BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INTELLIGENT SYSTEMS
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

SECURITY OF WIRELESS COMMUNICATION FOR IOT DEVICES

ZABEZPEČENÍ BEZDRÁTOVÉ KOMUNIKACE U ZAŘÍZENÍ IOT

PHD THESIS
DISERTAČNÍ PRÁCE

AUTHOR — **Ing. ONDŘEJ HUJŇÁK**
AUTOR PRÁCE

SUPERVISOR — **Doc. Dr. Ing. PETR HANÁČEK**
ŠKOLITEL

BRNO 2024

Abstract

The thesis explores the topic of security and privacy within the context of the Internet of Things. It particularly focuses on wireless networks designed for IoT devices, specifically LoRaWAN, Zigbee and Bluetooth Low Energy. The security weaknesses and vulnerabilities of these networks are analysed in detail and a security monitoring based on the NEMEA framework is proposed. A novel approach for monitoring Bluetooth networks is introduced. This approach is discussed in detail from theoretical design to experimental validation and eventual enhancement to production level.

Furthermore, the thesis outlines the security design of the proprietary IQRF network, which is part of its new standard. This design is based on the aforementioned security weakness analysis and significantly improves the security of this globally used network.

Finally, the thesis addresses privacy concerns by analysing the data transmission of four IoT gateways designed for home use. It also explores potential methods for anonymising IoT devices in various network types, identifying challenges and suggesting future research directions.

Abstrakt

Práce se zabývá tématem bezpečnosti a soukromí v kontextu internetu věcí. Zaměřuje se zejména na bezdrátové sítě určené pro zařízení internetu věcí, konkrétně LoRaWAN, Zigbee a Bluetooth Low Energy. Bezpečnostní slabiny a zranitelnosti těchto sítí jsou podrobně analyzovány a je navržen bezpečnostní monitoring na základě frameworku NEMEA. Pro monitorování sítí Bluetooth je představen nový princip, který je podrobně rozebrán od teoretického návrhu až po experimentální ověření a případné vylepšení na produkční úroveň.

Dále je v práci představen návrh zabezpečení proprietární sítě IQRF, který je součástí jejího nového standardu. Tento návrh vychází z výše uvedené analýzy bezpečnostních slabin a výrazně zvyšuje bezpečnost této celosvětově používané sítě.

V neposlední řadě se práce zabývá otázkami ochrany soukromí analýzou přenosu dat čtyř bran internetu věcí určených pro domácí použití. Zkoumá také možné metody anonymizace zařízení IoT v různých typech sítí, identifikuje problémy a navrhuje budoucí směry výzkumu.

Keywords

Internet of Things, Wireless Networks, Security, Privacy, Intrusion Detection, Bluetooth LE, IQRF, Monitoring, Anonymisation

Klíčová slova

Internet věcí, bezdrátové sítě, bezpečnost, soukromí, detekce průniku, Bluetooth LE, IQRF, monitorování, anonymizace

Reference

HUJŇÁK, Ondřej. *Security of wireless communication for IoT devices.* Brno, 2024. PhD thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Doc. Dr. Ing. Petr Hanáček

Rozšířený abstrakt

Množství komunikace v rámci Internetu věcí (IoT) již globálně překonalo tradiční datový provoz a tento trend stále více ovlivňuje naše každodenní životy. Senzory i jiné IoT zařízení pronikly nejen do veřejných, ale už i soukromých prostor, které propojují do celosvětové internetové sítě s cílem zvýšit pohodlí a automatizaci. Tato vzájemná propojenost však přináší i významné výzvy v oblasti bezpečnosti a ochrany soukromí, kterým je tato práce věnována.

Tato práce se zaměřuje na bezdrátové sítě, které byly vyvinuty s ohledem na specifické potřeby a omezení zařízení internetu věcí. Tyto sítě studuje z hlediska bezpečnosti a uvádí zranitelnosti, které v nich byly nalezeny. Do detailu jsou rozebrány specifika a bezpečnostní slabiny tří vybraných sítí - LoRaWAN, Zigbee a Bluetooth Low Energy (LE).

Protože aktualizace firmware zařízení je v prostředí IoT často nedostupná, zaměřujeme se v práci na bezpečnostní monitorování za účelem odhalení případných útoků na síť. Za tímto účelem představujeme modulární řešení postavené nad frameworkem NEMEA a detailně se zabýváme možnostmi takového monitorování v sítích Bluetooth LE. Pro tyto sítě v práci představujeme zcela nový princip, založený na nepřímém sledování spojení mezi zařízeními. Na tomto principu demonstrujeme celý vývojový cyklus od teoretického návrhu, přes experimentální ověření až po vylepšení detekčních schopností na produkční úroveň.

Kromě zvýšení bezpečnosti pomocí monitorování se věnujeme také problematice ochrany soukromí, kterou považujeme za nedílnou součást širší bezpečnosti. Zde provádíme analýzu čtyř IoT bran určených pro domácí použití a sledujeme množství dat která tato zařízení odesílají, stejně jako jejich strukturu a vzory. Dále se v oblasti soukromí věnujeme možnostem zajištění anonymizace IoT zařízení v různých typech sítí a identifikujeme jejich úskalí a možné další výzkumné směry.

Tyto výzkumné aktivity nakonec doplňujeme aplikačním výstupem, který je představen novým standardem pro celosvětově používanou proprietární IoT síť IQRF. Na základě zranitelností identifikovaných v ostatních sítích byly v rámci této práce vyvinuty bezpečností prvky pro tento standard, které výrazně zlepšují bezpečnost budoucích IQRF sítí.

Security of wireless communication for IoT devices

Declaration

I hereby declare that this doctoral thesis is my original work, conducted under the supervision of Doc. Dr. Ing. Petr Hanáček and Mgr. Kamil Malinka, Ph.D. I have thoroughly cited all the literary sources, publications, and other sources used during the preparation of this thesis.

.....
Ondřej Hujňák
August 31, 2024

Acknowledgements

I would like to express my gratitude to Doc. Dr. Ing. Petr Hanáček for his supervision, expertise, and invaluable consultations. I am also thankful to Mgr. Kamil Malinka, PhD., for his insightful tips and consultations. My sincere thanks go to Martin Vychodil for the collaboration with Espressif systems, and to Vladimír Šulc, PhD., for providing the opportunity to work on the successful real-life IoT network IQRF. Additionally, I extend my heartfelt thanks to my family for their continuous support during my research years for the PhD. Their encouragement and support were instrumental in the completion of this doctoral thesis. During the writing process, tools ChatGPT, DeepL and Grammarly were used for editorial and proofreading.

Contents

List of Figures

List of Tables

List of abbreviations

AEAD Authenticated Encryption with Associated Data.

AI Artificial Intelligence.

ANN Artificial Neural Networks.

Bluetooth LE Bluetooth Low Energy.

CBC-MAC Cipher Block Chaining Message Authentication Code.

CCM Counter with Cipher Block Chaining Message Authentication Code.

CRC Cyclic Redundancy Check.

CSMA/CA Carrier sense multiple access with collision avoidance.

CTR Counter.

DoS Denial of Service.

DPA Direct Peripheral Access.

FHSS Frequency-Hopping Spread Spectrum.

FRC Fast Response Command.

GAP Generic Access Profile.

HCI Host-Controller Interface.

IoT Internet of Things.

LoWPAN Low-Power Wireless Personal Area Network.

LPWAN Low-Power Wide Area Network.

LR-WPAN Low-Rate Wireless Personal Area Network.

M2M Machine-to-Machine.

MAC Message Authentication Code.

MitM Man-in-the-Middle.

MLP Multi-Layer Perceptron.

NBAD Network Behavioral Anomaly Detection.

NID Network Identification Number.

OTAA Over the Air Activation.

PoC Proof-of-Concept.

RIDX Rotation Index.

SIEM Security Information and Event Management.

SOHO Small Office/Home Office.

TDMA Time-Division Multiple Access.

TIQBC Time Quanta Bit Coding.

VRN Virtual Routing Number.

Chapter 1

Introduction

With the rapid development of information technology, computing devices have changed significantly, evolving from massive mainframes to compact laptops and embedded systems. In today's era of ubiquitous connectivity, these devices are increasingly connected by various networks that together form the Internet. The relatively recent integration of embedded systems into these global networks is often referred to as the Internet of Things (IoT). The limitations and specific requirements of some embedded systems prevent the use of traditional network approaches. Therefore, new connectivity methods are being developed, which brings new challenges, including guaranteeing security.

The Internet of Things (IoT) refers to a network of physical objects (Things) equipped with sensors, software, and other technologies, enabling them to connect and share data with other devices and systems via the Internet. IoT has significantly advanced several vital areas, including smart cities, Industry 4.0 with Industrial IoT (IIoT), smart transportation, smart buildings, and healthcare. These applications enhance efficiency, safety, and convenience by enabling data-driven decision-making and automation. The Things vary from robust systems utilising a full TCP/IP stack to constrained devices with limited CPU, memory, and power resources.

This work focuses on the constrained devices running specialised firmware and networks designed specifically for IoT applications because their needs and characteristics fundamentally differ from those of workstations and TCP/IP communications. These differences make applying the techniques developed for traditional computing systems directly impractical. Consequently, we will restrict the term IoT to refer to these constrained devices.

The continued growth of IoT networks, particularly wireless networks, is focused on providing connectivity for resource-constrained embedded devices. Due to the emphasis on minimizing power and processing requirements, security measures are often reduced or optimized. For devices for home use, this can lead to the absence of some basic security features for some vendors [98, 65].

IoT network security focuses primarily on three objectives: confidentiality, integrity and availability. Confidentiality aims to ensure that sensitive information that is being transmitted is not accessed by unauthorised parties. Integrity ensures that the information delivered is complete and has not been tampered with. Availability ensures that resources remain accessible to legitimate users.

As IoT networks evolve, security flaws in their protocols or use by vendors are emerging [97, 104]. Applying security patches to deployed IoT devices is significantly more challenging than for traditional devices due to the limitations of embedded systems and the low throughput of these networks. Consequently, vulnerabilities remain exploitable for ex-

tended time periods. For this reason, it is essential to be able to detect attacks in the IoT environment in order to act accordingly.

1.1 Motivation

Main motivation behind this work is the spread of IoT. According to multiple estimates the number of IoT devices will grow rapidly and affect every industry, e.g. IoT Analytics presumes that there will be over 29.7 billion IoT devices in the world by the year 2027 [107] (see the graph in Figure 1.1). The amount of IoT communication in global connectivity already surpassed the traditional traffic and this expansion of IoT devices is increasingly impacting our daily lives at every corner. Beyond industry and public spaces, it also penetrates the home environment with smart sensors, locks, and assistants. This connectivity interconnects private spaces with the global internet network, increasing convenience and automation but also raising concerns about security and privacy.

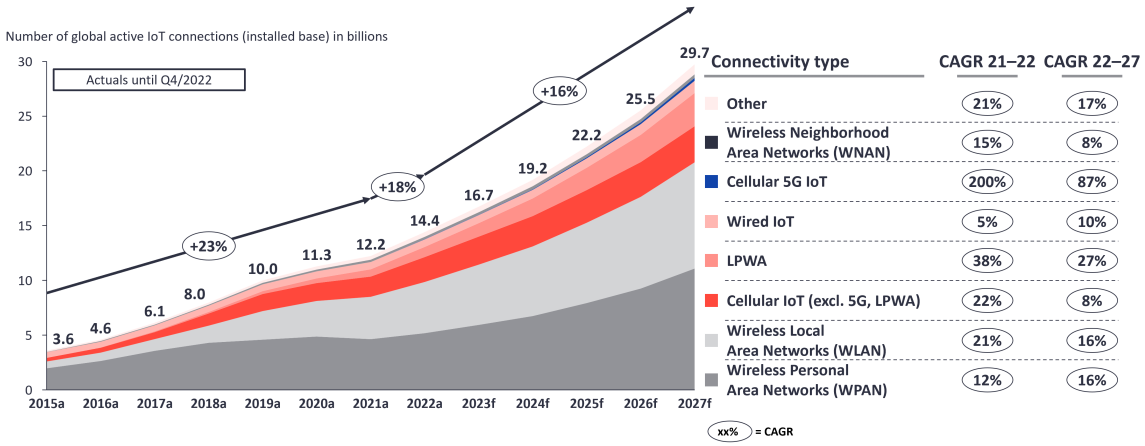

Figure 1.1: Global IoT market forecast (in billions of connected IoT devices) [107]

Wireless connections, in particular, have been shown to suffer from security problems similar to those of the first computer networks (see Chapter 3). Ensuring reliable security for these devices, which often have limited resources, is non-trivial and error-prone. In addition, manufacturers of household and consumer-oriented products tend to disregard security measures to reduce production costs and not take advantage of all available security features [65, 98]. But even when manufacturers adhere to security standards, the level of assurance and the ability to detect breaches are not always perfected, as illustrated by the following case study.

1.1.1 Smart lock example

Take the Danalock V3 smart lock [37] as an example. This IoT device is designed for wireless intelligent monitoring and control of rental homes and resorts in the sharing economy. The main advantage over traditional locks is the ability to control access by creating and removing digital keys for guests as needed. Guests can then unlock the property with any Bluetooth Low Energy (Bluetooth LE) enabled smartphone running the companion app. In addition to Bluetooth LE, the Danalock integrates with various other wireless IoT networks, increasing its compatibility within the smart home ecosystem.

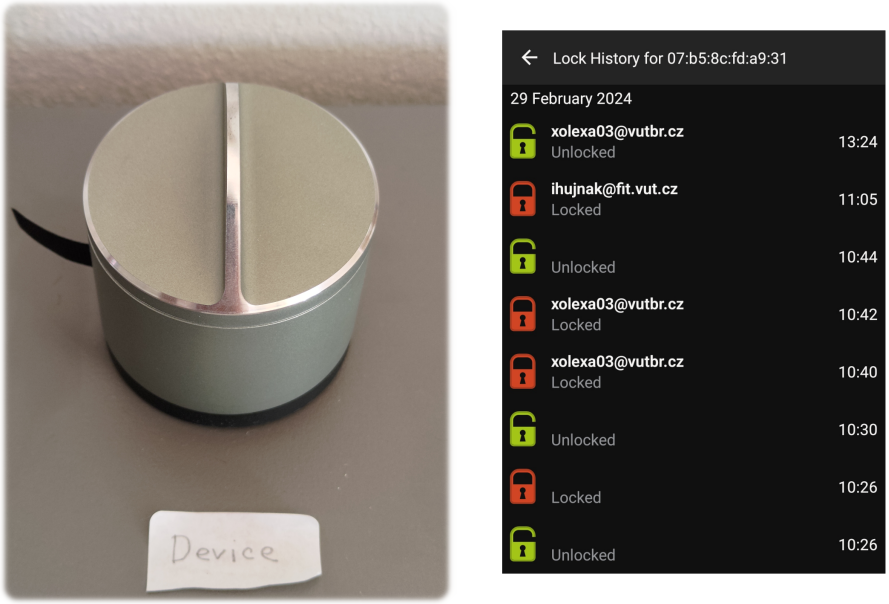

Figure 1.2: Danalock V3 with the Lock History log

As a home security device, the lock uses all the security features the communication protocols provide. However, our testing revealed monitoring limitations. The device provides the owner a Lock History log containing an audit trail of lock and unlock events. This log is stored exclusively in the cloud, and the responsibility for recording these events lies with the control application. Therefore, if the user is offline when unlocking the device, this action is not recorded in the log. This omission could be exploited by adversaries who would leave no trace of unauthorised access attempts in the log.

In addition, the Lock History log records only successful lock and unlock events without including any information about failed attempts, which allows an adversary to perform a covert attack. In addition, we found that when a guest is denied access, its identifying information is deleted from the log, complicating later analysis.

It is important to note that our testing did not extend to scenarios involving complex smart home integration. It is possible that better monitoring could be achieved by connecting the lock to a ZigBee/Z-Wave network. However, an attacker could always reduce such a scenario to ours by making the connection impossible, e.g., by jamming.

This case study illustrates that even IoT devices that are designed for object security often do not have all security aspects well covered. In particular, the area of monitoring or auditing is often omitted.

1.1.2 Privacy

In addition to security issues, the IoT also raises significant privacy concerns due to the vast amount of data collected by these devices. Analysing this big data can reveal previously undetectable patterns [78]. With the penetration of IoT into the home environment, these patterns can reveal information about users' private behaviour.

Currently, most home IoT ecosystems operate within a cloud infrastructure, which means that service providers always have access to at least some telemetry data or metadata. Ensuring user privacy is then mainly up to the service provider, and users have limited

options to minimise data tracking. Legislation such as the EU GDPR [5] seeks to address these privacy concerns. For this reason, in this paper, we also address the issue of privacy assurance and analyse the behaviour of home IoT gateways in terms of the data sent and the potential use of anonymisation in IoT networks.

1.2 Dissertation focus and contributions

The number of devices connected by IoT networks is growing rapidly, and this is expected to continue[1]. Given the growing popularity and the number of security incidents revealed in these networks [69], we consider it essential to improve the security level of these devices.

Based on the motivation presented, we decided to focus primarily on the smart home environment, given the growing popularity of smart home features. In addition, we take into account the rise of the sharing economy, such as peer-to-peer rentals (e.g. AirBnB), and Small Office/Home Office (SOHO) environments, which share many features with smart homes and often use the same technologies.

The work builds on these four main points that motivates us in the upcoming research:

- Wireless communication for IoT is known to be insecure.
- Security patches in IoT environment are very rarely applied.
- IoT devices often do not pose the computational power for proper security hardening based on complex schemes.
- To reveal security breaches in a wireless IoT environment without increasing the burden on battery-powered IoT devices, we need to be able to monitor the network from a mains-powered device.

There are two ways to improve the security of wireless IoT networks. The first approach involves enhancing the security features of IoT networks themselves. While this is a standard practice for all networks, it primarily addresses security for the future. Devices that are currently deployed may not support these updates, leaving them vulnerable until they are replaced. We demonstrate the process of developing such security mechanisms for a wireless IoT network in Chapter 6, where we discuss the security design of a proprietary IQRF network.

The second approach is through security monitoring, which enables users to detect incidents on networks or devices even if they are not up-to-date. This type of monitoring is not generally available in IoT networks, and our work aims to address this deficiency. In Chapter 4 we introduce an architecture of Secure gateway for IoT (SIoT), which provides such monitoring capabilities. Our goal is to provide consumers with a security mechanism that will enhance the security of their IoT environment and provide end-users with a secure, privacy-preserving platform.

Our work is providing significant contributions in the following areas:

- Development of an innovative monitoring technique for Bluetooth Low Energy (Bluetooth LE) networks. The full development cycle is shown, starting with initial concept design and ending with laboratory testing. For the testing phase, we have collected and published a dataset that includes a range of Bluetooth LE devices.

[1] https://iot-analytics.com/number-connected-iot-devices/

- Design of security features for an upcoming version of a proprietary IoT network used in real-world applications.

Other contributions of this work can be summarised as follows:

- We provide an overview of the security issues of selected contemporary wireless communication technologies.
- We performed analysis of network communication of home IoT gateways with focus on privacy leaks. We compare the amount of telemetry data and identify patterns, which can be used for fingerprinting or reveal dubious behaviour.
- We provide strategies for anonymising IoT devices in different types of networks.

The work presented in this dissertation is based on the following publications of the thesis author that have been accepted and published at international conferences:

- *Indirect Bluetooth Low Energy Connection Detection [57]*

  Proposal of a novel monitoring method in Bluetooth Low Energy networks, which is based on indirect monitoring of the device connections by listening to its advertisements. Validation of the proposed method on a proof-of-concept implementation.

  Author contribution: All parts of the process, from conceptualisation to writing.

  Published in: 2023 International Conference on Information Networking (ICOIN). 2023, p. 328–333

- *Security survey of the IoT wireless protocols [69]*

  Discussion of security issues of four currently used IoT network protocols - LoRaWAN, ZigBee, Z-Wave and Bluetooth LE. Also includes the evolution of security for these networks.

  Author contribution: Bluetooth and Zigbee network analysis, writing and presenting the paper.

  Published in: 2017 25th Telecommunication Forum (TELFOR). 2017, p. 1–4

- *Security Framework for IoT and Fog Computing Networks [108]*

  A security framework proposal for intrusion detection in IoT networks. The framework incorporates the principle of fog computing by enabling distributed execution with separate (pre-)processing.

  Author contribution: BeeeOn System and Bluetooth LE use-cases, writing and presenting the paper.

  Published in: 2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC). 2019, p. 87–92

- *IoT Gateways Network Communication Analysis [121]*

  Analysis of the communication patterns of four commercially available IoT gateways designed for home use. Comparison with similar studies highlighting differences indicating rapid evolution in device behaviour. Includes a dataset of measured data.

Author contribution: Conceptualisation, supervision of the experiment, results evaluation and publication preparation.

Published in: 2023 International Conference on Information Networking (ICOIN). 2023, p. 334–339

- *Survey of Privacy Enabling Strategies in IoT Networks [53]*

  This study explores the anonymization possibilities for IoT devices, focusing on both IP and non-IP networks. Two different strategies are proposed for IP devices based on the device capabilities. In the context of non-IP devices, the study explores LoRaWAN and proposes anonymization strategies for both devices and gateways. In the case of ZigBee, privacy issues are reduced to those of IP devices, with an emphasis on the problems associated with using public networks for Internet connectivity.

  Author contribution: IoT networks analysis and expertise, co-authoring the ideas and writing the publication.

  Published in: Proceedings of the 2017 International Conference on Computer Science and Artificial Intelligence (CSAI). 2017, p. 216–221.

Except for the academic publications at international conferences, the thesis author participated on the following technical paper from the practice:

- *IQRF Communication Standard*

  The new IQRF Communication Standard Specification that is an evolutionary step from the IQRF OS 4.06 specification. The author was responsible for the Security services (SES) layer.

  Author contribution: Chapter 10 – Security specification

In addition, this dissertation includes original research that is currently undergoing peer-review or publication. This includes the following publications:

- *Parallel BLE Advertising Monitoring*

  In this work we developed an advanced monitoring probe capable of simultaneously capturing data from all three primary advertising channels. The main contribution of this paper is a comprehensive dataset involving nine different devices, with a total of 510 samples captured by both the newly developed probe and a common controller. Upon comparing the data, we concluded that while the advanced capture technique slightly improves data quality, it does not substantially enhance connection detection capabilities.

  Author contribution: All parts of the process. Conceptualisation, probe development, dataset measurement and writing.

  Submitted to: The 57th International Carnahan Conference on Security Technology (ICCST), 2024.

- *Machine Learning Supported Bluetooth Low Energy Connection Monitoring*

  The paper contains a comprehensive evaluation of several detection methods, including statistical methods (ARIMA, GMM), outlier detection methods (LOF, iForest, and OCSVM), and a MLP classifier. For each method, we identified the universally

best performing parameters and compared the achieved results using the F1-Score as the primary metric. We can conclude, that the MLP classifier demonstrated exceptional ability to differentiate between Bluetooth connections and normal traffic.

Author contribution: All parts of the process. Conceptualisation, methods implementation, investigation and writing.

Submitted to: IEEE Consumer Communications and Networking Conference (CCNC), 2025.

1.3 Thesis structure

Following this introduction, Chapter 2 focus on presenting wireless networks within the context of IoT. It outlines the fundamental characteristics of these networks and categorises them based on area coverage and technology. Three selected networks – LoRaWAN, Zigbee, and Bluetooth – are then examined in detail.

