University of Zurich^UZH

BACHELOR THESIS – Communication Systems Group, Prof. Dr. Burkhard Stiller

Machine Learning-Assisted Shielding: Creating and Utilizing AirTag RSSI Datasets for Android Applications

Samuel Frank
Zurich, Switzerland
Student ID: 21-704-713

Supervisor: Katharina Müller, Prof. Dr. Burkhard Stiller
Date of Submission: January 04, 2025

University of Zurich
Department of Informatics (IFI)
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland

ifi

Bachelor Thesis
Communication Systems Group (CSG)
Department of Informatics (IFI)
University of Zurich
Binzmuehlestrasse 14, CH-8050 Zurich, Switzerland
URL: http://www.csg.uzh.ch/

Declaration of Independence

Abstract

Bluetooth Low Energy (BLE) basierte Tracker sind schon eine Weile auf dem Markt. Durch deren Integration in sogenannte Crowd-Sourced Offline Finding Networks (COFN) haben sie grosse Vorteile im Bezug auf Erschwinglichkeit und Batterielaufzeit verglichen mit alternativen Trackern. Seit dem Branchenriesen wie Apple ihre Implementierung dieser Technologie kommerzialisiert haben, hat dies zu einer andauernden Diskussion über Sicherheitsaspekte und Technologiemissbrauch geführt. Diese Arbeit befasste sich mit der Thematik ob Received Signal Strength Indication (RSSI) Daten kombiniert mit einem Machine Learning (ML) Ansatz benutzt werden können um den Schutz von Opfern von solchem Missbrauch zu verbessern. Spezifisch wurden AirTags benutzt, um einen beschrifteten Datensatz mit insgesamt 13'353 Einträgen zu sammeln. Dieser Datensatz wurde in einem nächsten Schritt verwendet, um verschiedene Klassifikationsmodelle zu trainieren, zu evaluieren und ein Modell in HomeScout, eine Schutzapplikation entwickelt von der Communication Systems Group der Universität Zürich, zu integrieren. Ein Decision Tree Klassifikator hat eine Leistungskennzahl (der die Genauigkeit, den F1-Score und eine Strafe für Overfitting berücksichtigt) von 84 % erreicht, und wurde erfolgreich in HomeScout implementiert.

Bluetooth Low Energy (BLE) based trackers have been on the market for a while now. Through their integration into so-called Crowd-Sourced Offline Finding Networks (COFN), they have great advantages in terms of affordability and battery life compared to alternative trackers. Since industry leaders such as Apple have commercialized their implementation of this technology, there has been an ongoing discussion about its security aspects and potential misuse. This thesis addresses the issue of whether Received Signal Strength Indication (RSSI) data combined with a Machine Learning (ML) approach can be used to improve the protection of victims of such misuse. Specifically, AirTags were used to collect a labeled dataset with 13'353 entries. This dataset was used in the next step to train and evaluate different classification models and finally integrate one model into HomeScout, a protection application developed by the Communication Systems Group of the University of Zurich. A Decision Tree classifier achieved a promising performance score (which incorporates the accuracy, the F1-score and an overfitting penalty) of 84% and was successfully implemented in HomeScout.

Acknowledgments

First and foremost, I would like to thank Katharina Müller for her support over the last 6 months. Her constant availability and helpful feedback during the process of the work was indispensable for me. I also thank my friend, Stefano Anzolut, for proofreading the thesis and providing valuable feedback. In addition, I thank the Communication Systems Group of the University of Zurich's Department of Informatics for the opportunity to write my bachelor thesis with them. I am grateful that the necessary hardware was made available to me and that I was able to learn a lot during this time.

Contents

Chapter 1

Introduction

The communication protocol Bluetooth Low Energy (BLE) was introduced as part of the Bluetooth core specification 4.0 in the year 2010 [1]. Due to its significant advantages in terms of energy consumption and ease of development during implementation, it has since become the standard communication protocol in the field of the Internet of Things (IoT) [2]. According to the 2021 market update released by Bluetooth SIG, Inc., it was predicted that nearly 4 billion devices capable of BLE communication would be shipped in 2021 [3]. The year 2021 was also the time that Apple released a new product called AirTag. AirTags are Apple's implementation of a BLE-based tracker that operate inside a Crowd-Sourced Offline Finding Network (COFN) [4]. Those BLE-based trackers leverage the advantages of the protocol by embedding the hardware inside a COFN and thus providing global item tracking with a battery lifespan typically around one year. BLE-based trackers were neither invented by Apple nor are they the only provider on the market. Other manufacturers such as Samsung [5], Tile [6] and Chipolo [7] offer similar implementations. However Apple has advantages over many other competitors in terms of the network effect. According to Apple, hundreds of millions of Apple devices are part of the network [8]. This work focuses specifically on the AirTag and its corresponding COFN.

Unfortunately, due to the described capabilities and affordability, these trackers are not only suitable for tracking important items but also individuals against their will. There have been multiple reports of AirTags being misused for stalking [9], [10], [11], [12]. This led to a class action lawsuit against Apple [13].

Initially, Apple developed an item safety alert (ISA) as an iOS exclusive protective mechanism against unwanted tracking. After criticism that Android users are not protected enough [14], Apple developed the Tracker Detect application for the Android market. Tracker Detect was also criticized for its inability to passively detect potential threats. You have to actively suspect that you are being tracked in order to initialize the manual scan [15], [16]. As part of recent developments, Apple has collaborated with Google in the form of an Internet draft submitted to the Internet Engineering Task force [17]. As a result of this collaboration, devices running iOS 17.5 and Android 6.0 or later will receive manufacturer-independent security notifications in the event of suspected unwanted tracking [18], [19].

In the context of the work of [20] and [15], an Android application called HomeScout was developed, which can passively search for the dangers described. Furthermore, HomeScout's protection is not limited to BLE-based trackers. The protection is generalized to BLE-based devices that are embedded in a COFN. This also includes smartphones, laptops, and headphones.

1.1 Motivation

This work focuses on investigating whether the HomeScout application can better detect malicious AirTags through a Machine Learning (ML) model that can predict proximity based on Received Signal strength Indication (RSSI) data. The motivation here is that although in theory the aforementioned collaboration between Apple and Google now makes passive scanning for maliciously placed trackers possible by default in Android, there is still a danger from other BLE-based devices, which HomeScout can scan and recognize. The collaboration and the resulting cross-manufacturer protection feature are new and have yet to undergo a thorough testing by external parties. Furthermore, Apple has sold over 55 million AirTags in the first 18 months [21]. This shows the spread of this technology. Improving the general functionality of HomeScout, which includes non-tracker-exclusive BLE protection, remains highly relevant.

1.2 Thesis Goals

The goal of this work is to gain insight into the possibility of proximity prediction based on RSSI data emitted by AirTags considering various ML models. In the context of this, different datasets are collected in different environments, which then serve as a basis for the models and the evaluation process. The performance of different models are investigated. Based on the performance insights, it will be discussed how and if integration into HomeScout can bring an improvement.

In addition, the goals of this work can be broken down further in the form of research questions:

(i) How does RSSI data emitted from AirTags correlate to distance and are there any underlying patterns?

(ii) Which machine learning model with which input features is best suited for predicting the proximity of a BLE emitting AirTag based on the RSSI values?

(iii) Can the in (ii) identified model be used to provide a reasonable shielding mechanism which further improves the accuracy of identifying malicious trackers?

1.3 Thesis Outline

Initially, the Fundamentals section explains important concepts such as BLE, RSSI and COFN in detail. In addition, existing literature is analyzed. Specifically, works in the field of RSSI data collection, preprocessing, and their applications in (ML-based) systems are discussed.

In the Design section of the thesis, the planning and methodology of the data collection experiments are looked at, a possible implementation in HomeScout is designed, and the chosen models are discussed. The process of determining the best performing model is also explained.

In the Results and Evaluation section, the performance of the models and the implementation is examined.

Ultimately, the work is summarized in the Final Considerations section together with an outlook on possible further research.

Chapter 2

Fundamentals

2.1 Background

This section aims to provide an overview of the main concepts within this thesis.

2.1.1 Bluetooth Low Energy

Bluetooth technology has been around since the year 2000. The first version of Bluetooth used at that time is called Bluetooth Basic Rate (BR). The initial idea was to enable wireless communication between two devices. Bluetooth BR made it possible to achieve a raw data rate of 1 million bits per second (1 Mb/s) on the physical layer. The initially most promising applications of Bluetooth BR were in the field of audio. A new technology called Bluetooth BR/Enhanced Data Rate (EDR) was then specified. This made a raw data rate of 2 Mb/s possible, but was still focused on communication between two devices [22].

In 2010, BLE was defined as part of the Bluetooth Core Specification 4.0 [1]. BLE was not intended to replace the above-mentioned versions of Bluetooth, but to complement them. For the first time in the history of Bluetooth, the technology could be used not only in a point-to-point topology but also in a broadcasting manner. The main design goal of BLE was to minimize the power consumption of the devices. Small devices with coin sized batteries should function for many weeks [22].

2.1.1.1 Key Features

This subsection focuses on the main changes introduced with Low Energy (LE) as part of the Bluetooth Core Specification 4.0. Specifically, the differences compared to BR/EDR that aim to enable longer battery life are examined. The source [23] is used. Instead of the terms 'master' and 'slave' used in the book, this thesis uses 'central' and 'peripheral'.

Frequency Bands

Like BR/EDR, the LE radio operates in the 2.4 GHz ISM band. Since this band is shared with several other wireless communication technologies, a mechanism called frequency hopping is used in both versions of Bluetooth. Frequency hopping counteracts interference. An important difference between BR/EDR and LE is that BR/EDR uses 79 channels for frequency hopping, while LE uses only 40 channels. LE also has dedicated channels for advertising and data transmission.

Mostly Off Technology

LE technology aims to keep devices off most of the time and send data only occasionally. This 'mostly off' strategy minimizes energy consumption and extends battery life, as the device is only switched on under certain conditions. For example, a BLE-based temperature sensor could be configured to transmit data only when the temperature reaches a certain threshold and otherwise be switched off.

Faster Connections

Connections based on LE take less time than BR/EDR connections as they only use 3 advertising channels for the connections, whereas BR/EDR uses 32 channels for scanning. This results in a BR/EDR connection taking around 20 milliseconds, while LE connections can be established in under 3 milliseconds. Fast connections make it possible to send data quickly and minimize energy consumption through shorter activation times of the radio module.

Reduced Functionality

In order to achieve long battery runtime, there were restrictions in the functionality supported by LE. This trade-off between energy efficiency and functionality means that LE complements the classic BR/EDR and does not replace it.

One of these restrictions includes the fact that devices that implement LE do not necessarily have to be transmitters and receivers. It is possible but not necessary. In addition, as LE is designed to transmit small amounts of data irregularly, it is not suitable for use cases such as headsets. This means that LE does not need to support the voice channel functionality. Unlike BR/EDR, LE does not allow the roles of central and peripheral to switch after a connection has been established. This simplification helps to keep the link layer much simpler. In contrast to BR/EDR, LE does not require continuous polling of the link. As the connection can be established significantly faster, this functionality can be dispensed with, further reducing energy consumption.

Shorter Packets

Shorter data packets require less energy for transmission, as the radio has to operate in high-power mode only for a short time. With longer packets, the silicon heats up, which changes the physical properties and can lead to frequency deviations that cause packets or connections to be lost. To prevent this, constant recalibration of the radio is necessary, which increases energy consumption and complexity. Shorter packets prevent such temperature changes, eliminating the need for recalibration.

Optimized Power Consumption of Peripherals

LE is designed so that there is an imbalance of energy consumption between the emitting (peripheral) and receiving (central) nodes. This is the case because the central node usually has more power available. The AirTag can be used here as an example of a peripheral device. The Airtag must send data to a central device, in this case a mobile phone. However, since the mobile phone can usually be charged daily, this power consumption is less important. Therefore, LE was designed so that the peripheral device has to be active for as little time as possible while the central device is constantly scanning for packets. This helps to save energy at the emitting nodes. Furthermore, in wireless radio communication, receiving is more energy-consuming than transmitting. This also closes the circle to above that LE based devices do not have to be transmitter and receiver at the same time, and thus further energy can be saved at the peripheral device.

Table 2.1: Summary of Key LE Features adapted from [23].

Connection Type	**Frequency Hopping Spread Spectrum**
Spectrum	2.4 GHz ISM Band. Regulatory range: 2400 - 2483.5 MHz.
Frequency Hopping	Across 40 RF channels. The channels are separated by 2 MHz.
Modulation	Gaussian Frequency Shift Keying (GFSK).
Maximum Data Rate	305 kbps (4.0), 800 kbps (4.2)
Maximum Data Packet Size	27 bytes (4.0), 251 bytes (4.2)
Typical Range	30 m to 100 m.
Topology	Central Peripheral architecture. The number of peripherals is limited only by the availability of resources on the central.
Connection Time	In the range of 2.5 milliseconds. LE supports a much lower connection time as compared to BR/EDR. So it's easier to just re-establish the connection and transfer data in case of LE instead of keeping the connection alive.
Data Security: Authentication Key	AES 128 bit key.
Data Security: Encryption Key	AES-128 (Stronger than BR/EDR)
Voice Channels	Not supported.
Applicability	Does not require line of sight. Intended to work anywhere in the world since it uses unlicensed ISM band.

2.1.1.2 Communication Basics

In order to gain an overview of the basics of LE communication, a few states of the link layer in the state machine diagram will now be discussed. The physical channel is also divided into time units, which are called events. Advertising events and connection events are being looked at. [23] is used as a source for this section. Any deviating sources are cited accordingly. Instead of the terms 'master' and 'slave' used in the book, this thesis uses 'central' and 'peripheral'.

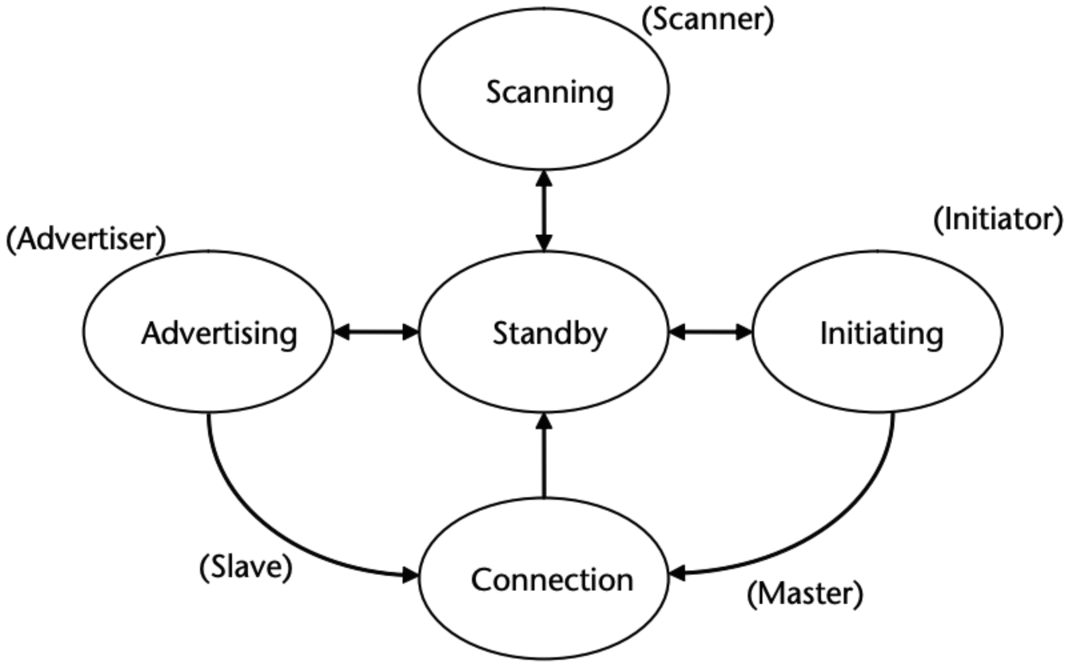


Figure 2.1: Statemachine of the Link Layer [23].

Advertising and Scanning

In the advertising state, the link layer sends advertising packets. Then it may also check which responses are available and respond to these devices accordingly. 2.1 shows, that this state can be reached from the standby state when the link layer decides it is time to advertise. An example is a thermometer that constantly advertises 'I am a thermometer.' It could possibly also advertise other data, for example, 'I have data to send'.

On the other hand, there is the scanning state. A link layer in this state listens for packets from surrounding advertisers. The scanning state can also be reached from the standby state when the link layer decides to scan. A scanner can ask for more information after receiving an advertising packet.

Advertising events are transmissions on the advertising channels. As seen in 2.2, at the beginning of each advertisement event, the advertiser sends an advertisement packet.

The scanner receives the packet and, depending on the type of advertising packet, sends a request back to the advertiser. The advertiser then responds in the same event if a request is received from the scanner. The event then closes, and the process starts all over again with the advertiser sending a packet at the start of the advertisement event.

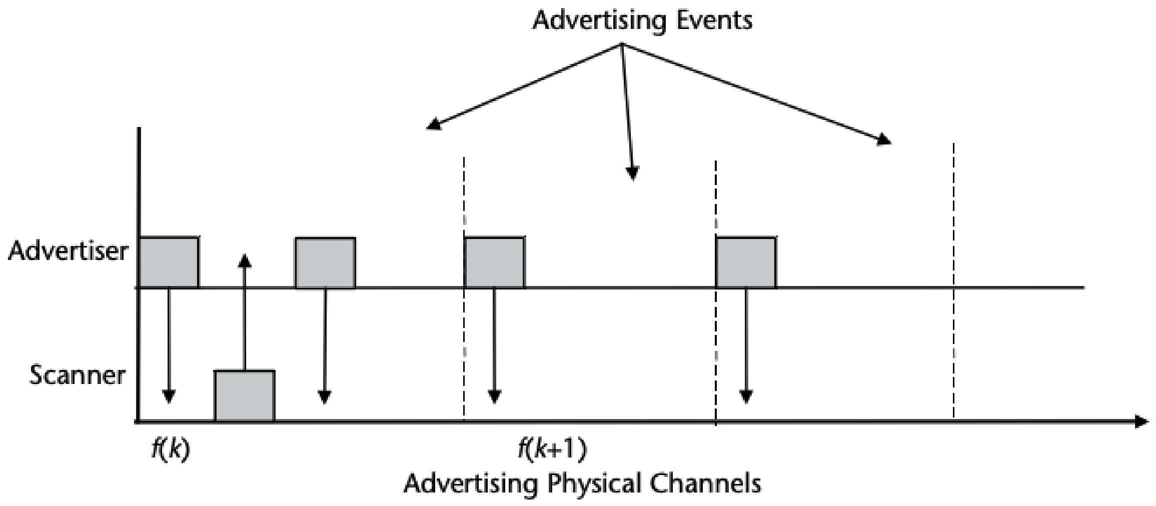


Figure 2.2: Advertising Events [23].

Connection

In the connection state, one device is connected to another. Two roles are defined. There is a central role and the peripheral role. The connection state can be reached either from the advertising state or from the initializing state. This is referred to as a peripheral or a central, respectively.

Connection events are used to exchange data packets between the peripheral device and the central device. As 2.3 shows, the start of a connection event is called an anchor point. At this anchor point, the central transmits a data channel Protocol Data Unit (PDU) to the peripheral. After this exchange, peripheral and central send packets alternately. The peripheral device must always respond to the central device. The central device may or may not respond to packets from the peripheral. All packets during a connection event are transmitted at the same frequency. Channel hopping occurs at the beginning of each connection event.

