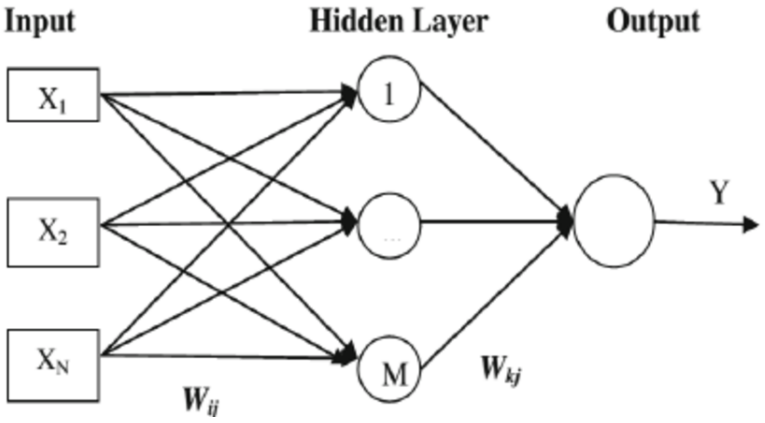

Figure 4.7: Example of an artificial neural network with one hidden layer

Neural networks can also learn more dynamically than just by the forward computation of more and more data. The concept of back propagation has improved ANNs by allowing the network to send back undesirable results to previous layers for readjustment and improvement of classifier values. In this model, the error rate from the last iteration through the hidden layers is sent back and factored into the next calculations and weight values. This in turn lowers the error rate of the next iteration, rapidly reducing the system's error and increasing reliability [96].

ANNs have many advantages, such as their decreased need for statistical training and the ability to be trained with multiple different algorithms. However, they aren't always the best tool for every job. They tend to be a "black box" in their analysis of data relationships [97]. They also are much more demanding computationally, especially when backpropagation is implemented. Finally, they remain prone to overfitting the training data provided to them, and must be well trained before being able to perform classification on live data effectively [97].

In investigating these classification algorithms and more, we have decided to proceed with the random forest algorithm in our implementation of a self-learning localization system with room-level accuracy. We believe our training data for signal strengths sampled room by room will lend itself well to the random forest model where random features are selected for each tree. The variability in signal strengths collected via 802.11 WiFi makes it important not to overfit training data, which an ANN design may be prone to do. Our concern with a kNN model is the impact that outliers in our data set may have when a tracking node might be blocked by structures or other unforseen issues. Random forest is more forgiving in these regards, which reassures us in proceeding with it as our supervised classification model.

4.2 Security and Privacy Protocols

In implementing our system, the responsibility for patient's personal health information would fall under the hospital's jurisdiction. The Health Insurance Portability and Accountability Act of 1996 (HIPAA) is the federal law put in place that requires the protection of sensitive patient data by health care providers, health plans such as insurance agencies, healthcare clearing houses, and business associates that handle private information during tasks such as data analysis and billing [98]. This protection must include the confidentiality, integrity, and availability of health information. The Healthcare facility must also do their best to detect and protect against any possible attacks to their information repositories. They must also perform due diligence to confirm no impermissible uses or disclosures of sensitive information occur. All employees at healthcare facilities are charged with these responsibilities [99].

Our priority as localization system engineers is mainly focused on securing the edge nodes and the access points that the nodes are directly communicating with. There are multiple ways in which a malicious actor could impede tracking accuracy or collect data to the detriment of our systems fidelity. Common techniques used against IoT systems like this include access point spoofing, side-channel analysis, and packet sniffing. We also touch on MAC randomization concepts in current devices and the dangers of large-scale data collection by large companies.

4.2.1 Access Point Spoofing

Access point spoofing is a common technique used by attackers to redirect web traffic and record personal information being transmitted unknowingly by victims. This act is performed by setting up a new access point device within the network's vicinity under attack. The access point's Service Set Identifier (SSID) and MAC address are set to be exactly the same as another access point in the

network. Since these are the two main characteristics that a wireless device is identified by, it can be difficult to find the fake. This device will go unnoticed and victims will begin connecting to it instead of the authentic access points [100].

Figure 4.8 [101] illustrates how an attacker can place a rogue access point within a network. If the attacker properly configures the rogue access point with the SSID and MAC ID of the original access point, a client could easily connect to the rogue network unknowingly. Once connected, the client is at the mercy of the attacker. In this type of man-in-the-middle attack, called Evil Twin, the attacker can now monitor all data packets, inject malware, or install backdoors into the client machine [102].

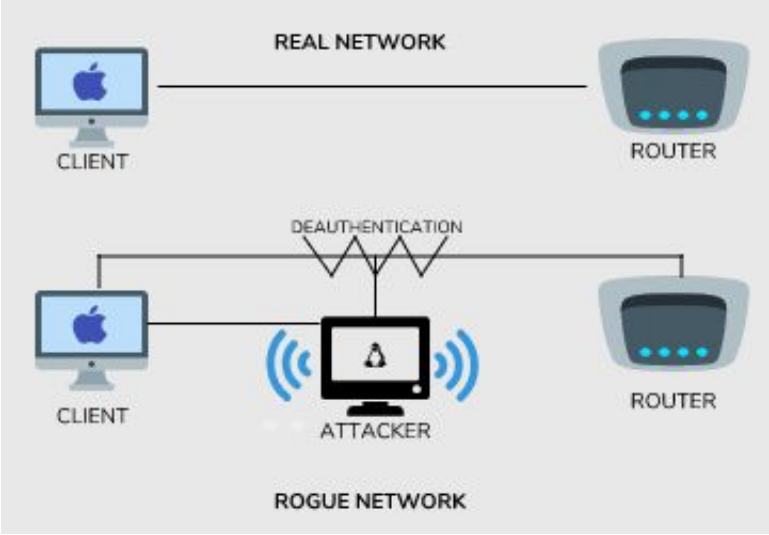

Figure 4.8: Access point spoofing to set up a rogue network.

Ahmad et al. propose a method of detecting illegitimate access points based on partitioning clustering [100]. If a rogue access point is suspected, they began recording beacon frames filtered for the SSID in question. The received signal strengths from the packets are then grouped into two clusters using K-means and K-medoids methods. If the RSS values are close to the centroids and medoids, the access point is legitimate. If the RSS values fall beyond the detection thresholds put in place, it is highly likely a rogue access point has been installed somewhere nearby. The team's detection system claims an accuracy rate of about 90% [100]. This process can continuously cycle through all SSID and MAC combinations in a large scale system to ensure all access points are free from malicious imposters. The researchers have installed this detection framework on the client devices themselves, allowing for fraud detection prior to any connections being established [100]. Placing the detection mechanism in the front-end of the system rather creates a more proactive

approach to data security, rather than a reactive approach if the monitoring was conducted in the back-end.

4.2.2 Side Channel Analysis

Side channel analysis is an attack against a system's cryptography that leaves no traces behind in the process. The attacker studies either the electromagnetic field generated by a computer's monitor to view information or measures the power consumption of a device to gain information leaked by the running processes [103]. This work focuses on the masking and randomization of running processes to prevent successful Simple Power Analysis (SPA) and Differential Power Analysis (DPA) attacks.

Simple power analysis gives attackers insight into a secure device's inner-workings by directly measuring the power consumption taking place during cryptographic operations. Armed with just an oscilloscope, an attacker can listen in on the frequencies being sent. Over time, signals will begin to show patterns and give details to the encryption process. For example, if a device is performing triple DES encryption, the oscilloscope can be tuned in finely enough to reveal the 1's and 0's being input as the encryption key [104]. An RSA decryption key can also be found, as demonstrated in figure 4.9 [105]. The peaks and valleys shown in the signal clearly spell out the binary form of the key.

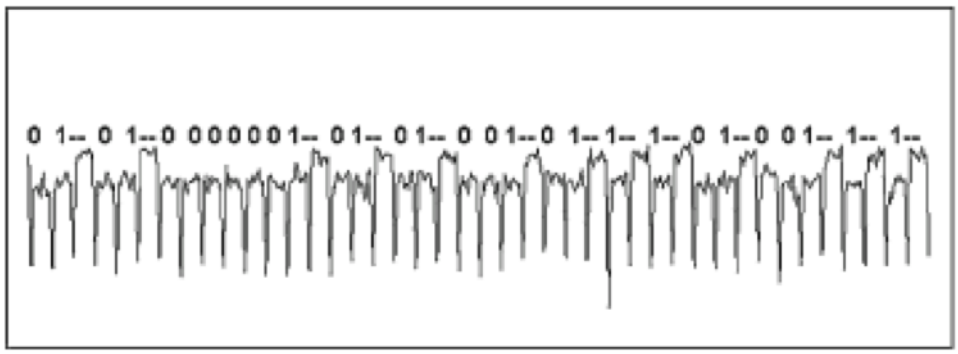

Figure 4.9: Simple power analysis measured by on oscilloscope

There is little that can be done to stop someone from picking up on these signals if their proximity can't be prevented. Instead of physical security, the two basic defenses add noise to the signal and desynchronize the function's instructions. Unfortunately, noise interference isn't usually enough on its own. Many filtering techniques are available to reduce the signal to noise ratios (SNR) and zero in on the desired frequencies [106]. Le et al. have elaborated on how the Guassian properties of noise can help remove it effectively from the non-Gaussian signals [106]. Because this is so well

studied, the more difficult desynchronization of signals has been put forward as another solution. In desynchronization, random delays are inserted into the code to disrupt any important functions' steady execution timing. Dummy code can also be added to change the time dimensions of the channels being monitored. Loops can iterate and add time shifts to throw off calculations. Also, code polymorphism has been proposed. This is the idea of executing different versions of binary code that are functionally identical. Their execution would have differing power consumptions, throwing off listeners [107].

Differential power analysis goes one step further than simple power analysis, using power consumption measurements to determine the success or failure of key block guesses. This requires an ample record of power traces from previous calculations done by the device. Traces are divided into two subsets, and each subset is averaged together. These two averages are then compared to inspect for any significant variation in power output by the device. This is called the Difference of Means attack. If any significance is found, data leakage can be revealed and secret keys can be predicted [108]. It is more challenging to stop differential power analysis than simple power analysis. Noise and desynchronization can help, but further actions must also be taken.

One major preventative measure to take is the reduction of the signal to noise ratio. This decreases the accuracy of a Difference of Means attack since the amplified averages will be less apparent, greatly increasing the number of data samples the attacker will need, thus slowing attacks dramatically. A second method put forward is power balancing data values and operations. This can be achieved by custom hardware or software. Maintaining the device's power output during computations can greatly reduce a third party's ability to intercept usable intelligence [103]. This also reduces the effectiveness of Difference of Means calculations. Finally, the best method for both simple and differential power analysis is the use of temporary keys. By limiting the number of transactions made with a single private or shared key, power analysis becomes a less viable tool for infiltration. This method is especially punishing against differential power analysis, since differential analysis takes much longer with its data collection phase. By the time it can analyze transmission history, the key will have changed and data collected would become obsolete [103].

4.2.3 Packet Sniffing

Packet sniffing is the act of monitoring packets of interest in a network as they travel to and from devices. System administrators can use it for beneficial reasons such as troubleshooting, but this practice can also have malicious intent. Packet sniffer programs such as Wireshark monitor network traffic on the data link layer of the OSI model, where MAC addresses are used [109]. From here, they

can capture data frames. Within the frames are packets that contain the TCP and UDP transport information about network sessions [110]. Existing so far down in the OSI layers gives these packet sniffing applications access to much of the data flowing to and from devices, especially if the data is irresponsibly unencrypted. Figure 4.10 [111] shows private information being poorly handled via HTTP. The username and password contained in the packet are being stored in plaintext, for anyone on the network to see. To prevent this, HTTPS should be used instead of HTTP to ensure the connection is encrypted. Otherwise, it is the application designers' responsibility to make sure all submissions over the network are encrypted.