After an initial introduction to networks, Chapter 3 provides an overview of the security issues identified in IoT networks. To aid comprehension, these issues are categorised within a taxonomy. The chapter first addresses general attacks that can impact multiple wireless networks, and subsequently focuses on specific vulnerabilities within the selected networks in individual sections.

In Chapter 4, we focus on the issue of monitoring for the purpose of intrusion detection. We introduce the basic concepts of intrusion detection in IoT networks and the challenges associated with fog computing. Additionally, we propose a security monitoring architecture based on the NEMEA system, a platform designed for real-time traffic analysis and anomaly detection.

Chapter 5 is the core of the dissertation, focusing on the security monitoring of a specific IoT network—Bluetooth. This chapter begins with an introduction to the issue and a discussion of the approaches chosen by other researchers. In Section 5.2, we explain in detail the necessary Bluetooth LE operations and the principles on which our monitoring approach is based. This is followed by a description of the development process, starting with proof-of-concept validation in Section 5.4, then the development of a specialised hardware tool in Section 5.5, and finally, the identification of a suitable detection method in Section 5.7.

The following chapter, Chapter 6, focuses on designing security features for a selected proprietary network, specifically IQRF, for its new standardised version. We begin by describing the specifics of the IQRF network as it existed prior to the establishment of the standard, which we will refer to as „IQRF Legacy.“ Section 6.2 then introduces the new IQRF Communication Standard and discusses the security features that were designed as part of this work. An evaluation of the impact of these security features is provided in section 6.3.

Although the main focus of the thesis is the security of IoT, we believe that the issue of privacy is inherently connected, and we address it in Chapter 7. To assess privacy leaks, we conduct an analysis of the communication of home IoT gateways, the description and results of which are presented in Section 7.1. We also delve into the area of anonymisation, where in Section 7.2, we propose various options for integrating IoT with anonymisation networks, highlighting potential issues and consequences.

Finally, in Chapter 8, we provide a comprehensive conclusion to the study by revisiting the main objectives and summarising the key findings and contributions made to the field.

Chapter 2

Wireless networks for Internet of Things (IoT)

To integrate various devices into the IoT ecosystem, some form of communication technology is imperative. This chapter discusses the different types of IoT networks and their unique characteristics. The primary distinction between IoT and traditional Internet communication is the prevalence of Machine-to-Machine (M2M) communication. Unlike human-centric communication, M2M communication typically comprises regular communication patterns with sparse connections in which only small amounts of data are exchanged.

A considerable number of IoT applications require the devices to be mobile or located in places that may be difficult to access, which requires the use of wireless networks. The concept of using wireless networks to connect sensors is not new; IoT wireless networks can be considered an evolution of wireless sensor networks (WSN) that have evolved and integrated with the Internet. However, nowadays, even static devices that could use wired connections often use wireless because of the convenience and lower cost of building a network. As a result, wireless networks represent the fastest-growing segment of IoT networks [107].

Unlike the traditional Internet, which uses a relatively small number of globally accepted connectivity technologies, wireless IoT networks represent a wide range of available solutions [87]. These networks are usually developed for narrowly focused applications, and their features are tailored to their use case. Even for similar use cases, there are often several different technologies available that use slightly different approaches. Until the different networks are unified into one or a few dominant ones, the effort to optimise them and solve security problems is split between them. Today, consumers who want to use IoT devices are faced with the choice of choosing a specific network, thereby limiting the number of compatible devices or creating a multi-network solution.

Because IoT-specific networks are designed to meet the needs of connected devices, they typically prioritise low power consumption and often include on-chip radio implementations with low data rates. Standard techniques also include a variety of battery-saving modes, such as extended hibernation between transmissions. Many of these networks operate in unlicensed frequency bands to ease market penetration, often using the ISM (Industrial, Scientific, and Medical) radio spectrum, which is subject to restrictions on the medium access control.

Below, we present two classifications of IoT wireless networks based on different criteria: the area of coverage and the technology used. We have chosen three representatives of IoT networks – LoRaWAN, Zigbee, and Bluetooth. For each representative, we provide

an overview of its basic characteristics and operation, with a particular emphasis on security measures. This detailed security information is vital for the subsequent chapters as the thesis focuses on security.

2.1 Division by area coverage

Computer networks are commonly classified based on their coverage area. Traditionally, this includes Wide Area Networks (WAN) that cover large geographical areas, such as countries. Metropolitan Area Networks (MAN) span cities or campuses, while Local Area Networks (LAN) typically cover a single building or office. The term Personal Area Network (PAN) is used for networks that connect devices within a short range.

In wireless IoT, networks are generally divided into two main groups: Low-Power Wide Area Network (LPWAN) and Low-Power Wireless Personal Area Network (LoWPAN) [16, 95]. The meaning of LPWAN in IoT is similar to the concept of WAN, while the definition of LoWPAN has strayed from PAN. The term „personal“ only remains due to historical reasons within the IEEE 802 task group naming, rather than indicating that LoWPANs are designed for personal use within a short range [24]. LoWPAN networks are often capable of covering much larger areas, similar to traditional LANs. Additionally, with the widespread use of mesh topologies, these networks can now span entire streets or neighbourhoods.

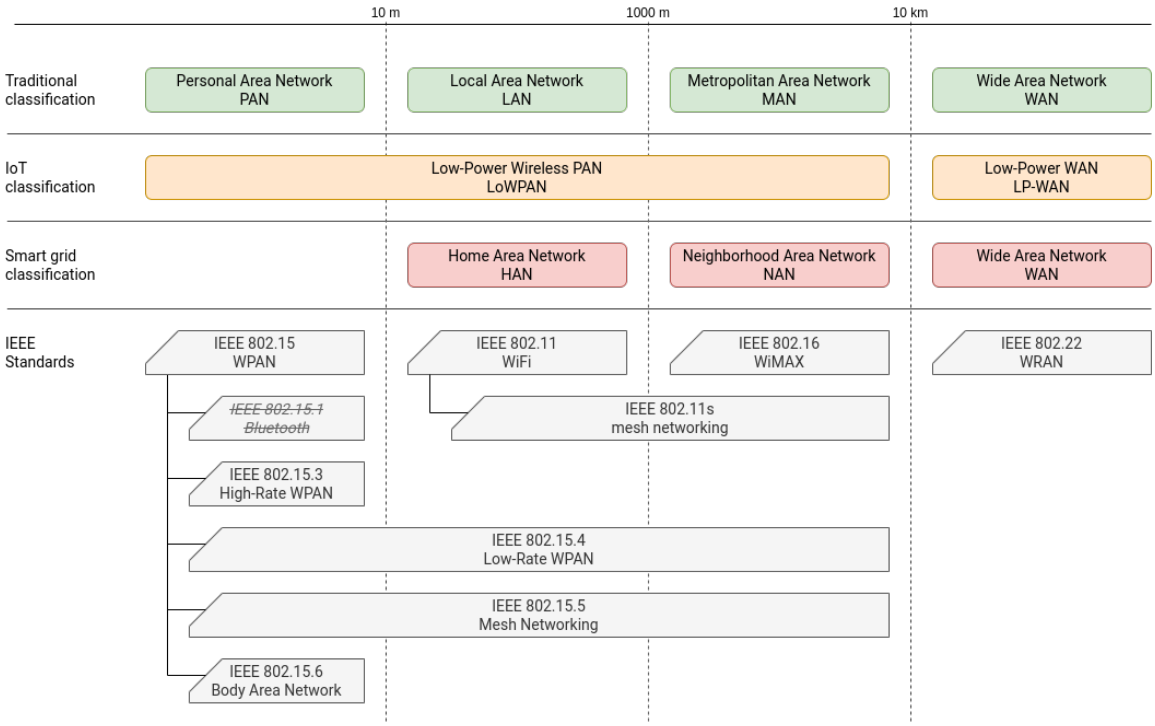

Figure 2.1: IoT Networks classification by area coverage

This discrepancy in using the term „personal“ in LoWPAN often causes confusion and inconsistencies. This is particularly noticeable in a smart grid environment, where the network architecture is structured from Home Area Networks (HAN) through Neighborhood Area Networks (NAN) to WAN [71], intentionally avoiding the term „personal“.

An overview of IoT network classifications is provided in Figure 2.1, along with the IEEE 802 task force standards for illustration.

2.1.1 Low-Power Wide Area Network

LPWANs are infrastructure networks with regional to global coverage that enables long-range communication at a low bit rate among connected devices, which can move freely in the network. In terrestrial networks, powerful base stations are spread throughout a locality and connect to a concentrating cloud that provides access to network services. In satellite networks, these base stations are substituted with satellites, resulting in higher latency and power consumption but extending coverage to remote areas. The infrastructure for LPWANs is usually owned by a service provider, which offers connectivity as a service.

Examples of LPWAN technologies include LoRaWAN [21], SigFox [125], Narrowband-IoT (LTE Cat NB1) [93], and LTE Cat M1 [54]. While these networks are primarily terrestrial, Narrowband-IoT and LoRa are also utilised for satellite connections [32, 117].

2.1.2 Low-Power Wireless Personal Area Network

LoWPAN encompasses a variety of different networks, from those that cover factories and households to those that connect just a few devices. The user typically owns and manages the network infrastructure, consisting of several nodes and a gateway (often called a hub). This gateway administers the network and provides connectivity between the specific IoT network and the wider LAN or Internet. Although the gateway can be connected to a cloud service for a better user experience, such a connection is generally not necessary for network operation. This category also includes ad-hoc networks temporarily created to connect nearby devices.

Many different networks are used in this IoT segment, the most widely used being Bluetooth [7], Zigbee [123] and Z-Wave [87, 45]. The most common standard many of these networks are based on is LR-WPAN [6].

2.2 Division by the technology

As the term IoT is very broad, it includes both devices using the same technologies as the traditional Internet, i.e. Internet Protocol (IP), and devices using protocols developed specifically for IoT. Such specialised protocols then require a translation layer for connectivity; otherwise, the devices within the network will not be accessible from the outside world.

2.2.1 Internet Protocol

If the Thing is powerful enough or provides a service that requires a direct user connection (e.g. webcams), it facilitates an IP network interface. Such a device can be directly connected to the traditional Internet via Ethernet, IEEE 802.11 (WiFi) or any other connection (see Figure 2.2). Communication with such devices is no different from regular Internet traffic, and such devices can be considered lightweight servers that provide services.

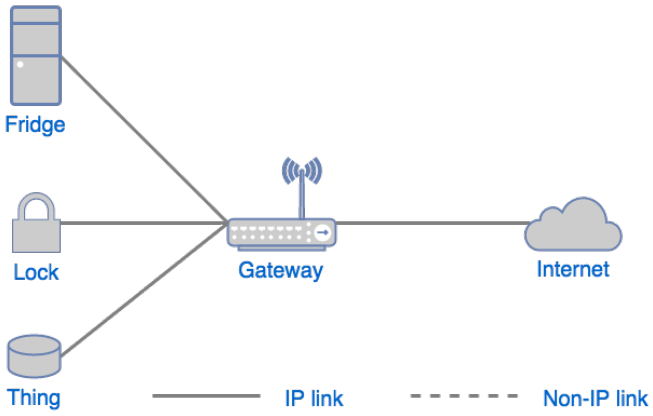

Figure 2.2: IP device network architecture

2.2.2 Custom protocols

Due to the constraints of IoT devices, specialized network protocols have been developed that reduce power consumption and computational demands or are designed for specific applications. These protocols are often incompatible with the Internet, thus requiring a translation layer. This layer is typically implemented in a gateway node between the IoT network and the Internet or as a cloud server, enabling communication between IoT devices and the outside world (as depicted in Figure 2.3).

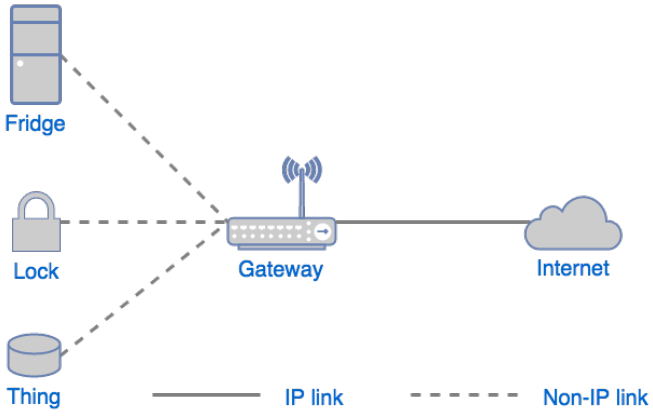

Figure 2.3: Non-IP device network architecture

While some of these protocols are publicly available as open standards, many remain proprietary and are maintained by their respective organisations. Examples of open standards are LoRaWAN, although based on the proprietary LoRa modulation technique, and Bluetooth. These protocols are described in detail later in this chapter. Zigbee serves as a hybrid example, which is based on the open LR-WPAN standard [6] but includes proprietary upper layers. Typical proprietary IoT networks include Z-Wave and SigFox.

In this thesis, Chapter 6 covers the proprietary mesh network IQRF. Designed for industrial applications, it is tailored for smart cities, buildings, and Industry 4.0. The reason for this specific focus is mainly due to the author's participation in the security design of its upcoming version.

2.3 LoRaWAN

We selected LoRaWAN as a representative of LPWAN networks due to its wide use and independence from mobile networks. This network, intended for large-scale IoT deployments and managed by the LoRa Alliance [21], uses a proprietary CSS (Chirp Spread Spectrum) modulation known as LoRa (Long-Range) with an adaptive spread factor [77]. LoRa operates in several sub-GHz ISM bands, which vary by country and can be further divided into multiple channels. Although low-frequency bands mean slower transmissions, they increase energy efficiency and range. LoRaWAN's adaptive data rate ranges from 0.3 kbit/s to 50 kbit/s, with its most significant advantage being its range, which can be up to 20 km, depending on the environment.

LoRaWAN uses the star-of-stars network topology shown in Figure 2.4, where end devices communicate directly with gateways using single-hop wireless connections. The device-to-infrastructure communication is sent as a broadcast message and all gateways that receive data forward it to a concentrating cloud Network Server over standard IP connections [51]. This allows for scalability, redundancy, and also localisation based on the known location of the gateways that received the broadcast. The cloud server then communicates with the Application Servers, and since the communication is bi-directional, the application can respond to the device. The Network Server or its APIs handle all interactions with devices.

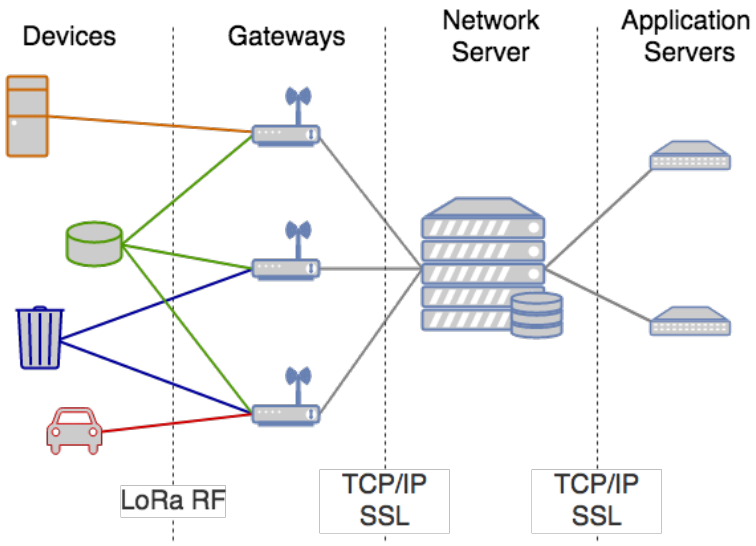

Figure 2.4: LoRaWAN architecture

- **Devices** – The devices in the network, which can move freely within the infrastructure.

- **Gateways** – Base stations that create the core of the network by relaying messages between Devices and the Network Server. The interface communicating with the Devices uses LoRa RF, while the upstream connection to the Network Server typically uses the Internet Protocol.
- **Network Server** – A central point where all Gateways communicate to transfer data between Devices and Applications. It concentrates data from all Gateways, deduplicates messages, and ensures communication with the Devices.
- **Application Server** – All required user functionalities are implemented through applications running on application servers. The application works on the basis of data received from the Device or sends back some instructions to the Device via the Network Server.

Depending on the power limitations and required availability for connections, LoRaWAN distinguishes three classes of end-devices [51]:

- **Class A** is the most restricted. These devices are in battery saving mode for extended periods of time and will only wake up to transmit when needed. After transmitting a message, they listen for incoming messages from the network before returning to battery saving mode.
- **Class B** represents devices that regularly wake up and listen to network messages. The *beacon* messages are used to synchronise the listening time windows with the gateways.
- **Class C** contains powered devices that are always online and therefore ready to receive messages.

Each device in LoRaWAN has several identifiers. DevEUI (Device Extended Unique Identifier) is a 64-bit long globally unique number used for unambiguous device identification. For addressing within the network, a network unique 32-bit DevAddr (Device Address) is assigned to the Device. There are two approaches how this address is obtained [56]:

- **Activation by Personalisation (ABP)** refers to the process by which all identifiers are preloaded into the device during provisioning. Such a device is locked to a specific network and to change the address, the device needs to be physically connected and reflashed, which is sometimes impossible after deployment.
- **Over the Air Activation (OTAA)** denotes the process by which devices acquire network-specific identifiers through a join request. This join request sent to the Network Server contains two identifiers - DevEUI and AppEUI[1] (stands for Application EUI), along with a nonce. After validating the request, the Network Server assigns the device a DevAddr. This mechanism allows the device to connect to different networks as needed, provided it is registered with the appropriate network server.

LoRaWAN security is designed to align with the general LoRaWAN design criteria of low power consumption, low implementation complexity and high scalability. The security framework is built on three pillars: payload encryption, message authentication code (MAC), and frame counter. The 128-bit AES block cypher was selected for both encryption and MAC calculation [48].

[1] Renamed to JoinEUI in version 1.1

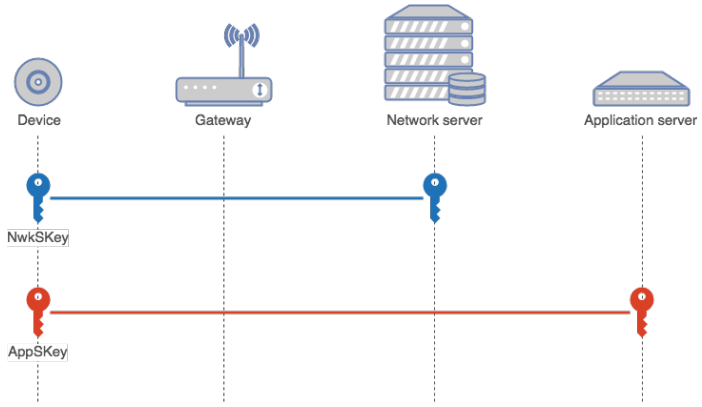

Figure 2.5: LoRa two stage encryption

Encryption in LoRaWAN networks is a two-step process, as shown in Figure 2.5. Both stages use the counter mode (CTR) of the AES cypher with different keys. The first stage, known as the network stage, uses NwkSKey (network session key) to ensure the integrity of all network frames and the encryption of network commands. The second stage involves End-to-End encryption of application data using a specific AppSKey (application session key). These keys are exchanged in the same manner as DevAddr addresses. In the case of OTAA, an additional key called AppKey (application key) is required. This key is never transmitted and is used to mutually authenticate the Device and the Network Server by calculating the MAC code of the connection request using this key. For LoRaWAN 1.1, the key management scheme was reworked to improve the security by separating key management from the Network Server into a special Join Server and introducing a NwkKey (network key) serving as a root key for derivation of NwkSKey and other special keys [56].

Each message includes a MAC calculated using CMAC (Cypher-based Message Authentication Code) mode with the NwkSKey to ensure the message has not been tampered with. In addition, frame counters are used to prevent replay attacks. Each device maintains an incrementing frame counter contained in the message header that the Network Server uses for message deduplication and replay attack prevention.

2.4 Zigbee

Zigbee is a well-established IoT wireless network focused on short-range connectivity, which is frequently used in both industrial applications and smart homes for local automation. The Zigbee specification, maintained by the Zigbee Alliance, was developed specifically for the IoT, with its main advantages being simplicity, low power consumption and mesh networking that allows it to cover larger areas. Figure 2.6 shows that the Zigbee protocol stack is built on top of the *IEEE 802.15.4* standard for LR-WPAN, which provides a physical and link layer.

2.4.1 IEEE 802.15.4

IEEE 802.15.4 [6] is an open standard for LR-WPAN that serves as a framework for this type of network by defining the physical and data link layers of the ISO/OSI model. The standard operates in the ISM bands, supporting both sub-GHz and 2.4GHz bands, which can lead to interference with IEEE 802.11 (WiFi) and other technologies. By supporting

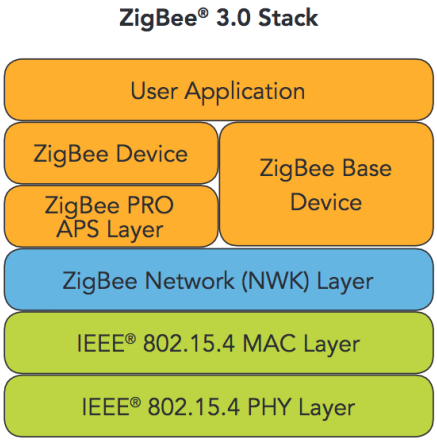

Figure 2.6: Zigbee protocol stack [8]

the 2.4GHz bands the standard supports connection speeds up to 250 kbit/s. To minimise the data loss caused by interference, collision avoidance (CSMA-CA) is used, so the devices test the medium before transmission and only send messages when the medium is clear.

The standard distinguishes between two device types:

- **Full Function Device (FFD)** is a device with full protocol support. Such a device is capable of communicating with all other devices and can relay messages to form a mesh network.
- **Reduced Function Device (RFD)** can only serve as an end device and communicate only with the FFD. These devices are usually very simple and battery powered.

These devices can be arranged in two topologies. In the star topology, all devices communicate with a central node (FFD) that coordinates the network. However, when multiple devices are involved in delivering messages, they form a peer-to-peer network, which enables flexible and scalable networks by eliminating a single point through which all data must pass.

Each IEEE 802.15.4 device is identified by a globally unique 64-bit EUI address, often referred to as a long address. This address is assigned at the time of manufacture and is immutable, so it can be used to identify across different IEEE 802.15.4 networks. For communication within the network, a 16-bit short address (also known as a node ID) is assigned when a device is connected to the network. This address reduces the overhead of network metadata and is sufficient if it is unique within a given network.