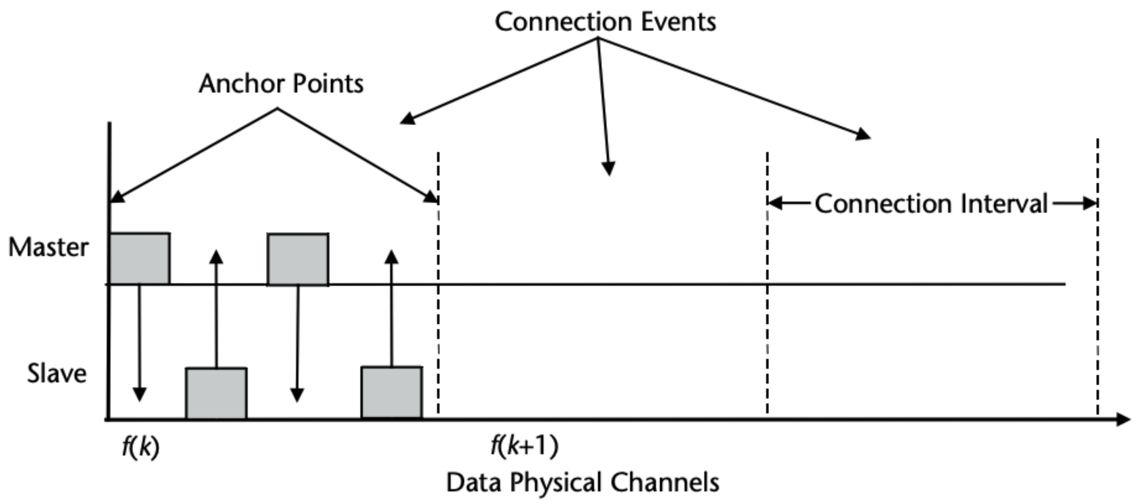


Figure 2.3: Connection Events [23].

2.1.1.3 BLE Protocol Stack Overview

This section is based on [22], and any other sources are referenced accordingly. It should provide a brief overview of the BLE protocol stack.

The LE protocol stack is constituted of a series of discrete modules and layers, some of which are optional and some of which are mandatory. These components are distributed across the two principal building blocks of the architecture, designated as host and controller. Figure 2.4 provides an overview of the LE protocol stack, illustrating the distribution of the protocol between the host and the controller. In addition, it includes a reference to the Open Systems Interconnection (OSI) model.

The separation of the host and the controller is frequently implemented in a manner whereby the host functions as an operating system, while the controller is a System On a Chip (SoC). However, the LE stack does not require this approach. The crucial aspect of the protocol separation is the division of logic into two distinct parts.

The two layers are connected by the Host Controller Interface (HCI), which serves as the foundation for communication between the host and the controller. The HCI is a logical construct, rather than a physical one, and can be implemented in different ways. The key implication of the HCI specification is that hosts and controllers from different manufacturers can work together, enabling interoperability.

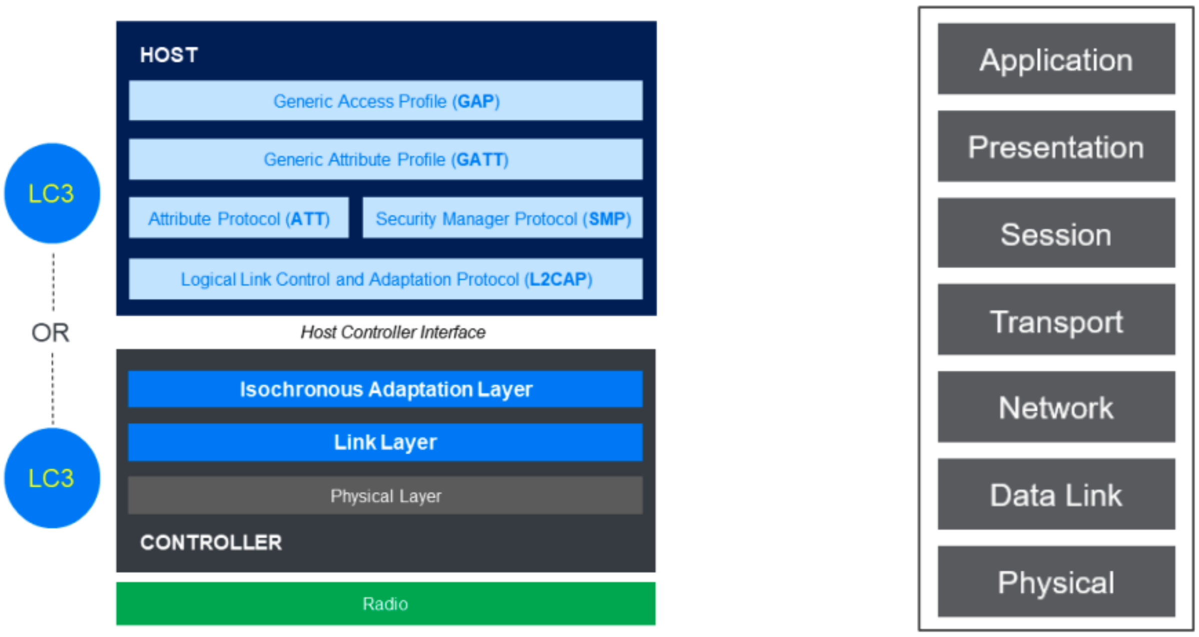

HOST
Generic Access Profile (GAP)
Generic Attribute Profile (GATT)
Attribute Protocol (ATT)
Security Manager Protocol (SMP)
Logical Link Control and Adaptation Protocol (L2CAP)
Host Controller Interface
Isochronous Adaptation Layer
Link Layer
Physical Layer
CONTROLLER
Radio
LC3
OR
LC3
Application
Presentation
Session
Transport
Network
Data Link
Physical

Figure 2.4: LE Protocol stack together with OSI model [22].

Physical Layer

The physical layer of BLE defines the manner in which radio waves are employed to encode and decode digital data at the transmitter and receiver, respectively.

BLE operates within the 2.4 GHz unlicensed band, with a frequency range of 2400 MHz to 2483.5 MHz, which is divided into 40 channels. In order to encode digital data from higher layers of the stack or to decode physical radio waves received from other nodes, a procedure known as Gaussian Frequency Shift Keying (GFSK) is used. Figure 2.5 illustrates the encoding of a 0 or 1 by GFSK, whereby the signal is shifted upward or downward above a certain threshold compared to the center of the relevant channel.

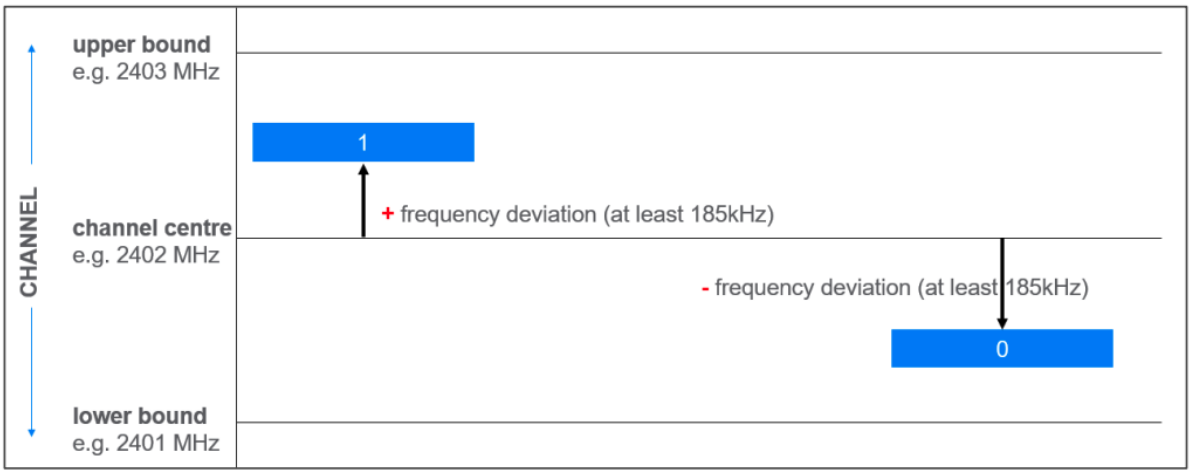


Figure 2.5: Frequency Shift Keying [22].

Link Layer

The second-longest section in the BLE part of the Bluetooth Core Specification 4.0 deals with the link layer. Only the part about HCI is longer. It is arguable that the link layer is the most complicated part of the entire BLE specification. The sophistication of the link layer is also one of the major reasons why BLE is so versatile.

The link layer enables both connection-based and connection-less communication. Furthermore, in addition to point-to-point communication, one-to-many communication with an unlimited number of receiving devices is also made possible.

As mentioned earlier, BLE uses 40 different channels in a 2.4 GHz frequency band. This is where the link layer plays a role and controls how these channels are used. Of the 40 available channels, three are designated for advertising purposes. These channels enable the discovery of devices, the establishment of connections, and the transmission and reception of initial data. The remaining 37 channels are used for data transmission from devices that are already in the connection state. To reduce the likelihood of interference, a technique known as adaptive frequency hopping is employed at the link layer. The 37 data channels are divided into two categories: those that are currently available for use and those that are not. A simple modulo algorithm, utilizing a random value between 5 and 16, is used to hop over the data channels, thereby minimizing the potential for interference [23].

Isochronous Adaptation Layer

The Isochronous Adaptation Layer is responsible for the transmission of time-sensitive data between the HCI and the link layer. This is particularly important for LE audio applications and other time-sensitive uses. As the communication of AirTags is not time sensitive in this sense, this layer will not be discussed in more detail.

Host Controller Interface

The HCI layer of the stack provides a standardized interface for communication between the host and the controller. This enables the host to send commands to the controller and the controller to respond with updates. The HCI concept is not limited to LE but is also used by BR/EDR.

In terms of functional specification, the HCI layers operate as follows. There are two defined terms, 'commands' and 'events'. These are instances of communication between the host and the controller layers. A command is sent from the host to the controller layer, whereas an event is sent in the opposite direction. An event is either an answer to a command that is being received or a command-independent event. Illustration 2.6 shows an example communication flow between the host and the controller, which is the main responsibility of the HCI.

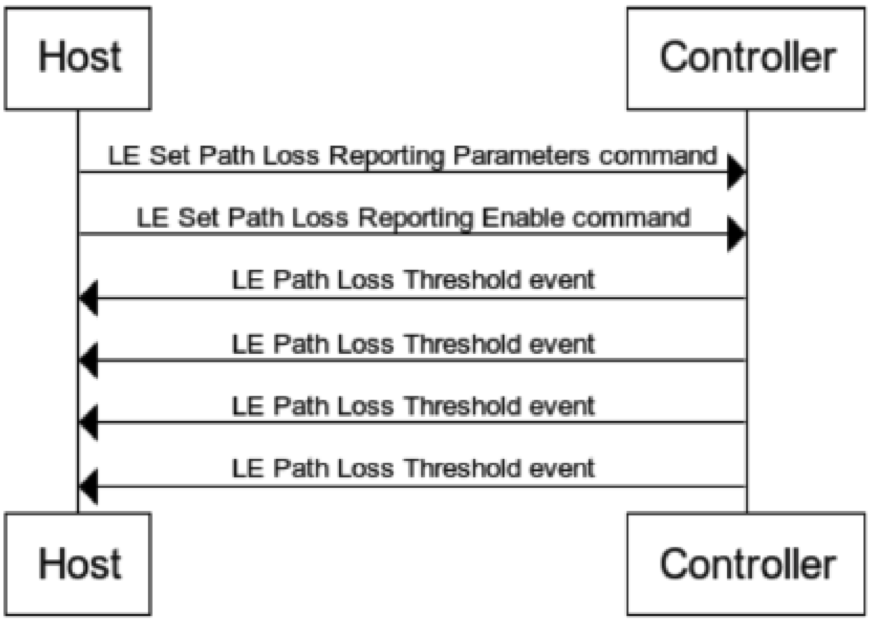

Host
Controller
LE Set Path Loss Reporting Parameters command
LE Set Path Loss Reporting Enable command
LE Path Loss Threshold event
LE Path Loss Threshold event
LE Path Loss Threshold event
LE Path Loss Threshold event
Host
Controller

Figure 2.6: Example HCI Communication [22].

Logical Link Control and Adaptation Protocol

The Logical Link Control and Adaptation Protocol (L2CAP) provides protocol multiplexing, flow control, segmentation, and reassembly of Service Data Units (SDU). To separate sequences of packets that pass between layers of the stack, L2CAP uses the concept of channels. These channels are fixed in the sense that they do not require resources to be allocated and are also associated with higher levels of the stack. Figure 2.7 illustrates the main responsibility of the layer.

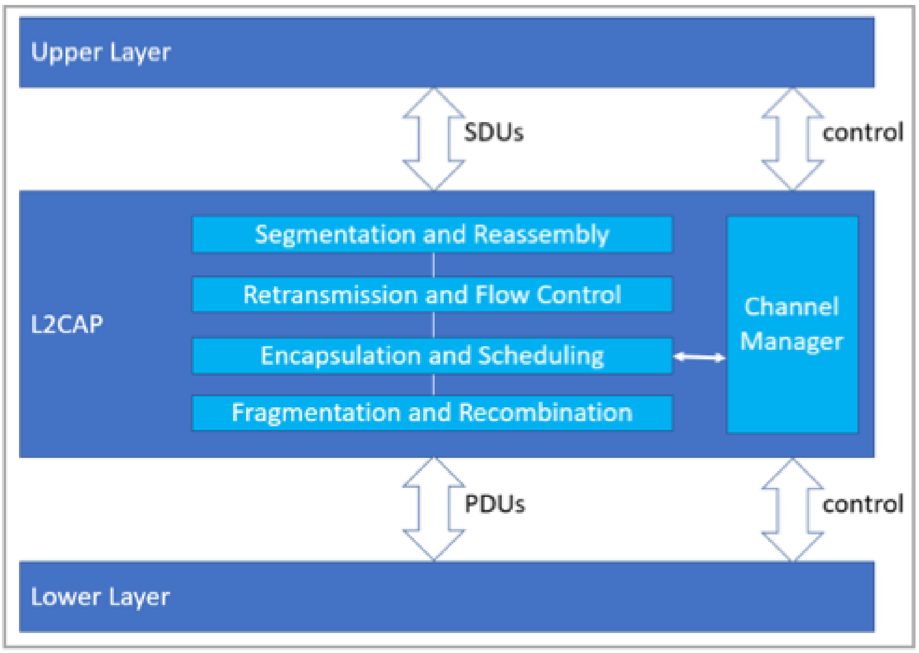

Upper Layer
SDUs
control
L2CAP
Segmentation and Reassembly
Retransmission and Flow Control
Encapsulation and Scheduling
Fragmentation and Recombination
Channel Manager
PDUs
control
Lower Layer

Figure 2.7: Main responsibility L2CAP [22].

The layer takes SDUs and embeds them into a Protocol Data Unit (PDU), which is then passed down and transmitted, where the L2CPU of the receiving node converts the PDUs back into SDUs [23].

The Attribute Protocol

The Attribute Protocol (ATT) is used by two devices. One acts as a server, and the other as a client. This layer thus adheres to the client-server architecture. The server keeps track of data elements called attributes in indexed attribute tables. ATT is one of the main mechanisms by which two devices connected via LE interact with each other. ATT defines 31 different types of PDUs that define the type of communication between connected devices. The 31 types are further divided into 6 broad categories. One of these is the response/request category. In this category, a request PDU is sent to the server device. The server device must then respond within 30 seconds. Otherwise, it is considered a timeout. This shows that the sequential form of communication is prominent throughout the layer. An example of a Request / Response PDU pairing is shown in Figure 2.8. ATT_WRITE_REQ and ATT_WRITE_RSP PDUs are used.

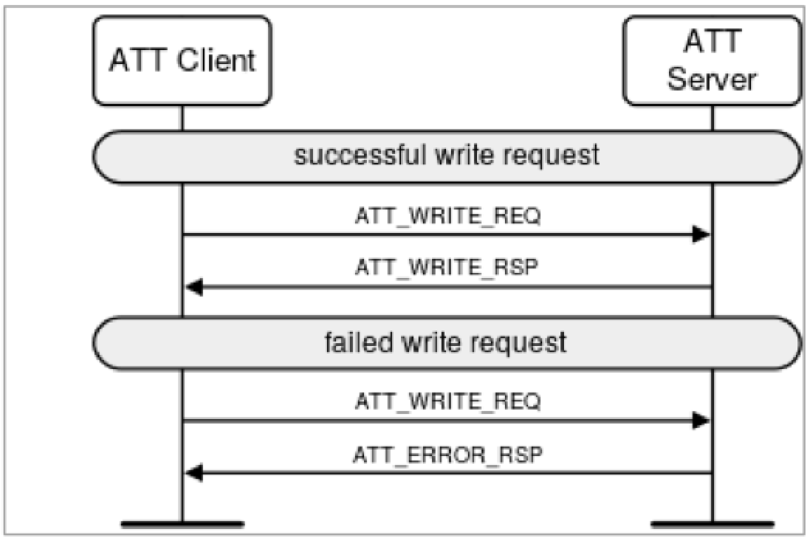


Figure 2.8: Example client server communication [22].

Generic Attribute Profile

The Generic Attribute Profile (GATT) takes data stored as attributes in the attribute tables of the ATT layer described above and transforms it into data types that can be used by the layers above GATT. These data types are called services, characteristics, and descriptors. It also defines how these data types can be used via the ATT. 2.9 shows this concept visualized.

Because the ATT defines a flat structure of attributes and relevant operations for those attributes which are hard to manage, the GATT breaks it down into the more manageable structures mentioned [23].

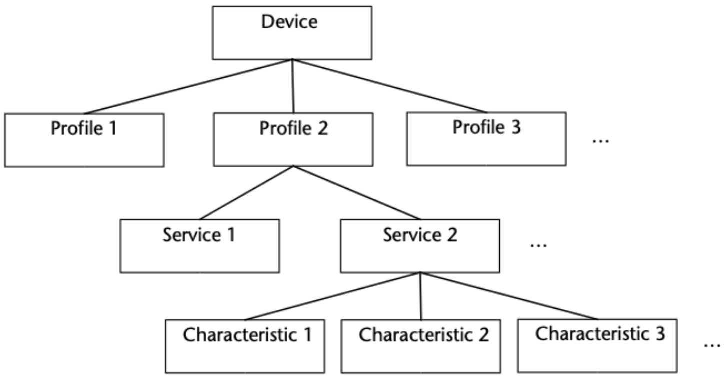


Figure 2.9: Visualization of the breakdown concept [23].

Services provide context for the characteristics they contain. Often, the services represent the most important features or capabilities of the devices. Characteristics are a type of state-based data. Characteristics consist of a type, a corresponding value, and a property that indicates how the data can be used. An example might be a data chunk that is read-only. Descriptors are a hierarchy below the characteristics and are responsible for providing metadata and means of control for the corresponding characteristic.

Generic Access Profile

The Generic Access Profile (GAP) is used to establish connections and discover other devices. In addition, connection less communication and the mechanism for the establishment of isochronous communication channels are also part of the GAP.

Although issues such as advertising, scanning, and connection establishment are at the core of this layer, it acts more as a coordinator. GAP sits on a high layer in the stack. Lower layers such as the link layer and the physical layer then implement it.

2.1.2 Received Signal Strength Indicator

RSSI is a parameter used to quantify the signal strength in wireless systems. It is usually measured in Decibel relative to Milliwatt (dBm). The higher the RSSI value, the stronger the corresponding signal [24]. In theory, RSSI measurements can range from 0 dBm to -100 dBm. However, the limits are rarely reached [25]. 2.10 gives a rough estimate of what the respective values indicate about signal strength.

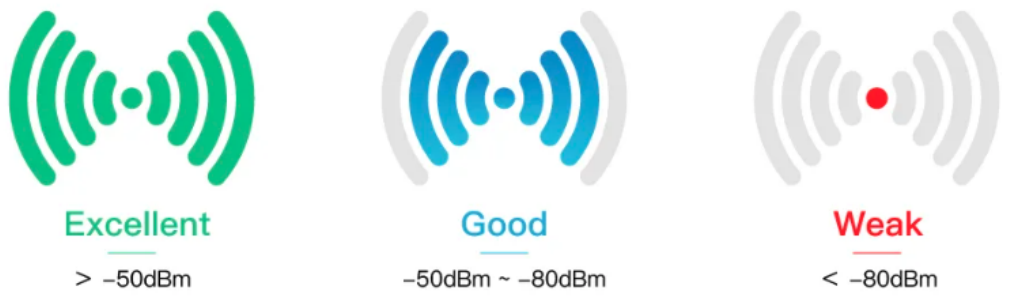


Figure 2.10: Overview RSSI mapping on signal strength [25].

For BLE, the RSSI value is important. RSSI measurements on periodic advertising channels are particularly important for localization, as mobile devices can scan these channels without establishing a connection. Due to the lower transmission power of BLE compared to classic Bluetooth, RSSI values are generally weaker at the same distance [25].