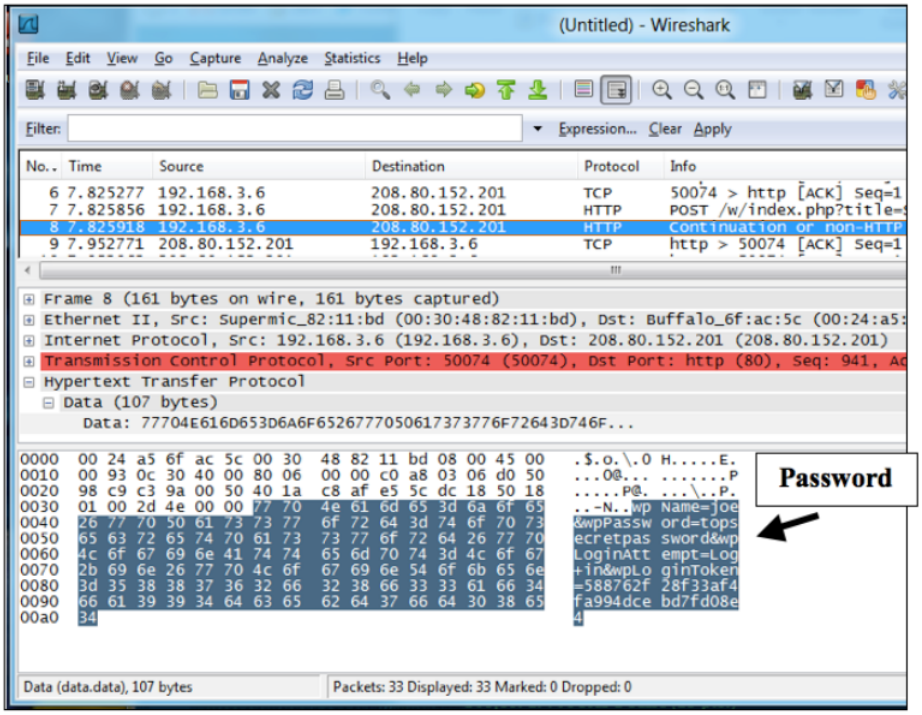

Figure 4.10: Wireshark analysis of an HTTP packet containing an unencrypted username and password provided

In our system model, packet sniffing is used in a benign and productive manner. It takes place on the patient's personal node strictly for localization purposes. The provided node collects beacon frames from many public devices on the network in order to calculate the position of the one device without looking into the contents of any packet traveling through the network. We will secure our localization data being submitted to the database via encryption so that no eavesdroppers may take advantage of our data collection.

4.2.4 MAC randomization

All devices capable of 802.11 network communication are identified by their Media Access Control address unique to only that device's network interface card. While it is useful in determining who was in a network, the transparency soon became a problem when businesses began tracking customers and users based on this unique identifier without their knowledge or explicit consent. Attackers could also easily pinpoint a certain person's location or web traffic data based on their device's MAC address [112].

Subsequent steps have been taken by mobile phone developers to prevent the ease of which an individual could be tracked by their smartphone. A timeline shown in figure 4.11 [113] shows how Apple and Android have gone about implementing preventative measures over recent years. By 2017, both companies had randomized the MAC addresses of phones while scanning and looking for possible network connections. In 2018, Android also included randomization while connecting to a network, and that became the norm for its operating system by 2019. Recently in 2020, Apple has made further strides by randomizing all MACs upon connection, as well as rotating randomized MAC addresses every day. The inclusion of this concept helps further deter attackers that can crack MAC randomization by making any successful efforts ephemeral [113].

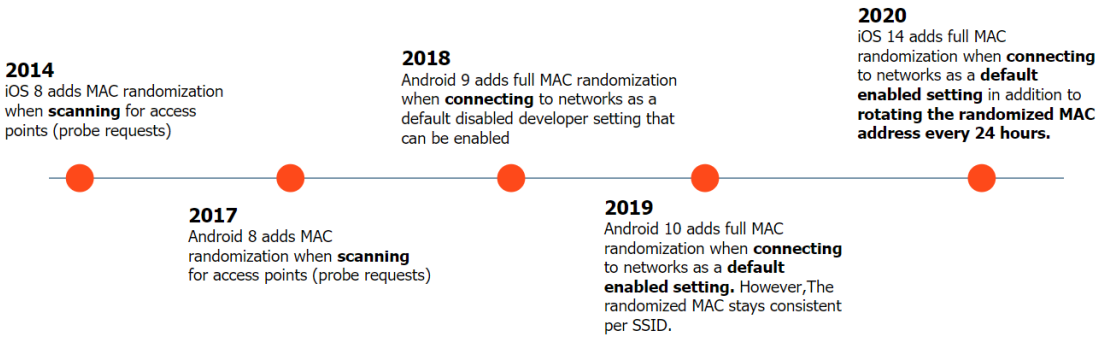

Figure 4.11: Timeline of recent MAC randomization improvements made by Apple and Android

The investigation on MAC randomization undergone by Martin et al. has left the concept in an uncertain place, especially in devices with operating systems less well-supported than those like Android and iOS smartphones [112]. Randomization is not sufficient in eliminating privacy concerns. There are multiple attack techniques that can crack the randomization methods in place today [112]. For our edge nodes, we will focus on the end-to-end encryption of our data, detection of possible rogue access points, and side channel analysis for now. Since our tracking nodes will only stay on a patient during limited time stays, initial randomization and rotating randomizations would seem

unnecessary since they are randomly assigned to patients to begin with.

4.2.5 Data Responsibility

As discussed in chapter 3, collecting data about a person and their location can be invasive when done irresponsibly. The major example is contact tracing apps for mobile phones. These apps' main purposes are to screen people for COVID-like symptoms, suggest actions to be taken, and keep track of their location history to help calculate who else may have come in close contact with a person flagged COVID-positive [114]. These are all important and necessary steps to be taken by responsible citizens, but the infrastructure in which these apps are used can be abused. Many groups that advocate for users' privacy rights have gone against the roll-out of these apps, citing concerns of overreach, discrimination, and lack of transparency of how else the data may be used. Both the American Civil Liberties Union and the Electronic Frontier Foundation have spoken up against this type of data collection [115] [116].

One of the main alarming points is that the Department of Health and Human Services does not deem that third party COVID tracking apps are bound by HIPAA policies [117]. It is up to congressional rulings for such privacy laws to come to fruition, and the law is far behind the pace of technology's constant innovations. Without policies in place, users of applications have no defense or legal recourse against a company or the government using the harvested data for any reason in addition to the contact tracing application. Apple and Google are trying to minimize this data hoarding by storing as much pertinent data locally on the user's phone or device. In any case, the app's use of Bluetooth for proximity detection leaves it open for reporting a high number of false positives and negatives with low accuracy in large and crowded areas [117].

Learning from these current issues, the data collected by our edge node in its enclosed system is reduced to the bare minimum for its intended functionality. The node only monitors the network for beacon frames of the hospital's devices broadcasting on the main network SSID. We filter the data packets down to only the timestamp, MAC ID, and received signal strength. The MAC ID will also be hashed further to lessen correspondence with the physical location of collection. The submission of this data will be encrypted end to end, and the database will only use the data for localization calculations. Since it will be integrated with the hospital's patient care system, we will work with hospital administration to make it HIPAA compliant and safe from misuse without legal recourse.

4.3 Chapter Summary

In this chapter, we investigated relevant machine learning techniques and security protocols pertinent to IoT systems. First, an overview of machine learning was presented, followed by an explanation of supervised learning algorithms versus their unsupervised counterparts. To finish that section, examples of supervised algorithms were given and elaborated upon due to their application in this work. Next, the security threats of access point spoofing, side channel analysis, packet sniffing, MAC randomization, and data responsibility are discussed. Relevant works are cited with examples of how researchers and security experts have gone about mitigating these attacks.

Chapter 5

Results and Evaluations

This chapter provides the results and analysis of the methods put forward for improving the accuracy and security of the localization system. Section 5.1 discusses the impact of the random forest algorithm on our data set, and the various ways in which we can evaluate the algorithm's accuracy. Section 5.2 contains the measures we have taken against the threats of access point spoofing, side channel analysis, and packet sniffing, as well as an analysis of their efficacy.

5.1 Machine Learning Implementation - Random Forest Algorithm

This section will explore the methods used and results found from implementing the random forest algorithm in Python on the RSSI data collected in Sentara RMH in February of 2020. We provide an overview of the methodologies used for data analysis and then continue to visualize and analyze the results.

5.1.1 Results

Accuracy Score

The primary metric used when evaluating a classification algorithm is the accuracy of the model. In the SciKit metric library for Python, an accuracy score method is available to compute the subset accuracy based on the given true and predicted data set values. The method returns a percentage result of how many samples were correctly classified based on the true or training data [118]. This can be used to quickly look at whether your training data is creating a model that can effectively predict and classify new data.

Figure 5.1 shows how we can quickly generate accuracy scores for various sized forests using Python. In our investigation, we used forests of sizes 1, 5, 10, 20, 30, 50, 70, 90, 125, 150, 175, and 200 to see at what rate the accuracy of our forest grew. The RandomForestClassifier library easily creates a model based on tree size and allows for the fitting of training data and predicting of test data. All forest sizes' accuracy scores can be computed at once and then plotted together by adding a loop.

```
treeValues = [1, 5, 10, 20, 30, 50, 70, 90, 125, 150, 175, 200]
accuracyValues = []

for i in range(len(treeValues)):
        model = RandomForestClassifier(treeValues[i], random_state=1)
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)
        accuracyValues.append(accuracy_score(y_test, y_pred))
print("*** Accuracy Values for tree sizes: \n")
print(accuracyValues)

plt.plot(treeValues, accuracyValues, label = "accuracy based on trees")
```

Figure 5.1: Python code for generating accuracy scores for various sized random forests.

Below in figure 5.2, the different forest sizes' accuracy scores are plotted. This data is based on a test size of only 10% of the overall data. This allowed 90% of the data to be used for training the model. We see a sharp increase in accuracy between 1 and 25 trees in the model. This is due to the tendency of a single tree to overfit a model, but the aggregate of many trees greatly reduces such bias. What is concerning about this graph's trend is the erratic behavior of the accuracy between 50 and 100 trees. The high spike at approximately 70 trees seems abnormal and probably more so an exception rather than the rule of how this trend should operate.

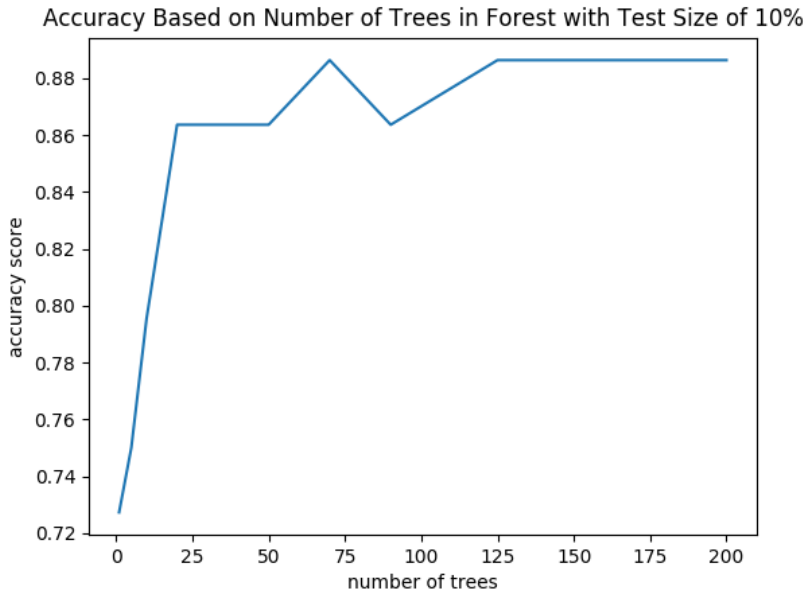

Figure 5.2: Random forest algorithm accuracy calculated with 10% test data.

We see more of the trend we would expect in the next graph, figure 5.3. While it looks lower in accuracy, the scale of the graph is a bit misleading. The accuracy at 50 trees is approximately the same, and increases from there to about 88% at just under 175 trees. This shows that while

it is good to have a large training data set, enough test data must be present in order to have an accurate test of the model when predictions are calculated.