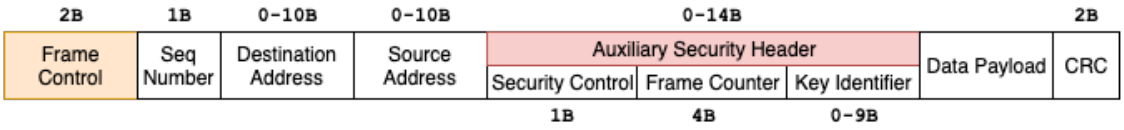

Figure 2.7: Detail of security header of IEEE 802.15.4

Security in the standard is enabled by a flag in the frame control byte and is managed through an auxiliary security header. This header, illustrated in Figure 2.7, contains the

security level, the identification of the key used, and the 32-bit frame counter. There are eight security levels [6]:

- **Level 0** means no security.
- **Levels 1-3** indicate that the frame is authenticated using a Message Authentication Code (MAC) to ensure data integrity. The length of the code varies from 32-bit to 128-bit, depending on the level.
- **Level 4** was used for payload encryption without authentication, but has been deprecated and shall not be used any more.
- **Levels 5-8** combine the payload encryption to ensure confidentiality with frame authentication. Again, the level corresponds to the length of the MAC tag.

The specification mandates the block cypher Advanced Encryption Standard (AES) with a 128-bit block length, as specified in FIPS Pub 197. Authenticated Encryption with Associated Data (AEAD) mode CCM* is used for encryption and authentication. This mode coincides with the original specification for the combined counter with CBC-MAC (CCM) mode of operation[2] and is extended to support all the specified security levels with different MAC tag lengths and encryption-only mode. The 104-bit nonce for CCM* is defined as:

$$Nonce = long\ address \mid\mid frame\ counter \mid\mid security\ control \text{ [6]}$$

The frame counter serves a dual purpose: ensuring the uniqueness of encryption nonces and preventing replay attacks. Each device possesses a unique outgoing frame counter and keeps a record of the frame counters of its neighbours. Only messages with valid frame counters are accepted.

2.4.2 Zigbee layers

Building on the IEEE 802.15.4 standard, Zigbee defines the network layer along with additional supplementary host layers [124], as shown in Figure 2.6. The primary responsibility of the network layer includes managing the multi-node PAN network, routing messages, and setting security protocols. The host layers provide application support through standardised data formats for commands, control, and data exchange between devices. Since the initial Zigbee standard, the host layers have evolved significantly. Initially, various sets of attributes were defined based on use cases and organised into the Zigbee Cluster Library [8]. Applications had to choose specific clusters based on their functionality, such as Zigbee Light Link (ZLL), Zigbee Home Automation (ZHA), or Zigbee Building Automation (ZBA). With the introduction of Zigbee 3.0, these clusters were deprecated and consolidated into a single standard to promote interoperability.

Unlike the underlying standard, Zigbee differentiates nodes based on their roles [44]:

- **Coordinator** is a FFD responsible for establishing and managing the network. There is always exactly one coordinator in Zigbee PAN and it can serve as a gateway between the Zigbee network and Internet.

[2]NIST SP 800-38C: Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality (https://csrc.nist.gov/pubs/sp/800/38/c/upd1/final)

- **Router** is also an FFD that is capable of continuously receiving and transmitting messages. Routers participate in networking by routing messages between devices and thus extending the range of the network.
- **End Device** is not involved in networking, so it does not need to continuously receive or transmit data, which ensures very low power consumption. However, it can be both RFD and FFD.

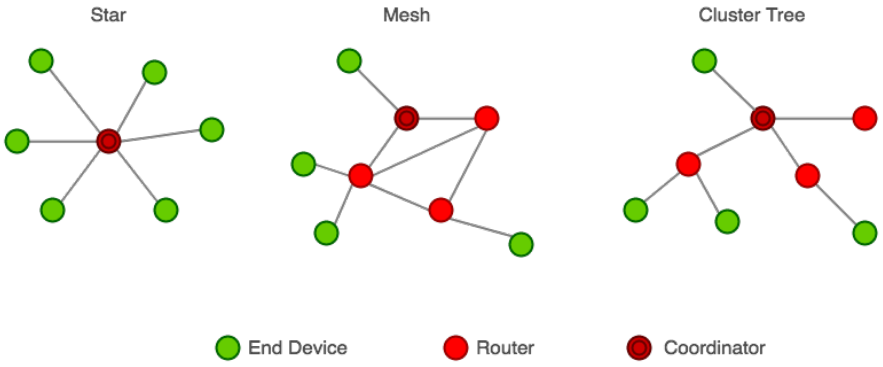

Figure 2.8: Zigbee topologies

Zigbee supports both star and peer-to-peer topologies, with the latter being renamed as mesh topology. Additionally, Zigbee defines another topology called the cluster tree [110]. This cluster tree topology organises the network in a hierarchical tree structure, which simplifies network management and routing. The overview of supported topologies can be found in Figure 2.8.

Although the protocol does not support nodes freely moving within the topology, it leverages the Ad-Hoc On-Demand Distance Vector (AODV) for routing [110]. This enables dynamic reconfiguration and recovery if some nodes become unresponsive. AODV routing uses special *router request (RREQ)* packets to determine the optimal path, and each router constructs its routing table with the next hop address based on the best cost calculated from the Received Signal Strength Indicator (RSSI) of responses.

In Zigbee, networks are identified by PAN ID and device addressing relies on IEEE short addresses (also known as NWK addresses). The coordinator has always address 0, while the device generates the address stochastically. To resolve conflicts, a particular post-join procedure *device_annc* is implemented, where device broadcasts a message containing its long and short addresses, and if a conflict is detected, all devices with conflicting addresses will re-generate their short addresses [123].

The Zigbee security architecture builds on the basic mechanisms defined in the IEEE 802.15.4 standard and primarily provides key management above them. In addition, some additional security features are added, such as the Touchlink protocol mentioned later in this section.

There are three types of keys used in ZigBee [124, 97]:

- **Master key** was initially a factory-issued, pre-loaded long-term key, which served for the exchange of keys through Symmetric-Key Key Establishment (SKKE) and also

secured Over-The-Air (OTA) upgrades. However, due to the leakage of some master keys, they were replaced by pre-loaded trust center link keys called install code.

- **Network key** is shared within the whole network and secure the network communication. This key is acquired in the join procedure and protected by the master key or a trust center link key.
- **Link key** secures unicast communication between two devices within a network. There are two types of link keys: trust center link keys, which protect communication between devices and trust centers, and application link keys, which protect communication between any two non-trust center devices.

Management and distribution of these keys is heavily dependant on the security model used. Zigbee supports two distinct security models [100] and every network has to select the model on creation. The security model cannot be changed once the network is established.

- **Centralised Security Model** contains a dedicated security management node known as the Trust Center (TC). The TC is responsible for key distribution, device authentication, and access control, ensuring that only authorised devices can join and communicate within the network. Each node must have a trust center link key for secure communication with the TC. This key is derived from the *install code* exchanged out-of-band prior to joining the network if the codes are required by the TC.
- **Distributed Security Model** is simplified and can be used for ad-hoc networks. The role of TC is shared among all routers and the nodes are authorised by the shared distributed trust center link key. The network key can be obtained from any router and the access control in this mode is limited.

Among the additional security protocols provided by Zigbee, we mention ZLL Touchlink. This protocol simplifies the process of connecting lights to Zigbee networks by using proximity-based pairing [97]. When devices are in pairing mode, the RSSI (Received Signal Strength Indicator) is checked to determine the distance between them. If the distance is sufficiently short, the new device is added to the network using a predefined Touchlink link key.

2.5 Bluetooth

Bluetooth is a short-range wireless communication link maintained by the Bluetooth Special Interest Group (Bluetooth SIG) [7]. Initially developed to replace wired serial links, Bluetooth was standardised as IEEE 802.15.1 but has since diverged from this specification. Its integration into smartphones facilitated its widespread adoption in the consumer market and made it a ubiquitous method of controlling consumer-oriented devices. In addition, this spread enables indoor localisation using Bluetooth beacons.

In 2010, the Bluetooth 4.0 specification introduced a new variant known as Bluetooth Low Energy (Bluetooth LE), also marketed as Bluetooth Smart. Bluetooth LE technology was developed specifically for power-constrained devices such as wearables and IoT sensors, which is why we focus on this version. The original Bluetooth technology is mostly referred to as Bluetooth Classic to distinguish it from this new variant. Although the two variants share the same radio and some features [2], they are incompatible with each other, as seen

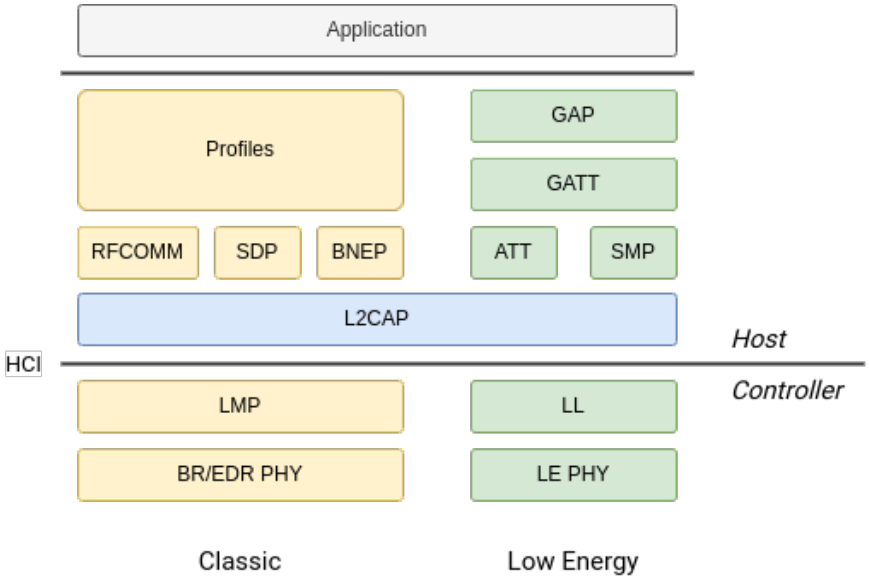

Figure 2.9: Bluetooth protocol stack

in the protocol stacks in Figure 2.9. Thus, if a device such as a smartphone aims to support all Bluetooth devices, it must implement a dual-stack.

Due to strict timing requirements, the lower OSI layers (physical and data link) are implemented within the Bluetooth controller. The upper layers are managed by the host, with the Host-Controller Interface (HCI) acting as a communication bridge between the two components [9]. The Logical Link Control and Adaptation Protocol (L2CAP) is shared in both variants. However, while Classic has specific profiles corresponding to the different functionalities defined in the upper layers, Low Energy unifies the stack into more generalised modules. It includes Attribute Protocol (ATT) and Security Manager Protocol (SMP), which are layered under Generic Attribute Profile (GATT) and Generic Access Profile (GAP).

Bluetooth operates in the international ISM 2.4 GHz band, which is shared with various other radio technologies [9]. To mitigate potential interference and manage multiple media access, Bluetooth employs a fast Frequency Hopping Spread Spectrum (FHSS), with hops occurring up to 1600 times per second [4]. Because of the speed of hops, a packet transmission can span over multiple hops. Hops are pseudo-random and follow Adaptive Frequency Hopping (AFH), meaning the channel map is exchanged, and insufficient-quality channels are skipped [4].

Both LE and Classic support the same basic modulation - Gaussian Frequency-Shift Keying (GFSK) with 1 Mbit/s in their physical layers (PHY). In Classic, this modulation is used in Basic Rate (BR) radio mode, while Enhanced Data Rate (EDR) supports two Differential Phase Shift Keying modulation techniques ($\pi/4$-DQPSK and 8DPSK), allowing for faster transmission speeds (2 and 3 Mbit/s respectively) [2]. While LE does not support different modulation, in Bluetooth 5 a faster 2 Mbit/s PHY was introduced, together with slower Coded PHY (500 or 125 kbit/s depending on the symbol size) utilising forward error correction for extended range [9].

BLE and Bluetooth Classic differ significantly in their subdivision of the frequency band into channels. Bluetooth Classic utilises 79 channels with 1 MHz spacing, all serving the

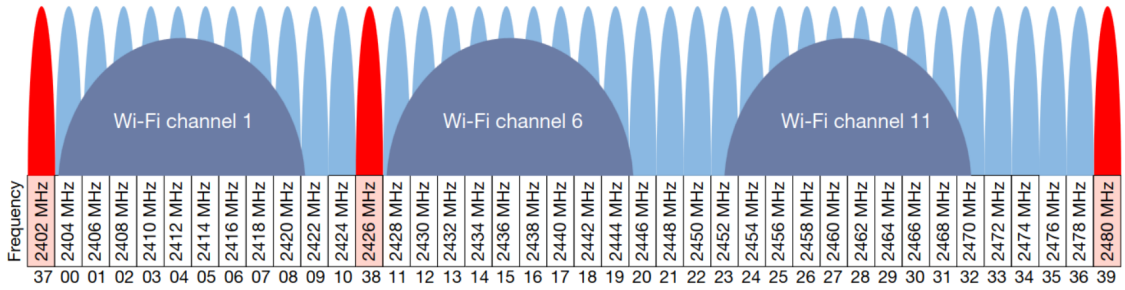

Figure 2.10: The 2.4GHz frequency band shared by BLE and 802.11. The red channels are the advertisement channels. [86]

same function. In contrast, BLE divides the spectrum into 40 channels with 2 MHz spacing. Within those 40 channels, 37 are designated for general-purpose use, while 3 channels are specifically reserved for advertising [9]. Figure 2.10 illustrates the BLE channel division with advertising channels highlighted in red. This difference is related to a different device discovery process. While Bluetooth Classic relies on paging on all channels, BLE introduced advertising on the dedicated channels to shorten connection times and increase energy efficiency.

The Bluetooth topology varies between Bluetooth Classic and Low Energy [1]. Bluetooth Classic uses a point-to-point connection, creating a star topology known as a piconet, where one master device manages the network and up to seven slave devices are connected to it. A device can act as a master in only one piconet but can participate as a slave in other piconets, leading to a more complex topology called a scatternet. In contrast, BLE initially supported point-to-point connections and broadcast, and there is no limit on the number of connections imposed by the specification. With the introduction of Bluetooth 5, a mesh networking profile was added to BLE, which significantly increased its range.

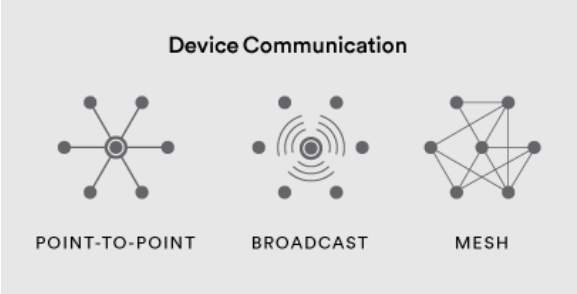

Figure 2.11: Topologies supported by Bluetooth LE [1]

Both the broadcast and mesh topologies utilise advertising for data exchange. In order to support larger payloads, extended advertising on general-purpose channels was added as supplementary to the primary advertising. If required, the primary advertising can direct a receiver to extended data on the general-purpose channels through the use of pointers.

Each device is identified by a unique 48-bit Bluetooth address (BD_ADDR) [11]. However, this address is not directly used in data frames during a connection. In a Classic Bluetooth piconet, the network is identified by an Access Code derived from the lower

24 bits of the master device's BD_ADDR. In contrast, Bluetooth Low Energy (LE) uses a 32-bit randomly generated Access Address for connection identification.

To address privacy concerns, as the BD_ADDR could be used for device tracking, Bluetooth LE Privacy Extension introduced two types of private addresses: resolvable and non-resolvable. Resolvable private addresses are generated using a hash function and a key, allowing devices with the key to recognise multiple addresses belonging to the same device. Non-resolvable private addresses, however, are generated randomly, preventing linking addresses to a single device. These private addresses can replace the BD_ADDR and change periodically, enhancing user privacy.

The Bluetooth security model includes five key features - pairing, bonding, device authentication, encryption, and message integrity [23]. Pairing is the process of creating shared secret keys, which are subsequently stored during bonding to establish a trusted device pair for future connections. Device authentication is crucial in preventing Man-in-the-Middle (MitM) attacks by confirming that the paired devices possess identical keys. Encryption ensures confidentiality of the messages, while message integrity protects against their forgery. As the security implementations vary between Bluetooth Classic and LE, we will focus on LE in this work. According to the Bluetooth Core Specification [7], none of the security features in BLE are mandatory; it is up to the vendor to select the appropriate settings. The LE security is expressed in terms of a security mode and security level defined as:

- **LE security mode 1** controls whether encryption is used for connections and which association model (type of pairing) shall be used.
  Levels:
  1. *No security* – no encryption or device authentication is used, devices connect without pairing.
  2. *Unauthenticated pairing with encryption* – uses the *Just Works* pairing method described later, which allows the devices to be connected without user interaction (as the authentication is missing).
  3. *Authenticated pairing with encryption* – uses other methods for pairing, which authenticate the devices through user interaction for MitM protection.
  4. *Authenticated LE Secure Connections pairing with encryption* – was added in version 4.2 together with LE Secure Connections pairing and denotes its usage.
- **LE security mode 2** defines the use of MAC for data signing in connections where encryption is not utilised. When LE security mode 1 levels 2-4 are defined, the security implications of this mode are automatically satisfied.
  Levels:
  1. *Unauthenticated pairing with data signing* – uses pairing method *Just Works* with no user interaction and device authentication.
  2. *Authenticated pairing with data signing* – uses pairing methods which require user interaction for device authentication.
- **LE security mode 3** – was introduced in Bluetooth 5.2 [23] for encryption of broadcasting isochronous data, such as LE Audio.
  Levels:

1. *No security* – the broadcasting does not use encryption nor authentication.
2. *Use of unauthenticated Broadcast_Code* – uses encryption of the broadcasting data with the key Broadcast_Code.
3. *Use of authenticated Broadcast_Code* – uses encryption of the broadcasting data, and if the key Broadcast_Code was not received via an authenticated method, an error shall be indicated.

The security levels are hierarchical, meaning that the higher level always has to satisfy the security requirements of the previous levels.

Based on the security levels, several association models are used for pairing, with *Just Works* being unauthenticated while the others require authentication. The pairing process, illustrated in Figure 2.12, consists of three phases. In the first phase, devices exchange pairing features that determine the selection of one of two possible second phases and the appropriate association model. Secure Connections, introduced in version 4.2, uses an Elliptic Curve Diffie-Hellman (ECDH) for key exchange and is preferred if both devices support it. A device can force this phase using the Secure Connections Only mode (also known as FIPS mode) [7]. The original method, now known as Legacy Pairing, relied on a simpler key derivation process. A third phase, protected by the key generated in the second phase, is used to distribute additional keys. If the distributed keys are stored for subsequent use, they are said to be bonded [3].

In Legacy Pairing there are three association models, which are used to obtain a Temporary Key (TK):

- *Just Works* provides no authentication and thus no MitM protection, but requires no user interaction and sets the TK to all zeroes.
- *Passkey Entry* requires one device to display a six-digit random number and the user to enter it on the other device. The TK is then this number padded with leading zeros.
- *Out Of Band* utilises a communication channel different from Bluetooth to exchange full 128-bit TK.

The TK is then used to calculate a confirm value, which is exchanged to verify that both devices have a valid TK. Once verified, a short-term key (STK) is derived to encrypt the channel.

Secure Connections (SC) improves pairing security by using FIPS-approved algorithms (ECDH over P-256 elliptic curve and AES-CMAC) [7]. This approach replaces the legacy key distribution method, which relied on a temporary key (TK), with a more robust approach. Initially, the devices exchange public keys and compute a shared Diffie-Hellman key (DHKey) using ECDH. Then, authentication is performed in two stages. The first stage, which is similar to Legacy Pairing, uses one of the association models to ensure that the paired device is indeed the intended one:

- *Just Works* – non-interactively verifies device public keys by exchanging a confirm value computed using AES-CMAC.
- *Numeric Comparison* – performs the *Just Works* method and additionally calculates a six-digit number from the verification parameters and the user must confirm that the numbers are the same on both devices.

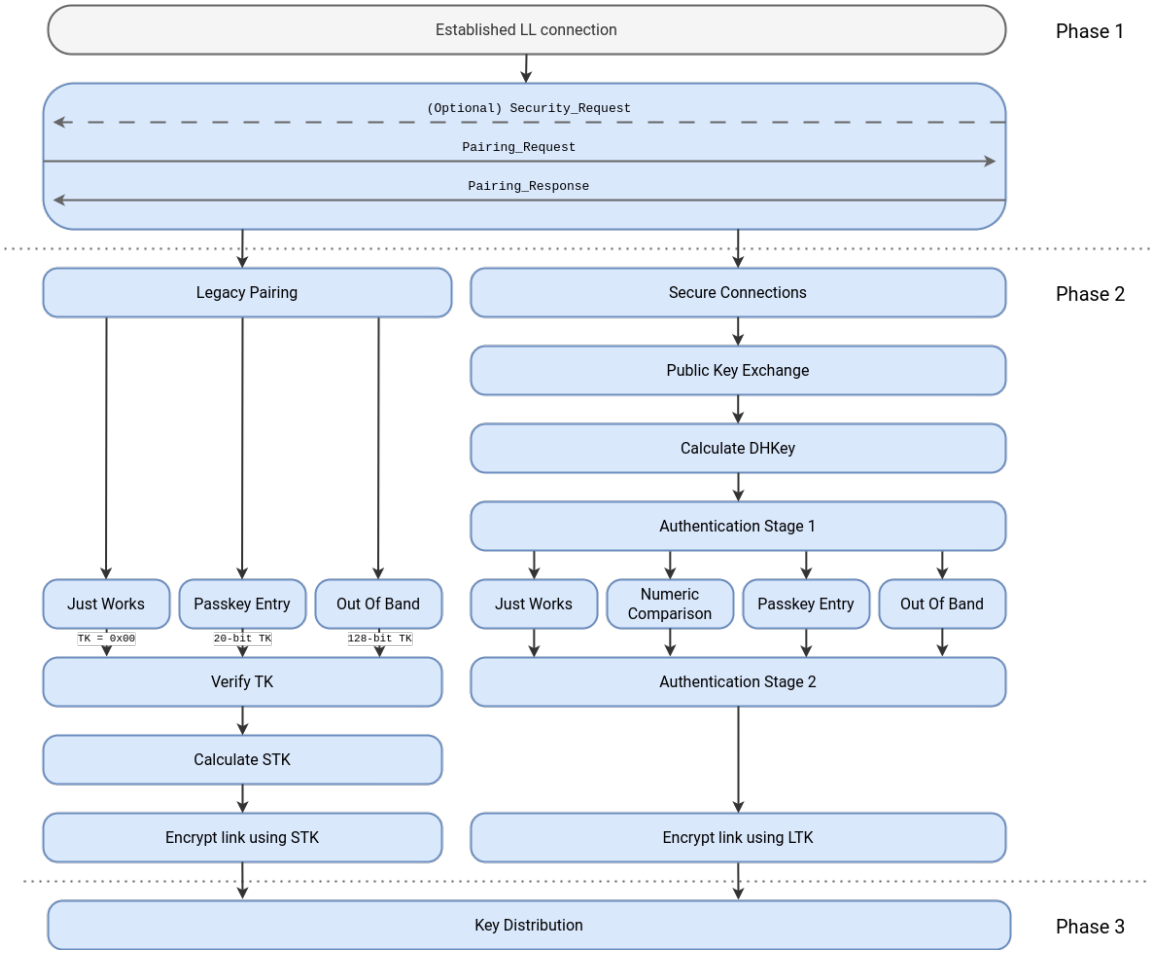

Figure 2.12: Bluetooth LE pairing flow

- *Passkey Entry* – requires the user to either enter the same passkey into both devices, or the passkey is displayed on one device and must be entered into the other device. The passkeys are then iteratively verified bit by bit using a similar process to *Just Works*.
- *Out Of Band* – process is split into an in-band exchange and a confirm value calculation, which is then exchanged out-of-band and verified.