2.1.2.1 RSSI based Distance Estimation

An important part of this thesis deals with the estimation of proximity based on the RSSI value. However, there are other methods to estimate this distance instead of the

ML approach chosen in this thesis. A more widely used approach in this context is the Log-Distance Path Loss Model.

$$d = 10^{\left(\frac{A - RSSI}{10 * \beta}\right)}$$

d : Distance between peripheral and central node
A: RSSI value at reference distance (1m)
β: Propagation constant

Figure 2.11: Log-Distance Path Loss Model adapted from [26].

This model leads to the equation shown in 2.11. This allows the proximity to be estimated from the RSSI value. However, the formula is based on two constants. One is an RSSI reference value, typically measured at a distance of one meter. The other is a propagation constant that varies according to the environment. A lookup table can be seen in 2.2 containing the values for the propagation constant corresponding to different environments [26].

Environment	**Path Loss Exponent**
Free-space	2
Urban area cellular radio	2.7 – 3.5
Shadowed urban cellular radio	3 – 5
In building Line Of Sight (LOS)	1.6 – 1.8
Obstructed in building	4 – 6
Obstructed in factory	2 – 3

Table 2.2: Path loss exponent for different environments, adapted from [26].

The Log-Distance Path Loss Model is particularly useful for indoor positioning applications. This is the case because the environment is stationary, and both constants can be set for the specific environment. However, the context of BLE-based tracker detection is more dynamic. This leads to problems if the formula was used in HomeScout during run-time. Finding the RSSI reference value as well as determining the propagation constant at run-time is very difficult, as the mobile phone can move constantly, making it difficult to adjust the formula on the run.

2.1.3 Crowd Sourced Offline Finding Networks

This subsection of the thesis explains how a COFN works. The theory is illustrated using Apple's specific implementation. Apple has named their implementation the Find My Network [27]. According to Apple, the Find My Network contains hundreds of millions of devices as of 2021. This includes mobile phones, laptops, AirTags and also third-party devices [8].

2.1.3.1 Introduction

The basic idea behind COFN is that so-called finder devices can detect the presence of lost offline devices in proximity. This is done using BLE. The finder device then reports the exact location of itself, which is also the approximate location of the lost and offline device, to the owner via the internet. 2.12 shows how this works from a high level [28].

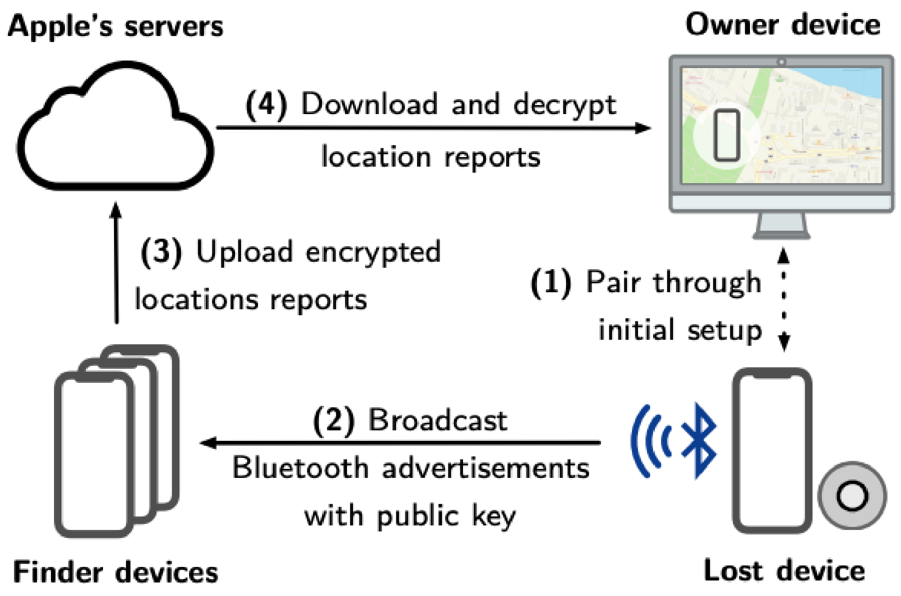

Apple's servers
Owner device
(4) Download and decrypt location reports
(3) Upload encrypted locations reports
(1) Pair through initial setup
(2) Broadcast Bluetooth advertisements with public key
Finder devices
Lost device

Figure 2.12: COFN technology from a high level [28].

Owner Device

Owner devices are devices that share an Apple ID. These devices can use the Find My application on iOS and macOS to access each other's location [28].

In the example of the AirTag, the owner device would be both the iPhone that initialized the AirTag and all other devices that share the Apple ID of this iPhone.

Lost Device

Lost devices are devices within the Find My Network that have started to broadcast BLE advertisements containing a public key. These advertisements are intended to be discovered by nearby devices. In the case of devices that can access the Internet, such as an iPhone or MacBook, the device is considered to be in the lost state when it loses its connection to the Internet. In the case of BLE-based trackers within the Find My network, a tracker is considered to be in a lost state when it loses the BLE connection to the owner's device [28].

Using the AirTag as an example, this would be the case if someone forgets a key paired with an AirTag at their workplace. Due to the interrupted BLE connectivity to the owner iPhone, the AirTag would start broadcasting BLE advertisement packets as a result of the transition into the lost state.

Finder Device

Finder devices are the heart of COFNs. These devices are able to receive the BLE advertisements, create an end-to-end encrypted location report and upload them to Apple's servers [28].

To stay with the example, the key forgotten at the workplace could advertise a BLE packet to an employee who also has an iPhone. This iPhone then uploads its location to the Apple servers.

Apple's Server

Apple's servers store location reports that are transmitted by Finder devices. Owner devices can retrieve these reports and decrypt them locally [28].

In the example, the worker who forgot the key could notice it at the front door, open the Find My app with the iPhone, and access the location of the keys. He would then realize that he needs to return to the office.

2.2 Related Work

2.2.1 RSSI Data Collection

Within the work of [29] a solid foundation for further research in the form of a labeled dataset with RSSI values containing 15.000 entries was created. Thereby 150 testing points and 15 fixed anchor nodes with a total of 11 mobile devices emitting BLE packets were deployed in a school building. By also providing the coordinates of the anchor nodes the set can be used for both fingerprint based and model based localization algorithms.

Similarly [30] collected a dataset of labeled RSSI data. They focused on a more harsh industrial environment, a harbor, with high humidity and other challenging conditions. As an underlying communication layer Low Power Wide Area Network (LPWAN) was used.

As part of the work of [31], the researchers gathered an RSSI dataset [32]. They used 13 iBeacons and an iPhone 6s as receiver. The 13 iBeacons were attached to the ceiling of the Waldo Library at Western Michigan University. They then divided the first floor of the library into a grid of 10x10 square foot squares. From these fields, both a labeled data set with 820 data points for training and a data set with 600 data points for testing were created using a self-developed application.

Paper	**Focus**	**Samples**	**Wireless technology**
[29]	RSSI dataset for improved indoor localisation.	15'000	BLE
[30]	RSSI dataset collected in a harsh harbor environment.	1'500	Long Range Wide Area Network (LoRaWAN)
[31], [32]	Improving indoor localization accuracy using RSSI data for smart city services.	1'420 labeled, 5'191 unlabled	BLE
This thesis	Collect data in different settings in order to investigate the influence of different features on accuracy.	13'353	BLE

Table 2.3: Summarizing RSSI data collection literature.

2.2.2 RSSI Data Processing Techniques

In the article [33] published in 2023, the researchers used the data set mentioned above [32] as the basis for their work. The focus lies on improving accuracy of RSSI-based indoor localization. They applied the weighted least squares method to the dataset and filtered it using a moving average filter. The use of the weighted least squares method solves the non-linearity problem of RSSI-based indoor localization and thus improves precision.

The work of [34] also points out that filtering techniques and linearization methods should be applied to an RSSI dataset before training machine learning models to counteract the strong deviations in the RSSI values.

This is also in line with other research. Smoothing out RSSI data with techniques like the Kalman filter [35, 36, 37] or the Mean and Median filter [38] is crucial before further using the data in order to achieve high accuracy in the following steps.

Paper	**Focus**	**RSSI Processing Techniques**
[33]	Improving RSSI-based indoor localization using ML.	Moving average filters, RSSI Linearization with weighted least-squares method.
[34]	Improving RSSI-based indoor localization using ML.	Pseudo-Linear solution (PSL).
[35, 36, 37]	Improving indoor localization using filtered RSSI.	Kalman filter
[38]	Improving RSSI-based indoor positioning.	Mean and Median filter.
This thesis	Proximity predictions based on RSSI values.	Investigate whether Processing techniques like the moving average filter might enhance the accuracy of the models.

Table 2.4: Summarizing RSSI processing techniques.

2.2.3 Machine Learning for Shielding and Pattern Identification

In the context of [39] it is mentioned that the use of machine learning approaches such as K-Nearest Neighbors (KNN), Support Vector Machines (SVM), Decision Trees, and Neural Networks can help address the lack of accuracy in RSSI based tracking. However, these solutions come with their own challenges, such as computational power and the required amount of data for such models.

After the mentioned processing [33] used the adapted RSSI dataset, trained and analyzed three machine learning models: KNN, SVM and Feed-Forward Neural Network (FFNN). Thereby, KNN achieved the highest accuracy of 85 %.

[40] have conducted research in the field of proximity analysis of BLE devices based on machine learning. They have chosen a random forest (RF) approach to estimate the distance between two devices. Centimeter-accurate Ultra Wide Band (UWB) ranging radios data, was used as ground truth for the training and test data set of the model.

Within the work of [41], the researcher analyzed proximity predictions based on BLE RSSI values together with a Gradient Boosted Machines (GBM) learning classification algorithm. The GBM model classified with a 92.85 % confidence. The context of the paper is the comparison of technologies used to analyze social distancing during the pandemic.

As part of [42], research was carried out in the area of locating goods in warehouses. BLE beacons and RSSI data were used to localize resources using a SVM approach.

Based on the literature review, a combination of RSSI data filtering together with a KNN, RF, Decision Tree or SVM model is best suited for the problem. This approach seems to be promising and suitable for predicting the proximity of a BLE-emitting AirTag.

Paper	**ML-algorithms**	**Accuracy**	**Application**
[33]	KNN, SVM, FFNN	85% (KNN), 84% (SVM), 76% (FFNN)	Improving indoor localization using RSSI
[40]	RF with 100 decision trees	Centimeter Level accuracy (90th percentile)	Proximity detection, contact tracing, social distancing using BLE/UWB
[41]	GBM	92.85 %	Comparison of social distancing classification models during the pandemic.
[42]	SVM	Up to 1.4m	Locating goods inside a warehouse.
This thesis	RF, KNN, Decision Tree, SVM	Aim for 90 %	Estimating the proximity of an AirTag towards the central device.

Table 2.5: Summarizing RSSI based ML-models literature.

2.2.4 RSSI Data in Tracking Systems

As discussed in [43], compared to the Global Positioning System (GPS), RSSI-based localization systems can have advantages in terms of accuracy in indoor environments. Furthermore, such systems effectively reduce the deployment and energy consumption costs during the creation and maintenance phase, respectively [44].

In the work of [45], the researchers developed an indoor tracking system based on RSSI values in a hospital corridor environment. They used two references and one moving target node. They applied a log-distance path-loss equation to the RSSI values in combination with several position estimation methods and an exponential weighted moving average filter to smooth out variations caused by fluctuating RSSI values. By doing so, they were able to accurately determine the position of the target node in the hospital corridor.

Similarly, [46] developed an indoor tracking system based on RSSI values. But they focused on a more harsh environment. Specifically, they have developed a system to track the position of workers in underground mines. This environment is particularly prone to fluctuations in RSSI values, at the same time accurate localization is extremely important for worker safety. Using a hybrid RSSI-based fingerprint algorithm that involves reference nodes together with dead reckoning and statistical averaging, they were able to track the nodes in the mine with an accuracy of 3.13 meters.

The paper [47] explores the use of RSSI data from multiple BLE beacons in an outdoor localization setting. The particular aim of the research is to help blind and visually impaired people cross complex intersections in large cities safely. The RSSI data was smoothed using a moving averaging filter. As part of a feasibility study, a KNN model was trained with the data and was able to achieve an accuracy of 99.8 % during classification.

2.2.5 Android Integrations of the AirTag technology

When Apple launched AirTags, BLE-based trackers, in 2021, security concerns were immediately raised, in particular, the lack of protection against stalking of Android users raised major questions [14]. In response, Apple released the 'Tracker Detect' app for the Android market. However, the main criticism of this app is that Android users have to actively search for potential threats [15], [16]. To close this gap, several alternatives to the Tracker Detect app have been developed.

One solution is the open source application AirGuard, developed as part of the work of [48]. Apple's iOS internal tracking protection feature was reverse engineered for the development of the AirGuard application, which enables passive scanning and detection of potential threats.

Another Android application, 'BLE-Doubt', automatically scans for BLE-based tracker beacons and can detect non-Apple trackers. This application uses a baseline of time and distance and their hybrid combination together with an improved topological classifier to avoid false positives [49].

Additionally, the HomeScout application, developed as part of the work of [15], uses parameters such as time, distance and the occurrence of BLE devices for detection. HomeScout is not limited to BLE-based trackers, but analyzes all BLE-based devices, including modern cell phones, laptops and more, for misuse.

Chapter 3

Design

This chapter provides an overview of the design process and methodologies used to generate a labeled RSSI dataset with AirTags, how it was used to train different machine learning models, and to evaluate their performance. Finally, it discusses the integration into HomeScout.

3.1 Data Collection

This subsection explains the design decisions made to collect the data used in this thesis. It provides information on both the hardware and software used. It also explains the data collection pipeline and the design choices made in relation to the methodology of the data collection process.

3.1.1 Hardware

3.1.1.1 nRF Board

The Nordic Semiconductor nRF52840 single-board Development Kit (DK) was used to capture and analyze BLE packets in this thesis. The DK has been provided by the Communication Systems Group (CSG). This DK is well suited for capturing packets emitted by AirTags. It supports Bluetooth 5.3 multi-protocol radio [50].

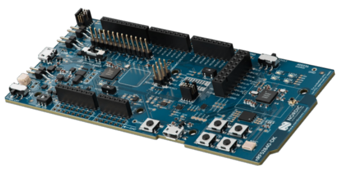


Figure 3.1: nRF52840 DK [50].

3.1.1.2 AirTags

The most important part of the hardware used in this thesis are the AirTags themselves. A total of 11 different AirTags have been provided by the CSG and were used to collect the data.

Figure 3.2: AirTag [4].

At the start of this work, the AirTags provided were analyzed. An overview of this analysis can be seen in the table 3.1. The ID column contains the value of a physical tag attached to each AirTag by the CSG. The information in the Serial Number and Owner columns could be determined by holding the AirTags with the white side against the top of an NFC-capable iPhone. An initial BLE packet analysis revealed that 5 of the 11 AirTags were not actually emitting any packets. After replacing the batteries and pairing the unpaired ones, this was resolved. All AirTags were in the lost state during the experiments, as they would be if they were misused as a stalking device and slipped to someone else. In this case, they would not be in proximity to the owner.

ID	Serial Number	Owner	Emitting Packets
CSG A1	HGJGMØUPPØGV	N/A	NO
CSG A3	HGJGMSSBPØGV	N/A	YES
CSG A4	HGJGMRWHPØGV	N/A	NO
CSG A5	HGHH3254PØGV	* * ** *4 03	YES
CSG A6	HGHH37XQPØGV	N/A	NO
CSG A7	HGHH37Y8PØGV	* * *1 28	YES
CSG A8	HGHH37RZPØGV	N/A	YES
CSG A9	HGQGMFSBPØGV	* * *9 32	YES
CSG A10	HGQGMHEMPØGV	* * *9 32	YES
CSG A11	HGQGMF7MPØGV	N/A	NO
CSG A12	HGQGMG3CPØGV	N/A	NO

Table 3.1: Summary of initial Device Packet Emission Status.

3.1.1.3 Battery Tester

The Kraftmax XT1 [51] was used to investigate the influence of the battery on RSSI values. This device was chosen because it can measure not only the voltage but also the internal resistance and a general capacity metric in percent.

Figure 3.3: Kraftmax XT1 [51].

3.1.2 Software

3.1.2.1 SEGGER Embedded Studio

SEGGER Embedded Studio version 8.14a was used as the IDE for programming the DK. This IDE facilitated the embedded development. Programs could be easily deployed onto the nRF DK during this process.

3.1.2.2 Data Collection

This subsection describes the development and architecture of the application designed for the DK to capture BLE packets emitted by AirTags along with the associated RSSI values.

The project[1] from this blog [52] was chosen as the baseline for the passive scanning in this paper. The author of the blog, in turn, took the nRF5 SDK BLE Blinky Application example [53] as a starting point. He modified the functionality by adding passive BLE scanning to the application. This effectively allows for the scanning of BLE packets and extracting the RSSI value along with the MAC address.

[1]The source code can be found here:
`https://github.com/jimmywong2003/nrf5-ble-scan-filter-example`

This was a very good starting point. But in order to start collecting data the following points had to be implemented:

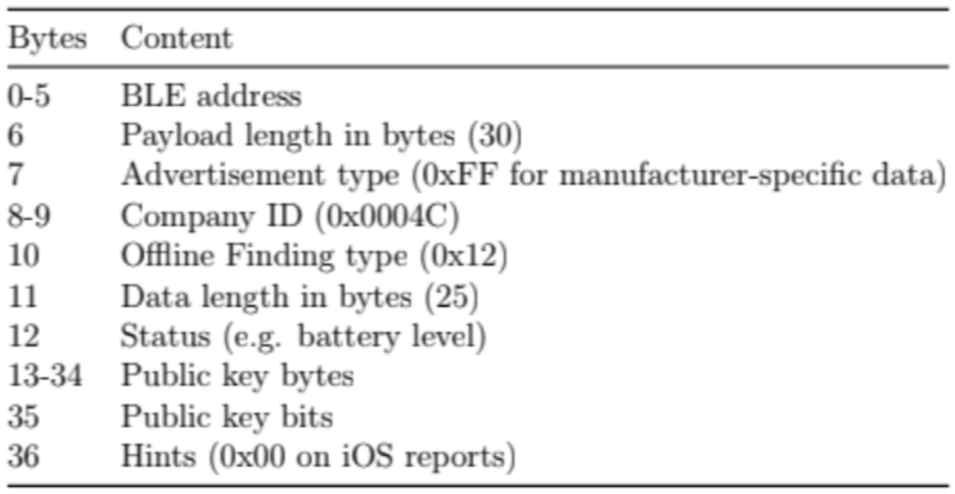
- Having a filter so that only MAC addresses and RSSI values corresponding to AirTag emitted packets are considered by the application.

- The ability to stop scanning to physically increase the distance between the DK and the AirTags, together with the need for a mechanism to infer the corresponding distance for each data point in post-processing to label the data.

AirTag filter

The theoretical basis for the AirTag filter was given by [20]. The authors in turn based their implementation on the work of [48] and the corresponding source code.

The following explanations of the filter in use are based on [20]. Deviating sources are cited accordingly.



Bytes	Content
0-5	BLE address
6	Payload length in bytes (30)
7	Advertisement type (0xFF for manufacturer-specific data)
8-9	Company ID (0x0004C)
10	Offline Finding type (0x12)
11	Data length in bytes (25)
12	Status (e.g. battery level)
13-34	Public key bytes
35	Public key bits
36	Hints (0x00 on iOS reports)

Figure 3.4: Advertisement format used by Apple [20].

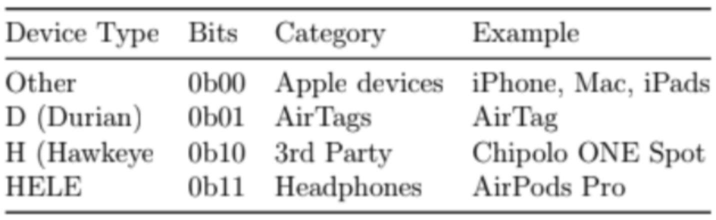
To identify packets from AirTags, it is important to filter for manufacturer-specific data containing Apple's Company ID. Figure 3.4 shows that this can be inferred from bytes 8 to 9 of the advertisement packet. A simple if statement as shown in 3.5 is used to filter for packets that meet this requirement. Additionally, byte number 10 (Offline Finding type) has been added as a check to the filter.

```
    // ... Further scanning logic

    uint16_t company_identifier = uint16_decode(&p_adv_report->data.p_data[data_offset]);

    if (company_identifier == 0x004C)
    {
        if (p_adv_report->data.p_data[data_offset + 2] == 0x12)
        {
            // ... Further scanning logic
        }

    }
            // ... Further scanning logic
```

Figure 3.5: First layer of the filtering applied.