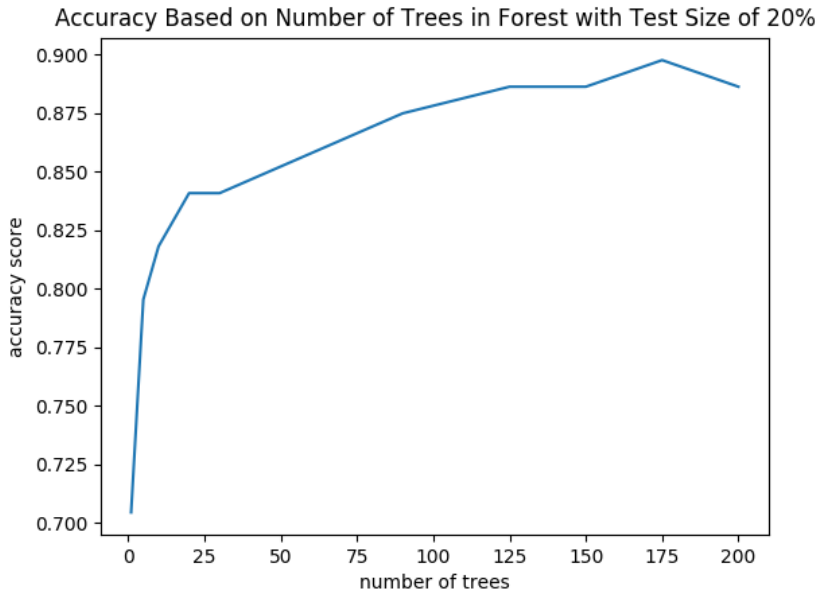

Figure 5.3: Random forest algorithm accuracy calculated with 20% test data.

We end our assessment of accuracy scores with a 30% test and 70% train split of our total data. Two of the most common splits are 20% or 30% allotments for the test data [119]. The graph in figure 5.4 shows a similar trend to the 20% model, but actually caps out at a lower accuracy overall, staying below the 87.5% mark regardless of forest size. This leads us to believe the better model design for our current data could possibly use the 80% train and 20% test data split. However, running many different trials of this assessment may yield slightly different results due to the random nature of the forests and thus their accuracy within this narrow range of 1 to 2% difference. For this reason, other aspects of the model must be considered to have a better understanding of its overall accuracy and reliability.

Classification Report

The classification report provided in the SciKit machine learning library gives key metrics to analyze a classification algorithm's success, in our case, being the random forest algorithm. It provides the precision, recall, f1-score, and support values. Precision is the count of how many data points are correctly classified for each class present in the data sampling. It is the ratio of true positives regarding the sum of all true and false positives [120]. The recall value denotes the fraction of the

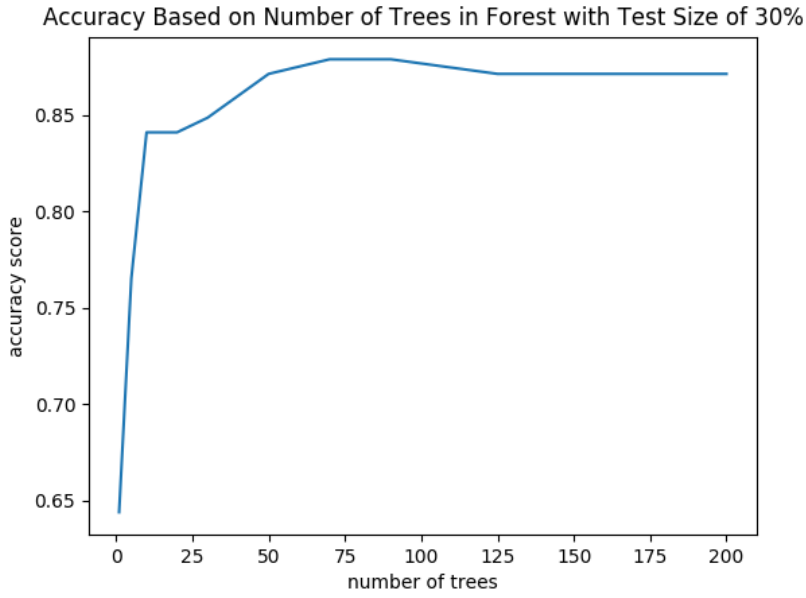

Figure 5.4: Random forest algorithm accuracy calculated with 30 percent test data.

total amount of relevant instances of a class that was retrieved. It is the ratio of true positives in relation to the sum of all true positives and false negatives [120]. The f1-score is the weighted average of the precision and recall scores, calculated by:

$$\frac{2 * (Recall * Precision)}{Recall + Precision}$$

This score is helpful when dealing with uneven class distribution since it takes both false positives and false negatives into account when evaluating the classification effectiveness. Finally, support is the actual number of specific class occurrences in the selected data set [120]. A highly imbalanced support score could indicate a poor sampling method in the chosen process.

Figure 5.5 shows the results from the random forest testing on our data. In this specific trial, a forest of 50 decision trees was generated with replacement and used 20% of the overall data as test data. This left the remaining 80% as training data. We see that our sampling's precision value at the bottom averages out to about 85%, which is right where we expected it given the accuracy scores we have discussed. We can see a similar average of the recall value being about 85%, which is also expected. The f-1 score has many 1.0 scores, showing that many classes are easily predicted successfully. However, some dip as low as 0.50 and 0.67. This is concerning and would hopefully be improved with more data in the model. The support values also show that the data isn't quite normalized. More data is sampled from some classes more than others. Locations like 101 and 103

only have one and two data samples, while others like 301 and 110 have eight and nine, respectively.

```
  'precision', 'predicted', average, warn_for)
              precision    recall  f1-score   support

         101       0.00      0.00      0.00         1
         102       1.00      1.00      1.00         1
         103       1.00      1.00      1.00         2
         104       1.00      1.00      1.00         5
         106       1.00      1.00      1.00         2
         107       0.62      1.00      0.77         5
         108       0.00      0.00      0.00         2
         109       1.00      0.86      0.92         7
         110       0.75      1.00      0.86         9
         112       1.00      0.50      0.67         2
         114       1.00      1.00      1.00         2
         115       0.67      1.00      0.80         2
         116       1.00      1.00      1.00         3
         118       1.00      1.00      1.00         2
         201       1.00      1.00      1.00         2
         202       0.67      1.00      0.80         2
         210       1.00      0.33      0.50         3
         211       0.67      0.67      0.67         3
         212       0.50      0.50      0.50         4
         213       1.00      0.50      0.67         2
         215       1.00      1.00      1.00         1
         216       1.00      0.50      0.67         2
         218       0.50      1.00      0.67         2
         301       1.00      0.88      0.93         8
         302       1.00      1.00      1.00         5
         303       1.00      1.00      1.00         4
         304       1.00      1.00      1.00         5

   micro avg       0.85      0.85      0.85        88
   macro avg       0.83      0.80      0.79        88
weighted avg       0.86      0.85      0.84        88
```

Figure 5.5: Random forest algorithm classification report calculated with 20 percent test data and 50 trees in forest.

Confusion Matrix

The final analysis performed on our random forest model is the calculation and visualization of confusion matrices. This model data can be generated using the SciKit library's confusion_matrix method. It calculates the number of true positives, false positives, true negatives, and false negatives present in the data sample. Figures 5.6 and 5.7 show the results for four different trials done. The matrices are cleanly visualized using the Seaborn library's heatmap package.

When evaluating a confusion matrix, the true positives and true negatives will fall upon the diagonal line from the top left of the matrix, down to the bottom right. In fairly normal data sets evaluated with properly trained algorithms, this is where you will see the highest number of data points. False positives will occur vertically above and below the accurate actual and predicted class

value. False negatives occur horizontally to the left and right of the expected and correct value [121].

Figure 5.6 shows two confusion matrices, the first depicting the analysis of predicted versus actual classification of data using 50 trees and a test data size of 20% of the total data. The matrix below is the same sample size but used 100 trees to classify the localization data. We see very similar results, with almost all data falling along the true positive and true negative line. Both matrices have minor amounts of false negatives, denoted by the 1's in various spots underneath the true positive and negative line. What is concerning is the number of false positives and their patterned clustering in both charts. The prominent example is class 9 being mislabeled as class 29 two times in the 50 tree matrix. This shows that our model may need more data to train on before being fully proficient in distinguishing between these two areas' RSSI values.

We repeated this same confusion matrix visualization in 5.7 with 70% of our overall data being used for training the model, and 30% used to test it. Again, we see most data is able to be classified correctly along the true positive and negative line. Also, we see the same grouping of false positives in the upper right corner. This sampling of 30% test size's 50 and 100 tree forests ended up sampling 30 of the possible 33 classes from the data, as opposed to only 29 in the 20% test size in figure 5.6. With this difference, we can see another trend occur in our model. Towards the bottom left, we see a handful of new false negatives, especially in the 50 forest tree. We attribute this to the reduced data size available to the model for the training phase. This is the fine line the data scientist must tow, giving enough data to train on to ensure proper classification, but enough to test the model with to ensure overfitting does not go unchecked.

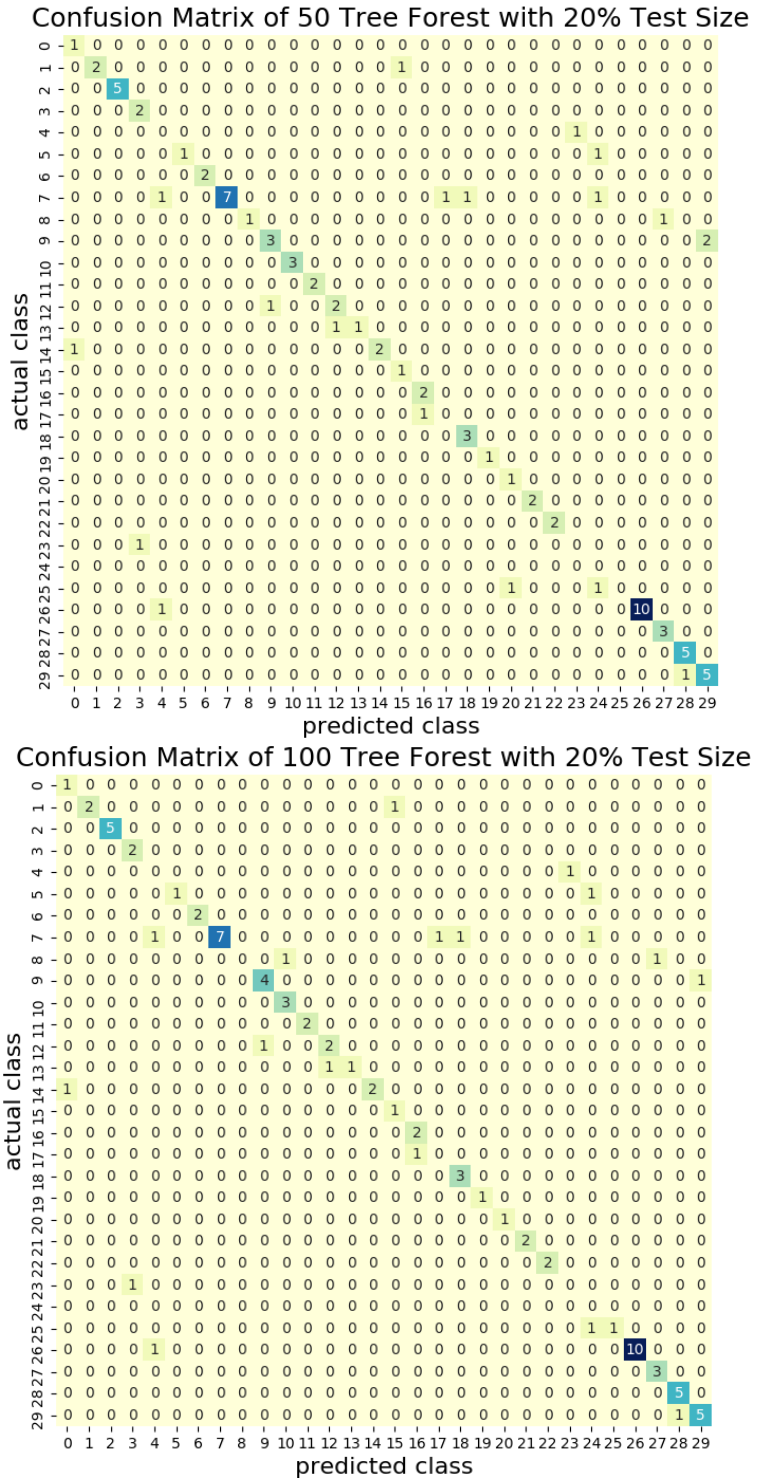

Figure 5.6: Random forest classification matrices with 20% test data samples and 50 and 100 trees in forests.