In the second authentication stage, the Long-Term Key (LTK) and MacKey are derived from the previously calculated DHKey. The MacKey is then used to create a check value used to verify that both devices have the same keys [3]. If this verification is successful, the LTK is used to secure the channel.

All encryption in Bluetooth LE is based on AES-128-bit block cypher as defined by FIPS-197[3] [7]. The authenticity of the packets transmitted over the encrypted link is protected using an authenticated CCM (Counter with CBC-MAC) encryption mode, similar to the security mechanisms used within IEEE 802.15.4 networks. The CCM algorithm relies on the Nonce, which is a combination of an initialization vector (IV) exchanged during the

[3]NIST Publication FIPS-197 (http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf)

encryption start procedure, and a packet counter that is reset on each connection [7], which also ensures the prevention of reply attacks.

There are multiple keys used in Bluetooth LE networks:

- *Temporary Key (TK)* is used in Legacy Pairing for authentication and derivation of STK.
- *Short Term Key (STK)* is used in Legacy Pairing for the protection of key distribution.
- *Long Term Key (LTK)* is the basis for a trusted device pair and is used to generate session keys for encrypting connections. If devices do not share an LTK, they create one during the pairing process. If a device loses its LTK (e.g. due to a power failure), a specific command *LL_REJECT_IND* is issued to force the devices to repeat the pairing process.
- *Connection Signature Resolving Key (CSRK)* is used for signing data over unencrypted link (Security mode 2).
- *Identity Resolving Key (IRK)* allows identification of Private Resolvable Addresses when LE Privacy feature is used.

Keys are exchanged during the key distribution process in pairing phase 3, with the exception of TK and STK, which are only used in legacy pairing. Notably, in Secure Connections pairing, the long-term key (LTK) is generated securely using the Diffie-Hellman method, eliminating the need for its exchange.

Chapter 3

Security issues in IoT networks

In the previous chapter, we discussed the specifics of wireless IoT networks and illustrated their operation with concrete examples. However, as highlighted in the introduction, the growing number of IoT devices, coupled with the critical importance of the data being transmitted, has made IoT systems a target for malicious activities such as manipulating decision-making processes in a wide range of applications we depend on or tracking user identity and behaviour [120]. In addition, IoT devices included in secure IT networks are attractive as an entry point in complex attack scenarios. Analysing known weaknesses and attacks on the technologies used is essential to enhance security and prevent security breaches in IoT systems. This chapter is based on our survey [69] of the IoT security landscape, with a particular focus on the three selected IoT networks: the LoRaWAN, ZigBee, and Bluetooth Low Energy. Building on the IoT Attack Taxonomy introduced by Chen et al. (Figure 3.1, [33]), we will examine reported vulnerabilities in wireless IoT networks.

The main challenge with the current IoT ecosystem is that most vendors focus on specific components rather than the entire system. Their priorities are often to deliver new features, fast time-to-market and ease of use, with insufficient focus on security and privacy [120]. This leads to serious vulnerabilities, as demonstrated by the number of documented attacks on IoT devices, especially those with IP capabilities. These devices are directly accessible via the Internet and can be easily located using search engines such as Shodan[1], making them prime targets for creating Botnets of Things (BoTs). These botnets can be further exploited for various attacks on targets on the Internet, such as the DDoS caused by the MIRAI botnet [38]. A study by Costin et al. [34] revealed severe security flaws in the embedded firmware of IoT devices, including several dozen hard-coded password hashes, many of which were weak and the passwords could be recovered, admin credentials affecting over 101,000 devices, and extractable private RSA keys along with their self-signed certificates used by approximately 35,000 online devices.

As the attack surface in IoT is vast, we use the IoT Attack Taxonomy provided by Chen [33], shown in Figure 3.1, for orientation. This taxonomy divides the surface into four architectural layers. The application layer is responsible for providing quality service to the end user. The middleware layer's task is to collect information and store it in the cloud and database, optionally preprocessing the data. The network layer ensures the connectivity of the IoT infrastructure. Finally, the perception layer is focused on identifying objects, measuring target information, and transforming this information into digital signals. In

[1] https://www.shodan.io/

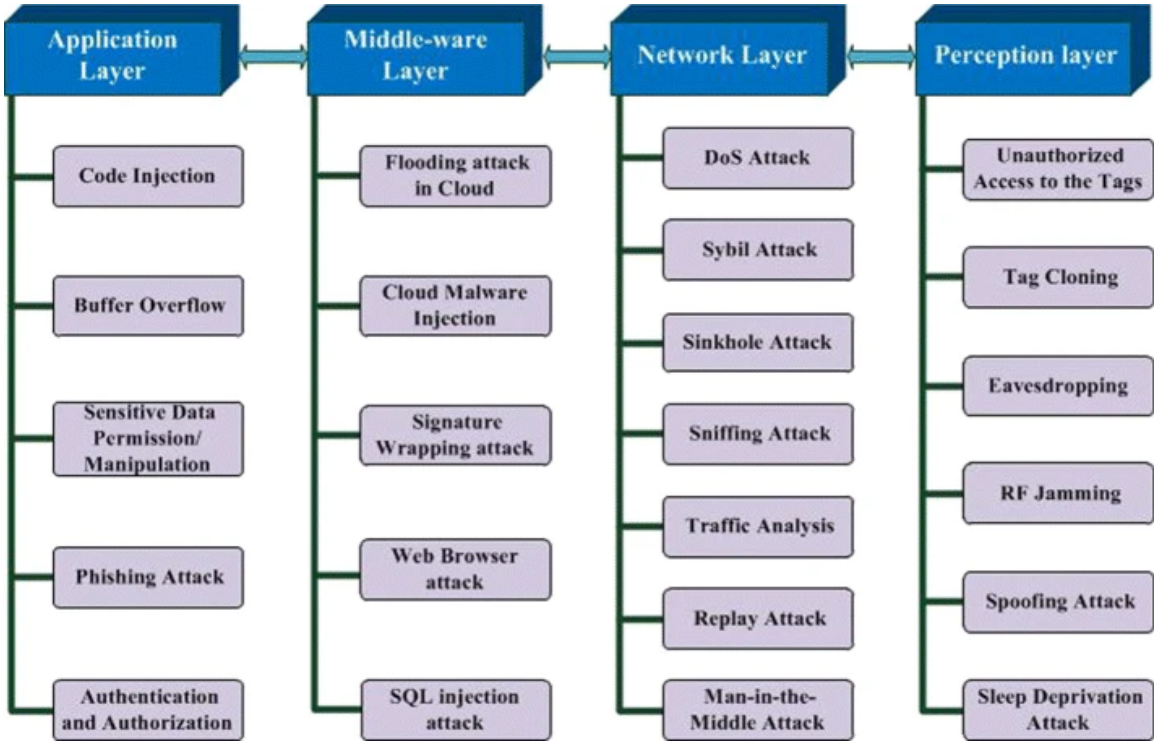

Figure 3.1: IoT Attack Taxonomy Based on Architecture [33]

this work, we focus primarily on the network layer, but since attacks on the perception layer can directly impact connectivity, we include them as well.

Before addressing the specific vulnerabilities of particular networks, it is essential to understand their general characteristics, especially in the area of wireless networks. We start with the perception layer and then move up to the network layer vulnerabilities.

One of the fundamental problems of wireless networks is the medium through which they operate [14]. These networks transmit data via electromagnetic waves through the air, a method that is inherently public domain, and there is no reliable method of preventing access to this medium. With the advent of software-defined radios (SDR), access to the airwaves has become more feasible than ever, making *eavesdropping* an ever-present risk. Therefore, encryption is a necessity in wireless network security to ensure the confidentiality of the information being transmitted.

Another consequence of transmitting data via air is susceptibility to interference, which can significantly disrupt communication. To overcome this problem, wireless communication protocols incorporate various techniques such as integrity checks to detect corrupted data, forward error correction codes to reconstruct original information, and methods like spectrum division and channel switching to mitigate noise. While these methods are effective against unintended interference, they are less successful against intentional *jamming* attacks.

Amin and Abdel-Hamid [14] created a taxonomy of attacks on the physical layer for IEEE 802.15.4 that can be applied to generic wireless communication. As can be seen in Figure 3.2, the attacks can be divided into three categories: *jamming*, where the attacker corrupts the message so it is not received; *message manipulation*, where the content is altered en route; and *steganography*, where hidden information is embedded within signal

patterns. While message manipulation can be prevented by authentication, defence against radio jamming is far more complex. For Wide-Band Denial attacks that target the entire spectrum, the only remedy is to switch to an unjammed frequency band. More subtle forms of jamming, such as Pulse-Band jamming, target specific channels or even individual messages, making them difficult to detect. These attacks, demonstrated by Brauer et al. [26], can effectively render particular communications impossible and are often employed in sophisticated attack strategies.

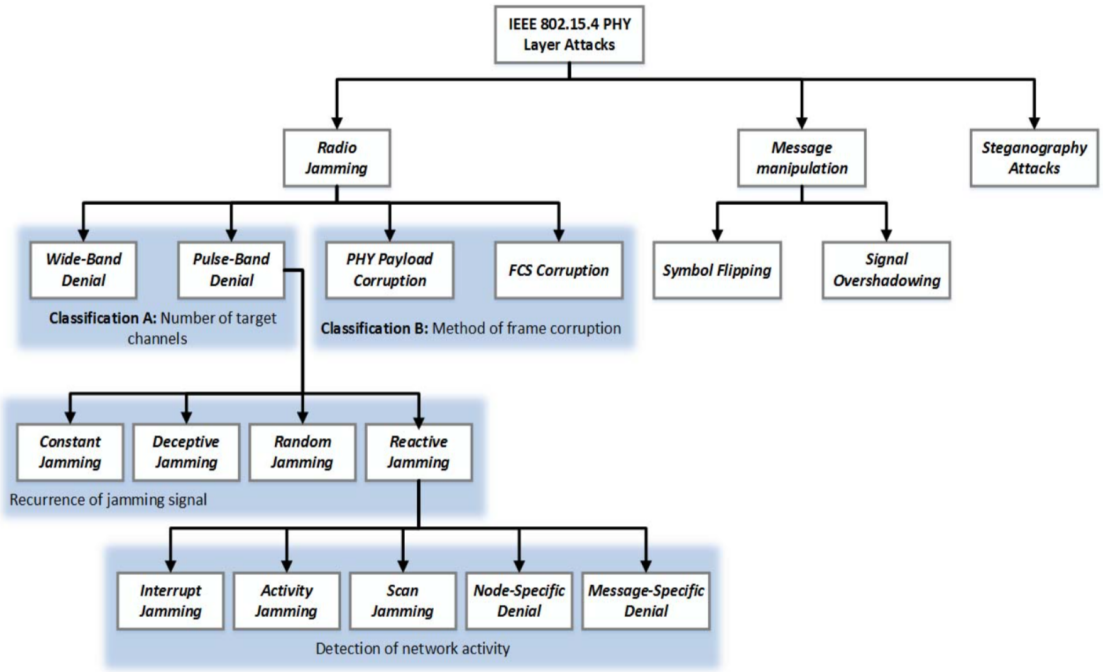

Figure 3.2: Classification of IEEE 802.15.4 PHY Layer Attacks [14]

Because the open nature of wireless communications allows anyone to not only eavesdrop on other communications but also to transmit data, ensuring the identity and authenticity of messages is a significant challenge. *Spoofing attacks* exploit this vulnerability by allowing an attacker to masquerade as a legitimate node, thereby manipulating the network infrastructure or the transmitted data. The main danger of spoofing is that it compromises the trustworthiness of network data and serves as a precondition for more sophisticated network layer attacks such as Sybil and replay attacks.

The last attack we present in the perception layer is classified as a *sleep deprivation attack* in the taxonomy used and targets battery-powered nodes. In fact, it can be considered as a specific case of energy depletion attacks [85] (which are absent in the taxonomy) in which an adversary tries to drain the battery of a device, leading to a denial of service (DoS). This is usually achieved by forcing the victim node to perform resource-intensive operations, such as processing huge amounts of messages. A concrete example of such attack, Ghost-in-Zigbee, is described later in the Section 3.2. In a sleep deprivation attack, the adversary prevents the victim device from going into a power-saving sleep mode and forces it to remain in a more power-intensive standby mode, thus accelerating the battery drain [85].

Network layer attacks focus on disrupting or compromising communication between devices. The basic types of these attacks include *denial of service (DoS) attacks*, *replay attacks*, and *man-in-the-middle (MitM) attacks*. DoS attacks on this layer typically overwhelm the network with excessive traffic, making it unavailable and disrupting the flow of information. Replay attacks involve an attacker eavesdropping on a message, intercepting it, and resending it at a different time, sometimes repeatedly. If the network does not recognise these messages as outdated, this can lead to unwanted actions, as found in the LoRaWAN message confirmation discussed in Section 3.1. In MitM attacks, an attacker intercepts communication between two nodes by acting as an intermediary. This allows the attacker to access and potentially manipulate all relayed data. This attack is possible in networks with weak authentication of communicating parties and messages, as the attacker must remain undetected while forwarding the information.

A more sophisticated attack on node authenticity is the Sybil attack, in which a malicious device impersonates multiple identities, either stolen or fabricated, to increase its influence. By impersonating a group of devices, an adversary can gain an inappropriate level of control, especially in peer-to-peer networks, and can significantly impact network topology, data integrity, and resource utilisation [92].

Other attacks on the network topology include *sinkhole* and *wormhole* attacks. Sinkhole attacks primarily target dynamic mesh networks with self-healing capabilities. In such an attack, an adversary manipulates the routing of the network by falsely reporting superior routing metrics, thereby artificially increasing its rank in the network [64]. This causes traffic in the surrounding area to be routed through the attacker's node, allowing the attacker to control or exploit the data. Conversely, wormhole attacks involve the attacker disrupting the network topology by creating a high-speed connection between two nodes, often both under his control. This can lead to routing race conditions, where the attacker can reorder or react to messages before the benign network delivers them. A particular case is then a relay attack, in which the malicious node forwards packets to remote nodes, which are then perceived as neighbours [64], for example, to bypass distance control (air gap).

Having reviewed the general issues, in the following sections we investigate the identified security issues of selected wireless IoT networks.

3.1 LoRaWAN

LoRaWAN uses CTR mode to encrypt application data, which is essentially a stream cipher. JungWoon at al. [73] argue that bit-flipping attack is possible in the communication between end-device and network server, because the plaintext and cyphertext have the same bit order. In this attack adversary can modify a part of encrypted message by changing directly the cyphertext if he knows the structure of the data being sent. Although this change would break the integrity check by MAC, JungWoon suggests that brute forcing its value is possible due to its low size of 4B. Yang in his thesis [116] suggests, that the same attack could be applied for the communication between network and application servers without the need of MAC brute-forcing.

Aras et al. [17] and Yang in his thesis[116] described a possibility of a replay attack. LoRaWAN uses *frame counter* as a protection against replay attacks and server validates that received message has higher value than the last one. As the frame counter value is a part of an unencrypted header, an adversary can easily learn its value. In the protocol there are three eventualities when the frame counter is reset. It can occur either when a device

joins a network in OTAA mode, if the device is reset, or when the value overflows. Because the device in OTAA mode during join procedure exchanges new encryption keys, it is not feasible for a replay attack and resetting an OTAA device effectively causes it to join the network again. But resetting a device using ABP mode will cause it to reset the frame counter while keeping the same keys, allowing an attacker to replay previously captured message in the right time (after message counters reset). If the frame counter is reset as a result of overflow, the keys were not updated which allowed the replay attacks. To mitigate this vulnerability, the devices supporting OTAA have since LoRaWAN 1.1 an option to undergo a re-join procedure, which updates the security keys [21].

Another attack allowed by frame counter reset is decryption of communication. Because the devices in ABP mode never change the encryption keys and LoRa uses CTR mode of encryption, after the reset of frame counter the messages will be encrypted by identical keystream [116]. Having multiple messages encrypted by the same keystream significantly increases the possibility of their decryption and allow eavesdropping on the communication.

During the OTAA mode, an end-device has to send a *JoinRequest* message to the network server to be added into LoRaWAN network. This unencrypted message contains device and application identifiers and a random number DevNonce. According to the specification [21], the server shall require DevNonce to be a random number and, to prevent replay attacks, compares this nonce from the JoinRequest with a sequence of already used nonces by the device. The length of the sequence is a system parameter that can be chosen [112] and if a replayed nonce is revealed, server can choose to reject the JoinRequest, or put the device to a blacklist and permanently exclude it from the network. The attack consists of storing join messages and after the nonce disappears from the server stored sequence, replaying join messages to the server and thus causing server ignore messages from the benign device. Moreover Tomasin et al. in [112] speculate about the possibility of this feature causing DoS without any adversary as true random nonce could lead to generation of the same number for consequent requests.

Another flaw opens a way to abuse the non-encrypted beacon messages broadcasted by the gateways. The beacon contains GPS coordinates of the gateway and also time that should be synchronized in end-devices. This way an attacker knows from beacon the exact location of the gateway and can perform some physical attacks. Moreover, it is possible to spoof the beacon messages with a wrong time and cause a DoS of the end-device. The end-devices can deplete their batteries faster when it receives many beacons with random time. [116]

A vulnerability was found also in message confirmation. In LoRa protocol every message is confirmed by ACK response from the server. The problem is that ACK message does not contain any identification of the message that is being confirmed and is simply used as acknowledgement of the last message sent [116]. If an adversary gains control of a gateway (or spoofs the gateway), he can store ACK messages for a certain device and might use it to confirm any other messages from the device. If the adversary manages to block outgoing messages from the device (eg. by jamming), he can then cause silent DoS attack as the device wouldn't be able to recognise that those messages weren't delivered.

3.2 ZigBee

The ZigBee standard has undergone an extensive development and testing both from the side of ZigBee Alliance and independent researchers. There were some vulnerabilities in the protocol or implementations identified and mitigated in new versions of the standard.

One of the biggest issues was found to be the key management that relied solely on secrecy of master keys. Up to the ZigBee 3.0, the master keys generated by the ZigBee Alliance were shared among all certified devices and used exclusively for network management including joining the network and network key exchange. The whole system was compromised when one of the keys leaked in 2015 [84]. As a result the standard introduces possibility to use *install codes* to exchange link keys with router prior to joining the network [43]. Such codes are exchanged out-of-band, usually by scanning QR code.

Another issue found in encryption was found within the life span of used keys. As the keys lifetime was not limited and ZigBee uses CTR mode for encryption, after the counter has overflown the messages were encrypted by the same keystream. Using the same keystream for encryption of multiple messages increases the possibility of decryption and because the counter is unencrypted in the header of every frame, it's feasible for an adversary to detect the exact moment when it overflows. As a countermeasure ZigBee allows regeneration of keys. In variable intervals TC or network managing node regenerates network keys[47] which causes the keystream to be different. To further mitigate the risk, there is hard boundary in ZigBee specification for maximum of 2^{61} messages encrypted with the same key and after this the key must be regenerated[123].Also the link key lifetime was unlimited which allowed the attacker to generate messages and cause counter overflow leading to reusing the keystream [47].

The well publicised is a vulnerability found in Atmel implementation of ZigBee which was used in Phillips Hue light bulbs [97]. This implementation failed to check the *Transaction ID* for *RESET_TO_FACTORY_NEW* command for an invalid value of 0, which leads to accepting the command and resetting the bulb. After reset the bulb tries to connect to a network and an adversary can make the bulb ignore the genuine network and force it to join his network instead. To avoid Touchlink protocol forcing the pairing devices to be nearby, attacker can make use of the compatibility feature of ZigBee and avoid this check. Moreover, the authors discovered that this implementation also does allow broadcast of the reset command, which is in contradiction with specification and further eases the attack. In this specific attack was also discovered master key securing the OTA update which allowed researchers to update the firmware and cause chain reaction of infected devices.

As hinted in the general information about energy depletion attacks, ZigBee protocol is susceptible to an attack labelled as *Ghost-in-ZigBee* [29]. This attack uses the fact that only the payload is encrypted in *IEEE 802.15.4*. Device address and counter value are not encrypted which allows attacker to generate messages for any selected device. The device have to process every message with the counter higher then its own and even if the MIC will cause the frame to be dropped thanks to integrity violation, if the adversary generates enough of those messages the device can deplete its energy resulting in DoS.

3.3 Bluetooth

Bluetooth has faced a number of security issues since its introduction, from access to encryption in the earliest versions through key exchange in Low Energy to implementation bugs. Each new version then comes with mitigations for identified vulnerabilities in the form of optional features to maintain compatibility. In its report on Bluetooth [90], NIST points out, among other things, that this approach allows for so-called downgrade attacks, where an attacker forces a victim to use an older method with known vulnerabilities, and recommends using SC Only mode to reduce this danger. The problem with downgrading to an older Bluetooth version is also acknowledged by Lonzetta et al. [75]. A significant

concern is also the limited ability to patch device software if vulnerabilities are discovered, as deployed Bluetooth controllers typically cannot be updated.

As Bluetooth technology has started to be integrated into security-critical components such as smart locks, it has attracted considerable attention from researchers. Lonzett's work [75] provides a comprehensive overview of the BLE security landscape, examining both Bluetooth Classic and Low Energy protocols and detailing various attacks on commercial products. Following this, Casar [36] provides a focused analysis of BLE, specifically outlining the weaknesses that have been discovered. Lacava [72] goes further by detailing specific vulnerabilities associated with their CVE identifiers and categorises attacks based on network topology, including single links, broadcast, and the relatively new mesh topology. In his PhD thesis [49], Hamby lists specific vulnerabilities and evaluates BLE devices' security before and after applying NIST's recommended prevention techniques [90].

The main vulnerability of the BLE specification lies in the Legacy Pairing scheme, which was the only pairing method available until version 4.2, leading to the introduction of Secure Connections. Legacy Pairing involves one device generating keys and transferring them to the other device, with security provided by a short-term key (STK) derived from a temporary key (TK) based on the association model. However, as NIST points out in its technical report [90], the TK generated by these methods is often not strong enough, making the process vulnerable to eavesdropping attacks that can result in key leakage. Moreover, depending on the negotiated parameters, the keys exchanged can be as small as seven bytes, and the number of authentication challenges is unlimited. This vulnerability becomes even more dangerous with the ability to force the device to re-pair [101]. Jasek [65] and Melamed [80] demonstrated practical attacks exploiting these weaknesses, such as transparent MitM attacks using the legacy *Just Works* method and forced re-pairing leading to encryption key recovery.

Despite improvements in later versions of Bluetooth technology, encryption and other security features remain optional and are often not implemented by many manufacturers. Security experts at Merculite conducted a survey of Bluetooth-enabled smart locks [98], which revealed that most of these devices suffer from serious security flaws. A common flaw found is the absence of BLE encryption, which was confirmed by S. Jasek [65], who said that 80% of the Devices he tested do not use BLE encryption. Although Jasek's study focused only on devices using the oldest BLE 4.0 standard, his results are concerning, given the continued widespread use of this standard. Without BLE encryption, devices lack network-level authentication and are vulnerable to MitM attacks, a risk that both Merculite and Jasek have demonstrated. While manufacturers often implement additional security measures at the application level, these can increase the likelihood of implementation flaws, often leading to MitM and replay vulnerabilities [98].