However, since this condition is not only fulfilled by BLE packets of AirTags, but by all Apple devices that communicate in the Find My network (Apple's COFN), the logic must be extended.

To distinguish between Apple devices, the status byte at the 12th position of the advertisement packet is looked at. It is being evaluated with a bit-wise AND operation using the value 0x30 (0011 0000 in binary). From this intermediate result, only the first four bits seen from the left are considered as the flag, which is sufficient to classify the corresponding device within the Find My network. Since the first 2 bits seen from left are always 0 due to the AND operation, the 3rd and 4th bits are actually decisive enough as a flag.



Device Type	Bits	Category	Example
Other	0b00	Apple devices	iPhone, Mac, iPads
D (Durian)	0b01	AirTags	AirTag
H (Hawkeye	0b10	3rd Party	Chipolo ONE Spot
HELE	0b11	Headphones	AirPods Pro

Figure 3.6: Lookup table for classification based on the computed flag bits [20].

A theoretical example calculation is shown in 3.7. The lookup table for classifying Apple devices based on the computed flag is shown in 3.6.

Given the following example status byte:

$$\text{status\_byte} = 0b0001\ 0011$$

Perform the AND operation:

$$\text{status\_byte}\ \&\ 0x30 = 0b0001\ 0011\ \&\ 0b0011\ 0000 = 0b0001\ 0000$$

By only considering the bits number three and four seen from the left, this gives us the flag bits:

$$\text{flag\_bits} = 0b01$$

According to the flag classification table, the packet corresponding to this status byte is classified as coming from an AirTag.

Figure 3.7: Sample classification of a status byte.

This led to the implementation used in this thesis and shown in 3.8.

```
    // Computing flag bits
    uint8_t status_byte = p_adv_report->data.p_data[data_offset + 4];
    uint8_t flag_bits = status_byte & 0x30;

    if (flag_bits == 0x10)
    {
        // ... AirTag identified
        // ... Further scanning logic
    }
```

Figure 3.8: Second layer of the filtering applied.

Distance Feature

In order to have the ability to stop scanning during the time used to increase the physical distance between the DK and the AirTags, the following idea was implemented. Two different global variables called m_logging_is_running and current_distance_index were introduced. Then the top left button of the DK was programmed to toggle the boolean variable m_logging_is_running to true and the top right button to false, which switches the scanning on and off and also gradually increases the variable current_distance_index in order to be able to re-create the distance for each measurement point in post-processing.

Conclusion

The described development process led to an application[2] that can passively scan advertisement packets emitted by AirTags and log both the MAC address and RSSI value. It is also possible to start and stop the scanning while keeping track of a distance index that will be used later to derive the distance for each measurement. Thus, forming the basis for collecting the dataset. The application was also used for the insights shown in Section 3.1.1.2. By approaching the AirTags one by one while keeping the others away from the DK, it was possible to determine whether each AirTag was transmitting advertising packets. In this way, the application's filtering mechanism was both confirmed and used to evaluate which AirTags were ready to be used in the data collection process at the beginning of the work.

[2]`https://github.com/samuelfrnk/BA_Samuel/blob/main/ble_app_uart_adv_scan/main.c`

3.1.3 Data Pipeline

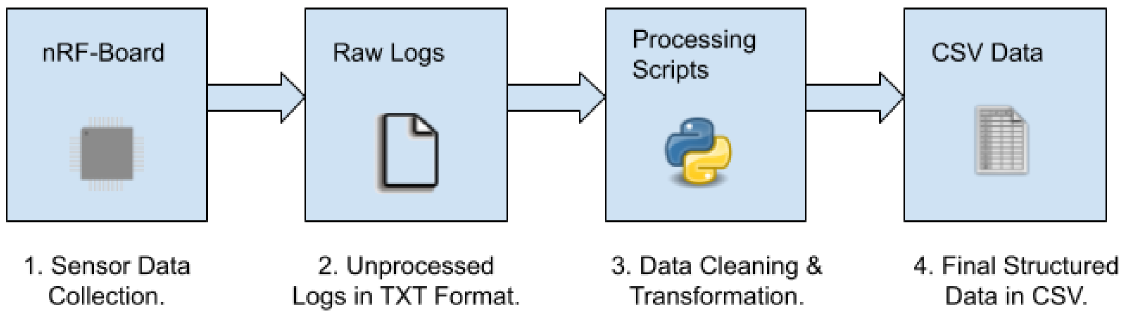

nRF-Board
Raw Logs
Processing Scripts
CSV Data
1. Sensor Data Collection.
2. Unprocessed Logs in TXT Format.
3. Data Cleaning & Transformation.
4. Final Structured Data in CSV.

Figure 3.9: Data Pipeline illustrated.

The pipeline shown in Figure 3.9 illustrates the process of extracting structured data from the scanning application. The first step is the sensor data collection, which was discussed in detail in Section 3.1.2.2. This process results in raw log text files[3]. A Python script [4] was developed and used to reconstruct the timestamp and distance information for each data point, and to produce structured data in CSV format leading to the final structure shown in the table 3.2.

RSSI-Value	MAC-Address	Timestamp	Distance
-24	D1:8A:6B:17:30:48	2024-10-30 14:14:16	0.0
-19	CA:2B:3D:A7:0C:C9	2024-10-30 14:14:16	0.0
-24	ED:7C:A0:C9:37:39	2024-10-30 14:14:16	0.0

Table 3.2: Sample structure of the CSV files returned by the script.

3.1.4 Methodology

The context of the data collection process used in this work was controlled and experimental. During all measurements the AirTags were placed at a fixed distance from the DK to ensure precise labelling. As the RSSI data decays rapidly with increasing distance, the distance primarily focused within the context of this work is 0 to 2 meters, using a granularity of 10 cm. In order to provide a controlled setting, the previously described battery tester was used. The goal here lies in ensuring consistency by evaluating the influence of battery drain on the RSSI values. Insights will further be drawn in the Results and Evaluation section.

[3]Raw log file examples can be found here: `https://github.com/samuelfrnk/BA_Samuel/tree/main/Experiments/Results/Raw_Data`

[4]The source code can be found here: `https://github.com/samuelfrnk/BA_Samuel/blob/main/Experiments/Evaluation/Conversion_Skripts/CSV_Conversion_and_overview.py`

In addition, all indoor data measurements were performed at the same location within the university building in Oerlikon. This consistency in location was intended to reduce environmental variations.

3.1.5 Challenges

When designing the data collection process, several steps were taken to ensure reproducibility and control of the experimental environment. One of the key challenges in working with RSSI data is its inherent sensitivity to interference and fluctuation. These issues were addressed through the use of controlled distances, the battery meter, consistent location and consideration of data smoothing techniques that can minimize noise and improve accuracy.

3.2 Model Selection

This section provides an overview over the design decisions taken while choosing models which are to be considered.

3.2.1 Decision Tree

A decision tree model was trained and evaluated within this thesis. Whilst there was no literature from the review that used this model, it is still interesting to consider. In particular to compare the performance to RF where several Decision Trees are used.

3.2.2 Random Forest

RF was chosen as the second model in this thesis. [40] was able to achieve centimeter-level accuracy in proximity prediction using RSSI data and RF with 100 decision trees.

In addition, RF is less prone to over-fitting compared with a single decision tree. Thus, using RF to perform proximity prediction is promising.

3.2.3 Support Vector Machine

A third model considered in this work is SVM. [33] achieved a quite high accuracy of 84 % using SVM paired with RSSI data.

3.2.4 K-Nearest Neighbors

KNN is the final model considered in this thesis. The reason for this decision is that within the work of [33] KNN achieved the highest accuracy of 85% and is therefore promising.

3.2.5 Overview

3.3 shows an overview table of the performance of the models from the existing literature measured in accuracy. These values are subsequently compared with the custom performance score in chapter 4.

Model	**Reference**	**Reported Performance**
Decision Tree (DT)	N/A	N/A
Random Forest (RF)	[40]	Centimeter-level accuracy
Support Vector Machine (SVM)	[33]	84% accuracy
K-Nearest Neighbors (KNN)	[33]	85% accuracy

Table 3.3: Overview of considered models, references, and their performance based on literature.

3.2.6 Classification vs Regression

Predicting a distance intuitively appears as a regression problem due to the continuous nature of distance. However, in the context of predicting the proximity of AirTags in this work, the problem is considered as a classification problem. The underlying assumption is that in order to develop a shielding mechanism against malicious AirTags, a classification into discrete distance buckets will simplify the problem. Thus, enhancing the performance of the models. The size of the buckets will be further elaborated in the Results and Evaluation section.

3.3 Design of Evaluation Approach

This section captures design decisions related to the evaluation of the ML models considered. It discusses approaches and metrics that help to decide which concrete model is best suited for the implementation phase. The theoretical basis for this section is [54], any derivative sources are cited accordingly.

3.3.1 Performance Metrics

Cross-Validated Accuracy

A first indication of how well a particular model has performed is the cross-validated accuracy score, which involves splitting the data into K folds, in this case 4. The average across all folds gives insight into how many predictions overall were correct. There are certain limitations of this metric, especially if the classes are not well balanced. However, since extensive balancing methods are applied to the dataset, this metric still gives an important insight into the performance.

Precision, Recall & F1-Score

Precision in our context gives an indication of how likely a classification is to be true, given that an RSSI record has been classified by the model as coming from a close proximity AirTag, and thus potentially malicious. Recall, on the other hand, gives an indication on how many of the entries that come from nearby AirTags are successfully identified as such.

An important third metric in this context is the F1-score. Since there is usually a trade-off between recall and precision, the F1-score combines both with a harmonic mean. As both false positive and false negative classifications are bad in the context of malicious AirTags identification, the F1-score can successfully capture this aspect of performance with respect to the mentioned trade-off.

Confusion Matrix

To get a performance overview of each model, confusion matrices are used. A confusion matrix compares the model classification with the actual classes. The columns are usually the predicted classifications, while the rows are the actual classes. In this way, the number of true negatives and positives as well as false negatives and false positives can be visualized appealingly.

Overfitting Evaluation

To determine how well the model generalizes to unseen data, the difference between training accuracy and testing accuracy is considered. This provides insight into whether a model is performing overfitting.

3.3.2 Selection Criteria

To find the best performing model, a custom performance metric is introduced. It combines the above metrics into a single score that is used to compare all models. The calculation can be seen in 3.10. The difference between training and test accuracy has to be normalized. The F1-scores and the cross-validated accuracies already range from 0 to 1.

The Performance Score is calculated as follows:

$$\text{Performance\_Score} = w_1 \cdot F + w_2 \cdot A - w_3 \cdot P$$

where:

$$\begin{aligned} F & \text{ is the F1-score} \\ A & \text{ is the cross-validated accuracy} \\ P & \text{ is the normalized overfitting penalty} \\ w_1, w_2, w_3 & \text{ are weights assigned based on the importance of each metric.} \end{aligned}$$

Figure 3.10: Custom Performance Score.

The selected weighting approach assigns 0.5 to cross-validated accuracy and F1-score and 1.0 to the overfitting penalty. This ensures that accuracy and F1-score are regarded equally, focusing on both general correctness while also counteracting accuracy flaws. The overfitting penalty is weighted with one in order to discourage models that memorize patterns from the training set and cannot generalize the classification. As a result, the overall strategy promotes the development of robust and generalizable models.

3.4 Shielding Design for HomeScout

As part of the work of [20], HomeScout, an Android application written in Kotlin that can passively scan and identify possible maliciously deployed BLE devices, was developed. The architecture of HomeScout as well as the underlying classification logic and the integration of the evaluated ML model is discussed in this section. [20] is used as the source in this section, and all deviating sources are cited accordingly.

3.4.1 HomeScout Architecture

As illustrated in 3.11, the architecture of the HomeScout application consists of three high-level layers: the User Interface (UI), the service layer, and the data layer.

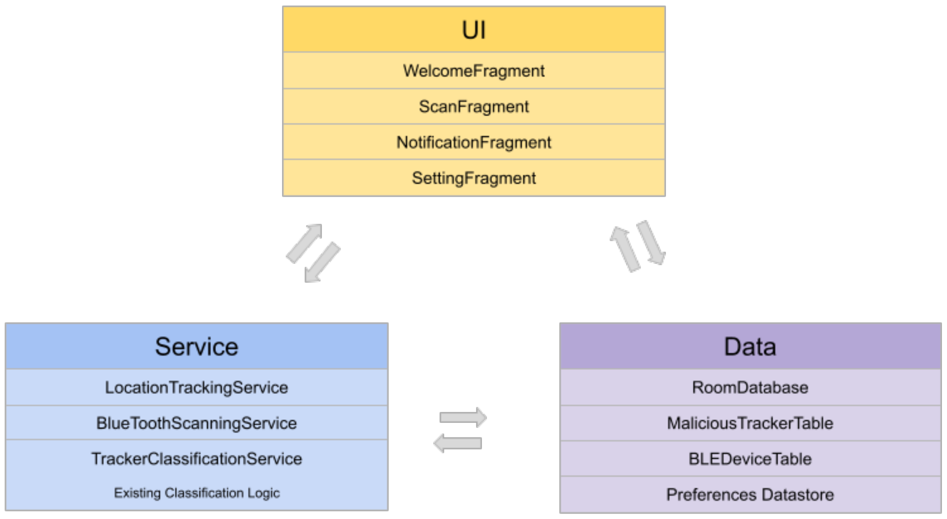


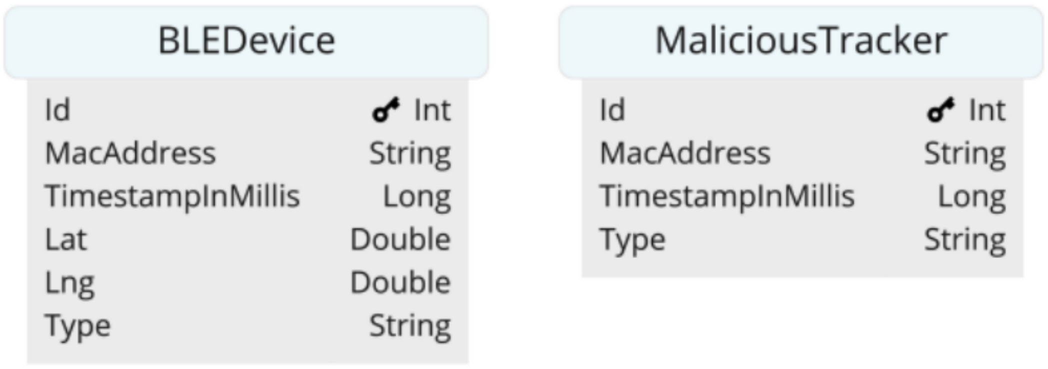
Figure 3.11: Overview of the HomeScout architecture including the three layers and the corresponding components.

3.4.1.1 User Interface Layer

The UI layer of HomeScout is responsible for interacting with the user. There are various fragments. The WelcomeFragment greets the user and clarifies the required permissions, including location access. The ScanFragment allows for real-time scanning for nearby BLE devices and performs their type classification. In the NotificationFragment, HomeScout gives the user an overview of all devices that have been classified as malicious, including their MAC address and type classification. In the SettingFragment, the user can manually set the desired thresholds for time, occurrences and distance.

3.4.1.2 Data Layer

Important data storages in the architecture of the application are the BLEDevice table and the MaliciousTracker table. The first contains all scan results, and the second contains all devices recognized as malicious by HomeScout. The schemes corresponding to these tables are shown in figure 3.12.



BLEDevice	
Id	🔑 Int
MacAddress	String
TimestampInMillis	Long
Lat	Double
Lng	Double
Type	String

MaliciousTracker	
Id	🔑 Int
MacAddress	String
TimestampInMillis	Long
Type	String

Figure 3.12: Schemes of the RoomDatabase from [20].

3.4.1.3 Service Layer

There are three different services within the HomeScout application. The TrackerClassificationService is the most important one for this work. Its main task is to classify entries from the BLEDevice data table into the MaliciousTracker data table, thereby identifying threats and alerting the user.

The classification logic runs every 30 seconds and iterates over all entries in the BLEDevice data table. All data points are grouped by MAC address and sorted by timestamps. Three different thresholds are used: time, occurrences, and distance. If the entries of a device corresponding to the MAC address in the data exceed all three thresholds, the device is classified as malicious and the user is being notified.

The distance threshold is worth discussing here. HomeScout initially focuses on the distance that the phone itself traveled while receiving BLE packets from a particular BLE device. This is done by monitoring the coordinates of the phone at the points where the BLE packets were scanned. However, this work shifts the focus to the proximity between the mobile phone and the tracking device. The underlying assumption is that if this proximity is consistently low, it indicates a malicious tracker.

3.4.2 Shielding Design

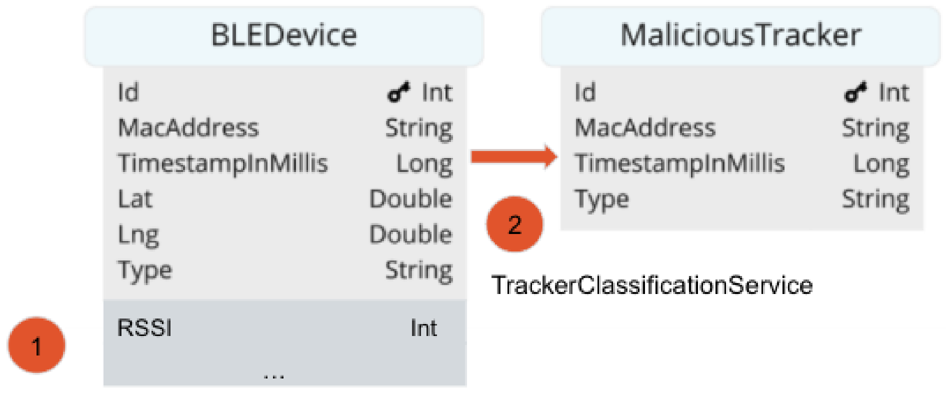


Figure 3.13: High level adjustments in the data and service layer, adapted from [20].

Overall, the implementation requires adjustments in all three layers of the application as indicated in red in 3.14. This includes the UI layer, the data layer and the service layer. The changes required for each of these layers will now be discussed as part of the design.

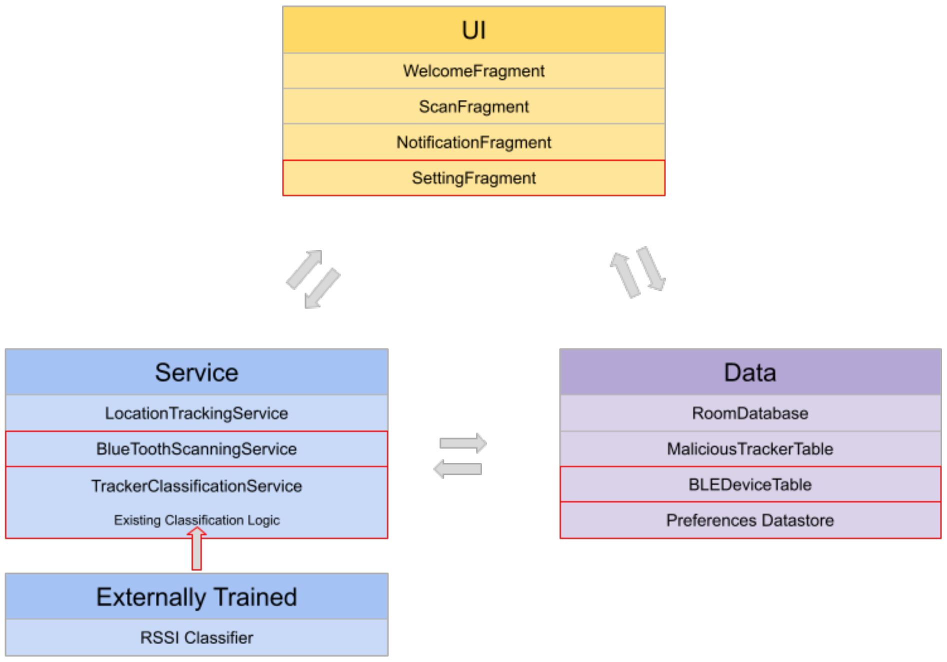


Figure 3.14: The HomeScout architecture including an overview of areas that need to be modified highlighted in red.