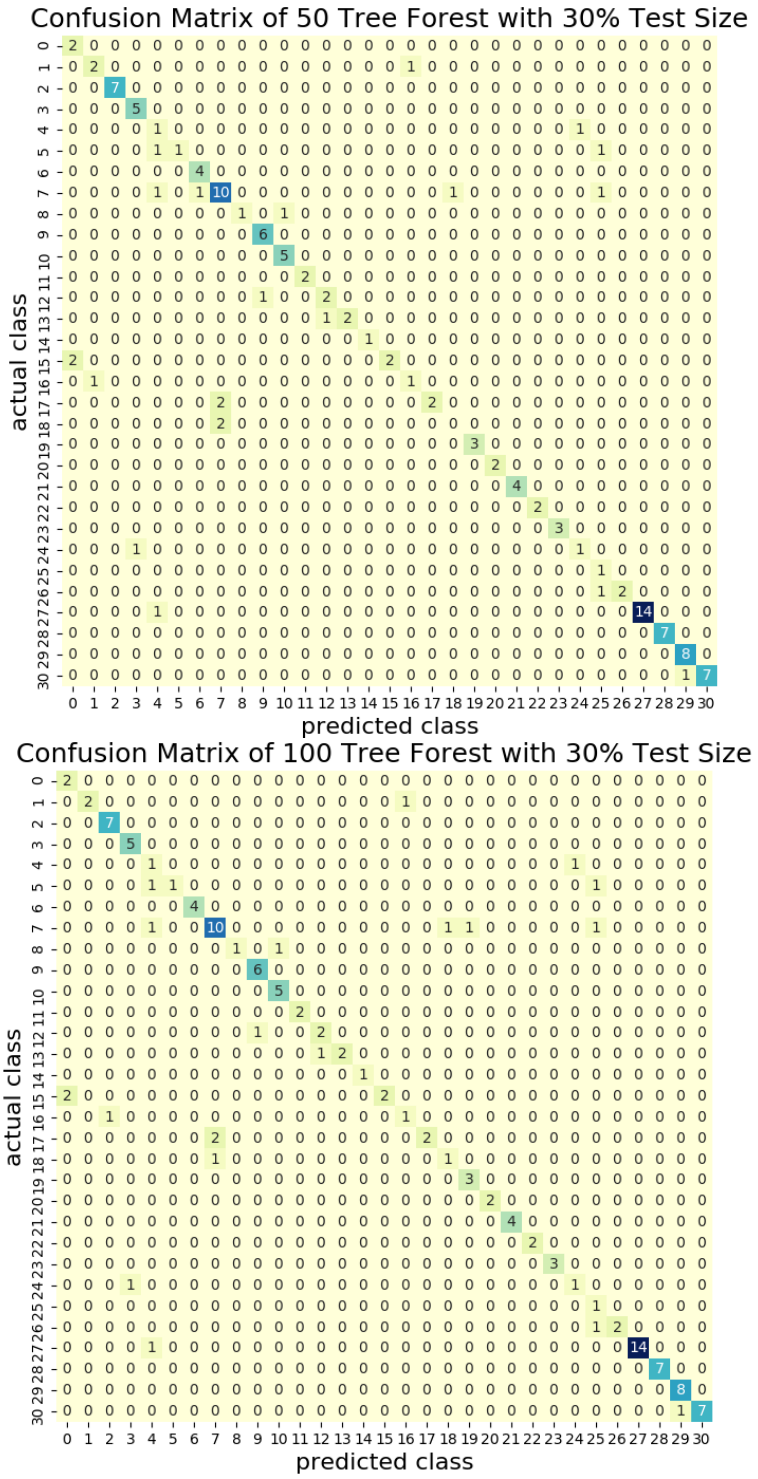

Figure 5.7: Random forest classification matrices with 30% test data samples and 50 and 100 trees in forests.

5.2 Security Countermeasures and Privacy Protocols

This section will explore the methods used and results found from implementing the security protocols mentioned in chapter 4. We begin with an analysis of access point spoofing in section 5.2.1

and how to stop potential attacks. We then look into side channel analysis and reduce data exposure from it in section 5.2.2. Finally, section 5.2.3 details how packet sniffing is mitigated in our localization system.

5.2.1 Access Point Spoofing

Access point spoofing could cause trouble on our network in two ways. First, a spoofed router could execute a man-in-the-middle attack to collect other users' data that connect to this rogue access point instead of the genuine one. Also, a rogue access point broadcasting beacon frames in the network but in a different area will feed false information to our database with tracking nodes recording those signals along with genuine ones. Section 5.2.3 below discusses how we thwart the ability for attackers to collect private data. This section will focus on rogue access points and how to stop them from impacting our localization system's fidelity.

After training, our random forest algorithm decides on the tracking nodes' location based on the nearby signal strengths of access points and other wireless devices on the network. If there are values for wireless devices not typically in range or values much higher or lower than expected, the model can be tricked into deciding on the wrong room classification for that node. If this goes unchecked or unnoticed for long enough, the model may begin to fail in properly classifying node locations. Another issue that must be avoided is allowing recorded data containing spoofed devices into future training data sets that aim to improve location accuracy.

To quickly identify anomalous behavior of known and relied upon access points, we will again look to confusion matrix values for help. In figures 5.6 and 5.7, we saw the trends of a fairly normal and accurate confusion matrix based off of random forest classification. In figure 5.8, we see the noticeable reduction in accuracy when predicted classifications fail to match up with expected ones. The red circles highlight were false negatives and false positives start to stand out in the model's attempt to classify the data. In this manner, we can run daily or weekly tests for each known location's recent data and test it against our original training data. If there are numerous false positives or negatives, investigations into the physical system's state can occur.

This type of inquiry also helps pinpoint the possible location of a rogue access point or other wireless devices. False positives and false negatives would no longer be miscalculations due to poor training. They would be calculations influenced by foreign data. By investigating the areas in and around where these inaccurate classifications are occurring, the location of a rogue wireless device can be narrowed down or even pinpointed.

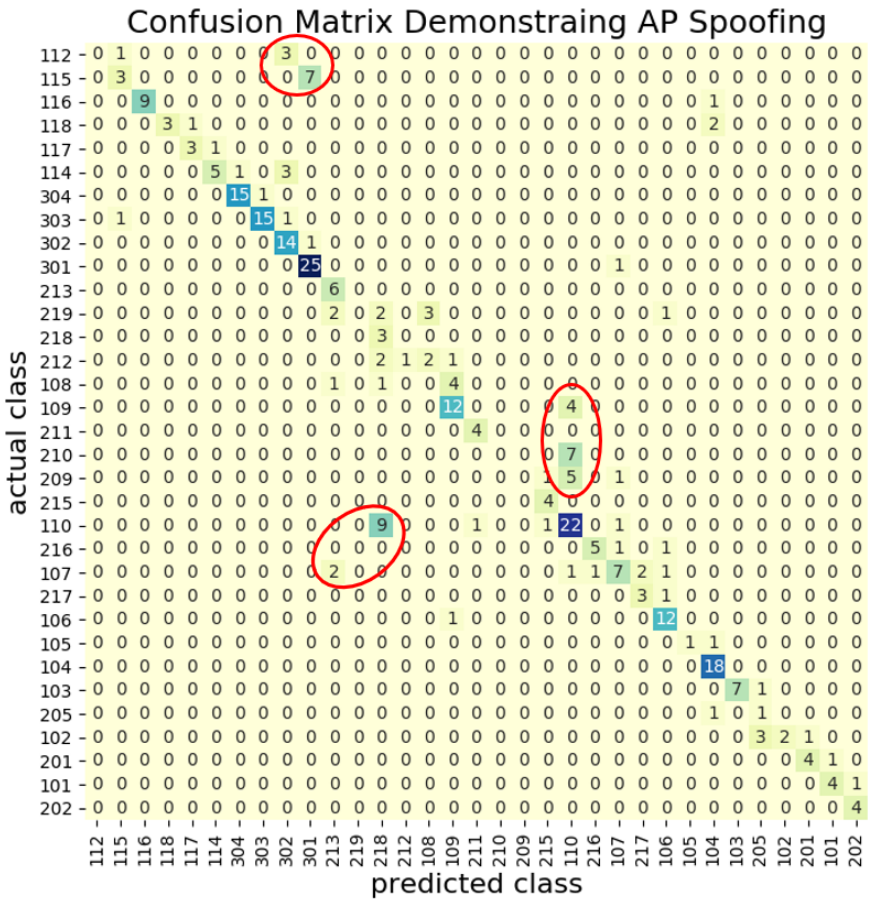

Figure 5.8: Confusion matrix demonstrating how to identify anomalous trends in a data sample.

5.2.2 Side Channel Analysis

As discussed in section 4.2.2, side channel analysis attacks are hard to prevent or detect if the attacker has physical access to a device. An oscilloscope can measure power consumption to decipher encryption keys in simpler systems via simple power analysis, or measure many function calls over time and record the successful and failed key block guesses of an encryption algorithm. This more advanced method of key block guessing is carried out in a differential power attack, and is beyond the scope of this research [103, 104]. We will focus our attention on creating noise to deter an attacker from carrying out a simple power analysis on our nodes. By doing so, a side channel attack would be inefficient and not worth pursuing for a typical attacker.

Our goal is to create noise on the node's processor simultaneously when the RSA encryption occurs during database submissions. If carried out correctly, an oscilloscope conducting a simple power analysis will not be able to easily distinguish the RSA private key that belongs to the node and is associated with the database. Figure 5.9 shows the method in which we created noise at the time

of encryption. Two functions were created. The "noiseFunction" is given a randomly generated RSA key and encrypts a message. The "dbConnection" function is passed the *ssl* dictionary, containing the location to the real certificate and private key. In this function, the actual encryption takes place and submits recorded data to the database. The important feature of this code is how the two functions are called. Multithreading is used such that each function is run in a separate thread, called simultaneously. They are then both joined back together when the functions both complete. This is the way in which noise is created, masking the true RSA key from easy observation.

```
import threading
import Crypto
from Crypto.PublicKey import RSA
from Crypto import Random
import MySQLdb
from hashlib import sha256
import os

#function added to create noise for side channel analysis prevention
def noiseFunction(pKey):
        print("in noise function")
        encrypted = pKey.encrypt('encrypt this message', 32)
        print 'encrypted message:', encrypted #ciphertext

#actual encrypted database submission
def dbConnection(ssl):
        conn = MySQLdb.connect(host = '192.168.221.134', port=3306, user = 'root', passwd = 'root', db = 'locatedb', ssl=ssl)
        cursor = conn.cursor()
        cursor.execute('insert into data (ID,Date,MAC,distance) values (%s,%s,%s,%s)',('88', 'todays date', 'MAC1', '5585'))
        conn.commit()
        conn.close
        print 'SUBMITTED TO DATABASE'

if __name__ == "__main__":
        #Prepare actual RSA/MySQL encryption
        ssl = {'cert':'/home/zach/client-ssl/client-cert.pem', 'key':'/home/zach/client-ssl/client-key.pem'}

        #Generate RSA key for noise function
        random_generator = Random.new().read
        key = RSA.generate(1024, random_generator)
        publickey = key.publickey()

        #call both functions in threads
        t1 = threading.Thread(target=noiseFunction, args=(key.publickey(),))
        t2 = threading.Thread(target=dbConnection, args=(ssl,))

        t1.start()
        t2.start()

        t1.join()
        t2.join()
```

Figure 5.9: Python code to perform simultaneous encryptions via multithreading.

Figure 5.10 shows the CPU usage with the original submission script. Only the encryption of the actual data is executing. The yellow arrows show the time at which three separate submissions took place from the node to the database. Each submission only activates approximately 15% of the processors' total power. The network history at the bottom also shows this increase of traffic too. While varying slightly between submissions due to other processes running, you can see the uptick of processor power used, which would be obvious to an attacker.

After adding the noise function, we see the CPU usage begin to change in figure 5.11. Each submission reaches or surges past 20% CPU usage. This demonstrates the noise function's usefulness in combating a simple power analysis. The processor is no longer handling just one encryption, but

Figure 5.10: CPU usage for regular submission.

two simultaneously. The power is split between the two encryption threads, creating more noise on the processor and making it harder to distinguish a single key. This technique will not fully protect the device, but rather act as a deterring factor against more straightforward attacks.