BLE is generally robust against jamming attacks due to its use of frequency hopping spread spectrum (FHSS), which allows it to rapidly switch frequencies and thus avoid interference on any single channel. However, the invention of advertising channels in BLE has introduced a new risk. Since every BLE communication begins on one of the three designated advertising channels, an adversary could perform a DoS attack by jamming these channels, effectively disrupting the entire network. Additionally, an attacker could eavesdrop on the initiation of communications by monitoring these channels. Brauer et al. [26] demonstrated that targeted disruption of BLE communication is possible. However, the effectiveness of such attacks diminishes significantly as the distance between the jammer and the receiver increases, leading to a rapid decrease in disruption rate.

Seri and Vishnepolsky revealed in their work [104] that the complexity of Bluetooth protocol leads to implementation errors. They inspected various Bluetooth stack implementations and discovered multiple vulnerabilities labeled together as *BlueBorne*. These vulnerabilities allow wide range of attacks from MitM to remote code execution on various platforms affecting billions of devices globally. The only way of protection is updating affected implementations to patched versions, which can prove to be next to impossible for some cases of deployed devices. As a follow-up, Garbelini et al. [46] contributed by developing a fuzzing-based testing framework that successfully uncovered several additional vulnerabilities in BLE implementations.

Lastly, Antonioli et al. [15] identified in their work a vulnerability in the pairing process affecting all security modes. During the first phase of pairing, device features are exchanged containing maximal supported entropy for the LTK. This can be abused to lower the entropy and, thus, the security of the resulting key, which, as a consequence, enables the adversary to brute force the key afterwards. The entropy can be lowered down to 1 byte for Bluetooth Classic and 7 bytes for Bluetooth LE. While using Secure Connections should result in a full 16 bytes of entropy, the authors state that, in reality, they were able to lower the entropy of SC-generated keys as well.

Chapter 4

Security measures

In the previous chapter, we illustrated the vulnerabilities of the IoT environment and highlighted the security challenges it faces. This problem is exacerbated by the variety of transport protocols used in these environments, leading to increased complexity and potential points of failure [120]. With fog computing, more and more data traffic is exchanged just inside local networks, where some nodes are provided with computing resources that enable faster and more efficient distributed processing at the edge of the network.

Securing IoT networks is critical due to the increasing sophistication of cyber-attacks, which can lead to unauthorised access, data breaches, and service disruptions. Effective IoT network security requires strong encryption, robust authentication mechanisms [33], continuous monitoring [67], and regular updates, though updating can often be a non-trivial task in deployed IoT devices [120].

We believe security monitoring is a key aspect, involving constant observation of network traffic and device behaviours to detect and respond to threats in real-time. Such monitoring is performed by Intrusion Detection Systems (IDS), which detect unauthorised access and malicious activities by analysing patterns in the data. According to the data source, we differentiate between Network-based IDS (NIDS) which monitors the traffic across the entire network, and Host-based IDS (HIDS) which use audits, logs and other data that originates from the host system [68].

Currently, there are many different IoT platforms[1]. However, they are mainly focused on data processing. Almost none use available computational resources for security analysis. The most common security solutions are still based on simple pattern matching, Deep Packet Inspection (DPI) and Access Control Lists (ACLs) at the network perimeter that are focused just on Internet Protocol (IP) devices. This simple approach is no longer possible due to the complexity and importance of network traffic among end devices that must be properly protected [120].

Nowadays, there is a big demand for a security solution that will respect modern IoT architectures using fog computing principles. These networks are very dynamic and securing network perimeter is not enough for the threat detection anymore. Moreover, IoT data comprises of IP and non-IP traffic (e.g Z-Wave, LoRa and Bluetooth Low Energy). The modern security solution should understand the normal behaviour of computer network composed by IP and non-IP nodes, analyse encrypted traffic without decryption, dynami-

[1] https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/making-sense-of-internet-of-things-platforms

cally adapt to network changes and send alerts in case of unexpected situations in network traffic. Based on created alerts it should automatically mitigate detected threats.

The very promising way of solving these challenges are Machine Learning (ML) and statistical methods. These methods are usually deployed in a cloud environment because they require high computational resources. However, this approach is not effective for modern IoT architectures respecting fog computing principles. IoT gateways that are placed at the network edge have a great insight into network communication because they have direct connectivity with end devices. On the other hand, these gateways usually have low computational and storage resources. More resources are available at upper layers of the fog computing architecture. However, the detail of direct insight into data is lower. Due to these reasons, it is necessary to modify a structure of anomaly detection modules so that it can run even on edge IoT gateways. Information gathered from gateways are then combined in upper layers of the fog computing architecture.

Our aim is to design and implement a network intrusion detection solution for modern IoT networks with respect to the fog computing architecture. To achieve the flexibility needed, we base our solution on NEMEA framework [31] described later, which provides platform with loosely coupled modules, which can be deployed across multiple nodes. Moreover, our design incorporates a software IoT gateway, BeeeOn[2], consolidating control and monitoring of the IoT network on one node. Because of this integration, we call our solution Secure gateway for IoT (SIoT).

4.1 Network Intrusion Detection

Network Intrusion Detection Systems (NIDSs) are responsible for detection of any unapproved activity that occurs within a network. There are many types of NIDS that differ both in their location within a network and the type of detection they perform. For detection in network environment those systems usually use packet inspection or flows that they analyse and try to decide whether the network has been compromised.

There are three major directions in the detections performed by NIDSs[27]. First type is *specification* based detection that relies on proper description of the correct use for a protocol. Every communication is then compared with this expected use and every deviation is reported as intrusion. Those systems have low false alarm rate and good efficiency, but development of detailed protocol descriptions that could be used is difficult and have to be reworked with every protocol update. Another type is *misuse* based detection which compares patterns of known attacks (called signatures) to events in the network and if a match is found, incident is reported. Having the exact attack patterns has advantage of high efficiency of such systems and possibility to identify an attack that appears in the network. Disadvantage is then recognition of only the described attacks. Last direction is *anomaly detection*. This detection creates behavioural models for nodes in the network and compares their current behaviour against this model. Deviations from this model are reported as anomalies and suspicion for intrusion. Computation of the model is often computationally excessive and requires large amount of data to be created (so called training data) and has high level of false positives. On the other hand this type of detection is able to recognise various problems in the network and even unknown attacks.

Anomaly detection is undergoing an extensive research as it seems to be the most promising direction for the future. Butun et al. created in [27] classification of anomaly

[2]https://beeeon.org

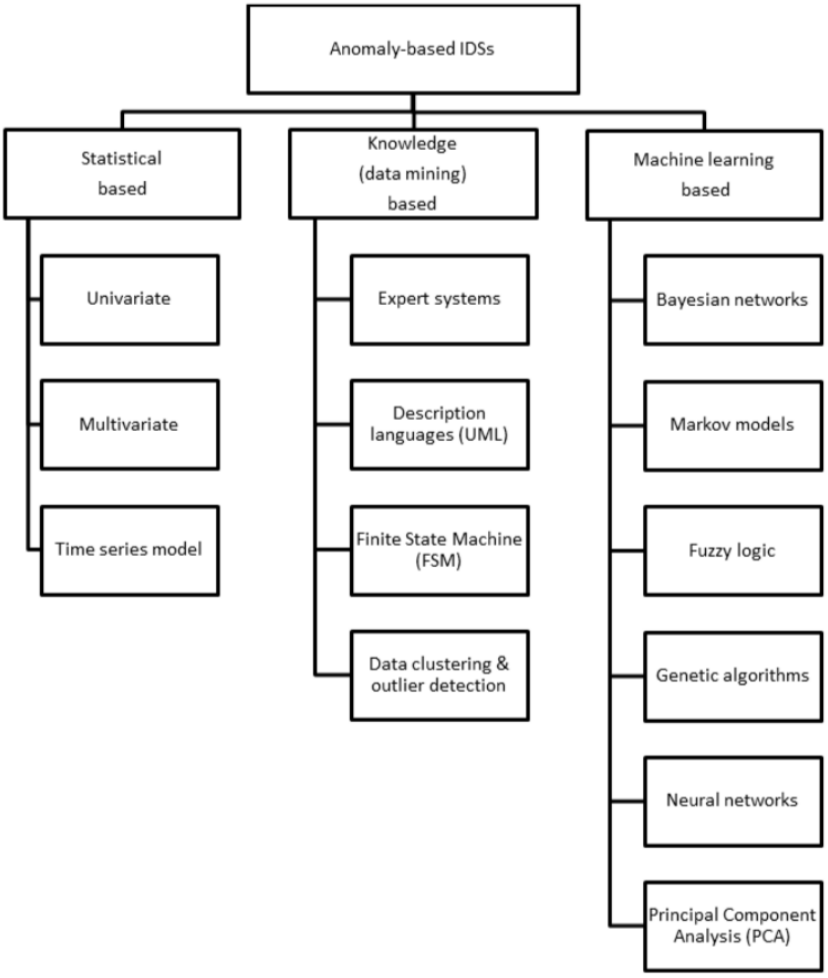

Figure 4.1: Classification of anomaly based IDS [27]

based IDSs that can be seen in Figure 4.1. Those systems can be divided into three categories based on the nature of the processing into statistical, data mining and machine learning techniques. Statistical models rely on profile with mean values for normal operation of the network. Network behaviour is then compared to this profile and is counted a score defining the difference from the profile and if the score is higher than a certain threshold, anomaly is reported. Data mining techniques use big data to create description of normal state and then check current traffic against it. The description can consist of generated rules in expert systems, diagrams in case of description languages, states and transitions creating a finite state machine or groups of clusters based on some similarity feature in which case detection consists of finding an outlier. Machine learning is a special set of methods that generate a model and then use it for recognising anomalies while periodically updating it to improve the results.

4.2 NEMEA Framework

NEMEA (Network Measurements Analysis) platform [126] is designed for real-time traffic analysis and anomaly detection in network environments. The NEMEA framework is composed of a series of independent, interconnected modules, each responsible for a specific task such as flow data preprocessing, filtration, anomaly detection, or logging and reporting of results. These modules are independent processes that communicate through unidirectional interfaces, enabling scalability as the modules can be deployed across multiple nodes.

A key feature of the NEMEA framework is its extensibility, allowing it to be easily expanded by integrating additional modules. This design not only supports a larger set of detection mechanisms but also accommodates different input sources, making it adaptable to various network environments. The modular nature of NEMEA also facilitates quick and easy implementation of new modules, making it a versatile tool for both production environments and as a common platform for researchers focused on network security and monitoring.

A crucial part of the framework is the Traffic Analysis Platform (TRAP) library, which serves as the backbone for communication and operational functionality. Notably, it defines the communication interfaces (IFCs) between modules. These IFCs are unidirectional, allowing each module to utilise multiple input and output interfaces. Additionally, these interfaces simplify the complexities of various underlying communication methods, such as UNIX sockets, TCP sockets, and file operations.

NEMEA uses a binary format called UniRec for efficient data exchange, which is structured around a template that defines a set of fields in a record. All records sent over a single IFC use the same template, ensuring consistency and facilitating fast access to the fields within each record. Moreover, this design minimizes overhead by eliminating the need to transfer metadata about the record structure with each transmission. To ensure compatibility of the modules, the compatibility of output and input IFC templates is checked upon connection. Examples of the UniRec templates used in our work can be found in Figure 4.3.

4.3 Secure gateway for IoT (SIoT)

The architecture of our secure gateway for IoT (SIoT) is split into NEMEA modules differentiated by their purpose. Specific modules for data acquisition are called collectors and their task is to use specific approaches and hardware tools to receive IoT frames from various physical layers of IoT technologies, parse the data and transform them into the standardised UniRec format used for module communication in NEMEA framework. Another type of modules is detector, which task is to analyse the collected frames and detect one or more traffic anomalies or specific network conditions. The output of detectors may be used in other modules in which case they serve as preprocessors. Moreover, modules can be connected to the cloud or fog computing devices where an additional evaluation is performed. Received data are essential for traffic classification, creation of ML models and further improvements of detection methods as depicted in Figure 4.2.

Our solution allows extension of existing modules with other unique features or inclusion of a new module. Effective upscale of the system is achieved by deploying modules to multiple nodes as every module may run separately and communicate with others via a network. Such a high level of scalability is necessary within the IoT concept. For example, even if the collector of LoRaWAN frames is deployed on a physically separated gateway, it still provides received frames to detectors running on another machine.

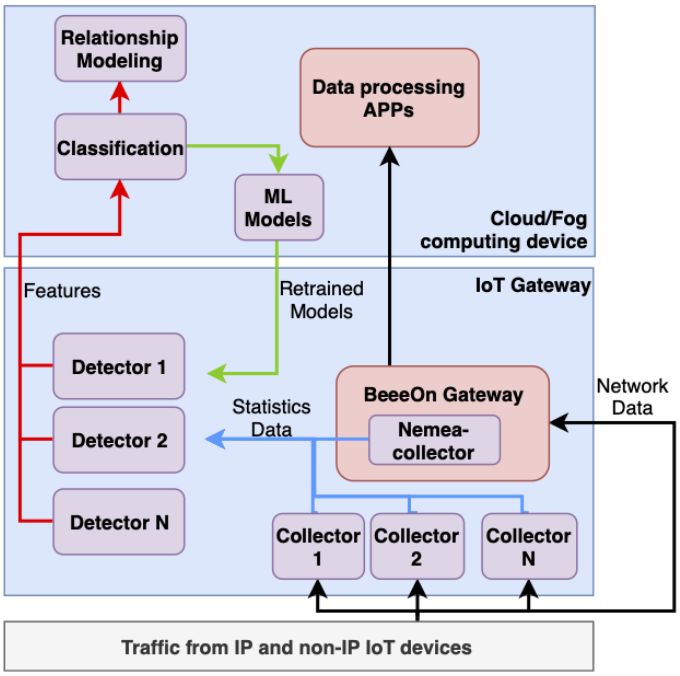

Figure 4.2: Proposed Security Architecture

Collector Message Format

Timestamp	Device ID/Mac Addres	Protocol Specific Fields

Detector Message Format

Timestamp	Device ID/Mac Addres	Alert Code	Alert Caption	Detector Specific Fields

Figure 4.3: Standardised Format of Messages for Collectors and Detectors

Although IoT traffic can vary significantly depending on the technology used, we have identified some common elements. We used the UniRec format, incorporating a predefined set of common items in the record, as shown in Figure 4.3. If needed, additional data specific to the protocol or detector can be added to this format. Each UniRec output record produced by collectors includes a timestamp and a device address. In addition, detectors also provide an alert code and a caption in their output. The alert code indicates the type of anomaly, especially when the detector can identify multiple types, while the caption provides a human-readable description of the anomaly to enhance user experience. Other fields are tailored to each specific technology. Standardising the output from collectors and detectors simplifies interfaces and allows for aggregation, providing a comprehensive overview of traffic and incidents.

4.4 SIoT utilisation in this thesis

In the previous section, we introduced a monitoring solution *SIoT* that can be flexibly deployed on one or more network devices, capable of running machine learning techniques even on edge devices with limited computational power. This solution was developed as part of the Secure Gateway for IoT project, in which the author of this thesis was involved. His

responsibilities included research work on security analysis of IoT networks and designing ways to monitor them, collaboration with BeeeOn and publishing. In particular, he delved into the issue of Bluetooth network monitoring, which shaped the direction of this thesis.

The key advantage of SIoT lies in its flexibility. It is composed of interconnected modules, each with a defined role, which can be easily enhanced or replaced. Moreover, due to the standardised format for data and metadata from both IP and non-IP traffic, developers are provided with a straightforward interface to develop detection methods within our system. Throughout the project, several modules were created to focus on both IP and non-IP network traffic, and the entire codebase was made available as an open-source project on GitHub[3].

In the subsequent chapters of this dissertation, we will be working within the framework of the proposed SIoT architecture. Chapter 5 describes the development of modules into SIoT that are used for security monitoring of Bluetooth communication. Following the modular architecture, the solution consists of two modules - a collector that acquires information from the Bluetooth network, and a detector that analyses and processes this information into alerts.

[3]https://github.com/CESNET/NEMEA-SIoT

Chapter 5

Bluetooth Low Energy (BLE) security monitoring

Bluetooth devices are facing increasing security challenges, especially with the rise of IoT technology in Small Office/Home Office (SOHO) and sharing economy environments. Firmware updates and security patches in this setting are not always reliable, leaving many devices vulnerable. Additionally, vendors often fail to implement security features properly, increasing the risk of using these devices. This is particularly concerning for security-critical applications like Bluetooth-controlled smart locks, where strong security measures are essential.

In this chapter, we tackle the challenge of detecting malicious activity in Bluetooth-enabled devices. We treat this issue as a component of the broader SIoT security solution discussed in Chapter 4. Our focus is on developing modules that enhance this solution, enabling users to incorporate Bluetooth networks into the intrusion detection system.

Intrusion detection is particularly important in scenarios such as the one described in motivation 1.1.1, where the limitations in the security logging of a smart lock were clearly exposed. If an adversary exploits vulnerabilities outside of the vendor's application or cloud services, for example through replay attacks, the built-in monitoring mechanisms may fail, allowing breaches to go undetected. This is particularly concerning given the rising popularity of smart locks in residential settings and the sharing economy, such as in Airbnb, where a breach could lead to unauthorised access to the property.

Our objective is to detect unauthorised manipulations with a device by independently monitoring its status, regardless of the vendor, and logging the times when it is accessed. These monitoring logs can be compared with vendor audit logs to identify any inconsistencies that may indicate tampering. They can also form the basis for more advanced detection methods, such as Network Behavioral Anomaly Detection (NBAD) or inclusion in Security Information and Event Management (SIEM). In situations where an adversary actively probes a system during the reconnaissance phase of an attack, this monitoring can function as an early warning and prevention system.

Security monitoring of Bluetooth networks is particularly challenging due to its primary use as a direct link between a controller, typically a smartphone or tablet, and an IoT device. This direct communication means that there is no centralised point, such as an access point in IEEE 802.11 networks, where monitoring tools could be easily integrated. Although the introduction of Bluetooth 5 and its mesh profile offers the potential for larger networks that might include monitoring nodes, this feature is still new and not widely

adopted. Many IoT applications will likely continue to use the direct link approach, so external monitoring of devices will be necessary. However, this method is complicated by the use of Frequency-Hopping Spread Spectrum (FHSS), which disperses communication across 40 channels, adding another layer of complexity to the monitoring process.

There are several options for how to employ such monitoring of Bluetooth networks. The most available are development tools such as nRF Sniffer for Bluetooth LE[1] or Ubertooth One[2]. These tools allow monitoring of a single device by following its connection across the channels. In order to be able to monitor multiple devices, it is possible to use powerful Software Defined Radio (SDR) based analysers like Ellisys Bluetooth Tracker[3] and RFcreations mini-moreph[4]. These analysers are industry-oriented with price tags exceeding $20.000 and thus not well suited for the SOHO use we decided to target. The last option would be embedding a monitoring option in the Bluetooth controller itself, but its inclusion would depend on the vendor and wouldn't address already deployed devices.

In this work, we focus on finding a cost-effective solution for efficiently monitoring multiple devices within a SOHO setting. We propose an alternative monitoring principle based on observing BLE Advertising channels. Unconnected devices periodically transmit their advertisements, and by analysing the timing of these advertising PDUs, we can detect when the monitored devices have been connected. The connections detected can then be crosschecked against vendor logs to detect discrepancies.

Adversary model

In this thesis, we are operating under the assumption that an adversary's goal is to misuse a Bluetooth LE end device while avoiding detection. We are specifically considering a scenario where the device is operating independently, not connected to the broader smart infrastructure through any channel other than Bluetooth. The attacker has the ability to eavesdrop on the device's communications passively and to actively send arbitrary packets on BLE channels. Additionally, we are assuming that the attacker possesses knowledge of how to exploit vulnerabilities in Bluetooth devices and the authentication process between the device and the user device.

5.1 Related work

As vulnerabilities in Bluetooth protocols and implementations have become increasingly prevalent, as discussed in Chapter 3, researchers have shifted their attention to developing tools for more effective security analysis of Bluetooth devices. Del Arroyo [19] proposed a methodology for security testing of these devices, which was later advanced by Ray [94] with the introduction of a semi-automated testing framework and further refined by Garbelini [46], who developed a fully automated testing framework.

Continuous monitoring of the BLE network is a non-trivial challenge because of the Bluetooth features. In the mentioned methodology, the authors note that a sniffer has to be deployed for every monitored device [19]. It is possible to monitor all Bluetooth channels at once, as was presented by Ossmann and Spill [89]; however, these solutions are either complex and expensive or have substantial limitations. Del Arroyo et al. come up with

[1] https://www.nordicsemi.com/Products/Development-tools/nrf-sniffer-for-bluetooth-le
[2] https://greatscottgadgets.com/ubertoothone/
[3] https://www.ellisys.com/products/btr1/index.php
[4] https://www.rfcreations.com/hardware-platform/mini-moreph

a compromise by monitoring primarily one connection but in gaps between transmissions, opportunistically sniffing other connections[18]. This approach allows a better overview of the network and monitoring of multiple connections at the cost of increased probability of missing some data packets and, in the worst case becoming out of sync and losing the connection. Heinrich, Stute and Hollick developed a tool called BTLEmap [52], whose main function is network mapping and device discovery with an emphasis on privacy. Developed for iOS and macOS devices, it scans advertisements and attempts to establish connections with nearby devices, presenting the user with a map of discovered devices with their characteristics. Since Apple devices restrict some information, the authors propose a probe that continuously monitors advertisements and reports them to the tool. Such a probe can be used as a source of continuous advertisement-based security monitoring.

Cayre et al. [30] developed an innovative intrusion detection method that is integrated directly into the BLE controller, enhancing its ability to preemptively identify security breaches.

The work of Wu et al. [114] is most similar to our proposal. The authors present BlueShield, a tool that aims to prevent spoofing attacks on users in Bluetooth networks by distinguishing between benign and malicious advertisements. They use three specially modified Bluetooth probes based on the Ubertooth One, where each collector monitors a single advertising channel, and compare the received advertisements with the profile created. An advertising profile consists of several properties, one of which is the advertising interval, which is also used in our work. BlueShield learns the advertising interval of benign communication, and if an advertisement is intercepted earlier than expected, it is flagged as a spoof. Cai et al. [28] are conducting research with a similar focus on spoofing attacks and have announced plans to create a significant dataset, BLE-SAD, aimed at aiding spoofing detection efforts.

Compared to others, we focus on security monitoring of BLE devices on common hardware (can run directly on IoT gateway). Unlike BlueShiled, we target misuse of endpoint devices, as opposed to protecting users against receiving spoofed information. We consider our work and that of BlueShield to be complementary. While our approach provides only basic metadata about the connections of the monitored devices, the result is a comprehensive view of network events.

5.2 Our approach to BLE connection monitoring

In Bluetooth LE technology, devices take on specific roles defined by the Generic Access Profile (GAP) [7] based on the type of communication, which can be either connectionless or connection-oriented. In connectionless communication, the Broadcaster role is responsible for transmitting data to any nearby devices through advertising or Broadcast Isochronous Stream (BIS), while the Observer role passively receives the data from the broadcaster. In connection-oriented communication, the roles correspond to those of Bluetooth classic - Peripheral and Central. The Peripheral device, typically a low-power, resource-constrained entity, advertises its availability to accept connections and functions as a server by providing data or services. Conversely, the Central device, usually more powerful and less constrained, initiates the connection, acting as a client that consumes the data or services the peripherals offer.