3.4.2.1 User Interface Layer

The required changes in the UI layer of HomeScout are rather simple. As 3.14 shows, they only affect the SettingFragment. Here a toggle must be introduced, which can activate and deactivate the RSSI shielding.

3.4.2.2 Data Layer

There are two different adjustments that must be made to integrate the shielding mechanism into the existing HomeScout logic. The first of which is that during the scanning phase, features needed by the model must be extracted in addition to existing attributes. Figure 3.13 shows with the highlighted number one, that the most important additional feature is the RSSI value. However, also features that are investigated and found to be useful for the performance of the models need to be extracted during the real-life scanning process.

The BluetoothScanningService class of the application is responsible for receiving BLE packets and extracting the relevant information for the entries in the BLEDevice data table. The application uses the BluetoothLeScanner class [5] during the scanning process. This class will then return objects of the ScanResult class [6] when it passively captures a BLE packet. As the method .getRssi() which returns an integer is part of the interface of the ScanResult class, adding the RSSI as an additional attribute for the entries within BLEDevice table is a straightforward step.

A more complicated step is to extract the additional features that are evaluated in this work. In particular, LOS / Non Line Of Sight (NLOS) and Indoor/Outdoor are challenging. Possible solutions are discussed in the Final Considerations chapter.

3.4.2.3 Service Layer

The next important point to modify is the TrackerClassificationService class. This change is displayed in 3.13 with the number two. This is the heart of the application. Adjustments here are more technical compared to the first step containing data level adjustments. The machine learning model will serve here as an addition to the existing three thresholds.

As the red arrow in 3.14 implies, an approach needs to be established that allows the model to be used within HomeScout at runtime. The models have been trained and evaluated using corresponding scikit-learn libraries [7].

A key step in integrating the evaluated model into HomeScout is Open Neural Network Exchange (ONNX). ONNX is an open-source project developed by a collaboration of Microsoft, Amazon and Meta. It simplifies the conversion of ML models from one framework

[5]https://developer.android.com/reference/android/bluetooth/le/BluetoothLeScanner

[6]https://developer.android.com/reference/android/bluetooth/le/ScanResult

[7]https://scikit-learn.org/stable/api/index.html

to another. The high-level process involves the conversion of the model into ONNX format using the Python library skl2onnx [8]. This converted file can be copied into the repository. As a next step the onnxruntime Maven dependency [9] can be used within Kotlin to perform predictions using the model at runtime [55].

3.4.3 Considerations

As explained by [39], on the one hand using RSSI data together within a machine learning approach can help counteract RSSI's inherent sensitivity to fluctuation and interference by capturing deeper patterns and thus counteracting the noise. On the other hand, machine learning itself comes with its set of challenges, such as the amount of resources and data required.

Therefore, it needs to be discussed whether the integration can provide a real benefit despite the challenges. A concrete alternative would be to use a similar threshold approach for RSSI as it is currently implemented with distance, occurrences and time in HomeScout. How this threshold could perform compared to the evaluated model and how the inherited problems with RSSI could be faced with the concrete implementation of such a fourth threshold needs to be investigated.

[8]https://github.com/onnx/sklearn-onnx

[9]https://mvnrepository.com/artifact/com.microsoft.onnxruntime/onnxruntime

Chapter 4

Results and Evaluation

This section of the thesis looks at the results of the dataset generation and the evaluation of the performance of the different models. An overview of the data is given and the performance of all models is discussed in detail. The best performing model is identified and further characterized.

4.1 Dataset

This section focuses on the dataset. First there is an overview of the dataset, then there is an analysis of the effect of the environments and the battery voltage on the RSSI values and finally a packet capture rate is calculated.

4.1.1 Final Structure

The datasets final structure that was used to train the models can be seen in 4.1.

Index	**RSSI-Value**	**MAC-Address**	**Timestamp**	**LOS**	**Indoor**	**Distance**
0	-19	F0:79:C0:40:FB:93	2024-09-12 09:45:00	1	1	0.0
1	-18	EA:25:00:4F:3C:4D	2024-09-12 09:45:00	1	1	0.0
2	-21	EF:0D:1E:F2:0C:B2	2024-09-12 09:45:01	1	1	0.0
3	-19	CB:08:AA:AA:C9:C6	2024-09-12 09:45:01	1	1	0.0
4	-19	ED:4F:A6:06:39:7A	2024-09-12 09:45:01	1	1	0.0
⋮	⋮	⋮	⋮	⋮	⋮	⋮
13352	-82	E8:9B:11:7B:D7:0F	2024-11-30 15:56:26	0	0	2.0

Table 4.1: Final Dataset structure.

4.1.2 Analysis

4.1.2.1 Overview

The final dataset[1], which was collected and later used to train and evaluate several models, consists of 13'353 individual data points. The distance column provides the label for the data. Data was collected in both indoor and outdoor environments, as well as LOS and NLOS and all four combinations of these.

The figure 4.1 illustrates the distribution of RSSI values and highlights the observed decrease in RSSI with increasing distance. What can be seen from the graphs is that in general there are a lot of RSSI measurements around -60 dBm. In addition, the RSSI values appear to decrease rapidly as the distance increases. At a certain point they level off. The error bars appear to increase with distance, which is an indication that the noise increases with distance as well.

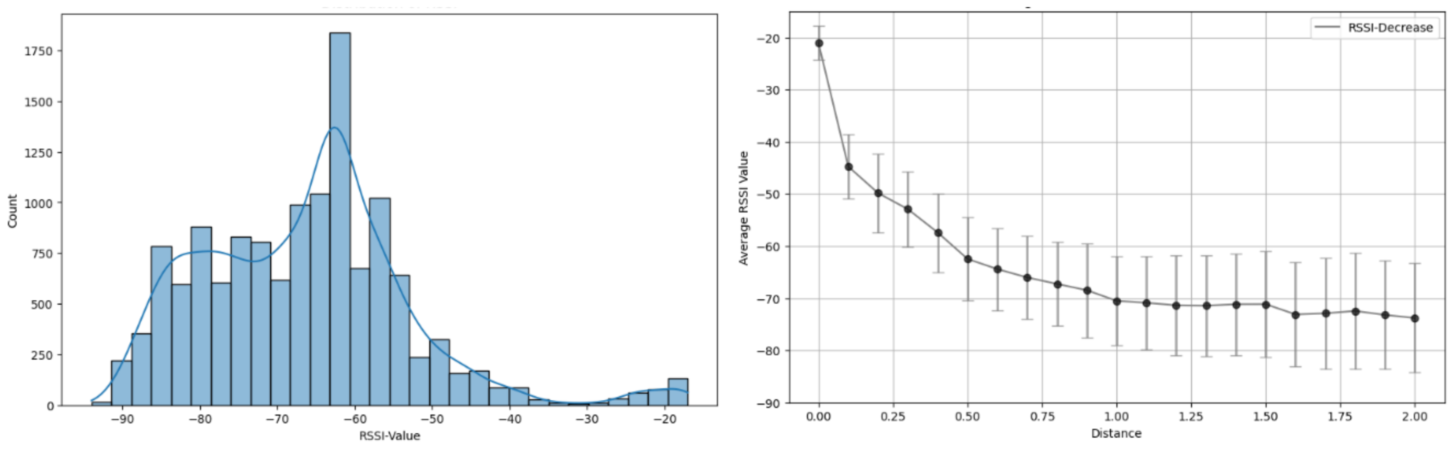


Figure 4.1: Overview Dataset; RSSI Distribution and Decay.

4.1.2.2 Environmental Effects

In order to compare the environmental influence the data points are grouped by indoor and outdoor measured and compared against each other. As Figure 4.2 shows, it is clear that RSSI data collected in the indoor environment is consistently higher than that collected in the outdoor environment. It clearly shows a direct relation between indoor and outdoor and higher and lower RSSI values.

[1]The combined and final dataset can be found here: `https://github.com/samuelfrnk/BA_Samuel/blob/main/Experiments/Results/Data_CSV/Combined_Data/combined_data.csv`

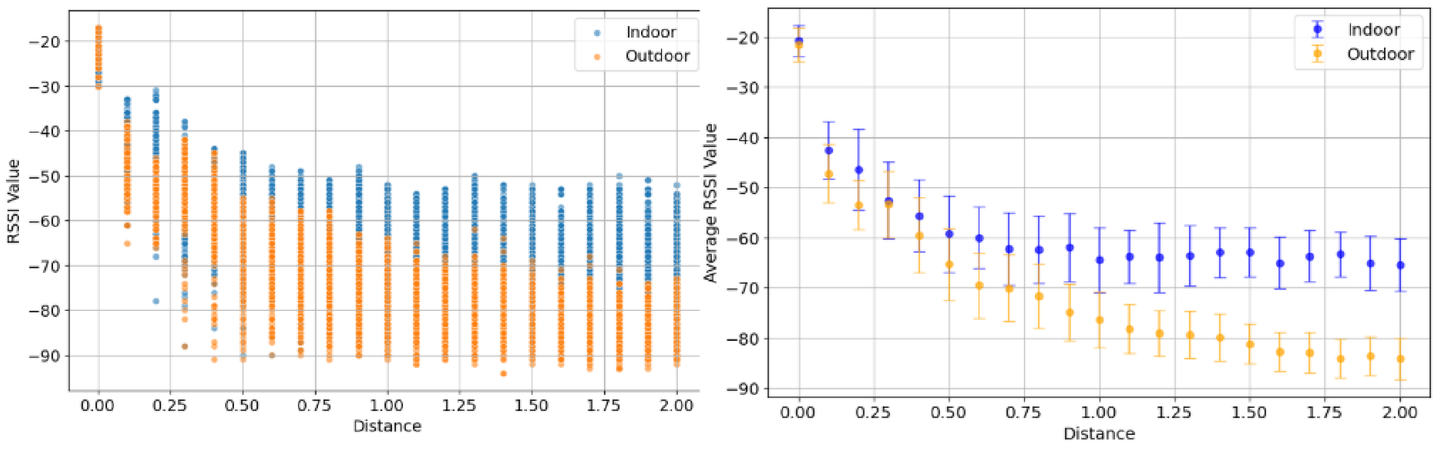


Figure 4.2: RSSI Decay grouped by Indoor (blue) and Outdoor (orange) data.

The influence of NLOS and LOS on the RSSI values is less clear. As the plots in 4.3 show, there seems to be no clear correlation between this feature and the RSSI values.

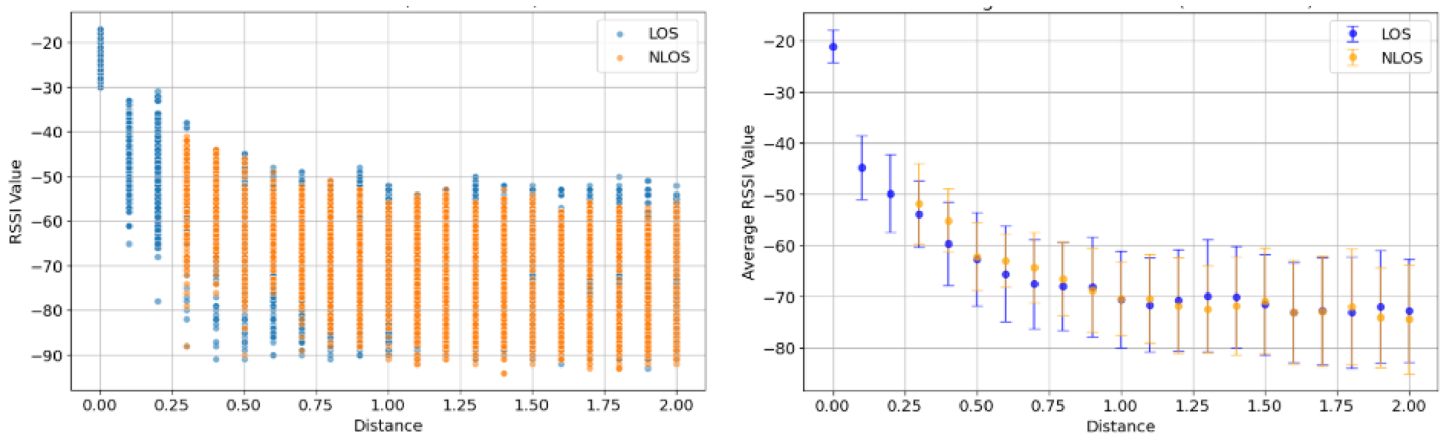


Figure 4.3: RSSI Decay grouped by LOS (blue) and NLOS (orange) data.

4.1.2.3 Battery Insights

In 4.2 the results of the measurements with the battery tester can be seen. The measurements were carried out on 16 October 2024. Of particular interest are the AirTags labeled CSG A8 and CSG A7. The battery in A8 has the lowest voltage, making it the least powerful battery. In contrast, the battery in A7 achieves the full 3 volts and is thus considered one of the most powerful batteries.

ID	Battery percentage	Battery Internal Resistance	Battery Voltage
CSG A1	95%	30 Ohm	2.99
CSG A3	95%	50 Ohm	3.00
CSG A4	95%	24 Ohm	3.00
CSG A5	95%	47 Ohm	2.98
CSG A6	95%	53 Ohm	3.00
CSG A7	95%	39 Ohm	3.00
CSG A8	40%	9 Ohm	2.69
CSG A9	80%	14 Ohm	2.82
CSG A10	80%	15 Ohm	2.84
CSG A11	95%	23 Ohm	2.93
CSG A12	90%	22 Ohm	2.92

Table 4.2: Battery Measurements with Kraftmax XT1.

Based on this, it was expected that AirTag A7 would exhibit a higher RSSI trend than AirTag A8. To test this hypothesis, both AirTags were moved close to the DK and the corresponding MAC addresses were recorded. The distance between the AirTags and the DK was then gradually increased, and in a subsequent post-processing step, the data was compared to the baseline, as shown in Figure 4.4 . Contrary to expectations, AirTag A7 consistently displayed lower RSSI values than AirTag A8. This result suggests that the power level does not influence the RSSI values, which is also an assumption within this work.

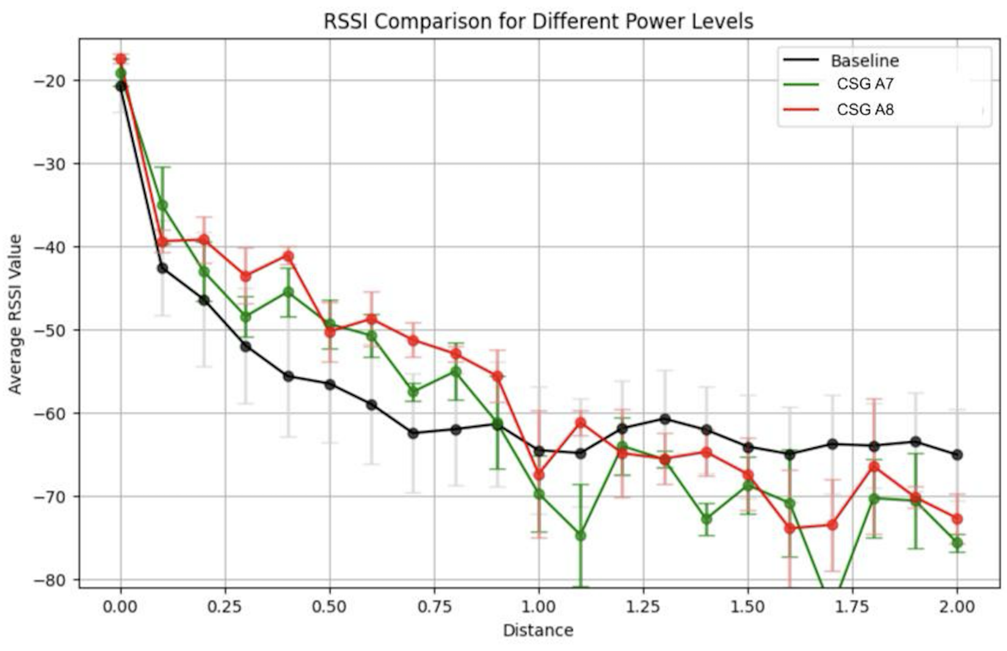


Figure 4.4: Comparison between AirTag with most powerful battery (green) and AirTag with least powerful battery (red).

4.1.3 Data Collection Evaluation

The following calculation is intended to evaluate the effectiveness of the application [2] presented in section 3.1.2.2, which was used to scan for BLE advertisement packets and collect corresponding RSSI data.

During Experiment 3, the emitted packets can be calculated as follows:

$$\text{Total\_Emitted\_Packets} = 11 \cdot 30 \cdot 37 = 12'210\,\text{packets.}$$

The scanning application captured:

$$\text{Captured\_Packets} = 5'869\,\text{packets.}$$

The capture rate is calculated as:

$$\text{Capture\_Rate} = \frac{\text{Captured\_Packets} \cdot 100}{\text{Total\_Emitted\_Packets}} = \frac{5'869 \cdot 100}{12'210} \approx 48\%.$$

Figure 4.5: Packet Capture Statistics inferred from Experiment 3.

According to [20], devices in Apple's Find My network emit a BLE advertisement packet every 2 seconds as soon as they are in the lost state. This information paired with the metadata from experiment 3[3] in which 11 AirTags were scanned for 37 minutes, the capture rate can be calculated. The capture rate of the application was found to be around 50%. Additional tweaking of the scan interval and scan window could improve this rate. However, in the context of this work and the relevant models being considered, this rate was sufficient. If the performance of deep learning models were to be analyzed, it would make sense to perform this tweaking and use more than 11 AirTags, as such models require more data.

4.2 Iterative Refinements

4.2.1 Class Consolidation Strategy

The model evaluation of [56] was used as a starting point for training and evaluating initial models. Within this work, an RSSI data set was collected and models were trained to predict distances based on RSSI values. The associated Jupyter notebook files were taken

[2]The application's source code can be found here: `https://github.com/samuelfrnk/BA_Samuel/blob/main/ble_app_uart_adv_scan/main.c`

[3]The metadata of Experiment 3 can be found: `https://github.com/samuelfrnk/BA_Samuel/blob/main/Experiments/Results/Overview_Data/Experiment_3.csv`

over and a first evaluation of the performance of the models with the newly collected data set was carried out.

[56] initially categorized the data into multiple discrete classes using the following binning strategy:

$$[0,1),\ [1,2),\ [2,4),\ [4,10) \text{ and } [10,\infty)$$

This multi-class approach, however, yielded poor predictive accuracy when applied to the newly collected dataset (see Figure 4.6). To address this issue, the classification problem was simplified to a binary classification. Specifically, the classes were merged into two categories using bins:

$$[0,0.5) \quad \text{and} \quad [0.5,\infty)$$

Adopting this binary classification modeling substantially improved both the overall accuracy and the weighted F1 score across all evaluated models (see Figure 4.6).