5.2.3 Packet Sniffing

As described in section 4.2.3, packet sniffing is the process of analyzing packets being sent over a network to glean information about a device's activities or data being transmitted. In our system, two types of packets are considered relevant for proper functionality. First, the beacon frames transmitted by the access points. These frames contain information such as the timestamp, SSID, RSSI, and other signal and network intricacies. Since they are broadcast by the access points and do not collect any personal information, we do not need further security measures placed on the access points beyond what the hospital system administrators have already provided.

Figure 5.11: CPU usage for submission using multithreading.

The second type of packets we need to increase security is the TCP data frames being sent from our nodes to the MySQL database over the hospital's WLAN. This is necessary to ensure the authenticity and integrity of the data. MySQL provides a solution to this with its Transport Layer Security (TLS) protocol [122]. It is an improvement upon the Secure Socket Layer protocol, which has been proven to have weaker encryption. In TLS, data can be sent over public networks while still confirming data is received safely This is due to the X.509 identity verification standard. This standard uses a Certificate Authority (CA) to assign electronic certificates to any user in the system that will be transmitting data [122]. In our system, the term "users" refer to all tracking nodes. The users also receive public and private keys for asymmetric encryption. A certificate possesses the owner's public key, and can be used to encrypt data. The owner of that key can then decrypt any messages sent to them via their private key that no one else can access. In our system, the MySQL database will assign electronic certificates to all nodes, while also generating each node's public and

private keys. In this way, a node can encrypt a message with the server's public key and send it safely to the database. The server can then decrypt the message using its private key.

Figure 5.12 shows the generated CA certificate created by the MySQL server. It contains basic information about the certificate like creation and expiration dates and version number, but also contains vital information like the public key. The server will share this certificate with all nodes in the network so that they can create a secure TLS connection with the server for the sending of encrypted data.

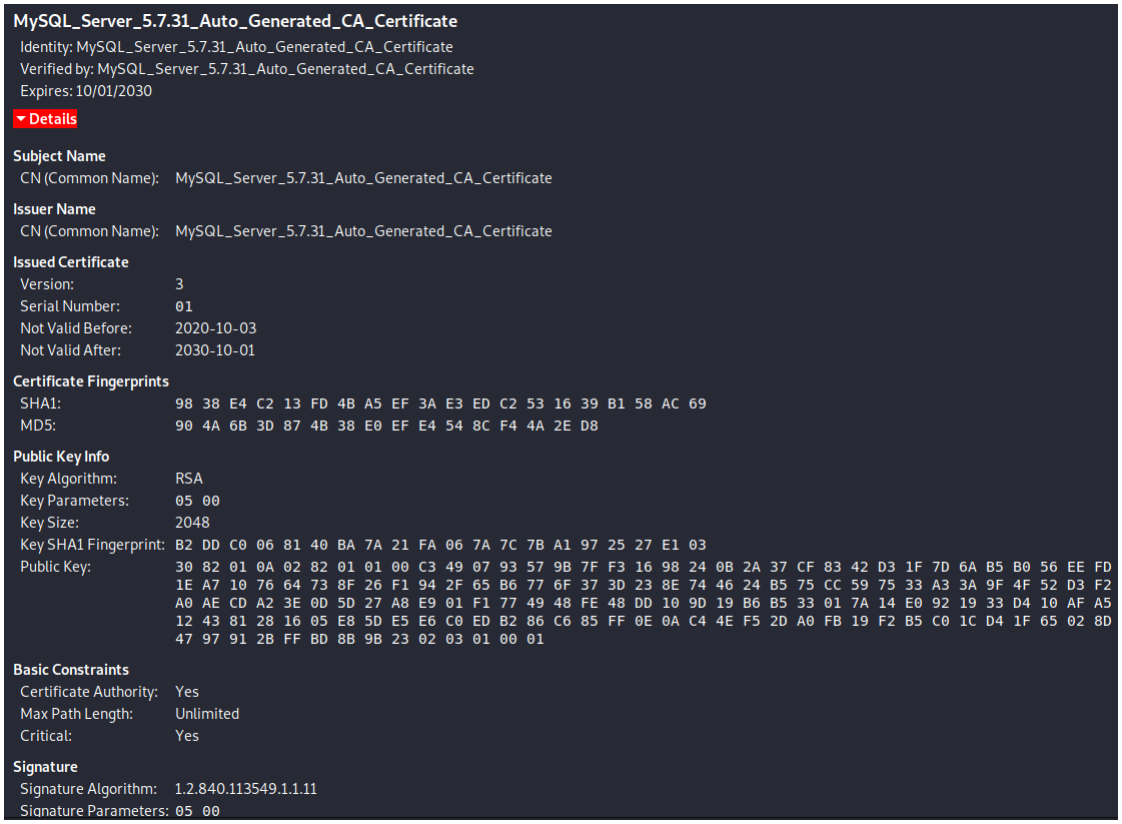

Figure 5.12: Certificate Authority certificate to be shared between client and server.

Figure 5.13 shows the generated CA certificate for the client, or edge node. We can see the client's public key towards the bottom of the certificate. This is shared with the MySQL server so that the server can use it to encrypt messages that the client can then decrypt using its private key. This is necessary for establishing the other side of the TLS secure connection.

The private key files do not easily show their information, as demonstrated in figure 5.14. The only details readily available in the private key file are the algorithm type, size, and SHA fingerprint values. Like the certificate files, it uses a .pem file format, which is used to contain Base64-encoded text rather than binary data [123].

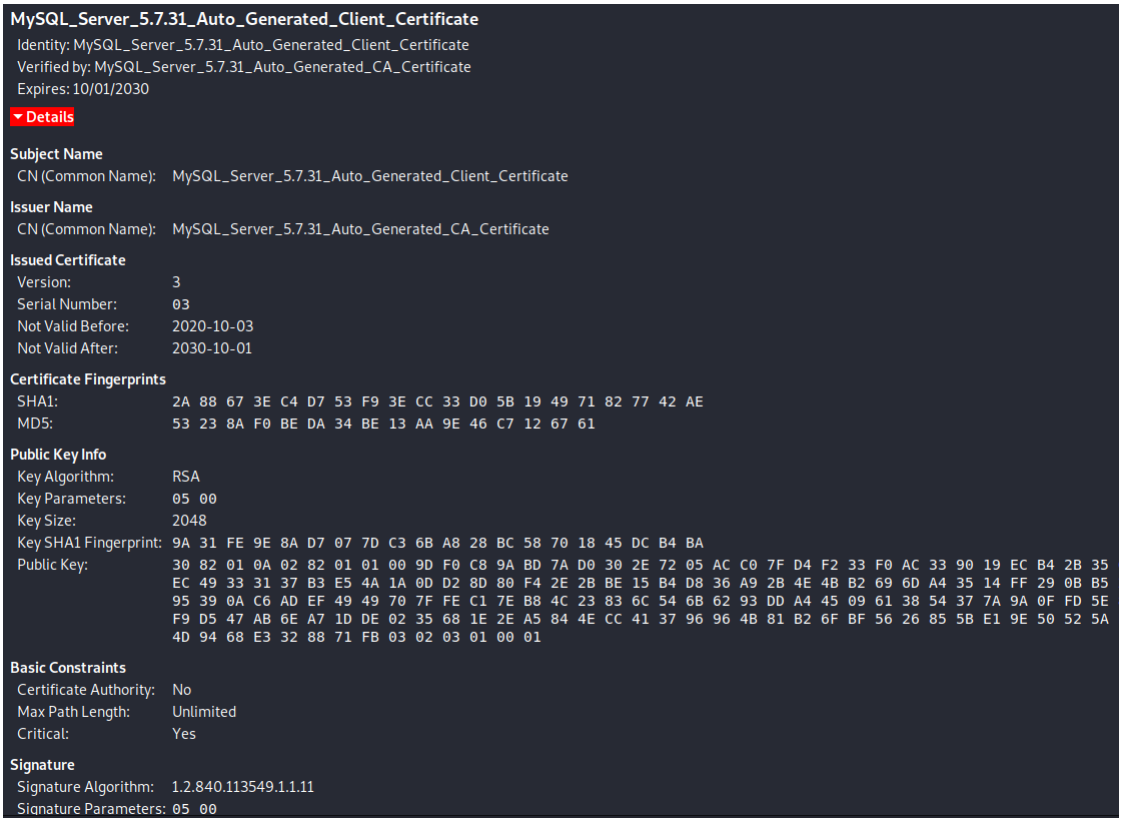

Figure 5.13: Client's unique certificate to show identity.

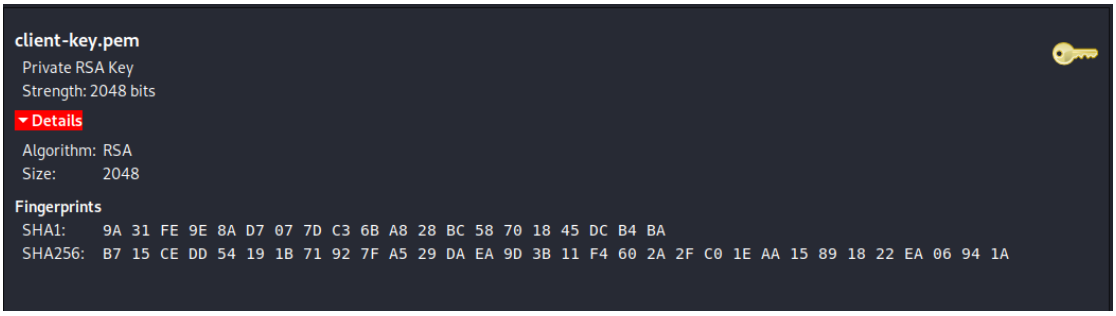

Figure 5.14: Client's unique secret key to prove identity.

After the certificates and keys were created, a secure connection via TLS could be established between the server and client. Figure 5.15 shows how this is done in Python. First the SSL information is set, providing the location of the client's certificate and key in the local memory. It is then used as an argument in the socket connection call. Our current database is locally hosted, so the 192.168.221.134 IP address is that of another Linux machine in our subnet. The port is set to the default 3306. An important factor in this connection is the *user* and *passwd* variables. In MySQL, the requirement of the SSL parameter on login can be user-specific. In our case, the root user has been required to use its SSL certificate and key in order to connect. If this script is run,

a secure connection over TLS is created, and the client can execute queries via the cursor. We see a query crafted to insert new ID, Date, MAC, and distance information into the table named data. By requiring the SSL argument in the connection call, any illegitimate attempts to add data to the database will fail upon attempting a connection with the database.

```
import MySQLdb
from hashlib import sha256

#set SSL certificate and key file locations
ssl = {'cert':'/home/zach/client-ssl/client-cert.pem', 'key':'/home/zach/client-ssl/client-key.pem'}

# Connect to the MySQL database
conn = MySQLdb.connect(host = '192.168.221.134', port=3306, user = 'root', passwd = 'root', db = 'locatedb', ssl=ssl)

#create cursor to carry out query
cursor = conn.cursor()
cursor.execute('insert into data (ID,Date,MAC,distance) values (%s,%s,%s,%s)',('88', 'todays date', 'MAC1', '5585'))
conn.commit()
conn.close
print 'SUBMITTED TO DATABASE'
```

Figure 5.15: Python code showing how a client would use his certificate and key to successfully access the MySQL database.

Successful use of the SSL argument when connecting to a server begins with the TLS handshake protocol. When a connection is requested and the login is successful, the server then responds with a Hello packet, containing the server's public key, certificate, and a request for an exchange. Figure 5.16 shows the client's response, providing the client's certificate, key exchange, and verifies the server's certificate. This handshake then creates a secured connection that remains active for as long as the client stays connected.

```
   12 0.037438390   192.168.221.133      192.168.221.134      TLSv1.2  1255
Transport Layer Security
  TLSv1.2 Record Layer: Handshake Protocol: Certificate
  TLSv1.2 Record Layer: Handshake Protocol: Client Key Exchange
  TLSv1.2 Record Layer: Handshake Protocol: Certificate Verify
  TLSv1.2 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
  TLSv1.2 Record Layer: Handshake Protocol: Encrypted Handshake Message
```

Figure 5.16: Key exchange between client and server.