The operation of Bluetooth LE is sub-divided into time units called events, with data transmitted between devices positioned in these events. The specification [7] recognises five types of these events:

- *Advertising* event includes transmissions on primary advertising channels, which are used for both connectionless communication and connection initiation. With the introduction of two additional advertising events in Bluetooth 5, this type of advertising is now referred to as Legacy Advertising.
- *Extended Advertising* increases the amount of data that can be included in advertising. It uses Legacy Advertising to signal the existence of an auxiliary packet containing the data transmitted on general-purpose channels, called secondary advertising channels in this context.
- *Periodic Advertising* consists of extended advertisements transmitted at regular intervals, enabling multiple devices to receive messages synchronously and conserve energy as they can power down their radios between transmissions.
- *Connection* event involves exchanging connection parameters and bidirectional communication between the central and peripheral devices.
- *Isochronous* events were introduced in Bluetooth 5.2 for time-sensitive data streams, especially audio. They use isochronous channels to ensure that data packets are delivered within strict timing constraints.

We focus on recognising the connection events because they contain bidirectional secure communication between devices.

5.2.1 Advertiser lifecycle

Peripherals that transmit advertising packets on the advertising channels are referred to as advertisers. To provide a clearer understanding of the advertiser's lifecycle, we present in this section an overview of the phases it undergoes. In the schematic in Figure 5.1 we split them by the channel type on the vertical axis, while on the horizontal axis are the non-connected and connected states. The starting state is represented by the dot in the initial node, indicating the beginning of the operation.

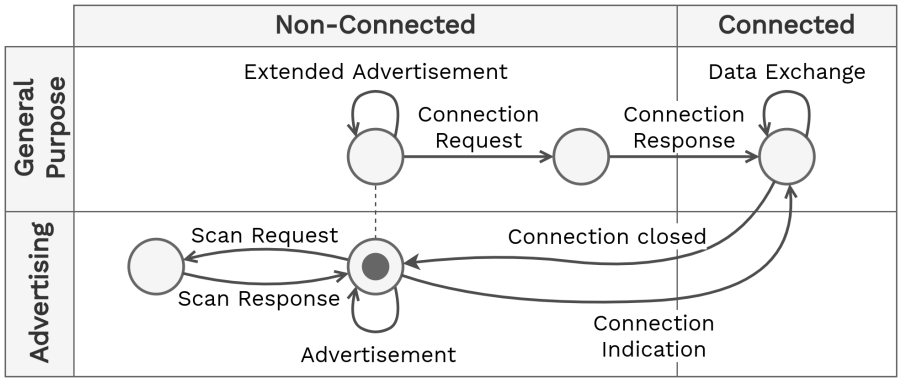

Figure 5.1: Lifecycle of a BLE Advertiser

After initialisation, when the advertiser is ready to provide a service, it starts advertising. Regardless of the advertising event used, this advertising involves periodically broadcasting *advertisements* on primary advertising channels, potentially supplemented by *extended advertising*. Scannable advertisers provide a brief exchange of information where

a *scan request* is responded to by a single-packet *scan response* on the same advertising channel. On the other hand, connectable advertisers allow initiators to engage in bidirectional communication by initiating a connection. In Legacy Advertising, a *connection indication* message on an advertising channel is used, while Extended Advertising performs this process already on general-purpose channels. *Data exchange* always occurs on general-purpose channels, and after the connection is closed, the device returns to advertising.

Depending on the capabilities of the device, we can observe two distinct patterns in advertising:

- *Intermittent pattern*
  In this pattern, illustrated in Figure 5.2, once the device establishes a connection, it ceases advertising until the connection is terminated. Intermittent pattern exhibits the typical limitation of Bluetooth IoT devices, which can maintain only a single connection simultaneously [80]. However, its advantage lies in simplifying the radio and saving resources.

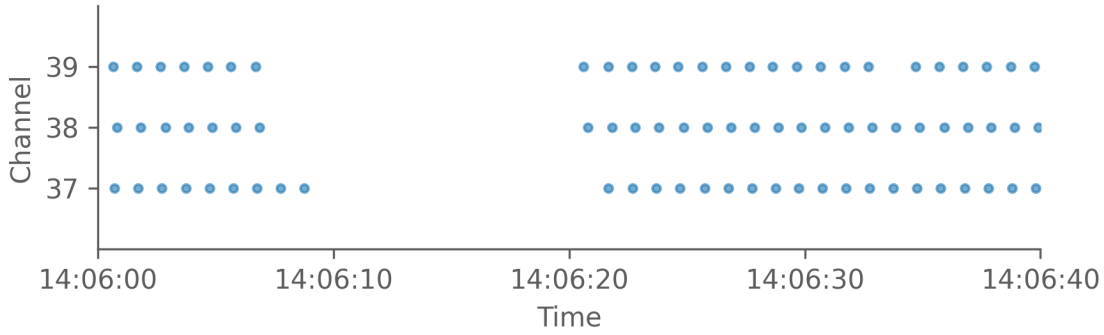

Figure 5.2: Identified **intermittent pattern** of Bentech FP3 lock.

- *Persistent pattern*
  Devices that can handle multiple connections at the same time follow a persistent pattern. This means the device continues to advertise even while connections are active (as shown in Figure 5.3). This approach requires more resources because the device must manage both the connection and advertising channels simultaneously.

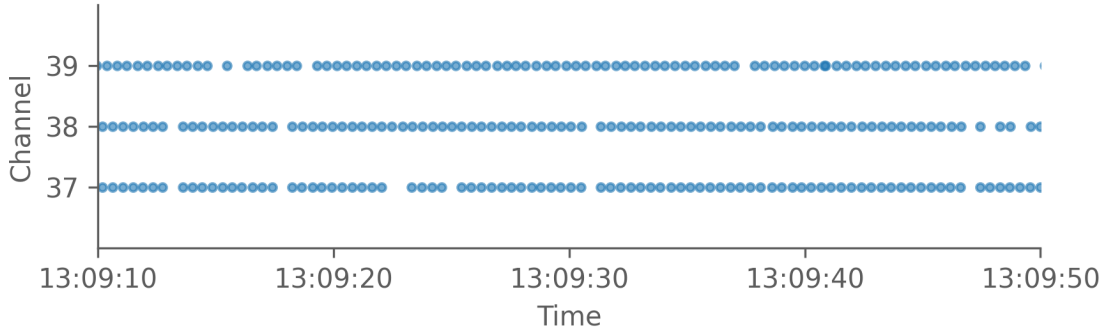

Figure 5.3: Identified **persistent pattern** of igloohome Padlock Lite.

Identifying these patterns poses a challenge since they cannot be directly detected from the packet data alone. Accurately recognising a device's advertising pattern requires ongoing monitoring of its behaviour over time to infer how its advertising relates to its connection status.

5.2.2 Advertising events

Our monitoring relies on Legacy Advertising, which is carried out by each Peripheral and Broadcaster, prompting us to examine it more closely. This advertising consists of regularly repeated Advertising events, during which a single advertising packet (depicted in Figure 5.5) is transmitted. This single packet can be transmitted on all the primary advertising channels or just a subset, as illustrated in Figure 5.4.

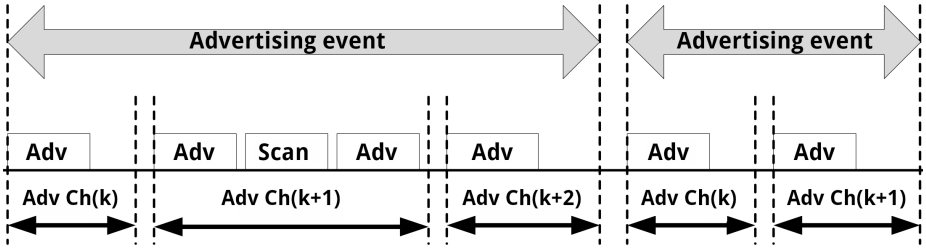

Figure 5.4: BLE Advertising Event [23]

There are several types of advertising packages transmitted on the primary advertising channels depending on their intent:

- `ADV_IND` indicates a device, which supports both scan requests and connections.
- `ADV_DIRECTED_IND` is intended for a specific device, which is allowed to connect.
- `ADV_NONCONN_IND` contains broadcast data of a device that does not support scan requests or connections.
- `ADV_SCAN_IND` announces the support for scan request, one exchange of scan request and response is shown in Figure 5.4.
- `ADV_EXT_IND` is part of extended advertising and points to an auxiliary advertising packet on a general-purpose channel.

The Advertising Interval, which is the time between two consecutive advertising events, can range from 20 ms to 10.24 seconds as per the *low duty cycle* specifications [7]. Additionally, for fast reconnections, the standard allows for a *high duty cycle* advertising, where a device can engage for up to 1.24 seconds in directed advertising with advertising intervals shorter than 3.75 ms. The Advertising Interval is application-specific and its setting affects the data broadcast rate or the speed of the connection establishment. On the other hand, since the advertiser can turn off the radio between events, this can be used to increase battery efficiency. To minimise network collisions, a (pseudo)random *advertising delay* ranging from 0 to 10 ms is added by the Link Layer after each advertising interval.

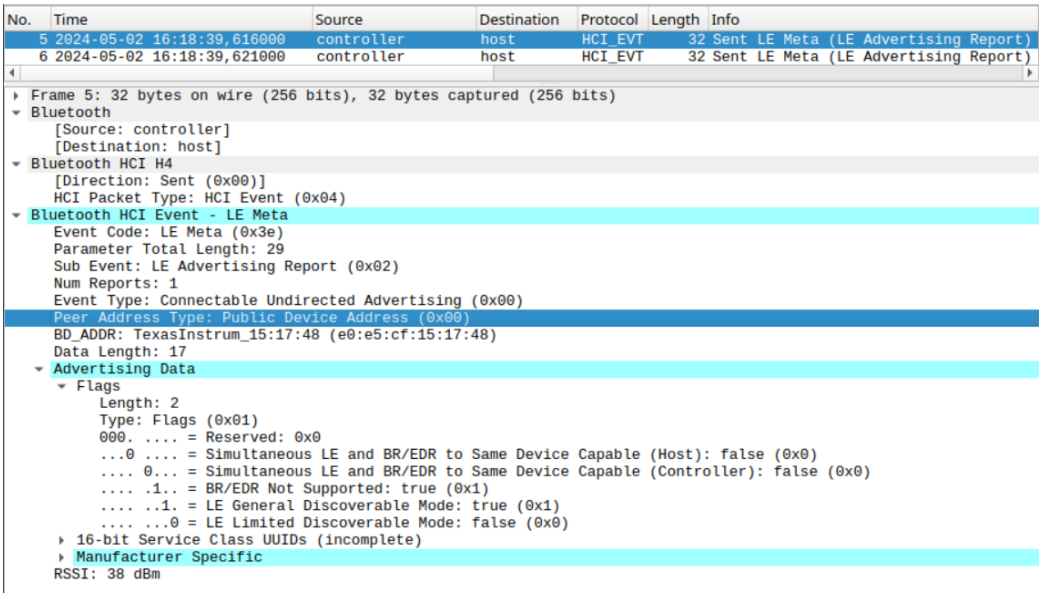

Figure 5.5: Advertisement report as received from HCI interface.

5.2.3 Detection principle

In order to include BLE devices in security monitoring solutions, we need to be able to retrieve some characteristics of their behaviour. This characteristics can be obtained either by directly tapping one of the nodes, usage of broadband signal analysers, monitoring one device at a time or relying on vendor notifications.

We propose an indirect method for monitoring intermittent Bluetooth LE device connections in the vicinity by observing activity on the primary advertising channels [57]. All Bluetooth LE devices that offer services announce their presence on these channels. When a device is connected and follows an intermittent advertising pattern, it temporarily disappears from the primary channels for the duration of the connection. By continuously monitoring Legacy Advertising, the periodic appearance of advertisements from immobile devices can be observed, with only minor variance due to anti-collision delays mentioned in 5.2.2. If a device is absent from the advertising channels for an extended period of time, it can be inferred that the device is engaged in a connection event. Although this method is less reliable than directly listening to connection establishment packets, it has the significant advantage of being compatible with standard Bluetooth chips without requiring modifications. Furthermore, the use of Extended Advertising complicates such connection monitoring, as the establishment can occur on general-purpose channels.

To consider the variability of Advertising Intervals across different applications, we implement a per-device model that captures the unique characteristics of each device. The operation, contained in the schematics in Figure 5.6, could be split into two phases. In the initialisation phase, the model is created, and the Advertising Interval `t`, along with other relevant characteristics, is learned. Once the model is established, it is used to distinguish between the device's passive state, during which it advertises, and a connection event. When the initiator establishes a connection, the device ceases advertising, causing a notable deviation in the expected interval `t`. This deviation does not align with the established model, thus a connection is reported.

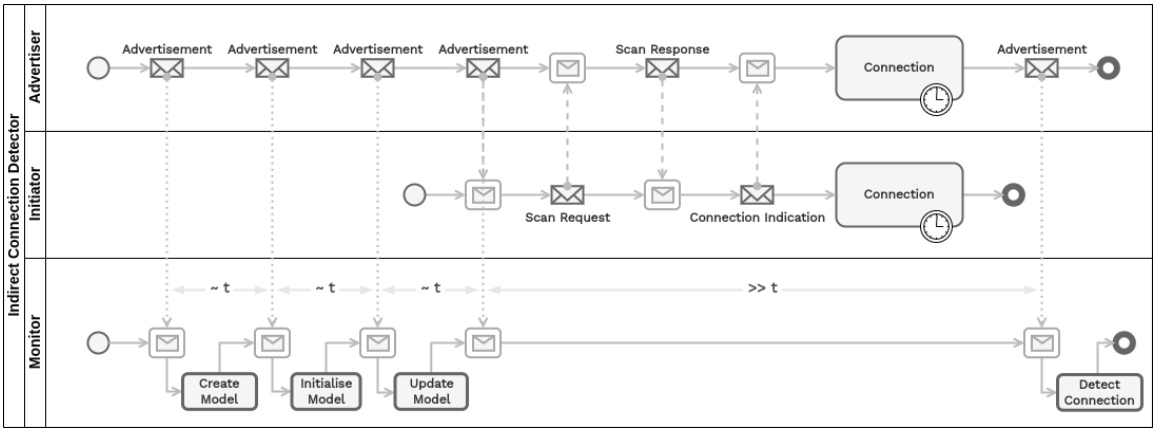

Figure 5.6: The schematics of the proposed connection detector

In the initialisation phase, we expect the device to be in a non-connected state. Even if this were not the case, since the Advertisement Interval is learned from consecutive advertisements, this would not affect the generated model, only delay its initialisation for a period of time after the connection is complete. If the connection event occurred during the initialisation, it would bias the generated model. In order to overcome this problem, the model is updated over time, ensuring adaptation to changing conditions and refining the model by capturing similar errors.

Based on this formulated principle, we focus on identifying the appropriate models and detection methods that are best suited to distinguish between passive advertising and connections.

5.3 Verification methodology

To verify our idea for connection monitoring in Bluetooth LE networks, we conducted several research phases that addressed the key research questions that needed to be answered. The first research question focuses on the fundamental verification of the functionality of this approach, leading us to formulate the following research question:

> **RQ1**: *Can the proposed connection monitoring principle be effectively applied to track activity of Bluetooth LE devices with intermittent advertising pattern?*

In order to answer this research question, we conducted an experiment as detailed in Section 5.4. We chose five Bluetooth LE devices that represent different types of IoT devices. We implemented a proof-of-concept solution consisting of a collector to gather advertising packets and a detector to create a device model and identify connections based on the collected data. Using the collector, we gathered a dataset of the advertising activity of these devices. We manually inspected the dataset and confirmed that some devices exhibited an intermittent pattern, and we could identify connections by observing interruptions in advertising activity. We then proposed two basic detection methods using simple statistics, incorporated them into the detector, and evaluated their performance for our specific problem.

We demonstrated that the concept was valid and capable of detecting connections on a standard Bluetooth LE controller. However, since the quality was insufficient, we have

decided to subject this issue to further investigation in the areas we have identified as problematic. The first area we examined was the advertising collection, where frequent dropouts in advertising messages reduced the accuracy of the statistical methods used for detection. Based on this, we defined the following research question:

> **RQ2**: *Would optimised advertising collection hardware improve detection efficiency?*

Answering this question required us to develop a custom advertising sniffer, which is described in detail in Section 5.5. This sniffer is capable of parallel collection of advertisement packets from all three advertising channels. By employing this parallelism, we address the challenge of channel hopping typically performed by standard controllers, aiming to minimise packet dropouts. Moreover, as this collector is able to identify on which channel the advertisement was captured, it allowed us to delve deeper into the collection process and better understand its behaviour and shortcomings.

We used this sniffer to compile an advanced dataset, described in Section 5.6. The dataset includes a representative sample of nine devices from four functional classes. We captured it simultaneously using both our custom sniffer and the standard controller originally employed to compare the quality of the capture. By testing the same detection methods as in the proof-of-concept solution on data captured by the sniffer, we could address research question *RQ2*.

After addressing the capture phase, our attention shifted to identifying higher-quality methods for detecting connections from the measured data. To achieve this, we utilised the compiled advanced dataset. Initially, we approached the problem as an outlier detection task based on the premise that a connection would likely behave as an outlier compared to the passive state of the device. Consequently, we formulated the research question as follows:

> **RQ3**: *Can outlier detection methods enhance the identification of connections in device advertising activity?*

For the sake of completeness, we decided to test the suitability of the artificial intelligence approach in addition to outlier detection methods and formulated the following question:

> **RQ4**: *Can artificial intelligence methods enhance the identification of connections in device advertising activity?*

To address these questions, we once again designed an experimental procedure. We undertook the following steps, which are detailed in Section 5.7. First, we used our dataset, focusing on the data from intermittent devices measured by our sniffer. We manually labelled this data to allow us to assess the accuracy of the methods being tested. From the field of outlier detection, we selected five methods that represent different categories of algorithms: ARIMA and GMM from the statistical-based category, LOF from the nearest-neighbour-based category, OCSVM from the classification-based category, and iForest from the projection-based category. In the domain of AI, considering the anticipated simplicity of the problem, we chose to use a basic multi-level perceptron with minimalist neural networks. We implemented all methods in two steps: training and testing. To identify suitable parameters, we employed a grid search approach.

We then evaluated all the methods on data measured in a laboratory environment and in the relevant office environment. Ultimately, we identified a method with the necessary quality for the final product, confirming the discovery of a novel principle for Bluetooth LE security monitoring, enhancing the security of SOHO environments.

5.4 Proof-of-concept verification of the monitoring principle

In this section, we focus on verifying the feasibility of the proposed Bluetooth LE connection monitoring principle. To gain a deeper knowledge of applicability in a real environment, we implemented a Proof-of-Concept (PoC) solution consisting of two modules, a collector and a detector, which can be integrated into the framework from Chapter 4. This solution corresponds to the *Monitor* swim-lane of the Schematics 5.6 given in the description of the monitoring principle in Section 5.2.3. This PoC solution was used to create a fundamental dataset containing labelled records of advertisement occurrences, which was subsequently analysed and utilised for connection detection experimentation.

5.4.1 Proof-of-Concept solution

Our PoC solution consists of two modules: the collector and the detector. These modules correspond to the basic building blocks of the NEMEA Framework described in Chapter 4.2 and are visualised in the system architecture shown in Figure 5.7. The Advertising Collector scans advertising channels and reports all detected advertisements, which can be stored to create an advertisement dataset. The Connection Detector analyses these advertisements and notifies of detected connections. Since the detector is separate from data collection, it can be run repeatedly over the collected data. A significant advantage of this modular design is that the modules can operate independently. For instance, the detector can be integrated into a multi-tier solution, such as fog computing, where it can run on more powerful hardware while the collector is deployed within the monitored network.

In the overview in Figure 5.7, the advertisements from BLE nodes are captured by the *Advertising collector* and reported upwards as a stream of advertising information. Three missed advertisements can be noticed (red underscores) for Node 1, which are then collectively identified by the *Connection Detector* for the connection and reported as information that the device was in use at a certain time and for a certain period of time (highlighted in red).

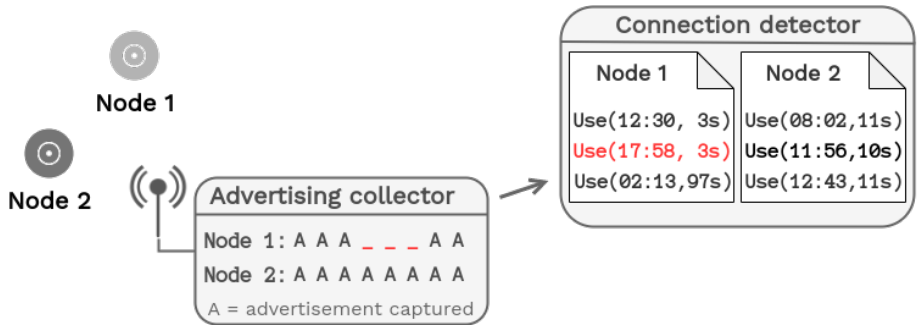

Figure 5.7: Architecture of the implemented solution

The PoC solution was developed to operate on a standard Host-Controller Interface (HCI) and is therefore compatible with any common Bluetooth controller. We created

the solution using the Python programming language and made the source code publicly available. You can download the source code from the following URL: https://nextcloud.fit.vutbr.cz/s/T7bg4tQs6RpeMAR

Collector

The collector module is based on several key libraries. It uses a modified version of the `PyBT` library[5], which provides an abstraction layer for communicating with the Bluetooth controller. The module depends on the well-known packet manipulation library `Scapy` for processing packets. To ensure compatibility with the NEMEA Framework [126], the collector module uses TRAP as its communication interface, enabling the exchange of messages in the UniRec data format.

The collector supports only one parameter, which is the output TRAP interface where the advertisements shall be reported. Because the communication with the Bluetooth socket requires elevated rights, this module ought to be run with root privileges.

When started, the collector first sets up the Bluetooth socket for passive scanning (resetting it if necessary) and then begins the scanning procedure. Each notification received through the HCI interface is converted into a UniRec structure with four entries – BDADDR containing the device address, TIME the timestamp of the received notification, RSSI signal strength and ATYPE the device address type (public or random). This structure is then sent via the specified interface for further processing (by the detector).

Detector

The detector receives advertising records in UniRec format from the collector and processes them using the selected model based on the configuration. For each recognised device, identified by its unique BDADDR, the detector initialises a model specifically for that device. All advertisements from the same device are then processed using this model. Once the model is initialised, it detects extended periods without advertisements and interprets these intermissions as potential connections, which are reported accordingly.

The output of this detector is two files – an *alert* CSV file containing the detected connections and a *model* CSV file containing the intermediate state of the model after each received advertisement for thorough inspection.

We have implemented two simple testing models for initial evaluation, both based on basic statistical methods – *SimpleStatisticsModel* and *SlidingWindowModel*.

- `SimpleStatisticsModel`

SimpleStatisticsModel, descriebed in Algorithm 1, uses a moving average of silence intervals between subsequent occurrences of advertisements. For every advertisement we calculate the silence interval from the last seen advertisement and difference from the mean interval. The model consists of the moving average and threshold, which is set as the maximal observed difference. If the difference is more than twice as large as threshold, a connection is reported and model not updated.