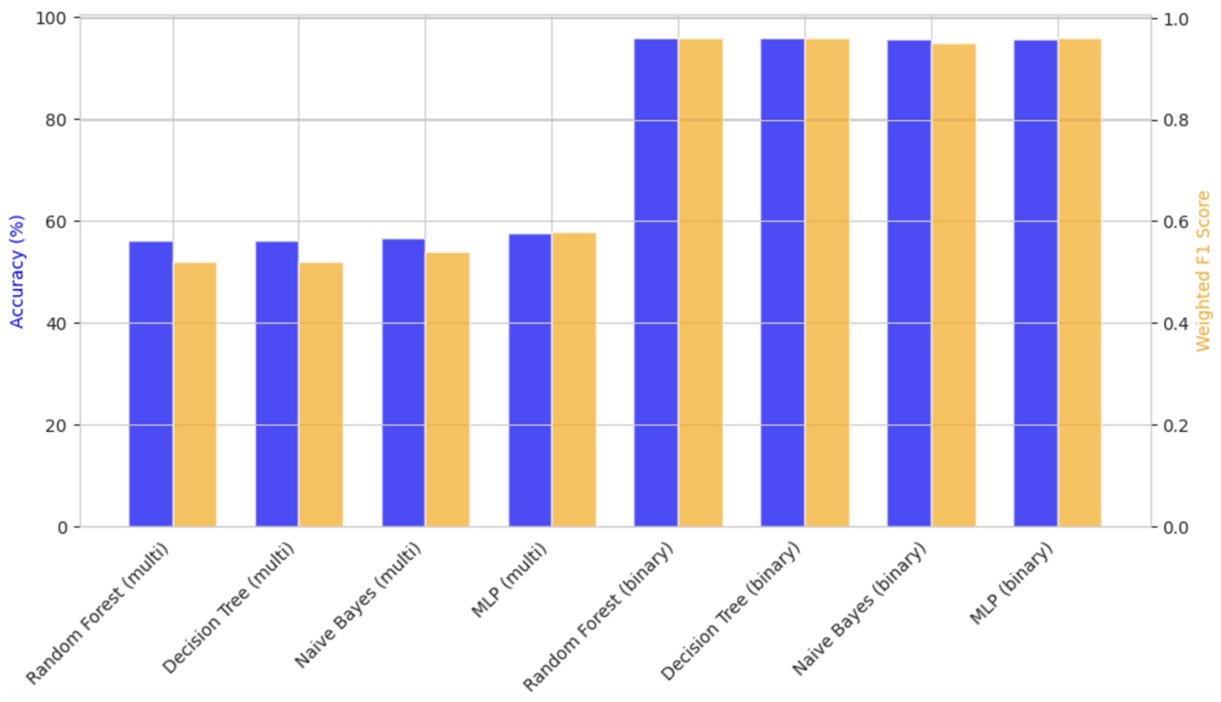


Figure 4.6: Comparison of multi- and binary-level classification performance using models from [56] combined with the collected dataset.

This initial evaluation [4] has shown that breaking down the classes and turning the problem into a binary classification problem can significantly improve both accuracy and F1 score. As a consequence, all further training and evaluation was carried out using this binary

[4]The complete results using multi-level buckets can be found here: `https://github.com/samuelfrnk/BA_Samuel/blob/main/ML_Analysis/Darios_Notebook/Experiment_3/ML_Analysis_Experiment3.ipynb` The complete results using binary buckets can be found here: `https://github.com/samuelfrnk/BA_Samuel/blob/main/ML_Analysis/Darios_Notebook/Experiment_3/Binary_Bins/ML_Analysis_Experiment3_BinaryBins_Smoothend.ipynb`

classification approach. The underlying assumption is that if a tracker is closer than 0.5 meters to the central mobile phone, it is more likely to be malicious. In order to counteract imbalanced classes and their implications on the model Synthetic Minority Oversampling Technique (SMOTE) is being applied.

4.2.2 Features

4.2.2.1 RSSI Smoothening

Since several sources [33, 34, 35, 36, 37] suggest applying smoothing techniques to RSSI data as a post processing step, smoothed RSSI value is considered as a possible input feature. The smoothening process should thus counteract the noise inherent in the RSSI data. A moving average filter with a window size of 5 was used and the result is visualized in the figure 4.7. The smoothed RSSI values have been added to each data point so that in a next step the importance of the features can be analyzed and it can be decided which features to use for the models.

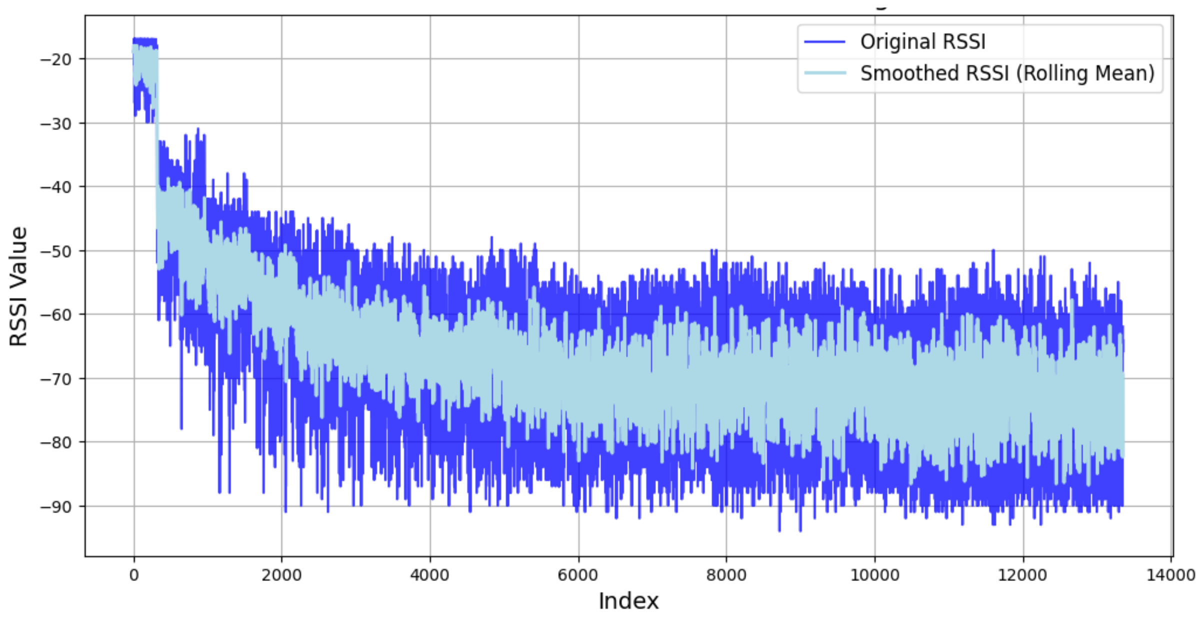


Figure 4.7: Visualized results of the moving average filter.

4.2.2.2 Feature Selection

As seen in 4.1, there are several possible features for the classification input. Indoor and outdoor as well as LOS and NLOS were extracted together with the RSSI value during the collection experiments. Smoothed RSSI has been added with a post processing calculation. To get an idea of how beneficial these features are to the performance of the models, a feature importance analysis was performed for both the decision tree classifier and the random forest classifier. In addition, a permutation importance analysis was performed for KNN and SVM.

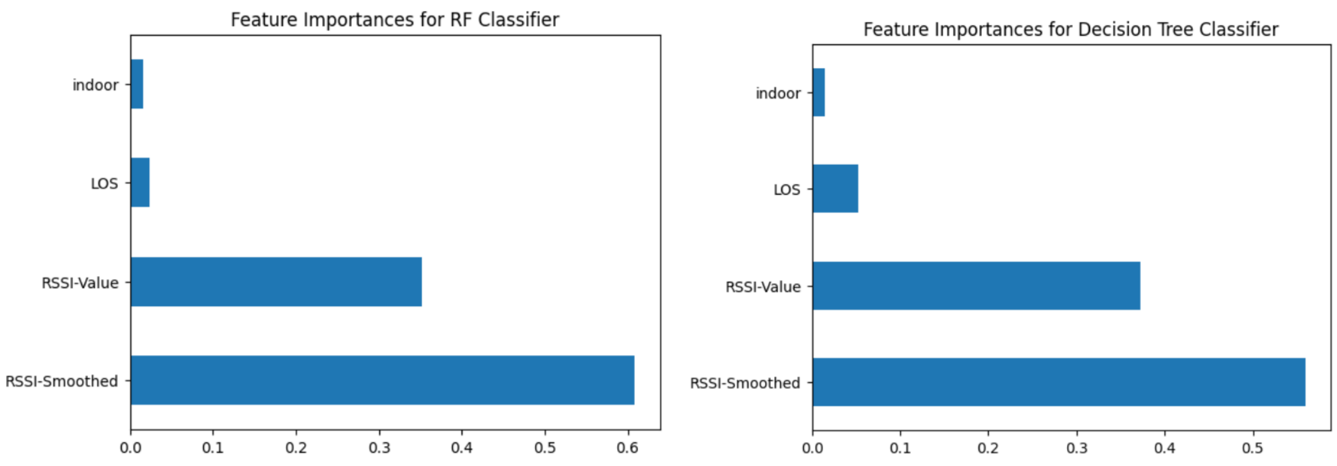


Figure 4.8: Feature Importance Analysis of RF and Decision Tree Classifier.

According to the analysis of feature importance 4.8, the Smoothed RSSI appears to be essential for reducing impurity and therefore has the highest feature importance score. This is also consistent with the permutation importance analysis 4.9 where again the smoothed RSSI has the highest score for both models.

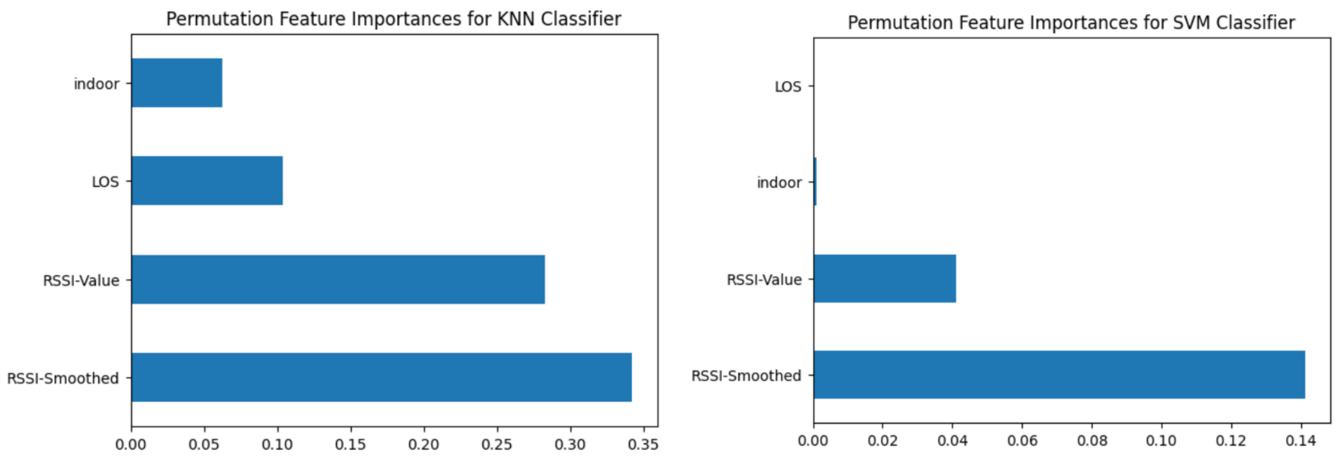


Figure 4.9: Permutation Importance Analysis of RF and Decision Tree Classifier.

As the smoothed RSSI outperformed all other features in all initial models, it is added as an input feature to the classifier. The indoor and LOS features do not appear to be as important in all models. They are both added because they are still helpful for the performance of KNN.

4.2.3 Final Model Outline

The iterative refinements presented above culminate in the final model structure shown in 4.10. In the following section, all models are evaluated based on this structure.

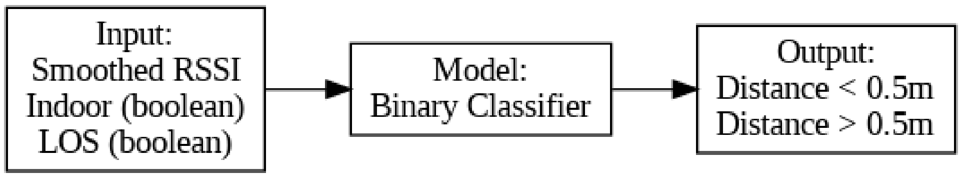


Figure 4.10: Final Model Structure.

4.3 Performance Evaluation

This section provides insight into how the selected models performed [5] in the context of the outlined structure. The initial step is to conduct an overview of all models. Subsequently, the model that has been identified as the most promising in terms of the defined performance score will be the subject of closer examination.

4.3.1 Model Overview

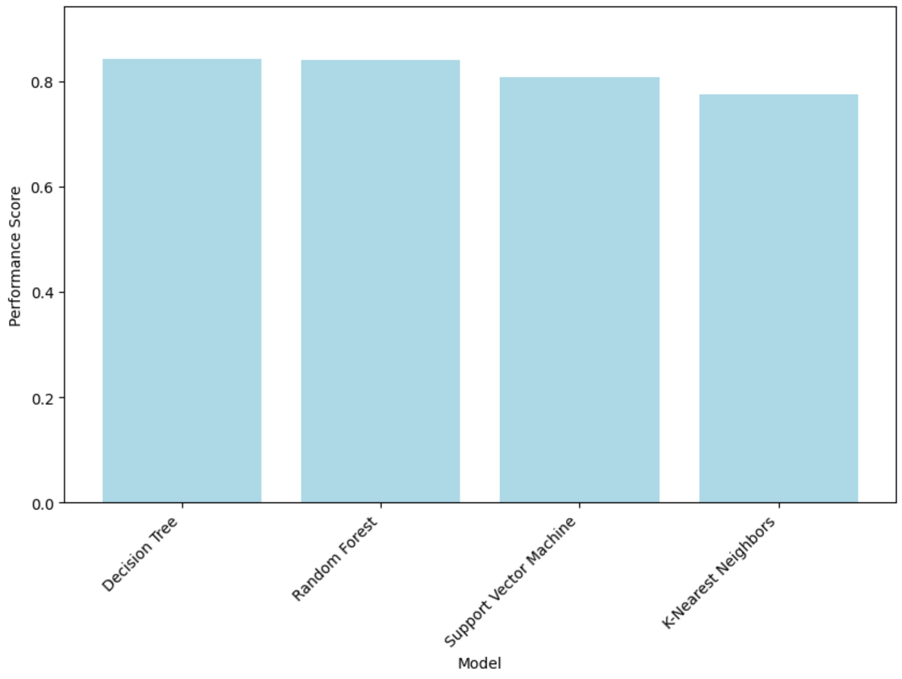


Figure 4.11: Performance Scores Across Models.

[5]The complete Evaluation notebook can be found here: `https://github.com/samuelfrnk/BA_Samuel/blob/main/ML_Analysis/This_Work/Final_Evaluation/ML_Analysis_BA.ipynb`

The definition of the performance score defined by this thesis can be found in section 3.3.2. As figure 4.11 illustrates, the decision tree classifier demonstrated the highest overall performance, with a score of 0.8412. This was closely followed by the RF classifier, which exhibited a marginally lower performance, with a score of 0.8409. The SVM classifier achieved a performance score of 0.8070, while the KNN classifier, the worst performing classifier, achieved a score of 0.7755.

A summary of the values leading to the associated performance scores can be found in 4.3. It is noteworthy that the KNN classifier demonstrated satisfactory accuracy and remarkable performance in terms of the weighted F1 score. However, due to the defined weights of the components of the score and the relatively high discrepancy between the test and training accuracy of KNN, as well as the inherent risk of overfitting, the classifier did not exhibit a good overall performance.

Classifier	Cross-Validated Accuracy	Weighted F1 Score	Normalized Overfitting Penalty	Performance Score
Decision Tree	0.8535	0.8882	0.0297	0.8412
Random Forest	0.8531	0.8882	0.0297	0.8409
Support Vector Machine	0.8411	0.8980	0.0626	0.8070
K-Nearest Neighbors	0.8187	0.9064	0.0870	0.7755

Table 4.3: Comparison of classifier performance metrics.

To build a bridge to the design section and the literature review, 4.4 compares the achieved performance of the models in the literature with the performance scores of the individual models that were achieved in the context of the thesis.

Model	Reference	Reported Performance	Performance Score (This Work)
Decision Tree	N/A	N/A	84.12%
RF	[40]	Centimeter-level accuracy	84.09%
SVM	[33]	84% accuracy	80.70%
KNN	[33]	85% accuracy	77.55%

Table 4.4: Comparison of models, references, and their performance results from literature versus our findings.

4.3.2 Best Performing Model

As mentioned above, the decision tree classifier performed best. In this section, the classifier is now analyzed in more detail.

4.3.2.1 Metrics

Its confusion matrix, shown in 4.12, reflects the class distribution present in the test data, which was not altered by SMOTE. Since SMOTE was applied only to the training data to prevent data leakage, the original class imbalance remains visible in the confusion matrix of the test set. Within the classification, label 0 corresponds to class proximity greater than 0.5 meters, while 1 corresponds to class proximity less than 0.5 meters, and is therefore classified as malicious.

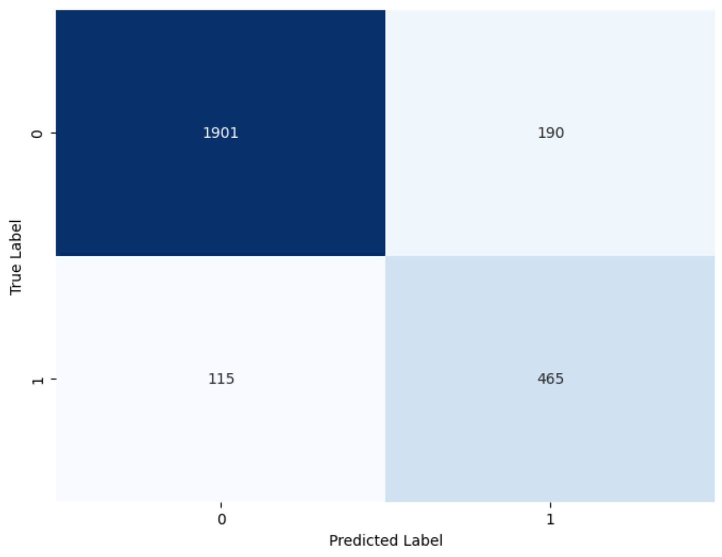


Figure 4.12: Confusion Matrix of the Decision Tree classifier.

As seen in 4.5, the decision tree achieves a high test accuracy of about 88.6 % and a strong weighted F1 score of about 0.89, which indicates a balanced performance across all classes. The accuracy difference of 0.0297 is relatively small, indicating that the model does not exhibit drastic overfitting. Overall, the model's performance score of 0.8412 emphasizes its robust effectiveness in this classification task.

Metric	**Value**
Training Accuracy	0.8595
Test Accuracy	0.8858
F1 Score (Test)	0.8882
Mean Cross-Validation Accuracy	0.8535 (± 0.0103)
Normalized Accuracy Difference	0.0297
Performance Score	0.8412

Table 4.5: Summary of key metrics for the Decision Tree classifier.

The high and consistent accuracy score across different folds, as seen in 4.13, further underlines the robustness of the classifier and its ability to generalize to unseen data.

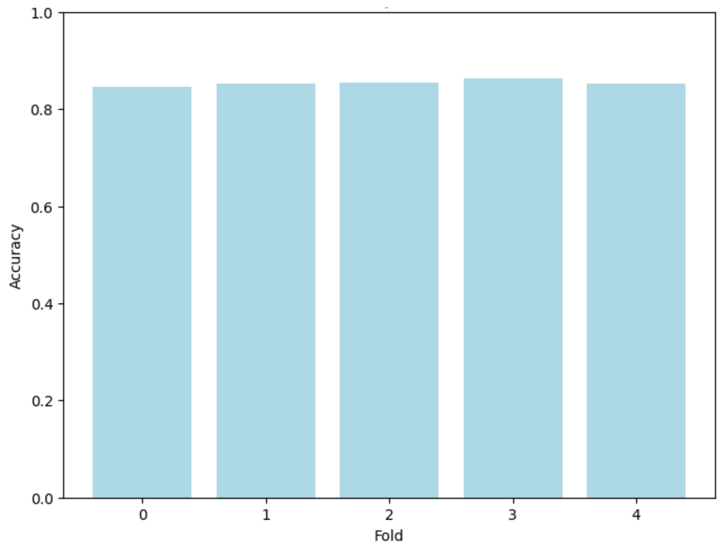


Figure 4.13: Cross-validation Accuracy Scores for Decision Tree.

4.3.2.2 Conclusion

It was not necessarily expected that the Decision Tree classifier would perform best. The fact that the classification problem was simplified with binary classes probably plays a role. In addition, the number of input features is manageable, which reduces the complexity of the feature space. This means that even simpler models can make good predictions. Another big advantage of this result is that a simpler model leads to a more lightweight integration in HomeScout.

4.4 Porting to HomeScout

This section explains the implementation [6] of the shielding feature in HomeScout on the three layers UI, service, and data.