We can confirm the success of our TLS connection by looking at the Wireshark packet capture in figure 5.17. Under the Transmission Control Protocol, we see the addition of the Transport Layer Security. The protocol belongs to the mysql application and has properly encrypted the sent data. Instead of seeing the packet information in plain text, it is fully encrypted, shown at the bottom of the figure.

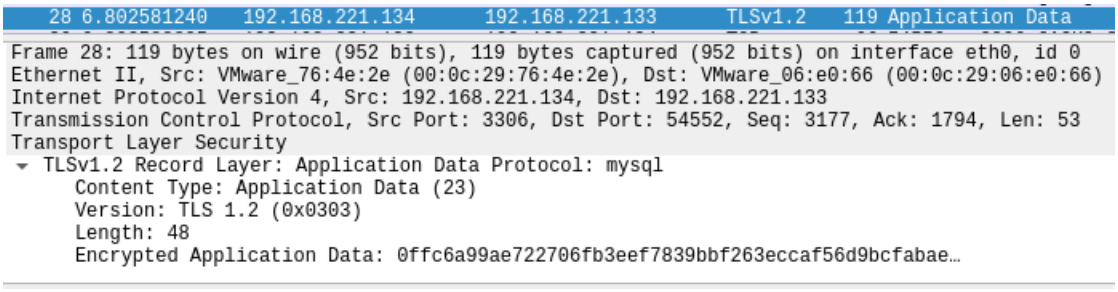

Figure 5.17: MySQL data being sent securely over TCP/TLS connection

5.3 Chapter Summary

This chapter reviewed the contributions made in improving the system's accuracy via machine learning and security via programming and networking techniques. We investigated the accuracy values, classification report, and confusion matrices generated by data run through the random forest algorithm, showing its effectiveness in classifying RSSI data. We also discussed the confusion matrices generated by the random forest algorithm to help identify and locate potential rogue access points in our system. Side channel analysis was then discussed, and how a simple power analysis could be mitigated via a noise function. Finally, packet sniffing was addressed and how to reduce data leakage over a network by implementing an SSL/TLS encryption requirement on all data submissions to the database.

Chapter 6

Conclusion and Future Work

In this thesis, we evaluated and implemented methods to improve the accuracy and security of an 802.11 WLAN localization system. In Chapter 2, we discussed the evolution of localization, beginning with outdoor technologies, then reviewed the techniques used in indoor localization. These techniques were: Received Signal Strength Indicator (RSSI), Channel State Information, Angle of Arrival, Time of Flight, Time Distance of Arrival, Return Time of Flight, and Phase of Arrival. Along with these techniques, the technologies that implement them were reviewed. These technologies were WLAN RSSI, Radio Frequency Identification Device, Bluetooth, Ultra-wideband, ZigBee, LED sensors, and acoustic sensors. The chapter also discussed machine learning and pertinent techniques: Artificial Neural Networks, Extreme Learning Machines, Support Vector Machines, and the Random Forest Algorithm. The chapter concluded with current works regarding the security aspect of edge nodes, focusing on encrypting wireless packets.

In Chapter 3, we discussed the prevalence of IoT systems and the need for edge computing. This leads us to the explanation of our own system's architecture. After an overview of the system, the details of our node's hardware were given. This entailed explaining the capabilities of the Raspberry Pi Zero W, the TP-Link Nano USB dongle, as well as the LiPo SHIM adapter to allow the node to interface with a lithium battery. We then discussed the channel hopping concept so that the node could record beacon frames on all possible broadcasting frequencies. Next, the software was explained to show exactly how the node successfully records, parses, and submits RSSI data. We then provided details on the data's cloud storage services and how data analysis was conducted via MySQL queries and visualized in Python. The chapter concluded with a discussion of the viability of a system like this in a healthcare environment, as well as its potential to help during pandemics with contact tracing.

In Chapter 4, we discussed machine learning techniques and security protocols. We examined both supervised and unsupervised machine learning methods, and how they go about improving their models with data. We then explained the processes of the k-nearest neighbor, random forest, and artificial neural network algorithms. Next, we turned to look into security protocols. This

involved learning more about access point spoofing, side channel analysis, packet sniffing, and MAC randomization, including steps security experts have taken to prevent these attacks. The chapter ends by covering data responsibility for both system engineers and healthcare professionals, and how our system would require collaboration between the two.

Chapter 5 includes the proposed solutions and results put forward in attempting to improve the localization system. First, the implementation of the random forest algorithm on the data collected at Sentara RMH in Harrisonburg, Virginia is discussed. Next, the increase in localization accuracy via random forest is shown through multiple statistical methods and visualizations. We then show how a rogue access point could be detected using the same data analysis, namely a confusion matrix, comparing historical data against the new anomalous data being created by a spoofed access point. Side channel analysis is covered next. We demonstrated how noise could be added to processor simultaneously to help protect an encryption key from being easily captured during simple power analysis. Finally, we walked through the set up of a secure SSL/TLS connection between the node and database to prevent any eavesdroppers on the WLAN from recording our data or injecting false data into our database.

We have contributed to the IoT and localization community with the results included in this work. We have put forward a complete end-to-end system that can successfully perform room-level localization based on historic or training data using random forest machine learning techniques. This work also goes beyond the baseline requirements of localization functionality by including multiple security features often forgotten or ignored in the IoT technology development cycle. It demonstrates all necessary components needed for a ready-to-use IoT system. The agile nature of the back end database also provides the option to implement features like contact tracing in high-risk buildings like hospitals, providing further support for staff and patients.

Our future work would involve returning to Sentara RMH safely and responsibly to collect more data and conduct further localization trials. This is necessary to provide more data to the random forest algorithm for increased accuracy. We would also like to explore ways in which the edge nodes could be reduced in size. This would require printing customized circuit boards, that can replicate the Raspberry Pi, which would be able to be fashioned into comfortable wearables, such as lanyards or bracelets. Localization is still a growing field that has several open-research questions.

Appendix A

Channel Hopping Script

```
#!/bin/bash
# Channel hopping shell script
# GPLv2
# Portions of code graciously taken from Bill Stearns defragfile
# http://www.stearns.org/defragfile/
#
# jwright@hasborg.com

# Defaults
BANDS="IEEE80211B"
DWELLTIME=".25"

CHANB="1 6 11 2 7 3 8 4 9 5 10"
CHANBJP="1 13 6 11 2 12 7 3 8 14 4 9 5 10"
CHANBINTL="1 13 6 11 2 12 7 3 8 4 9 5 10"
CHANA="36 40 44 48 52 56 60 149 153 157 161"

requireutil () {
while [ -n "$1" ]; do
if ! type -path "$1" >/dev/null 2>/dev/null ; then
echo Missing utility "$1". Please install it. >&2
return 1 #False, app is not available.
fi
shift
done
return 0 #True, app is there.
```

```
} #End of requireutil

fail () {
while [ -n "$1" ]; do
echo "$1" >&2
shift
done
echo "Exiting." >&2
echo
exit 1
} #End of fail

usage () {
fail 'chanhop.sh: Usage:' \
 "$0 [-i|--interface] [-b|--band] [-d|--dwelltime]" \
 '-i or --interface specifies the interface name
 to hop on [mandatory]' \
 '-b or --band specifies the bands to use for
 channel hopping, one of' \
 'IEEE80211B  Channels 1-11 [default]' \
 'IEEE80211BINTL Channels 1-13' \
 'IEEE80211BJP Channels 1-14' \
 'IEEE80211A Channels 36-161' \
 '    Use multiple -b arguments for multiple channels' \
 "-d or --dwelltime amount of time to spend on
 each channel [default $DWELLTIME seconds]" \
 ' ' \
 "e.x. $0 -i ath0 -b IEEE80211BINTL -b IEEE80211A -d .10"
 } #End of usage

# main
requireutil sleep whoami iwconfig || exit 1
```

```
if [ `whoami` != root ]; then
echo "You must run this script as root, or under \"sudo\"."
usage
fail
fi


while [ -n "$1" ]; do
case "$1" in
-i|--interface)
INTERFACE="$2"
shift
;;
-b|--band)
ARG_BANDS="$2 $ARG_BANDS"
shift
;;
-d|--dwelltime)
ARG_DWELLTIME="$2"
shift
;;
*)
echo "Unsupported argument \"$1\"."
usage
fail
;;
esac
shift
done

if [ -z "$INTERFACE" ]; then
usage;
exit 1
```

```
fi

# Test the sleep duration value
if [ ! -z "$ARG_DWELLTIME" ] ; then
sleep $ARG_DWELLTIME 2>/dev/null
if [ $? -ne 0 ] ; then
fail "Invalid dwell time specified: \"$ARG_DWELLTIME\"."
fi
DWELLTIME=$ARG_DWELLTIME
fi

# If the user specified the -b argument, we use that instead of default
if [ ! -z "$ARG_BANDS" ] ; then
BANDS=$ARG_BANDS
fi

# Expand specified bands into a list of channels
for BAND in $BANDS ; do
case "$BAND" in
IEEE80211B|IEEE80211b|ieee80211b)
CHANNELS="$CHANNELS $CHANB"
;;
IEEE80211BJP|IEEE80211bjp|ieee80211bjp)
CHANNELS="$CHANNELS $CHANBJP"
;;
IEEE80211BINTL|IEEE80211bintl|ieee80211bintl)
CHANNELS="$CHANNELS $CHANBINTL"
;;
IEEE80211A|IEEE80211a|ieee80211a)
CHANNELS="$CHANNELS $CHANA"
;;
*)
fail "Unsupported band specified \"$BAND\"."
```

```
;;
esac
done

echo "Starting channel hopping, press CTRL/C to exit."
while true; do
for CHANNEL in $CHANNELS ; do
iwconfig $INTERFACE channel $CHANNEL
if [ $? -ne 0 ] ; then
fail "iwconfig returned an error when
setting channel $CHANNEL"
fi
sleep $DWELLTIME
done
done
```

Appendix B

Data Parsing Python Script

```
import MySQLdb
import time
import datetime
import random
import sys
import smtplib
import email.utils
from email.mime.text import MIMEText
import math



#Find one
def find_between( s, first, last ):
    try:
        start = s.index( first ) + len( first )
        end = s.index( last, start )
        return s[start:end]
    except ValueError:
        return ""
#Find 2
def find_between_r( s, first, last ):
    try:
        start = s.rindex( first ) + len( first )
        end = s.rindex( last, start )
        return s[start:end]
```

```
    except ValueError:
        return ""

#Connect and push



def push1(old):
    with open("boottest4.txt") as file:

            first = file.read(1)
            Date = find_between( first, "", "EST")
            #print find_between_r(line,  "", "UTC")
            MAC = find_between( first, "\t", "\t")
            #print find_between_r(line,  "UTC", "-")
            #Signal = find_between( line, "-", "")
            s = find_between_r(first,  "\t", ",")
    if not first:
            print "Itteration Failed"
            time.sleep(5)
            push1(old)

    else:
                print "STARTING PUSH 1"
                print''
                print''
                with open("boottest4.txt") as file:

                    lines = file.readlines()
                    last = lines[-1]
                for line in lines:
                    if line is last:
                            print "Itteration Failed"
                            print "Last Line"
```

```
            print''
            print''
            time.sleep(15)
            continue
    else:
            ID ='1.1'
            print "NEW PACKET"
            Date = find_between( line, "", "EST")
            print Date
            #print find_between_r(line,  "", "UTC")
            MAC1 = find_between( line, "\t", "\t")
            print MAC1
            #print find_between_r(line,  "UTC", "-")
            #Signal = find_between( line, "-", "")
            s = find_between_r(line,  "\t", ",")
            check = s
            s = int(s)
            s = abs(s)
            d = (0.161*(2.71828**(0.0808*s)))
            df = ("%.0f" %d)
            print df, "Feet"
            difference = abs((old) - (d))
            print difference
            print "PACKET COMPLETE"
            print ''
            print''

    if difference >= 15 and d <= 50:
        try:
            print 'STARTING PUSH'
            conn = MySQLdb.connect('AWS URL',
                                   'user',
                                   'PASSWORD',
```

```
                    'E_Health_Fall_2017_2')
            cursor = conn.cursor()
            #cursor.execute ('SELECT*FROM AccessPoints')
            #for row in cursor.fetchmany(2):
             #   print row
              #  letters = str(row)
               #MAC2 = find_between(letters, " '", "'")
               # print MAC2
                #if MAC1 == MAC2:
                 #   print 'Correct MACID'
                  #  print Name
                   # break
                #else:
                 #   print 'Checking Next MACID'
                  #  continue



            cursor.execute('insert into
            test3(ID,Date,MAC,distance)
            values (%s,%s,%s,%s)',(ID,Date,MAC1,df))
            conn.commit()
            conn.close
            print 'PUSH COMPLETE'
            print ''
            print ''
            old = d
            time.sleep(0.5)
        except:
            time.sleep(5)
            continue
    else:
            z = 1
```

```
def main():
    old = 0
    push1(old)
main()
```

Bibliography

[1] T. W. Post, "With fitness trackers in the workplace, bosses can monitor your every step — and possibly more," https://www.washingtonpost.com/business/economy/with-fitness-trackers-in-the-workplace-bosses-can-monitor -your-every-step--and-possibly-more/2019/02/15/75ee0848-2a45-11e9-b011-d8500644dc98_story.html.