- `SlidingWindowModel`

The second method utilised a sliding window for a better statistical overview of the data, which is why we call it SlidingWindowModel. We have tried several simple statistical

[5] https://github.com/mikeryan/PyBT

Algorithm 1 SimpleStatisticsModel detection

```
threshold
moving_mean
last_adv_t                                          ▷ Last advertisement timestamp

procedure PROCESS_ADVERTISEMENT(t)                  ▷ t is the advertisement timestamp
    interval = t − last_adv_t
    difference = |interval − moving_mean|
    if difference > 2 * threshold then
        output Connection detected
    else                                            ▷ Update the model
        threshold = MAX(threshold, difference)
        moving_mean = (moving_mean+interval) / 2
```

methods but the mean showed the best results. As described in Algorithm 2, we use the mean of the window and its standard deviation to allow for fluctuations in the silence intervals. If the silence interval is more than twice the average plus the deviation, we can conclude that at least one Bluetooth advertising event was missed and proclaim it a connection.

Algorithm 2 SlidingWindowModel detection

```
window = [ ]
last_adv_t                                          ▷ Last advertisement timestamp

procedure PROCESS_ADVERTISEMENT(t)                  ▷ t is the advertisement timestamp
    interval = t − last_adv_t
    mean = MEAN(window)
    std_dev = STANDARD_DEVIATION(window)
    if interval > 2 * mean + std_dev then
        output Connection detected
    else                                            ▷ Update the model
        POP(window)
        APPEND(window, interval)
```

The detection can be improved by omitting clearly incorrect data from the model. Even though we tested only BLE devices with a low duty cycle, which can never have the interval between advertisements shorter than 20 ms, the controller sometimes reported advertisements with an interval of just a few ms. These erroneous data are ignored when fed into the SlidingWindowModel.

5.4.2 Environment setup

We have conducted the experiment in two different environments. At first, we captured the data in an office with common interference factors (other Bluetooth and Wifi devices) to represent real-world environment. The data from this environment was preprocessed before running the PoC detector so that only the communication of the monitored device was filtered out. Both the collector and detector was run on Raspberry Pi 4 model B, connected via the Ethernet port to minimise interference on the baseband chipset.

The second part of the experiment was performed in a shielded room with only the monitor and tested device present to ensure data purity. A laptop ThinkPad T14 gen 2 has been used because of its portability and easy operability.

In both environments the test was run on internal Bluetooth controllers using the built-in HCI commands. It was experimentally verified that both chips detect advertisements on all channels, likely achieved by opportunistically switching the scanned channels. The setup always consisted of the device running the detector, exactly one device under test and the smartphone that controlled it. All three nodes were placed approximately 50 centimetres apart in a roughly triangular shape.

A total of five consumer-ready devices with different operating characteristics were included in the experiment. A total of three devices were mains powered, namely two light bulbs and a power outlet. Two devices were battery powered, namely the thermometer and the tracker. The concrete devices are stated in Table 5.1. Using a diversity of devices provides a comprehensive view of how different devices behave and whether our monitoring principle can be applied.

5.4.3 Dataset capture

We used the collector module to gather the dataset, focusing on a range of typical use cases for each tested device based on their specific capabilities. Each device includes at least one passive use case where the device remains unconnected and no interaction occurs. Other use cases were designed to encompass various interactions with the devices, including turning them on and off (for light bulbs and power plugs), adjusting the colour or brightness of lights, or retrieving the current state, such as temperature and humidity.

The capture itself took place in a defined environment, where only the collector (running either on RPi or laptop) and the device under test were present. All other devices from our dataset were switched off for the duration of the test. The test always consisted of capturing just one device and a use case in a separate file that was labelled accordingly. Each test was repeated several times to eliminate fluctuations and accidental interference.

To capture a sample, the collector has been started with the output sent to a named UNIX socket (the collector requires root privileges for managing Bluetooth socket):

```
# ./ble_adv_scanner.py -i "u:adv_scan"
```

And a `logger` from NEMEA modules[6] has been used to capture the UniRec messages from the socket and store it into a CSV capture file:

```
$ logger -t -i "u:adv_scan" > {date}_{device}_{usecase}_{run}.csv
```

This process allows for a later replay of the stored messages back into the NEMEA system by another module – `logreplay`, or a different analysis or preprocessing of the stored data thanks to the accessible CSV format.

We have published the full dataset[7] containing not only the raw captured advertising data, but alongside also the cleaned data, which are filtered to contain only messages from the target device (filters out crosstalk from other devices) and data converted to XLSX files for manual inspection and analysis.

[6] https://github.com/CESNET/Nemea-Modules/tree/master/logger
[7] https://nextcloud.fit.vutbr.cz/s/oCHpFXXYYeQNcHs

Table 5.1: Identified characteristics of the monitored devices

Device	Advertising interval	Standard deviation
Phillips Hue White	324 ms	159
Revogi Delite 2	55 ms	24
Phillips Hue Smart plug	323 ms	165
Mi Temperature and Humidity Monitor 2	1689 ms	1053
FIXED Smile	3072 ms	1909

5.4.4 Data analysis

We decided to perform the analysis in two steps. At first, we manually inspected the data to verify whether the data flow corresponds with the expected behaviour. We focused on identifying the *advertising interval* and verifying whether the connection is distinct enough from this interval. The general statistical methods and tools were used for this verification.

Afterwards, we ran the implemented connection detectors over the dataset to evaluate the proposed statistical models. We evaluated the identified connections against expected outcome depending on the use case and observed the internal behaviour of the model.

In the experiment, we have captured a dataset containing 118 distinct behaviour samples of five IoT devices. These samples contain 19 passive captures, 74 single action (connection) captures, and 25 captures contain two actions. In total, 59 captures were taken by collector running on Raspberry Pi in the office environment, and 40 were taken by laptop in the shielded room. This dataset gives us a base to evaluate the device behaviour and method applicability.

We manually derived the advertising interval as the most frequent interval between consecutive advertisements and noticed that extended intervals that are a multiple of the advertisement interval occurred quite frequently. This is most likely due to missed advertisements in the capture, which we believe to be caused by the vendor's implementation of advertisement scanning. The deduced advertising interval and standard deviation of the intervals are stated in the Table 5.1.

From the data we verified that the typical connection is noticeable as it is longer than three times the normal advertising intervals, which can be noticed even if the capture is impure with some advertisements omitted (see Figure 5.8). This supports our idea that this channel can be used to reveal the connections.

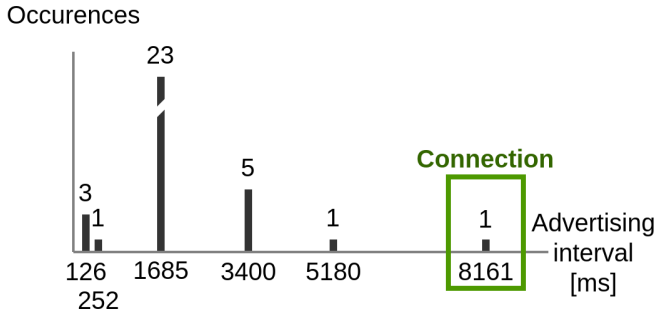

Figure 5.8: Distinct connection of Mi thermometer

However, there are significant differences in the behaviour of the devices and the way the control app uses the Bluetooth connection. Some apps use the short connection to exchange a single command, while others keep the connection alive. The short connections lead to challenging detection when the scan of the advertisements is not flawless as the duration is only slightly longer than the advertising interval (see Figure 5.9). The battery-powered devices, however, use longer advertising intervals to conserve energy. This may result in the connection not lasting significantly longer than the outage of a single advertising message and thus make it more challenging to recognise.

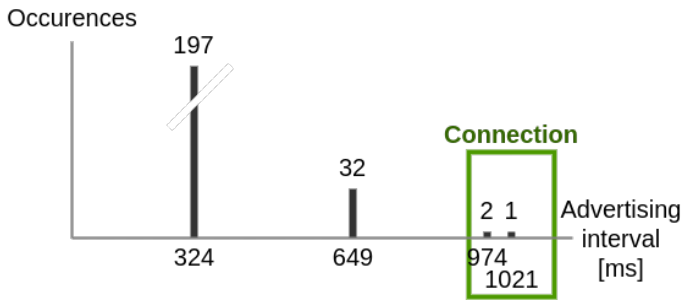

Figure 5.9: Phillips Hue connection indistinguishable from advertisement dropouts

5.4.5 Evaluation

After the analysis we can answer the research question *RQ1*. The proposed method works on devices that follow an intermittent advertising pattern with a connection time at least four times greater than the advertising interval. If the connection length is sufficient, it is possible to unambiguously differentiate between accidentally missed advertisements and a real connection. This need for connection length in relation to the advertising interval arises because there are often dropped single advertisements and also dropped two consecutive advertisements.

Unfortunately, the simplistic detection methods based on average are too weak to differentiate a connection from the more significant fluctuation or missed advertisement. They are able to recognise the distinct connections, but show subpar results for devices with shorter connection times. The conditions based on the standard deviation fail when the sampling window contains values with small variability and a larger fluctuation occurs, while the condition based on maximal difference fails when there is a more significant fluctuation in the setup phase and omitted advertisements occur before the actual connection.

As a result, we formulated research questions RQ2-4, as was stated in Section 5.3. These questions aim to improve detection efficiency by addressing the identified shortcomings. First, we focus on enhancing the quality of the collector to minimise the advertisement drop rate. Then, we work on identifying better detection methods that can accurately distinguish between omitted advertisements and actual connections.

5.5 Parallel monitoring probe

In this section, we discuss the development of a custom monitoring probe designed to improve the quality of collected data compared to collecting advertising messages from a standard Bluetooth controller. In order to minimise the omission of advertisements, we

propose that our probe monitors all three primary advertising channels simultaneously, loosely following the model proposed by Wu [115]. Unlike Wu, who utilises Ubertooth One, we have built our probe on common ESP32 chips. Apart from being more readily available and cost-effective, the ESP32 offers an HCI interface that enables us to collect data similar to that obtained from Bluetooth controllers. In contrast, the Ubertooth is a more specialised device that provides access to full frames, including the Link Layer, which remains inaccessible via HCI. By using ESP32, we obtain data that is comparable to that from a Bluetooth controller, allowing our findings to be directly applicable to solutions using common controllers.

The monitoring probe, whose schematic is shown in Figure 5.10, consists of three collectors, each dedicated to monitoring a specific primary advertising channel. These collectors are connected via an USB hub to a central monitor, represented by Raspberry Pi 3B+ (RPi). This central monitor is responsible for collecting the data and storing it in a CSV file. The resulting CSV file follows the same structure as those obtained by the PoC collector, ensuring compatibility and comparability between datasets.

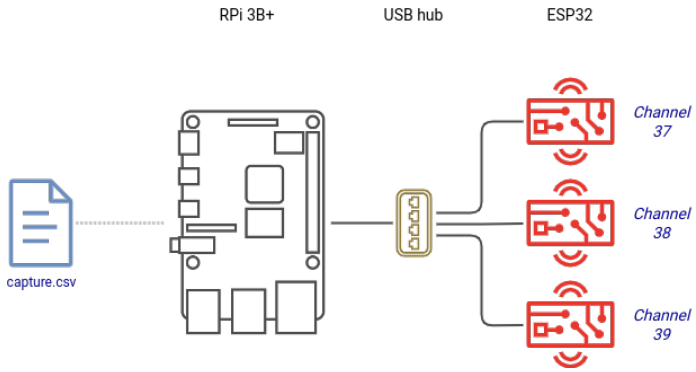

Figure 5.10: Scheme of the monitoring probe.

For the central monitor, we used the same model of Raspberry Pi (RPi) as in the PoC solution to ensure consistency across studies and facilitate effective cross-validation of results. We developed Python software to manage the reception of advertising reports via serial connections and their subsequent storage in the dataset. Additionally, thanks to the use of RPi, it was possible to run the PoC collector on its Cypress CYW43455 Bluetooth controller (RPi controller), allowing us to measure the same data simultaneously with both approaches for direct comparison.

Each collector was implemented as a stand-alone ESP32 chipset [42], running custom firmware that we developed to utilise the internal Bluetooth controller. This firmware was designed to listen to a single advertising channel and relay the captured advertising reports upstream via a serial link. Since the ability to lock onto a single channel is not typically available, we collaborated with Espressif to gain access to the necessary low-level API. By employing the Espressif-supplied extension, we successfully configured the Bluetooth radio to listen exclusively on a selected channel. To verify the correct behaviour, we implemented a Bluetooth advertiser that operated on only one channel during Advertising events and confirmed that the collector only reported Advertising reports transmitted on the chosen channel.

Our monitoring probe, depicted in Figure 5.11, was designed to capture precise data, and accurate timestamping of advertisement captures was a significant concern, as our

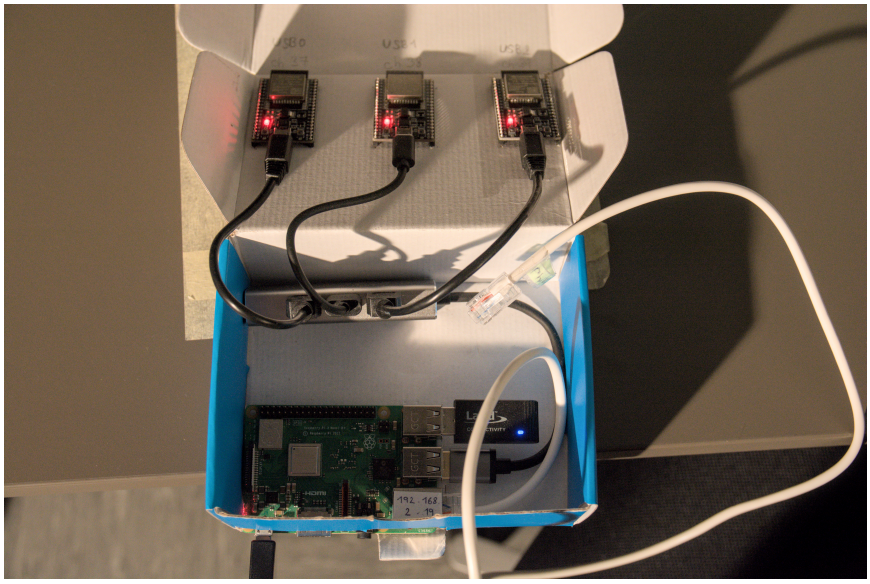

Figure 5.11: Photo of the built monitoring probe.

monitoring principle relies on the timing of advertising reports. Due to potential errors introduced by communication delays and the fact that the central monitor doesn't operate on a real-time OS, we decided to timestamp the advertisements directly on the ESP chips. However, since these chips lack real-time clocks, synchronisation between collectors and the central monitor is essential for observing inter-channel behaviour. To address this, we synchronised the start times of all ESP32 chips using the Data Terminal Ready (DTR) signal and then translated the collector timestamps to real-time on the central monitor. More details are provided in Appendix A. Upon testing, we found that the internal clocks of the ESP chips exhibited a time skew of approximately 2 milliseconds over 10 minutes, which is acceptable given that our dataset samples are 1 minute long and a random anti-collision delay of 10 seconds is applied to each advertising event.

5.6 Dataset on BLE advertising behaviour

To evaluate the effectiveness of the monitoring probe described in the previous section, we conducted a dual advertisement collection process. This process involved using both the newly developed monitoring probe and the PoC solution operating on the Bluetooth controller of the Raspberry Pi 3B+ (RPi BLE controller). This dual collection approach allowed us to directly compare the measurements obtained from each method, which facilitated a thorough assessment of the probe's performance. As a result, we compiled a unique dataset called BLE-ARD, which contains samples of advertising flows from various representative devices. By making this dataset available to the public[8], we aim to support researchers in enhancing detection methods that focus on advertising interval analysis.

The BLE-ARD dataset comprises measurements from nine different devices in two environments - a shielded chamber and an office. For each environment, 210 single action samples and 45 passive samples are included, for a total of 510 samples corresponding

[8]https://github.com/hujon/BLE-ARD/

to 8.5 hours of data per collector. This provides a robust foundation for analysing the behaviour of the devices and the connection detectors.

Each sample in our dataset is stored in a capture file named after the action it contains and includes records of received advertising reports. Every record within these files contains the reception timestamp, the Bluetooth address of the transmitting device, the type of address (public or random), the type of advertising, and the received signal strength (RSSI). Additionally, for data collected by our probe, the device name (if available) and the channel on which the advertisement was received are also recorded. The capture files themselves are organised in a hierarchical tree structure, where the highest level denotes the collector (monitoring probe or the RPi BLE controller), the middle level specifies the environment, and the lowest level identifies the specific device.

5.6.1 Devices

Given the dataset's focus on security monitoring, we selected a representative sample of currently available consumer-ready IoT devices. This includes various types of door locks, padlocks, light bulbs, power plugs, and sensors, essential for a thorough analysis of security vulnerabilities in smart home environments.

The devices are organised into functional classes such as Locks (two door mounted locks and two padlocks), Lighting, Actuators, and Sensors, each addressing specific security needs and operational characteristics. This division is depicted in Table 5.2.

Additionally, the devices can be categorised by the power source. Philips Hue devices and Revogi bulb are mains powered, while the majority of the included devices are battery powered. This distinction directly affects the advertising features as the battery powered devices have to balance the availability and power consumption. Longer advertising interval increases the connection time or data availability, but because such devices go into a deep sleep between the advertisements and turn off their radio modules, this extends battery life.

Table 5.2: Devices contained in the dataset

Device class	**Device**
Lock	**Bentech** FP3
	Danalock V3-BTZE
	igloohome Padlock Lite
	Nuki Smart Lock 3.0
Lighting	**Philips** Hue **white**
	Revogi Bluetooth LED bulb
Actuator	**Philips** Hue Smart **plug**
Sensor	**Mi Temp**erature and Humidity Monitor 2
	BeeWi Motion Sensor

5.6.2 Monitored actions

In order to observe the behaviour of the devices as they are used, we had to derive a set of actions to be performed and recorded. For each device, we studied all the user actions

that can be performed through their respective control mechanisms and created a set of common actions for each functional class (given in Table 5.3).

For locks, there are lock and unlock actions (except for Bentech devices, which cannot be locked via the app) and retrieval of the device's activity log (available for Nuki and igloohome devices). Bulbs offer a range of actions from simply turning on/off, adjusting the luminance level or colour, to selecting a predefined effect. The power plug only offered basic on/off actions and the sensors only allowed for data extraction.

Table 5.3: Functional class device actions

Device class	**Actions**
Lock	lock, unlock, log
Lighting	on, off, luminance, colour, effect
Actuator	on, off
Sensor	data

5.6.3 Measurement methodology

The measurement methodology for creating the dataset involved a dual capture approach, where data collection was conducted simultaneously by the RPi BLE controller and a monitoring probe. This dual capture setup was crucial for ensuring comprehensive data collection and verification.

Each measurement was conducted in two distinct environments: *a shielded chamber* and a typical *office setting*. The shielded chamber provided a controlled environment free from external interference, allowing for the collection of clean data. Conversely, measurements in the office environment provided valuable real-world data, which included noise and interference from the surroundings. This provided insight into how the equipment performs under typical operating conditions. To ensure the accuracy of the results, the office measurements were conducted during working hours when more people were present and actively using WiFi and Bluetooth devices.

In both measurement environments, the monitoring setup included three key components: one target device, an Android device (Lenovo TB-8704X with Android 8.1.0) equipped with the necessary control application, and a monitoring probe. These devices were arranged about 50 cm apart on a tabletop, positioned to form an equilateral triangle without any physical obstructions between them as depicted in Figure 5.12. The monitoring probe itself was managed from a laptop connected via an Ethernet cable, with all potential radio interference from WiFi and Bluetooth disabled to maintain the integrity of the data collection process.

The measurement protocol was designed so that each session lasted for one minute. After the initial 20 seconds of collecting baseline data, a specific action (as identified in the previous section) was manually performed by the operator. Each session focused on a single action to ensure clarity and consistency in the results. To further validate the data, passive measurements were also taken, where no actions were performed and the control application was not running. Each type of measurement, both active (involving actions) and passive, was repeated five times. This repetition was crucial for minimising measurement errors and ensuring the reliability of the dataset's findings.

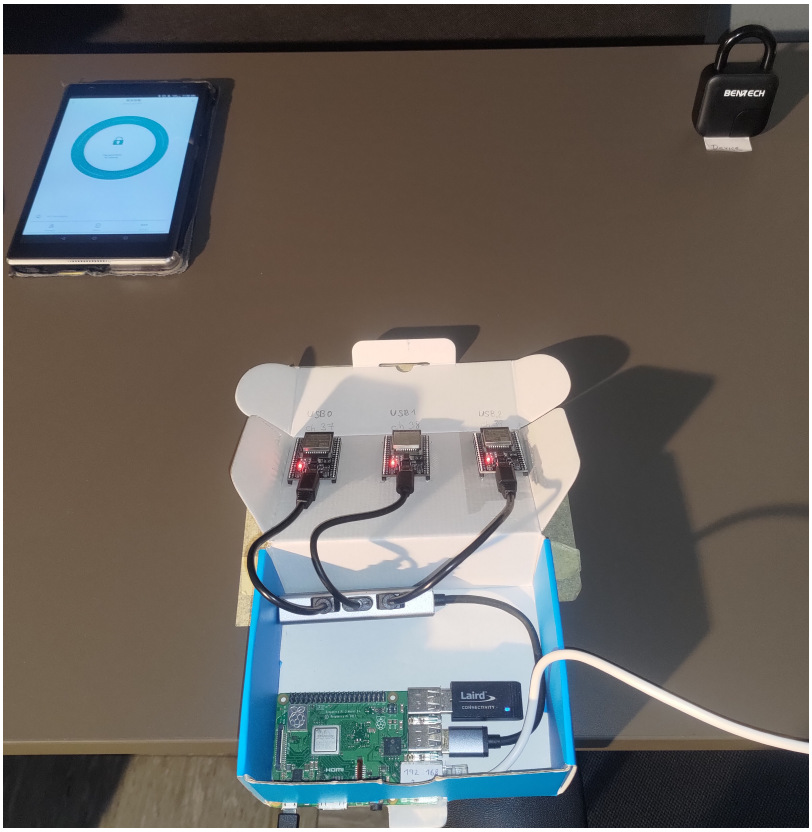

Figure 5.12: Monitoring setup in the shielded chamber environment.

5.6.4 Processing

The dataset obtained through the initial measurement methodology required further processing to enhance its clarity and usability. From the original data we derived two modified versions by applying two processing steps - data cleanup and data filtering.

In the *data cleanup* step, we addressed issues with Advertising Records that had identical timestamps, addresses, and channels but slightly different RSSI values, filtering out these as duplicates to streamline the dataset. Additionally, the asynchronous arrival of advertising reports due to the monitoring probe's parallel nature required reordering to achieve chronological consistency.

In the *data filtering* step, the focus was on isolating relevant advertisements. The capture files were specifically filtered to only include advertisements that originated from the Bluetooth addresses of the devices identified in the study.

5.6.5 Analysis

We conducted an analysis of the dataset to understand its features and evaluate whether the monitoring probe enhanced the collection of advertisements. Our objective was to assess the dataset's ability to accurately reveal the advertising intervals of the monitored devices, determine the drop rate of received advertisements, and distinguish actual device connections from advertisement omissions.