4.4.1 User Interface Layer

From a UI perspective, the user needs a way to switch the shielding feature off and on. This also leads to an adjustment that is needed in the UI layer, which affects the setting fragment. The shielding feature in HomeScout should be a binary decision. Either the user wants to activate the shield or not. This means that a toggle is needed in the settings for this choice. Additionally, two more toggles have been implemented for the input features

[6] The complete source code of HomeScout, including the additional RSSI AirTag shielding feature, is available at: `https://github.com/samuelfrnk/BA_Samuel/tree/main/HomeScout`

alongside RSSI; indoor and LOS. Figure 4.14 shows these implemented changes [7] in the UI layer.

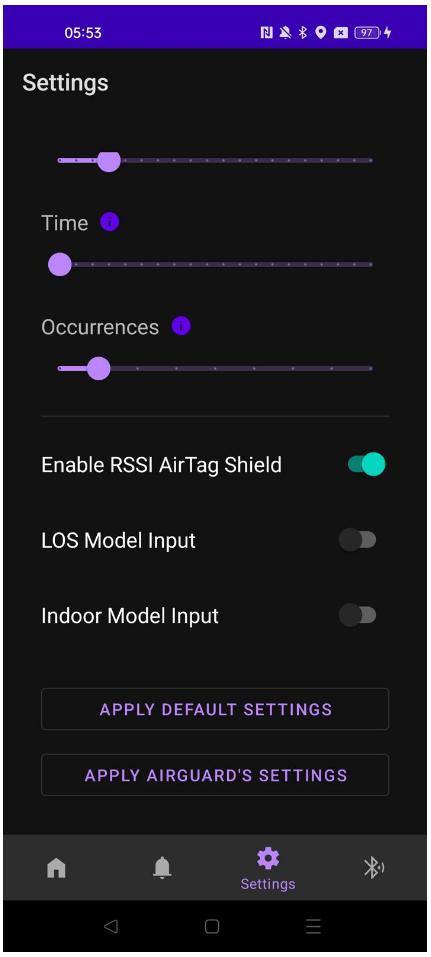


Figure 4.14: UI adjustments within the setting fragment including toggles for the shielding, and the input features.

4.4.2 Data Layer

As discussed in the design section, the changes in the data layer affect mainly the BLEDevice database. An additional feature RSSI has been added here, which is saved for each entry during scanning within the BluetoothScanningService class. Both of these changes are visible in 4.15, they are rather small and were directly realized according to the design of the implementation. In the design section, it was further mentioned that additional input features also have to be extracted at runtime, which turns out to be a challenge. This is why these environmental features were implemented using a manual toggle. Whether these two features, LOS and indoor, could also be extracted at runtime is outside the scope of this work and will be considered for future work.

[7] The complete change log of adjustments related to the UI is available in the following commit : `https://github.com/samuelfrnk/BA_Samuel/commit/f71c37c95453a3b40974e2e68257f89c1e5df403`

```

    data class BLEDevice(
        var macAddress: String? = null,
        var timestampInMilliSeconds: Long = 0L,
        var lat: Double,
        var lng: Double,
        var type: String,
        // RSSI value added:
        var RSSI: Int
    ){
    // ... further BLEDevice logic
    }

    class BluetoothScanningService : LifecycleService() {

        // ... further scanning logic

        val timestampInMilliSeconds = Calendar.getInstance().
           timeInMillis
        val lat = it.latitude
        val lng = it.longitude
        val deviceType = DeviceTypeManager.identifyDeviceType(result).
           type
        // extraction of the RSSI value:
        val rssi = result.rssi

        val bleDevice = BLEDevice(
            mac,
            timestampInMilliSeconds,
            lat,
            lng,
            deviceType,
            // insertion of the RSSI value:
            rssi)

            scanResults[mac] = bleDevice

        // ... further scanning logic

    }
```

Figure 4.15: Data level adjustments including the addition of RSSI data and its extraction within the BLEDevice.kt and BluetoothScanningService.kt class respectively.

4.4.3 Service Layer

The most important change for the integration into the existing tracker detection logic of HomeScout is within the service layer of the application. The existing thresholds regarding distance, occurrences and time have been extended with a further component which is explained in this section.

4.4.3.1 Initialization

First, the evaluated decision tree classifier was extracted with ONNX and made accessible in the HomeScout framework in the form of an ort file within the assets of the application. As shown in 4.16, the onCreate method of the TrackerClassificationService class then references the exported Classifier.ort file. This file is used to initialize an OrtEnvironment and an OrtSession object, which are subsequently utilized for performing the ML classifications. In addition, the user's desired settings, including both the newly introduced boolean values as well as the existing thresholds, are also available as fields.

```

    class TrackerClassificationService : LifecycleService() {

        // ... further classification logic

        private var distance : Float? = null
        private var timeinMin : Float? = null
        private var occurrences : Float? = null
        private var isRssiShield: Boolean ?= null
        private var isIndoor: Boolean ?= null
        private var isLos: Boolean ?= null

        private lateinit var ortEnvironment: OrtEnvironment
        private lateinit var ortSession: OrtSession

         // ... further classification logic

        override fun onCreate() {

        val modelPath = "Classifier.ort"
        val assetManager = assets
        val modelInputStream = assetManager.open(modelPath)
        val modelBytes = modelInputStream.readBytes()
        ortSession = ortEnvironment.createSession(modelBytes)

        }

        // ... further classification logic

    }
```

Figure 4.16: Lifecycle method to initialize OrtEnvironment and OrtSession for classification using the exported Classifier.ort model file.

4.4.3.2 Classification

The classification logic is implemented in the startTrackerClassification function located in the TrackerClassificationService class, as shown in 4.17. It begins by iterating over all recognized devices, including their corresponding individual scans. The three thresholds, occurances time and distance are then checked with the existing logic step by step in this order. If a threshold is not exceeded, the iteration of the for loop is continued and the associated device is not considered malicious.

```kotlin
class TrackerClassificationService : LifecycleService() {

    // ... further classification logic

    private fun startTrackerClassification() {

        // ... further classification logic

        val scansOfThisDevice = hashMapBleDevicesSortedByTime[key]!!

        // ... further classification logic

        if ( scansOfThisDevice.size == 1 || scansOfThisDevice.size < occurrences!!) {continue}

        // ... further classification logic

        if (diffBetweenYoungestAndOldestScan < timeThresholdInMillis ) { continue }

        // ... further classification logic

        if (distanceFollowed < distance!!) { continue }

        //RSSI Shield:
        if (scansOfThisDevice[0].type == AirTag().type && isRssiShield == true) {
            var closeTrackerCount = 0
            for ((index, scan) in scansOfThisDevice.withIndex()) {
                val inputData = floatArrayOf(
                    scan.RSSI.toFloat(),
                    if (isIndoor == true) 1f else 0f,
                    if (isLos == true) 1f else 0f
                )
                val inputTensor = OnnxTensor.createTensor(ortEnvironment, arrayOf(inputData))
                val results = ortSession.run(Collections.singletonMap("input", inputTensor))
                val outputTensor = results[0].value as LongArray
                if (prediction == 1L) {
                    closeTrackerCount++
                }
            }
            if (closeTrackerCount < occurrences!!) {
                continue
            }
        }
    }
}
```

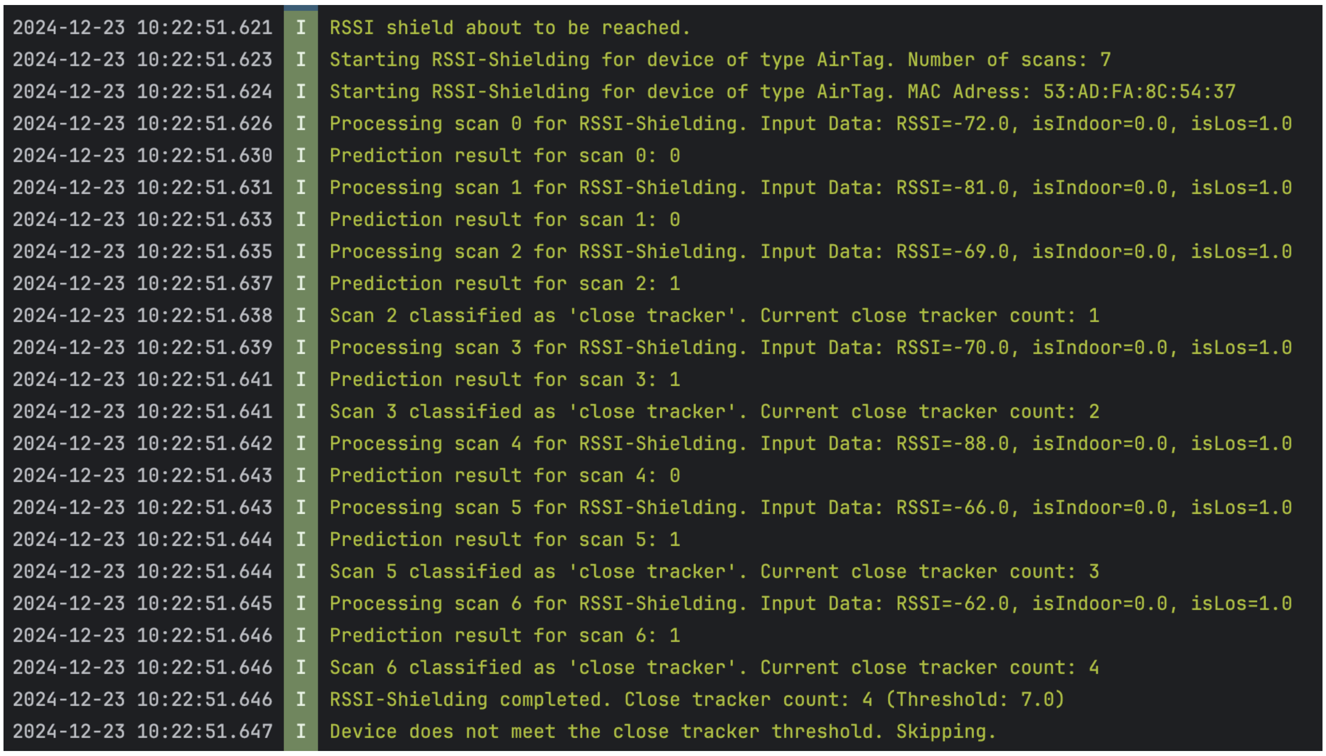
Figure 4.17: The implemented shielding logic within the TrackerClassificationService class.

The ML shielding logic, seen in 4.17 and starting at line 24, is activated only under specific conditions. Specifically, if the RSSI shield is enabled in the settings and the device currently being analyzed is an AirTag, as the model was specifically designed for this classification. If both conditions are met, the code iterates over all scans that are assigned to this device and performs a classification for each entry. This process takes place between lines 27 and 34. The input data consists of the RSSI values from the database and the LOS and indoor features from the settings. The model classifies each scan as either a 1 if the device is regarded closer than 0.5 meters or a 0 if it is regarded as further away. Each time the model classifies a 1, the closeTrackerCounter initialized in line 25 is incremented by one. After all entries have been classified, the counter is compared with the threshold value for the number of occurrences. If the value is lower, the loop is continued and the device is classified as unsuspicious. However, if the counter exceeds the threshold, the loop is terminated, the device is classified as malicious and the user is informed via a notification.

This integration enhances the threshold occurrence so that not all occurrences are taken into account and compared with the threshold, but only those occurrences that are classified as close by the model.

4.4.4 Resource consumptions considerations

4.18 shows the logs corresponding to a classification of an AirTag with seven BLE scans and related sub classifications. Only four of the seven entries are classified as near, which is not enough for the threshold value of 7, which is why the AirTag as a whole is not considered malicious.



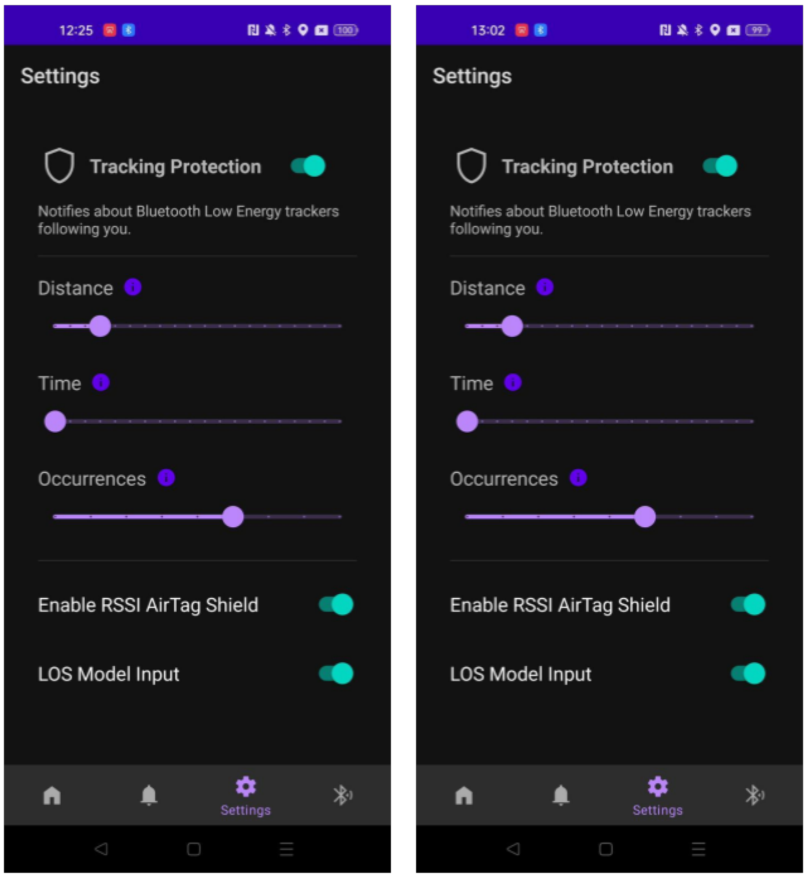
```
2024-12-23 10:22:51.621 I RSSI shield about to be reached.
2024-12-23 10:22:51.623 I Starting RSSI-Shielding for device of type AirTag. Number of scans: 7
2024-12-23 10:22:51.624 I Starting RSSI-Shielding for device of type AirTag. MAC Adress: 53:AD:FA:8C:54:37
2024-12-23 10:22:51.626 I Processing scan 0 for RSSI-Shielding. Input Data: RSSI=-72.0, isIndoor=0.0, isLos=1.0
2024-12-23 10:22:51.630 I Prediction result for scan 0: 0
2024-12-23 10:22:51.631 I Processing scan 1 for RSSI-Shielding. Input Data: RSSI=-81.0, isIndoor=0.0, isLos=1.0
2024-12-23 10:22:51.633 I Prediction result for scan 1: 0
2024-12-23 10:22:51.635 I Processing scan 2 for RSSI-Shielding. Input Data: RSSI=-69.0, isIndoor=0.0, isLos=1.0
2024-12-23 10:22:51.637 I Prediction result for scan 2: 1
2024-12-23 10:22:51.638 I Scan 2 classified as 'close tracker'. Current close tracker count: 1
2024-12-23 10:22:51.639 I Processing scan 3 for RSSI-Shielding. Input Data: RSSI=-70.0, isIndoor=0.0, isLos=1.0
2024-12-23 10:22:51.641 I Prediction result for scan 3: 1
2024-12-23 10:22:51.641 I Scan 3 classified as 'close tracker'. Current close tracker count: 2
2024-12-23 10:22:51.642 I Processing scan 4 for RSSI-Shielding. Input Data: RSSI=-88.0, isIndoor=0.0, isLos=1.0
2024-12-23 10:22:51.643 I Prediction result for scan 4: 0
2024-12-23 10:22:51.643 I Processing scan 5 for RSSI-Shielding. Input Data: RSSI=-66.0, isIndoor=0.0, isLos=1.0
2024-12-23 10:22:51.644 I Prediction result for scan 5: 1
2024-12-23 10:22:51.644 I Scan 5 classified as 'close tracker'. Current close tracker count: 3
2024-12-23 10:22:51.645 I Processing scan 6 for RSSI-Shielding. Input Data: RSSI=-62.0, isIndoor=0.0, isLos=1.0
2024-12-23 10:22:51.646 I Prediction result for scan 6: 1
2024-12-23 10:22:51.646 I Scan 6 classified as 'close tracker'. Current close tracker count: 4
2024-12-23 10:22:51.646 I RSSI-Shielding completed. Close tracker count: 4 (Threshold: 7.0)
2024-12-23 10:22:51.647 I Device does not meet the close tracker threshold. Skipping.
```

Figure 4.18: Logs with timestamps of a classification for an AirTag with seven corresponding scans.

It is also interesting to look at the timestamps. The classification starts with the first log at 10:22:51.621 and ends with the last log at 10:22:51.647. There are only 26 milliseconds in between, indicating that it works well in terms of time to classify each individual scan of a device.

Furthermore, the RSSI shield only takes effect when the existing three thresholds consider a device to be malicious, which limits the number of classifications required. The logic that initiates classification and scanning has also not been changed, which was initially designed to be as resource-efficient as possible.

A test was carried out to get an idea of how much battery the application consumes. To do this, the mobile phone on which HomeScout was running was first charged to 100 %. Then the RSSI shielding was activated together with the default settings, and the mobile phone was carried around together with several AirTags. 4.19 shows that the phone activated tracking protection for a total of about 40 minutes from 12:25 to 13:02 and was moving around. It drained one percentage battery in the entire time.


12:25 100

Settings

Tracking Protection

Notifies about Bluetooth Low Energy trackers following you.

Distance

Time

Occurrences

Enable RSSI AirTag Shield

LOS Model Input

Settings

13:02 99

Settings

Tracking Protection

Notifies about Bluetooth Low Energy trackers following you.

Distance

Time

Occurrences

Enable RSSI AirTag Shield

LOS Model Input

Settings

Figure 4.19: Battery test during 40 minutes of movement and tracking protection activated with the RSSI shielding. Including the battery loss of one percentage.

4.4.5 Conclusion

This subsection provides an overview of the porting of the classifier in HomeScout. The integration has worked. The final implementation strengthens the existing occurrence threshold parameter. If the shield is activated, not all scan entries count towards the occurrence counter, but only those that have been classified as close by the model. According to the mentioned initial evaluations of the resource consumption of the feature, the new shielding seems to be lightweight enough to work in the mobile environment of HomeScout. The extraction of the additional features besides the RSSI value turned out to be a challenge as anticipated in the design section of the thesis. In the current solution, these are manually set by the user in the settings. How this situation could be tackled is discussed in the Future Work section.

Chapter 5

Final Considerations

The thesis is concluded in the final chapter. In addition, an overview is given of possible areas where the results of this work could be used as a basis for further research in this field.

5.1 Conclusions

The following points were defined as the thesis objectives at the beginning of this work. These will now be examined in the context of the conclusion of the thesis.

(i) How does RSSI data emitted from AirTags correlate to distance and are there any underlying patterns?

(ii) Which machine learning model with which input features is best suited for predicting the proximity of a BLE emitting AirTag based on the RSSI values?

(iii) Can the in ii) identified model be used to provide a reasonable shielding mechanism which further improves the accuracy of identifying malicious trackers?

Research question (i) was investigated by collecting data sets and subsequently analyzing them. The decay of the RSSI values with increasing distance was observed. The values were high at short distances. However, as soon as the distance became greater, they fell rapidly. This was observed in data corresponding to all environments, indoor and outdoor, as well as in LOS and NLOS measurements. In addition, a high degree of fluctuation and variance was observed throughout the entire data. This was to be expected and serves as a motivation for looking at a ML approach that might counteract precisely these RSSI inherent fluctuations. In addition, interesting findings were made in the comparison between indoor and outdoor measurements. The RSSI measurements indoors were consistently higher than the outdoor measurements. This was not necessarily to be expected and would be interesting to investigate further.

In a further step, the collected data could be used as a basis to answer research question (ii). Through initial experiments and evaluations of classifications, the problem was modeled as a binary classification problem with three input features. A custom performance score was defined that takes into account various aspects of the evaluation of classification models. This includes the accuracy of the model as well as the ability to correctly classify unseen data and thus the ability to generalize in the context of the problem. For this reason, the F1-score, which takes precision and recall into account, was also included in the performance score. In addition, a penalty consisting of the difference between training accuracy and testing accuracy was introduced to identify a model that exhibits as little overfitting as possible. In the context of this evaluation, the decision tree performed best with a promising score of 84%. That simpler models like decision tree or RF have done so well is probably a consequence of the design decision to model the classification binary and thus simpler. A simple model is also better suited for a mobile environment, such as the HomeScout application.