[2] I. D. of Public Health, "Contact tracing," https://www.dph.illinois.gov/covid19/contact-tracing.

[3] E. Manley, H. A. Nahas, and J. Deogun, "Localization and tracking in sensor systems," *IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC'06)*, vol. 2, pp. 237–242, 2006.

[4] N. Perugini, *Navigating waters before GPS: Why some mariners still refer to Loran-C*, 2018 (accessed August, 2020). [Online]. Available: https://www.nauticalcharts.noaa.gov/updates/navigating-waters-before-gps-why-some-mariners-still-refer-to-loran-c/

[5] Skybrary, *LORAN-C*, 2018 (accessed August, 2020). [Online]. Available: https://www.skybrary.aero/index.php/LORAN-C

[6] W. Blanchard., "The genesis of the decca navigator system," *The Journal of Navigation*, 2015.

[7] S. Upreti and M. Kumar., "Perspectives of global positioning system (gps) applications," in *Seminar cum Workshop, March 2008*, 2008.

[8] I. Getting., "Perspective/navigation-the global positioning system," *IEEE Spectrum*, vol. 30, 1993.

[9] S. Yucer, F. Tektas, M. V. Kilinc, I. Kandemir, H. Celebi, Y. Genc, and Y. S. Akgl, "Rssi-based outdoor localization with single unmanned aerial vehicle," *ArXiv*, vol. abs/2004.10083, 2020.

[10] M. I. M. Ismail, R. A. Dzyauddin, S. Samsul, N. A. Azmi, Y. Yamada, M. F. M. Yakub, and N. A. B. A. Salleh, "An rssi-based wireless sensor node localisation using trilateration and multilateration methods for outdoor environment," 2019.

[11] J. Trogh, D. Plets, E. Surewaard, M. Spiessens, M. Versichele, L. Martens, and W. Joseph, "Outdoor location tracking of mobile devices in cellular networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, pp. 1–18, 2019.

[12] R.-Z. Li, X.-L. Luo, and J.-R. Lin., "Weighted centroid correction localization in cellular systems," *American Journal of Engineering and Applied Sciences*, vol. 4, 2011.

[13] Z. Farid, R. Nordin, and M. Ismail., "Recent advances in wireless indoor localization techniques and system," *Journal of Computer Networks and Communications*, vol. 2013, Aug 2013.

[14] Y. Wang, M. Li, and M. Li., "The statistical analysis of ieee 802.11 wireless local area network–based received signal strength indicator in indoor location sensing systems," *International Journal of Distributed Sensor Networks*, vol. 13, 2017.

[15] T. D. McAllister, S. El-Tawab, and M. H. Heydari, "Localization of health center assets through an iot environment (locate)," in *2017 Systems and Information Engineering Design Symposium (SIEDS)*. IEEE, 2017, pp. 132–137.

[16] A. Salman, S. El-Tawab, Z. Yorio, and A. Hilal, "Indoor localization using 802.11 wifi and iot edge nodes," in *2018 IEEE Global Conference on Internet of Things (GCIoT)*. IEEE, 2018, pp. 1–5.

[17] A. Hilal, M. Khalil, A. Salman, and S. El-Tawab, "Exploring the use of iot and wifi-enabled devices to improve fingerprinting in indoor localization," in *2019 IEEE Global Conference on Internet of Things (GCIoT)*. IEEE, 2019, pp. 1–6.

[18] F. Zafari, A. Gkelias, and K. K. Leung., "A survey of indoor localization systems and technologies," *IEEE Communications Surveys & Tutorials*, vol. 21, 2019.

[19] D. Zhang, F. Xia, Z. Yang, L. Yao, and W. Zha., "Localization technologies for indoor human tracking," in *5th International Conference on Future Information Technology*, 2010.

[20] O. Hashem, K. A. Harras, and M. Youssef, "Deepnar: Robust time-based sub-meter indoor localization using deep learning," in *2020 17th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, 2020, pp. 1–9.

[21] K. Muthukrishnan, G. Koprinkov, N. Meratnia, and M. Lijding., "Using time-of-flight for wlan localization: feasibility study," *South African Computer Journal*, vol. 29, 2017.

[22] W. Sakpere, M. Adeyeye-Oshin, and N. B. Mlitwa., "A state-of-the-art survey of indoor positioning and navigation systems and technologies," *South African Computer Journal*, vol. 29, 2017.

[23] B. Wang, S. Zhou, W. Liu, and Y. Mo., "Indoor localization based on curve fitting and location search using received signal strength," *IEEE Transactions on Industrial Electronics*, vol. 62, p. 572, Jan 2015.

[24] P. F. e Silva and E. S. Lohan., "Room-level indoor positioning with wi-fi and rfid fingerprints," in *23rd International Conference on Advances in Geographic Information Systems*, 2015.

[25] C. Gomez, J. Oller, and J. Paradells., "Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology," *Sensors*, vol. 12, 2012.

[26] F. S. Danis and A. T. Cemgil., "Model-based localization and tracking using bluetooth low-energy beacons," *Sensors*, vol. 17, Oct 2017.

[27] L. Zwirello, T. Schipper, M. Harter, and T. Zwick., "Uwb localization system for indoor applications: Concept, realization and analysis," *Journal of Electrical and Computer Engineering*, vol. 2012, May 2012.

[28] M. Aykaç, E. Erçelebi, and N. B. Aldin., "Zigbee-based indoor localization system with the personal dynamic positioning method and modified particle filter estimation," *Analog Integrated Circuits and Signal Processing*, vol. 92, May 2017.

[29] S. Shao, A. Khreishah, and I. Khalil., "Retro: Retroreflector based visible light indoor localization for real-time tracking of iot devices," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, 2018.

[30] R. Jia, M. Jin, Z. Chen, and C. J. Spanos., "Soundloc: Accurate room-level indoor localization using acoustic signatures," in *2015 IEEE International Conference on Automation Science and Engineering (CASE)*, 2015.

[31] N. Hernandez, M. Ocana, J. M. Alonso, and E. Kim., "Continuous space estimation: Increasing wifi-based indoor localization resolution without increasing the site-survey effort," *Sensors*, vol. 17, no. 1, p. 147, Jan 2017.

[32] V. C. Cheng, H. Li, J. K. Ng, and W. K. Cheung., "Fast setup and robust wifi localization for the exhibition industry," in *32nd International Conference on Advanced Information Networking and Applications*, May 2018, pp. 1795–1800.

[33] M. Youssef, M. Mah, and A. Agrawala, "Challenges: device-free passive localization for wireless environments," in *Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, 2007, pp. 222–229.

[34] C. Pereira, L. Guenda, and N. B. Carvalho., "A smart-phone indoor/outdoor localization system," in *2011 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 2011.

[35] T. Kulshrestha, D. Saxena, R. Niyogi, V. Raychoudhury, and M. Misra., "Smartits: Smartphone-based identification and tracking using seamless indoor-outdoor localization," *Journal of Network and Computer Applications*, vol. 98, September 2017.

[36] Z. Li, X. Zhao, F. Hu, Z. Zhao, J. L. C. Villacrés, and T. Braun., "Soicp: A seamless outdoor–indoorcrowdsensing positioning system," *IEEE Internet of Things Journal*, vol. 6, October 2019.

[37] F. T. Alaoui, D. Betaille, and V. Renaudin., "Pedestrian dead reckoning navigation with the help of a*-based routing graphs in large unconstrained spaces," *Wireless Communications and Mobile Computing*, July 2017.

[38] E. S. Team, *What is Machine Learning? A definition*, 2020 (accessed August, 2020). [Online]. Available: https://expertsystem.com/machine-learning-definition/

[39] Y. Li, Z. Gao, Z. He, Y. Zhuang, A. Radi, R. Chen, and N. El-Sheimy., "Wireless fingerprinting uncertainty prediction based on machine learning," *Sensors*, vol. 19, January 2019.

[40] YuanLan, Y. Chai, and SohGuang-BinHuang., "Ensemble of online sequential extreme learning machine," *Neurocomputing*, vol. 72, August 2009.

[41] H. Zou, X. Lu, H. Jiang, and L. Xie., "A fast and precise indoor localization algorithm based on an online sequential extreme learning machine," *Sensors*, vol. 15, no. 1, pp. 1804–1824, Jan 2015.

[42] A. S. AL-Khaleefa, M. R. Ahmad, A. A. M. Isa, M. R. M. Esa, Y. Aljeroudi, M. A. Jubair, and R. F. Malik., "Knowledge preserving oselm model for wi-fi-based indoor localization," *Sensors*, vol. 19, May 2019.

[43] T. Evgeniou and M. Pontil, "Support vector machines: Theory and applications," in *Advanced Course on Artificial Intelligence*. Springer, 1999, pp. 249–257.

[44] K. Anusha, R. Ramanathan, and M. Jayakumar, "A support vector regression approach to detection in large-mimo systems," *Telecommunication Systems*, vol. 64, pp. 709–717, 2017.

[45] N. Donges, *A COMPLETE GUIDE TO THE RANDOM FOREST ALGORITHM*, 2020 (accessed August, 2020). [Online]. Available: https://builtin.com/data-science/random-forest-algorithm

[46] X. Guo, N. Ansari, and H. Li., "Indoor localization by fusing a group of fingerprints based on random forests," *IEEE Internet of Things Journal*, vol. 5, 2018.

[47] S. Walczak and N. Cerpa., "Artificial neural networks," *Encyclopedia of Physical Science and Technology*, 2003.

[48] H. Rizk, M. Abbas, and M. Youssef, "Omnicells: Cross-device cellular-based indoor location tracking using deep neural networks," in *18th Annual IEEE Conference on Pervasive Computing and Communications (PerCom). IEEE*, 2020.

[49] Z. Farid, R. Nordin, M. Ismail, and N. F. Abdullah., "Hybrid indoor-based wlan-wsn localization scheme for improving accuracy based on artificial neural network," *Mobile Information Systems*, vol. 2016, May 2016.

[50] R. Sharma, *Gaussian distribution: Why is it important in data science and machine learning?*, 2019 (accessed August, 2020). [Online]. Available: https://medium.com/ai-techsystems/gaussian-distribution-why-is-it-important-in-data-science-and-machine-learning-9adbe0e5f8ac

[51] R. D. A, P. Kristalina, and A. Sudarsono., "Secure data transmission scheme for indoor mobile cooperative localization system," in *2017 International Electronics Symposium on Engineering Technology and Applications (IES-ETA)*, 2017.

[52] C. Bradley, S. El-Tawab, and M. H. Heydari, "Security analysis of an iot system used for indoor localization in healthcare facilities," in *2018 Systems and Information Engineering Design Symposium (SIEDS)*. IEEE, 2018, pp. 147–152.

[53] aniketbote, *RSA Algorithm in Cryptography*, 2019 (accessed August, 2020). [Online]. Available: https://www.geeksforgeeks.org/rsa-algorithm-cryptography/

[54] P. Nohe, *The difference between Encryption, Hashing and Salting*, 2018 (accessed August, 2020). [Online]. Available: https://www.thesslstore.com/blog/difference-encryption-hashing-salting/

[55] B. Varghese, N. Wang, S. Barbhuiya, P. Kilpatrick, and D. S. Nikolopoulos, "Challenges and opportunities in edge computing," in *2016 IEEE International Conference on Smart Cloud (SmartCloud)*. IEEE, 2016, pp. 20–26.