In a final step, the evaluated decision tree model was integrated into HomeScout as a shielding mechanism. This was done as part of the research question (iii). Using ONNX, the model was successfully transferred from the evaluation framework to the HomeScout framework, which is an Android application developed in Kotlin. For the application, this results in a new shielding feature that can be either switched on or off. There were several challenges. One of these was the extraction of the other binary input features along with the observed RSSI values. Specifically, indoor and LOS, which in the implementation are not automatically extracted by the application itself at runtime, but have to be manually set by the user in the settings. In addition, the question of the advantages of the chosen approach compared to a simpler threshold approach, as was originally the case with the existing components of the HomeScout classification, occurrences, time and distance values, could not be fully clarified.

5.2 Future Work

Based on the findings from the data collection and data analysis in this thesis, there are several areas that would be interesting to explore further. One of these would be the correlation between lower RSSI values and an outdoor experiment setting. This could be reproduced with different hardware and the correlation could be confirmed in a different setting. In addition, the collected labeled data set is also suitable for other research in connection with RSSI data and ML. It would also be interesting to compare it to RSSI values from devices other than BLE trackers.

In the context of the identified ML models and their training, there are also exciting areas that could be further investigated. On the one hand, this would be the possibility of systematic hyper-parameter tuning and investigating its influence on the performance scores achieved by the optimized models. One could also look at more complex models than those considered in the thesis. In particular gradient boost and neural networks. These more complex models could be used to investigate which performance scores are possible when the problem is modeled in a more complex way compared to a binary classification.

In relation to the integration in HomeScout, there are also exciting additional areas and questions that have opened up. In particular, the possibility of extracting all input features at runtime instead of leaving this to the user manually. Exciting approaches consist of the combination of the coordinates that are already in the BLEDevice database for each scan and geolocation APIs, which together could make it possible to extract the input feature indoor at runtime. A starting point here could be the Google Places API [57]. This API makes it possible to analyze surrounding locations based on coordinates, such as those in the scan results. The Nearby Search endpoint could be used. In the request, the parameter includedPrimaryTypes could be filled with environments that suggest an indoor position. Such places could be libraries, restaurants, grocery stores, and other similar locations. The API then returns the distances to the surrounding places in a selectable radius. These distances could be compared to a small threshold value. If one of the distances falls below this threshold, HomeScout could label the corresponding scan as indoor. This solution raises important questions about the privacy of sensitive data, the effectiveness of the approach, and the performance implications for HomeScout, which could be fully clarified and evaluated. The extraction of the input feature LOS is a greater challenge. It is not necessarily possible to derive this information from the existing data in HomeScout. A technically sophisticated idea could possibly be found. An alternative would be to make an assumption that when a victim is stalked by an AirTag, it is usually hidden in a handbag, backpack, or under the car, which is usually not in LOS to the victim's mobile phone. Based on this assumption, the input feature LOS could be hardcoded as false.

It would also be very interesting to compare experiments between tracker classifications with and without the implemented shield. Field experiments leading to confusion matrices could be carried out for both cases to further explore the feature and potential benefits. Another point is the comparison of the ML shielding developed in this thesis and a simpler threshold approach. Again, field experiments with confusion matrices would give very exciting insights into how much value the model in HomeScout really gives and how effectively ML can be used in this problem context to counteract RSSI inherent challenges.

Bibliography

[1] Bluetooth SIG, "Core Specification 4.0," [Online]. Available: https://www.bluetooth.com/specifications/specs/core-specification-4-0/, Last visit October 11, 2024.

[2] A. Barua, M. A. Al Alamin, M. S. Hossain, and E. Hossain, "Security and Privacy Threats for Bluetooth Low Energy in IoT and Wearable Devices: A Comprehensive Survey," *IEEE Open Journal of the Communications Society*, Vol. 3, pp. 251–281, 2022.

[3] Bluetooth SIG, "202 Market Update," [Online]. Available: https://www.bluetooth.com/wp-content/uploads/2021/01/2021-Bluetooth_Market_Update.pdf, Last visit October 11, 2024.

[4] Apple Inc., "Apple introduces AirTag," [Online]. Available: https://www.apple.com/newsroom/2021/04/apple-introduces-airtag/, Last visit October 11, 2024.

[5] Samsung Electronics, "Galaxy SmartTag," [Online]. Available: https://www.samsung.com/ch/mobile-accessories/galaxy-smarttag-black-ei-t5300bbegeu/, 2024, Last visit October 13, 2024.

[6] Tile Inc., "How Tile works," [Online]. Available: https://www.tile.com/how-it-works, 2024, Last visit October 13, 2024.

[7] Chipolo Inc., "Find your everything," [Online]. Available: https://chipolo.net/de, 2024, Last visit October 13, 2024.

[8] Apple Inc., "Apple's Find My network now offers new third-party finding experiences," [Online]. Available: https://www.apple.com/pt/newsroom/2021/04/apples-find-my-network-now-offers-new-third-party-finding-experiences/, 2021, Last visit October 13, 2024.

[9] K. Langley, "State police arrest suspect in AirTag stalking case," [Online]. Available: https://www.nbcconnecticut.com/news/local/state-police-arrest-suspect-in-airtag-stalking-case/3214713/, 2024, Last visit October 13, 2024.

[10] L. Zobel, M. Muldofsky, N. Mastrangelo, D. Kim, A. Ball, R. Wenzlaff, and I. Pereira, "Apple AirTags causing major security concerns over reports of stalking," [Online]. Available: https://abcnews.go.com/US/apple-airtags-causing-major-security-concerns-reports-stalking/story?id=96531871, 2023, Last visit October 13, 2024.

[11] E. Mooney, "'It was ruining my life', Irish actress reveals month of terror after psycho's sick tracker ploy dashed Hollywood dream," [Online]. Available: https://www.thesun.ie/news/12724844/aine-oneill-actress-stalker-terror-apple-airtag-tracker-hollywood/, 2024, Last visit October 13, 2024.

[12] B. McLaren, "Apple AirTags: Love Island star says device used to stalk her," [Online]. Available: https://www.bbc.com/news/newsbeat-65030359, 2023, Last visit December 26, 2024.

[13] C. Miller, "Apple must face class action lawsuit over AirTag stalking, judge rules," [Online]. Available: https://9to5mac.com/2024/03/16/airtag-stalking-lawsuit-judge/, 2024, Last visit October 13, 2024.

[14] T. Mayberry, E. Fenske, D. Brown, J. Martin, C. Fossaceca, E. C. Rye, S. Teplov, and L. Foppe, "Who Tracks the Trackers?: Circumventing Apple's Anti-Tracking Alerts in the Find My Network," *Proceedings of the 20th Workshop on Privacy in the Electronic Society (WPES).* New York, NY, USA, ACM, 2021, pp. 181–186.

[15] K. O. E. Müller, L. Bienz, B. Rodrigues, C. Feng, and B. Stiller, "HomeScout: Anti-Stalking Mobile App for Bluetooth Low Energy Devices," *2023 IEEE 48th Conference on Local Computer Networks (LCN).* IEEE, 2023, pp. 1–9.

[16] B. Roston, "Apple's New Tracker Detect App Helps Android Users Find Hidden AirTags," [Online]. Available: https://www.slashgear.com/apples-new-tracker-detect-app-helps-android-users-find-hidden-airtags-14702343/, 2021, Last visit July 23, 2024.

[17] B. Ledvina, Z. Eddinger, B. Detwiler, and S. P. Polatkan, "Detecting Unwanted Location Trackers," [Online]. Available: https://datatracker.ietf.org/doc/draft-detecting-unwanted-location-trackers/, 2024, Last visit October 13, 2024.

[18] Apple Inc., "Apple and Google deliver support for unwanted tracking alerts in iOS and Android," [Online]. Available: https://www.apple.com/mg/newsroom/2024/05/apple-and-google-deliver-support-for-unwanted-tracking-alerts-in-ios-and-android/, 2024, Last visit October 13, 2024.

[19] E. Kay, "3 ways unknown tracker alerts on Android help keep you safe," [Online]. Available: https://blog.google/products/android/unknown-tracker-alert-google-android/, 2023, Last visit October 13, 2024.

[20] L. Bienz, "HomeScout: A Modular Bluetooth Low Energy Sensing Android App," Master's Thesis, Communication System Group CSG, Department of Informatics IfI, Universität Zürich, January 2023, supervisor: Katharina Müller.

[21] C. Naysmith, "Apple AirTags and Bluetooth Trackers Are Officially a Billion-Dollar Industry â Here's What To Know, Trends, and the Best Ways To Invest," [Online]. Available: https://finance.yahoo.com/news/apple-airtags-bluetooth-trackers-officially-175911565.html, 2022, Last visit October 13, 2024.

[22] Bluetooth SIG, "The Bluetooth® Low Energy Primer," [Online]. Available: https://www.bluetooth.com/bluetooth-resources/the-bluetooth-low-energy-primer/, Last visit December 26, 2024.

[23] N. Gupta, *Inside Bluetooth Low Energy*, second edition Edt. Artech House, June 2016.

[24] C. Udekwe, "Understanding RSSI in Wireless Signals," [Online]. Available: https://www.baeldung.com/cs/rssi-wireless-signal, 2024, Last visit November 29, 2024.

[25] Y. Huang, "Die Rolle von Bluetooth RSSI bei der Indoor-Positionierung," [Online]. Available: https://www.mokosmart.com/the-role-of-bluetooth-rssi-in-indoor-positioning/, 2024, Last visit November 29, 2024.

[26] M. Phunthawornwong, E. Pengwang, and R. Silapunt, "Indoor Location Estimation of Wireless Devices Using the Log-Distance Path Loss Model," *TENCON 2018 - 2018 IEEE Region 10 Conference*. Jeju, Korea (South), IEEE, 2018, pp. 0499–0502.

[27] Apple Inc., "Find My: One app to find it all," [Online]. Available: https://www.apple.com/icloud/find-my/, 2024, Last visit October 13, 2024.

[28] A. Heinrich, M. Stute, T. Kornhuber, and M. Hollick, "Who Can Find My Devices? Security and Privacy of Apple's Crowd-Sourced Bluetooth Location Tracking System," *arXiv preprint arXiv:2103.02282*, 2021.

[29] Y. Assayag, H. Oliveira, M. Lima, J. Junior, M. Preste, L. Guimaraes, and E. Souto, "Indoor Environment Dataset Based on RSSI Collected with Bluetooth Devices," *Data in Brief*, Vol. 55, p. 110692, 2024.

[30] A. Moradbeikie, M. Zare, A. Keshavarz, and S. I. Lopes, "RSSI-Based LoRaWAN Dataset Collected in a Dynamic and Harsh Industrial Environment with High Humidity," *Data in Brief*, Vol. 53, p. 110120, 2024.

[31] M. Mohammadi, A. Al-Fuqaha, M. Guizani, and J.-S. Oh, "Semisupervised Deep Reinforcement Learning in Support of IoT and Smart City Services," *IEEE Internet of Things Journal*, Vol. 5, No. 2, pp. 624–635, 2018.

[32] M. Mohammadi and A. Al-Fuqaha, "BLE RSSI Dataset for Indoor localization and Navigation," [Online]. Available: https://www.kaggle.com/datasets/mehdimka/ble-rssi-dataset/, 2018, Last visit December 26, 2024.

[33] M. W. P. Maduranga, V. Tilwari, and R. Abeysekera, "Improved RSSI Indoor Localization in IoT Systems with Machine Learning Algorithms," *Signals*, Vol. 4, No. 4, pp. 651–668, 2023.

[34] M. W. P. Maduranga, R. Abeysekera, and V. Tilwari, "Improved-RSSI-Based Indoor Localization by Using Pseudo-Linear Solution with Machine Learning Algorithms," *Journal of Electrical Systems and Information Technology*, Vol. 11, No. 1, pp. 10–20, 2024.

[35] L. Alsmadi, X. Kong, K. Sandrasegaran, and G. Fang, "An Improved Indoor Positioning Accuracy Using Filtered RSSI and Beacon Weight," *IEEE Sensors Journal*, Vol. 21, No. 16, pp. 18 205–18 213, 2021.

[36] D. R. D. Ainul, S. Wibowo and M. Siswanto, "An Improved Indoor RSSI Based Positioning System Using Kalman Filter and MultiQuad Algorithm," *2021 International Electronics Symposium (IES).* IEEE, 2021, pp. 558–564.

[37] M. H. Dwiputranto, D. J. Suroso, and N. A. Siddiq, "Kalman filter for RSSI-based indoor positioning system with min-max technique," *AIP Conference Proceedings*, Vol. 2968, No. 1, 2023.

[38] V. R, V. Mittal, and H. Tammana, "Indoor Localization in BLE using Mean and Median Filtered RSSI Values," *2021 5th International Conference on Trends in Electronics and Informatics (ICOEI).* IEEE, 2021, pp. 227–234.

[39] R. M. M. R. Rathnayake, M. W. P. Maduranga, V. Tilwari, and M. B. Dissanayake, "RSSI and Machine Learning-Based Indoor Localization Systems for Smart Cities," *Eng*, Vol. 4, No. 2, pp. 1468–1494, 2023.

[40] S. Debnath and K. O'Keefe, "Proximity Estimation with BLE RSSI and UWB Range Using Machine Learning Algorithm," *2023 13th International Conference on Indoor Positioning and Indoor Navigation (IPIN).* Nuremberg, Germany, IEEE, 2023, pp. 1–6.

[41] Z. Su, K. Pahlavan, E. Agu, and H. Wei, "Proximity Detection During Epidemics: Direct UWB TOA Versus Machine Learning Based RSSI," *International Journal of Wireless Information Networks*, Vol. 29, No. 4, pp. 480–490, 2022.

[42] H. Zadgaonkar and M. Chandak, "Locating objects in warehouses using BLE beacons & Machine Learning," *IEEE Access*, Vol. 9, pp. 1–1, 2021.

[43] K. Filus, S. Nowak, J. DomaÅska, and J. Duda, "Cost-effective Filtering of Unreliable Proximity Detection Results Based on BLE RSSI and IMU Readings Using Smartphones," *Scientific Reports*, Vol. 12, No. 1, p. 3263, 2022.

[44] D. Biswas, S. Barai, and B. Sau, "New RSSI-fingerprinting-based Smartphone Localization System for Indoor Environments," *Wireless Networks*, Vol. 29, No. 3, pp. 1281–1297, 2023.

[45] A. Booranawong, P. Thammachote, Y. Sasiwat, J. Auysakul, K. Sengchuai, D. Buranapanichkit, S. Tanthanuch, N. Jindapetch, and H. Saito, "Real-time Tracking of a Moving Target in an Indoor Corridor of the Hospital Building Using RSSI Signals Received from Two Reference Nodes," *Medical & Biological Engineering & Computing*, Vol. 60, No. 2, pp. 439–458, 2022.

[46] M. Cavur and E. Demir, "RSSI-based hybrid algorithm for real-time tracking in underground mining by using RFID technology," *Physical Communication*, Vol. 55, p. 101863, 2022.

[47] K. Shin, R. McConville, O. Metatla, M. Chang, C. Han, J. Lee, and A. Roudaut, "Outdoor Localization Using BLE RSSI and Accessible Pedestrian Signals for the Visually Impaired at Intersections," *Sensors*, Vol. 22, No. 1, p. 371, 2022.

[48] A. Heinrich, N. Bittner, and M. Hollick, "AirGuard - Protecting Android Users from Stalking Attacks by Apple Find My Devices," *Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec).* New York, NY, USA, ACM, 2022, pp. 26–38.

[49] J. Briggs and C. Geeng, "BLE-Doubt: Smartphone-Based Detection of Malicious Bluetooth Trackers," *2022 IEEE Security and Privacy Workshops (SPW).* IEEE, 2022, pp. 208–214.

[50] Nordic Semiconductor, "nRF52840 DK," [Online]. Available: https://www.nordicsemi.com/-/media/Software-and-other-downloads/Product-Briefs/nRF52840-DK-product-brief.pdf, Last visit November 14, 2024.

[51] Kraftmax, "Kraftmax XT1 Revolution Profi Batterietester inkl. Innenwiderstandsmessung," [Online]. Available: https://kraftmax.eu/batterien/zubehoer/2010/kraftmax-xt1-revolution-profi-batterietester-inkl.-innenwiderstandsmessung, 2024, Last visit November 14, 2024.

[52] J. Wong, "BLE Scanner with RSSI and MAC Address," [Online]. Available: hhttps://jimmywongiot.com/2021/05/31/ble-scanner-with-rssi-and-mac-address/, 2021, Last visit November 14, 2024.

[53] Nordic Semiconductor, "BLE Blinky Application - nRF5 SDK v17.1.0," [Online]. Available: https://docs.nordicsemi.com/bundle/sdk_nrf5_v17.1.0/page/ble_sdk_app_blinky.html, 2024, Last visit November 14, 2024.

[54] A. Géron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, first edition Edt. Sebastopol, CA, USA, O'Reilly Media, Inc., 2017.

[55] S. Panchal, "Deploying Scikit-Learn Models In Android Apps With ONNX," [Online]. Available: https://towardsdatascience.com/deploying-scikit-learn-models-in-android-apps-with-onnx-b3adabe16bab, 2022, Last visit November 27, 2024.

[56] D. A. Monopoli, "Dataset Generation for ML Personal Tracker Detection with a Focus on RSSI Shielding Approaches," Beachelor's Thesis, Communication System Group CSG, Department of Informatics IfI, Universität Zürich, July 2024, supervisor: Katharina Müller.

[57] Google, "Google Maps Platform: Places API Documentation," [Online]. Available: https://developers.google.com/maps/documentation/places/, 2024, Last visit December 30, 2024.

Abbreviations

API	Application Programming Interface
ATT	Attribute Protocol
BR	Basic Rate
CSG	Communicating Systems Group
dBm	Decibels relative to Milliwatt
DK	Development Kit
EDR	Enhanced Data Rate
FFNN	Feed-Forward Neural Network
GFSK	Gaussian Frequency Shift Keying
GAP	Generic Access Profile
GPS	Global Positioning System
GBM	Gradient Boost Machines
HCI	Host Controller Interface
KNN	K-Nearest Neighbours
LOS	Line Of Sight
L2CAP	Logical Link Control and Adaption Protocol
LE	Low Energy
LPWAN	Low Power Wide Area Network
NLOS	Non Line Of Sight
ONNX	Open Neural Network Exchange
OSI	Protocol Data Unit
PDA	Open System Interconnection
SDU	Service Data Units
SVM	Support Vector Machine
SMOTE	Synthetic Minority Oversampling
SOC	System On a Chip
UWB	Ultra Wide Band
UI	User Interface

List of Figures

List of Tables

Appendix A

Contents of the Repository

The repository [1] of this thesis contains 4 different sub folders: Experiments, HomeScout, ML_Analysis and ble_app_uart_adv_scan. These are explained individually.

A.1 Experiments Folder

A.1.1 AirTagEvaluation Folder

This sub folder contains all the results of the measurements and experiments relating to the AirTags. This includes the battery measurements as well as the results of the serial number analysis and BLE packet advertisement emitting tests.

A.1.2 Results Folder

This sub folder contains the results of the data collection. It contains the combined data set, a hash of the combined dataset, the individual data sets, plots and metadata.

A.2 HomeScout Folder

A.2.1 App Folder

This folder contains the source code of the new version of HomeScout, which has integrated the ML-based RSSI shielding feature developed as part of this thesis in addition to the existing features.

[1] `https://github.com/samuelfrnk/BA_Samuel`

A.3 ML_Analysis Folder

A.3.1 Darios_Notebook Folder

This folder contains Jupyter notebooks which have been taken from the work of Dario [2] and fed with the data of the experiment 3 of this thesis. This is the initial analysis has led to the problem being modeled as a binary classification problem.

A.3.2 This_Work Folder

This subfolder contains the ML analysis that selected the Decision Tree Classifier through the performance analysis in the form of a Jupyter notebook. It also contains the exported export classifier, which has been implemented in HomeScout as a .ort file.

A.4 ble_app_uart_adv_scan

A.4.1 Main.c File

This subfolder contains the source code for the application developed and used on the nRF board to filter for BLE ADV packets and log the respective MAC address and RSSI values.

[2] `https://github.com/dariomonopoli-dev/Bachelor_thesis_code`