[56] Inovex, *The Edge is Near: An Introduction to Edge Computing!*, 2019 (accessed September, 2020). [Online]. Available: https://www.inovex.de/blog/edge-computing-introduction/

[57] I. Beavers, "Intelligence at the edge part 1: The edge node."

[58] R. P. Foundation, *Raspberry Pi 3 Model B*, 2018 (accessed August, 2020). [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

[59] Newegg, *TP-LINK TL-WN722N Wireless N150 High Gain USB Adapter*, 2020 (accessed September, 2020). [Online]. Available: https://www.newegg.com/tp-link-tl-wn722n-usb-2-0/p/N82E16833704045

[60] R. P. Foundation, *Raspberry Pi OS*, 2020 (accessed August, 2020). [Online]. Available: https://www.raspberrypi.org/downloads/raspberry-pi-os/

[61] W. Electronics, *ARM Architecture*, 2020 (accessed August, 2020). [Online]. Available: https://www.watelectronics.com/arm-processor-architecture-working/

[62] S. Compare, *RaspberryPI models comparisone*, 2020 (accessed September, 2020). [Online]. Available: https://socialcompare.com/en/comparison/raspberrypi-models-comparison

[63] R. P. Foundation, *Raspberry Pi Zero W*, 2020 (accessed September, 2020). [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-zero-w/

[64] Newegg, *TP-LINK TL-WN725N Nano Wireless N150 Adapter, 150Mbps, IEEE 802.11b/g/n*, 2020 (accessed September, 2020). [Online]. Available: https://www.newegg.com/tp-link-tl-wn725n-usb-2-0/p/N82E16833704141

[65] Pimoroni, *LiPo SHIM*, 2020 (accessed September, 2020). [Online]. Available: https://shop.pimoroni.com/products/lipo-shim

[66] Metageek, *Why Channels 1, 6 and 11?*, 2020 (accessed September, 2020). [Online]. Available: https://www.metageek.com/training/resources/why-channels-1-6-11.html

[67] E. Tech, *How to Boost Your Wi-Fi Speed by Choosing the Right Channel*, 2020 (accessed September, 2020). [Online]. Available: https://www.extremetech.com/computing/179344-how-to-boost-your-wifi-speed-by-choosing-the-right-channel

[68] Man7, *crontab(5) — Linux manual page*, 2020 (accessed September, 2020). [Online]. Available: https://man7.org/linux/man-pages/man5/crontab.5.html

[69] J. Geier, *How to: Assign 802.11b/g Access Point Channels*, 2020 (accessed September, 2020). [Online]. Available: http://www.wireless-nets.com/resources/tutorials/assign_ap_channels.html

[70] Y. Hanawa, *chanhop.sh*, 2020 (accessed September, 2020). [Online]. Available: https://gist.github.com/hnw/6fbd3ac3bb59d0c93fc0bd2a823cf5cb

[71] Wireshark, *Tshark Man Pages*, 2020 (accessed September, 2020). [Online]. Available: https://www.wireshark.org/docs/man-pages/tshark.html

[72] N. P. Networks, *802.11 Data Types and Wireshark filters*, 2020 (accessed September, 2020). [Online]. Available: http://www.netprojnetworks.com/802-11-data-types-and-wireshark-filters/

[73] A. W. Services, *About AWS*, 2020 (accessed September, 2020). [Online]. Available: https://aws.amazon.com/about-aws/

[74] T. Healthy, *This Is Why Doctors Still Use Pagers*, 2020 (accessed September, 2020). [Online]. Available: https://www.thehealthy.com/healthcare/doctors/hospital-pagers/

[75] SHRM, "Monitoring employee productivity: Proceed with caution," https://www.shrm.org/hr-today/news/hr-magazine/pages/0615-employee-monitoring.aspx.

[76] H. Business and Technology, "How patient-tracking technology is improving care at hospitals," https://www.healthcarebusinesstech.com/patient-tracking-hospitals/.

[77] Z. Li, Y. Yang, and K. Pahlavan, "Using ibeacon for newborns localization in hospitals," in *2016 10th International Symposium on Medical Information and Communication Technology (ISMICT)*. IEEE, 2016, pp. 1–5.

[78] A. Association, "Sleep issues and sundowning," https://www.alz.org/help-support/caregiving/stages-behaviors/sleep-issues-sundowning/.

[79] J. Schmidt, M. R. G. Marques, S. Botti, and M. A. L. Marques., "Recent advances and applications of machine learning in solid-state materials science," *Computational Materials*, vol. 5, no. 83, 2019.

[80] J. H. Lee, J. Shin, and M. J. Realff., "Machine learning: Overview of the recent progresses and implications for the process systems engineering field," *Computers and Chemical Engineering*, vol. 114, pp. 111–121, June 2018.

[81] J. Brownlee, *Supervised and Unsupervised Machine Learning Algorithms*, 2016 (accessed September, 2020). [Online]. Available: https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/

[82] JavaTPoint, *Regression vs. Classification in Machine Learning*, 2016 (accessed September, 2020). [Online]. Available: https://www.javatpoint.com/regression-vs-classification-in-machine-learning

[83] MathWorks, *Machine learning technique for finding hidden patterns or intrinsic structures in data*, 2020 (accessed September, 2020). [Online]. Available: https://www.mathworks.com/discovery/unsupervised-learning.html

[84] Stanford CS221, "K-means," 2013, [Online; accessed September, 2020]. [Online]. Available: https://stanford.edu/~cpiech/cs221/handouts/kmeans.html

[85] G. Developers, "Clustering algorithms," https://developers.google.com/machine-learning/clustering/clustering-algorithms.

[86] T. Srivastava, "Introduction to k-nearest neighbors: A powerful machine learning algorithm (with implementation in python & r)," https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/.

[87] L.-Y. Hu, M.-W. Huang, S.-W. Ke, and C.-F. Tsai., "The distance function effect on k-nearest neighbor classification for medical datasets," *SpringerPlus*, vol. 5, Aug 2016.

[88] M. Sarkar and T. Leong, "Application of k-nearest neighbors algorithm on breast cancer diagnosis problem." in *AMIA Annual Symposium*, 2000, pp. 759–763.

[89] A. Chakure, "Decision tree classification," https://towardsdatascience.com/decision-tree-classification-de64fc4d5aac.

[90] M. Borcan, "Decision tree classifiers explained," https://programmerbackpack.com/decision-tree-explained/.

[91] J. Ali, R. Khan, N. Ahmad, and I. Maqsood., "Random forests and decision trees," *International Journal of Computer Science Issues*, vol. 9, Sept 2012.

[92] L. Breiman and A. Cutler, "Random forests," https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm#intro.

[93] W. Koehrsen, "Random forest simple explanation," https://medium.com/@williamkoehrsen/random-forest-simple-explanation-377895a60d2d.

[94] A. Krogh, "What are artificial neural networks?" *Nature biotechnology*, vol. 26, no. 2, pp. 195–197, 2008.

[95] A. Yılmaz, C. Aci, and K. Aydin, "Mffnn and grnn models for prediction of energy equivalent speed values of involvements in traffic accidents/trafik kazalarında tutulumunun enerji eşdeğer hız değerleri tahmininde mffnn ve grnn modelleri," *International Journal of Automotive Engineering and Technologies*, vol. 4, no. 2, pp. 102–109, 2015.

[96] A. Al-Masri, "How does back-propagation in artificial neural networks work?" https://towardsdatascience.com/how-does-back-propagation -in-artificial-neural-networks-work-c7cad873ea7.

[97] J. V. Tu., "Advantages and disadvantages of using artificial neural networks verus logistic regression for predicting medical outcomes," *Journal of Clinical Epidemiology*, vol. 49, 1996.

[98] CDC, *Health Insurance Portability and Accountability Act of 1996 (HIPAA)*, 2018 (accessed September, 2020). [Online]. Available: https://www.cdc.gov/phlp/publications/topic/hipaa.html

[99] U. D. of Health, H. Services *et al.*, "Summary of the hipaa security rule," *Health Insurance Portability and Accountability*, 2018.

[100] N. M. Ahmad, A. H. M. Amin, S. Kannan, M. F. Abdollah, and R. Yusof, "Detecting access point spoofing attacks using partitioning-based clustering," *Journal of Networks*, vol. 9, no. 12, p. 3470, 2014.

[101] hash3liZer, "How to setup fake (rogue) access point on linux — hostapd," https://www.shellvoide.com/wifi/setting-up-fake-access-point-or-evil-twin-to-hack-wifi-rogue-ap/.

[102] R. Orsi, "Understanding evil twin ap attacks and how to prevent them," https://www.darkreading.com/attacks-breaches/understanding -evil-twin-ap-attacks-and-how-to-prevent-them-/a/d-id/1333240.

[103] J. Porup, "What is a side channel attack? how these end-runs around encryption put everyone at risk," 2019 (accessed September, 2020), "https://www.csoonline.com/article/3388647/what-is-a-side-channel -attack-how-these-end-runs-around-encryption-put-everyone-at-risk.html.".

[104] N. T. Courtios, "All about side channel attacks," http://www.nicolascourtois.com/papers/sc/sidech_attacks.pdf.

[105] anysilicon, "Side-channel attacks: How differential power analysis (dpa) and simple power analysis (spa) work," https://anysilicon.com/side-channel-attacks-differential-power-analysis-dpa-simple-power-analysis-spa-works/.

[106] T.-H. Le, J. Clediere, C. Serviere, and J.-L. Lacoume., "Noise reduction in side channel attack using fourth-order cumulant." *IEEE Transactions on Information Forensics and Security*, vol. 2, pp. 710 – 720, November 2007.

[107] K. M. Abdellatif, D. Couroussé, O. Potin, and P. Jaillon, "Filtering-based cpa: A successful side-channel attack against desynchronization countermeasures," in *Fourth Workshop on Cryptography and Security in Computing SystemsAt: Stockholm*, 2017, pp. 161–174.

[108] O. Lo, W. J. Buchanan, and D. Carson., "Power analysis attacks on the aes-128 s-box using differential power analysis (dpa) and correlation power analysis (cpa)," *Journal of Cyber Security Technology*, vol. 1, Sept 2017.

[109] S. Ansari, S. G. Rajeev, and H. S. Chandrashekar., "Packet sniffing: a brief introduction," *IEEE Potentials*, vol. 21, Jan 2003.

[110] G. for Geeks, "Layers of osi model," https://www.geeksforgeeks.org/layers-of-osi-model/.

[111] S. Bowne, "Stealing passwords with a packet sniffer," https://samsclass.info/123/proj10/p3-sniff.htm.

[112] J. Martin, T. Mayberry, C. Donahue, L. Foppe, L. Brown, C. Riggins, E. C. Rye, and D. Brown., "A study of mac address randomization in mobile devices and when it fails," *Proceedings on Privacy Enhancing Technologies 2017*, March 2017.

[113] ElevenOS, "How mac address randomization can affect the wi-fi experience," https://blog.elevensoftware.com/how-mac-address-randomization-can-affect-the-wifi-experience.

[114] S. McGrail, "Virginia rolls out contact-tracing app for covid-19 exposure," https://hitinfrastructure.com/news/virginia-roll-outs-contact-tracing-app-for-covid-19-exposure.

[115] ACLU, "Aclu issues governance principles for covid-19 contact tracing technologies," https://www.aclu.org/press-releases/aclu-issues-governance-principles-covid-19-contact-tracing-technologies.

[116] A. Crocker, K. Opshal, and B. Cyphers, "The challenge of proximity apps for covid-19 contact tracing," https://www.eff.org/deeplinks/2020/04/challenge-proximity-apps-covid-19-contact-tracing.

[117] C. Shachar, "Protecting privacy in digital contact tracing for covid-19: Avoiding a regulatory patchwork," https://www.healthaffairs.org/do/10.1377/hblog20200515.190582/full/.

[118] scikityb developers, "sklearn.metrics.accuracy score," https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html.

[119] J. Brownlee, "Train-test split for evaluating machine learning algorithms," https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms/.

[120] scikit-yb developers, "Classification report," https://www.scikit-yb.org/en/latest/api/classifier/classification_report.html.

[121] scikityb developers, "sklearn.metrics.confusion matrix," https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html#sklearn.metrics.confusion_matrix.

[122] MySQL, "Using encrypted connections," https://dev.mysql.com/doc/refman/5.7/en/encrypted-connections.

[123] FileInfo, "Pem file extension," https://fileinfo.com/ extension/pem.html.