As a preliminary step in our statistical analysis, we calculated advertising deltas for each record across all samples. These advertising deltas represent the time elapsed since the

last advertisement from the same device on the same channel. Calculating these intervals allowed us to assess the regularity of advertising behaviours, providing a basis for more detailed analysis of the data.

For each sample, we calculated four key indicators: the advertising interval, the percentage of records matching this interval, the estimated error, and the connection length for samples involving actions. These metrics helped us assess the quality of each sample and observe variations across different environments and measurements. To evaluate the performance of the two collection methods, we averaged these indicators for all samples from each device. The resulting values, expressed in milliseconds, are detailed in Tables 5.4 and 5.5, organised by the collection method used.

Table 5.4: Dataset summary (RPi BLE controller)

Device	**Adv. interval**	**Error estimate**	**Conn. length**
Bentech	1006 (79%)	70	13944
Danalock	505 (88%)	20	3926
igloohome	410 (80%)	18	1884
Nuki	317 (60%)	39	3504
Philips white	859 (85%)	42	2318
Revogi	56 (86%)	1	6691
Philips Plug	326 (79%)	14	1278
Mi Temp	2249 (85%)	188	15659
BeeWi Motion	869 (56%)	60	4401

Table 5.5: Dataset summary (Monitoring probe)

Device	**Adv. interval**	**Error estimate**	**Conn. length**
Bentech	1003 (89%)	59	14142
Danalock	500 (93%)	15	3900
igloohome	414 (88%)	12	*Persistent adv.*
Nuki[9]	*339 (57%)*	37	3136
Philips white	856 (92%)	28	*Persistent adv.*
Revogi	54 (83%)	1	6800
Philips Plug	327 (88%)	23	*Persistent adv.*
Mi Temp	2191 (89%)	124	15735
BeeWi Motion	865 (61%)	47	4294

The advertising interval was determined by taking the median value of the advertising time differences assuming that the majority of the advertisements are received at intervals consistent with the device's set advertising interval.

To assess the dropout rate, we calculated the percentage of matching records that fall in the same cluster as the specified advertising interval, as opposed to multiples of that interval. We used half the distance between two intervals as a discriminating criterion.

[9] Oddities in the advertising interval explained in Section 5.6.6

For active samples, the connection length was determined by identifying the maximum advertising delta. This maximum value indicates the longest time gap between received advertisements, suggesting the duration of a connection event.

Finally, we evaluated the reliability of the data by calculating an estimate of the standard error of the advertising deltas. First, the standard sample deviation of the deltas was calculated, deliberately excluding identified connections. The error estimate was calculated from this deviation to provide a measure of the variability of the deltas, which helps to assess the potential rate of dropout.

5.6.6 Observations

The advertising intervals determined from the data measured by our monitoring probe and the RPi BLE collector differed only marginally, the common difference was under 2% which we consider negligible. This suggests that we revealed the real advertising profile of the devices, and the measurements made by the common RPi controller are sufficient for this profiling. Thus, it turns out that using dedicated hardware and parallel channel tracking does not benefit our principle based on advertising interval tracking.

The exception was Nuki Smart Lock, which displayed several distinct advertising intervals throughout our observations. It primarily advertised at intervals of approximately 1016 ms with short bursts of faster advertising at the rate 51 ms lasting around 5 seconds. Interestingly, in passive captures in office environment Nuki exhibited a third distinct advertising interval of 417 ms. This variable behaviour caused a loss of precision in the statistical processing of the device.

Even if our probe monitored continuously all three primary advertising channels, the rate of omissions dropped on average only by 5% further indicating that the common RPi controller manages to capture the advertisements satisfactorily.

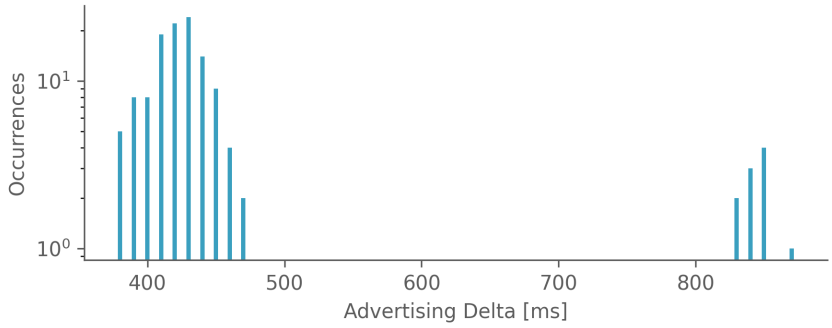

Figure 5.13: Persistent advertising pattern histogram [igloohome]

We have identified a persistent advertising pattern in the igloohome and both Philips devices. In Figure 5.13, a histogram illustrates an unlock capture from the igloohome device, showing that the advertising deltas cluster around the established advertising interval and its multiples. For comparison, Figure 5.14 presents a histogram for the same action performed on the Danalock device. As this device follows the intermittent pattern, we can identify distinct clusters around the advertising interval and its multiples on the left side, with the connection distinctly separated on the right side by a noticeable gap.

The Nuki Smart Lock also exhibits signs of a persistent advertising pattern, although the variable advertising rate mentioned earlier reduces the certainty of this observation.

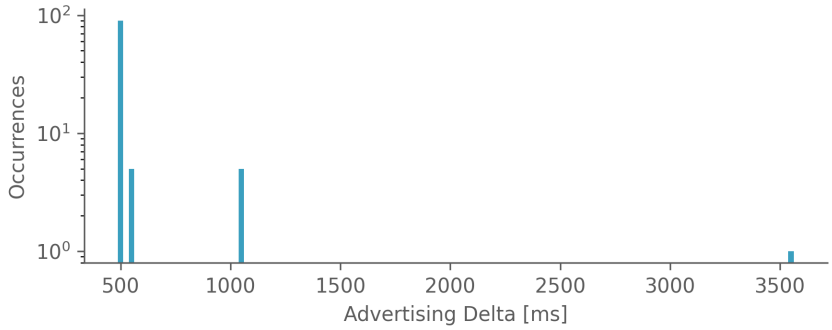

Figure 5.14: Intermittent advertising pattern histogram [Danalock]

We believe that the bursts of shorter advertising intervals may occur following a connection, serving to enhance the device's response time after an action is performed. This characteristic could potentially be used to detect connections in this specific device.

We observed distinct behaviours driven by the specifics of their control applications. Devices such as the Danalock and Mi thermometer initiate connections in response to user actions, making these actions easily detectable by our monitoring system. In contrast, devices like the Bentech lock and Revogi bulb establish a continuous connection once their application is activated, which generally renders user actions invisible to our monitoring efforts. However, we are still able to monitor the presence of the application connection itself. Given that it's unlikely for users to keep the control application running continuously and close to the device, this level of monitoring may be adequate for practical purposes.

5.6.7 Dataset evaluation

We managed to successfully compile a significantly larger dataset than the one used in Section 5.4, with 510 samples per collector compared to the 118. This dataset not only expanded in quantity but also broadened the scope of device coverage from 5 to 9, specifically focusing on security-oriented IoT devices.

We conducted a direct comparison between the data captured by the RPi BLE controller and data from our monitoring probe to answer the research question *RQ2*. After thorough analysis we can conclude that enhanced collection method, which involves simultaneous monitoring of all three advertising channels, achieves slightly better but overall comparable data quality to that obtained using a common Bluetooth controller. As the data captured by the two methods are analogous, no significant improvement in the performance of the original detection methods can be expected. However, to rigorously confirm and definitively answer the question, we test the original detection methods over this dataset in the following section.

Our analysis revealed that even the improved collection method still remains susceptible to message dropouts, indicating that detection methods must be designed to accommodate this limitation. On the positive side, with further refinement, these detection methods would not require any specialised hardware since its benefit is negligible.

As a side note, during our analysis, we manually identified the advertising patterns. However, it would be preferable and potentially more efficient for future research to develop a classifier-based approach that can automatically differentiate between these patterns. This

is crucial, especially since the proposed monitoring principle is effective only for devices that exhibit the intermittent advertising pattern.

5.6.8 PoC detection methods evaluation

This section aims to definitively answer the research question *RQ2* by applying the PoC solution detection methods described in Section 5.4.1 to the BLE-ARD dataset. By comparing their behaviour when applied to RPi BLE collector and monitoring probe measured data, we can discover whether the detection efficiency improved.

To assess the quality of these detection methods, we primarily used two metrics - precision and recall. Precision measures the proportion of correctly identified connections among the connections reported by the method. Recall then measures the proportion of correctly identified connections by the method among all actual connections. These metrics can be represented by the following formulas:

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

Summarised results per collection method and environment are presented in Table 5.6. In general, the devices with high difference between advertising interval and connection length perform better, as expected, but the used methods are highly susceptible to fluctuations. In particular, the sliding window reports more alerts if the data is more regular, because then even a minor dropout will cause the outlier threshold to be exceeded.

Table 5.6: Detection methods comparison

	SimpleStatistics		*SlidingWindow*	
Environment	**Precision**	**Recall**	**Precision**	**Recall**
RPi BLE controller				
Shielded room	24,68%	50,37%	37,46%	65,37%
Office	24,42%	59,83%	26,12%	75,00%
Monitoring probe				
Shielded room	44,04%	72,89%	23,41%	87,22%
Office	32,90%	75,96%	23,69%	84,22%

The results showed that the recall improved in the data measured by our probe. The *SimpleStatistics* method was able to correctly detect 19% more connections then when applied to the data collected by the RPi BLE controller, while the *SlidingWindow* improved by 16%. On the other hand the precision is compromised by a high number of false positives which we attribute to the simplicity of the used methods. The *SimpleStatistics* method exhibited in precision a general improvement of 14%, while the *SlidingWindow* precision decreased by 8%, which could be caused by an incorrectly chosen window size.

Because these methods do not achieve acceptable precision even when using a specially designed collector, more appropriate methods need to be devised for reliable connection identification.

5.7 Investigation of advanced detection methods

After dismissing the idea of improving detection quality through better data collection, we redirected our efforts toward developing more advanced methods for detecting connections. Given that the monitoring principle is based on characterising the advertising pattern and identifying connections as instances with significantly longer delays between advertisements, we frame it as an outlier detection problem. For this purpose, we consider connections to be outliers in the device's normal advertising behaviour. Consequently, we believe outlier detection methods will deliver the required performance, as outlined in our research question *RQ3*. Additionally, we acknowledged the suitability of Artificial Intelligence for pattern recognition and included a simple AI method for comparison in research question *RQ4*.

To verify the applicability of potential detection methods, we performed a series of measurements to assess the effectiveness of selected techniques using our BLE-ARD dataset. Specifically, we evaluated five outlier detection methods based on different core principles to ascertain their suitability for connection detection. Furthermore, we deployed a simple neural network to explore the potential of AI in this context. After extensive testing with various configurations, we identified the best-performing parameters for each method. This approach is designed to reliably identify the most suitable techniques for detecting Bluetooth connections based on device advertising behaviour.

5.7.1 Outlier detection methods

Outlier detection identifies data points or observations that deviate significantly from the majority of the data. Initially, it was used for data cleansing to remove outliers and enable smoother fitting of parametric statistical models. However, outliers often represent crucial information, such as cyber-attacks or mechanical faults, which lead to extensive research in high-performance outlier detection techniques for various real-life applications.

Outlier detection methods can be categorised based on the availability of input data labels into supervised and unsupervised methods. Supervised outlier detection uses labelled training data to build predictive models and is often viewed as a binary classification problem with imbalanced data. Unsupervised outlier detection uses unlabelled data to calculate outlier scores either by building models or directly from the data.

Boukerche et al. [25] present a comprehensive overview of the current state of outlier detection methods and provide a taxonomy. Samara et al. [102] then focused explicitly on outlier detection in IoT and classified the techniques into seven categories based on their fundamental principle, as shown in Figure 5.15. In Bluetooth, Roth [99] deals with the problem of removing the outliers from Bluetooth vehicle speed data by using a time series analysis method - ARIMA. Moghaddam et al. [83] explored indoor device localisation and designed a two-step outlier method to improve accuracy. Moreover, by separating the data by advertising channel, they improved the accuracy of detecting the outliers by 18.42%.

We have selected five representative outlier detection methods that cover various categories of detection techniques. We are testing GMM and ARIMA from statistical methods, LOF from nearest-neighbor techniques, and OCSVM from classification methods. Additionally, we are testing the iForest method, which Boukerche categorises as a projection method. Since we are using an AI-based method for classification rather than outlier detection, it will be described in the Section 5.7.2. This selection provides us with an understanding of the suitability of each basic technique for the problem of detecting BLE connections.

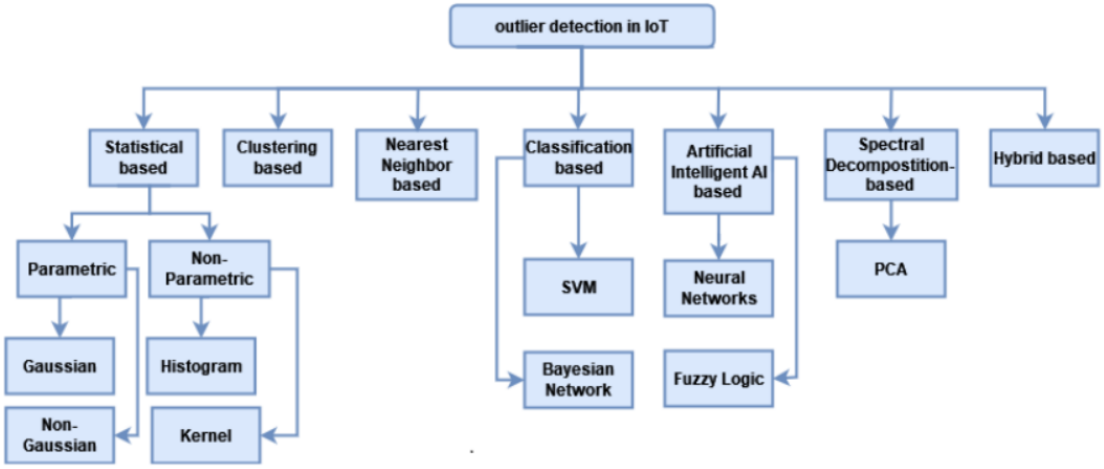

Figure 5.15: Outlier detection techniques for IoTs. [102]

Autoregressive integrated moving average

ARIMA is a well-known time series statistical forecasting model [106]. The model consists of three key components: autoregression (AR), differencing (I), and moving average (MA). The autoregression component uses past values, or lags, to predict the current value in a time series. The moving average component adds variability by incorporating a linear combination of past error terms, also known as residuals. Since ARMA models can only effectively describe stationary series, which have no trend, the differencing component is applied to transform a non-stationary series into a stationary one, ensuring that the model can be appropriately used.

Models are represented as *ARIMA(p, d, q)*, where p is the autoregressive part, d is differencing, and q is the moving average, it can be characterised by the following equation:

$$y'_t = \phi_1 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \epsilon_t + \theta_1 \epsilon_{t-1} + \cdots + \theta_q \epsilon_{t-q},$$

- y'_t represents the d-times differenced series,
- $\phi_1, \phi_2, \ldots, \phi_p$ are AR coefficients,
- $\theta_1, \theta_2, \ldots, \theta_q$ are the MA coefficients,
- ϵ_t is the white noise,
- $\epsilon_{t-1}, \epsilon_{t-2}, \ldots, \epsilon_{t-q}$ are the error terms.

To use ARIMA forecasting for outlier detection, Yongle [76] proposed (for ARMA) outlier detection based on computing residuals between real data and forecasted data and using Chauvenet's criterion to determine outliers. In our implementation we decided to use a criterion based on squared errors e. We compute the residuals as the difference between the forecasted values and the actual data. We identify an outlier if the squared residual exceeds the threshold defined as the mean squared error (MSE) plus standard deviation σ multiplied by a parameter s of the squared residuals. This threshold is computed in the model fitting phase and then used for evaluation.

$$Threshold = \frac{1}{n}\sum_{t=1}^{n} e_t^2 + s \cdot \sigma$$

Gaussian mixture model

A Gaussian Mixture Model (GMM) [96] is a type of finite mixture model that assumes the data can be represented by multiple Gaussian distributions, known as components. Each component is defined by its weight w (indicating the proportion of the data it represents), mean (μ), and covariance matrix Σ. The model is then composed of a weighted sum of M component Gaussian densities as given by the equation

$$p(x|\{\omega, \mu, \Sigma\}) = \sum_{c=1}^{M} \omega_c \; g(x|\mu_c, \Sigma_c)$$

where x is the measurement of features and $g(x|\mu_i, \Sigma_i)$ are the component Gaussian densities. This model aligns with the distribution of the advertising delta, as the components can effectively capture the variations in the advertising intervals and omissions.

Blanco et al. [22] utilise the GMM for anomaly detection by calculating the probability of occurrence for the tested data for each component. They then apply a voting scheme across different features to identify anomalies.

In our work, we focus on a single feature, namely advertising delta. Because of this the covariance matrix Σ is simplified to the variance σ^2. To detect outliers, we determine whether the given advertising delta lies outside of the distribution of any component c with a margin given by a parameter s:

$$x \notin [\mu_c - s \cdot \sigma_c, \mu_c + s \cdot \sigma_c]$$

The parameter s allows us to modify the bounds of the target interval, which affects the degree of confidence that the tested point belongs to the distribution. Commonly used in statistics is the constant $s = 3$, also known as the *three-sigma rule*.

Local outlier factor

The core idea of *Local Outlier Factor* (LOF) method is to detect how isolated a data point is with respect to its neighbours. For each point p evaluated, an outlier factor is calculated, which determines the degree of abnormality [79]. To calculate the outlier factor, the local reachability densities lrd of the given point and its neighbours are computed as the inverse of the average reachability distance based on the $MinPts$ nearest neighbours of p:

$$lrd_{MinPts}(p) = 1/\left(\frac{\sum_{o \in N_{MinPts}(p)} dist_{MinPts}(p, o)}{|N_{MinPts}(p)|}\right)$$

where N denotes the $MinPts$-distance neighbourhood and the reachability distance $dist$ is based on the distance (euclidean, minkowski, etc.) between the points p and o.

The outlier factor of object p is the average of the ratio of the local reachability density of p and those of p's $MinPts$-nearest neighbours:

$$LOF_{MinPts}(p) = \frac{\sum_{o \in N_{MinPts}(p)} \frac{lrd_{MinPts}(o)}{lrd_{MinPts}(p)}}{|N_{MinPts}(p)|}$$

A high outlier factor suggest lower local reachability of the given point p when compared to its neighbours. This implies that the given object is isolated which represents an outlier.

Isolation forest

Isolation Forest (iForest) builds an ensemble of binary search trees (iTrees) for a given data set, where the trees isolate (meaning separate one instance from the others) data points based on a selected feature. The main idea is built on the assumption that anomalies are a minority and have very different features compared to normal data [74].

To construct the iForest, the required number of trees are created. Every tree is constructed from a randomly selected partition of data (sub-sampling) by recursively splitting this partition. The data are split by randomly selecting a feature f and a pivot value between f_{min} and f_{max}. Any given object p is fully isolated once there is a sufficient amount of splits to separate it from all other objects.

The anomaly score is calculated based on the path length, which is defined as the number of edges from the root node to the point where the traversal ends at an external node. Due to randomisation, it is expected that any anomaly in the data will require fewer data splits to completely separate it from the rest of the objects. Objects with higher densities, based on their features, need more splits, resulting in longer average decision paths for the tree.

Since individual trees are generated with different sets of partition the path lengths are averaged over the iForest for better generalisation of the model.

Support vector machines

The One-Class Support Vector Machines differs from the previously described methods because it is used for novelty detection rather than anomaly detection [103]. The method is based on a hypothesis that if the data are separable, there exists a hyperplane separating the data. Such hyperplane represents the frontiers of initial observations and if the given point is a novelty, the probability that it lies outside of the estimated region described by the hyperplane is larger.

The training data represent the regions that define the borders of initial observations in the n-dimensional space. Any object p that lies outside of this region can be determined to be a novelty. One-class SVM is similar to conventional SVM approach as it utilises a predefined *kernel* function, which maps the original data points into suitable n-dimensional space based on the similarity property defined by the function.

5.7.2 Artificial Intelligence approach

Artificial Intelligence (AI) is a broad term encompassing a wide range of approaches that mimic human behaviour. AI can be leveraged for outlier detection methods discussed in the previous section, where this term mainly refers to the use of deep learning based on Artificial Neural Networks (ANN) or fuzzy logic [102]. For outlier detection, ANNs such as autoencoders or LSTM are used to model normal behaviour and by comparing the errors between predicted and real values (reconstruction error in case of autoencoders) the system can identify an outlier. This principle is fundamentally similar to the approach we used for ARIMA, as described in the previous section.

Due to this similarity, we opted not to employ AI for outlier detection and thus separated it in this section. We decided to narrow down the term AI to methods employing ANNs and use them for the problem of classification, particularly to distinguish between Advertising

events and Connection events. In the rest of this section, we explain basic principles of ANNs and the selected network type.

Artificial neural networks

Artificial Neural Networks (ANN) are parallel computational models consisting of densely interconnected nodes usually called neurons or perceptrons. Each neuron performs a simple mapping function consisting of a linear transformation and an activation function. The linear transformation accepts n inputs $x_1, \ldots, x_n$ and to each input x_i a weight w_i is assigned. The output y is then calculated as [70]:

$$y = \sum_{i=1}^{n} w_i x_i$$

The result of this transformation y is then translated to the output of the neuron by an activation function. The examples of the activation function include *Unit Step*, where the input y is transformed to a binary values $1, 0$ based on a threshold, or some more sophisticated functions such as Tanh or ReLU. By adding a nonlinear activation function, the ANN will be able to perform an arbitrarily complex function that maps inputs to outputs [105].

In order to increase the computational strength and ability to learn patterns, the neurons are organised into ANNs. Such network is a directed graph, where vertices are neurons and edges are connections [70]. Typically, the neurons in ANNs are organised into layers and outputs of one layer are used as inputs for the next layer. We can distinguish two fundamental types of neural networks. If the network graph does not contain any loops, the network is called a *feed-forward* network. In such networks data flows through the layers only in one direction. On the other hand, *recurrent* networks contain loops in the data processing flow and outputs of one layer can be coupled back to inputs of preceding layers [70].

Every ANN needs to be trained to perform any meaningful tasks. Initially, the weights w_i are set to a random value. During training, these internal parameters are iteratively optimised through back-propagation to minimise the difference between the predicted output and the actual target using an optimisation algorithm such as gradient descent or its variants, like Adaptive Moment Estimation (Adam).

Multi-layer perceptron

Multi-Layer Perceptron (MLP) is the most widely used form of ANNs [70], which exhibits a strictly layered structure, where connections can exist only between the neurons of consecutive layers. Moreover, each layer in MLP is fully connected to the next one. Every MLP exhibits an input layer, one or more hidden layers, and an output layer. The neurons in each layer typically use the same activation function.

We have selected MLP as a candidate for a detection method because of its relative simplicity and efficiency, as it is a feed-forward network. We focus on simplistic architectures to achieve low computational requirements so the processing can be performed at the IoT devices locally following the fog principle. We believe even simple MLP networks shall be able to learn the distribution of advertising deltas and recognise connections reliably.

5.7.3 Methodology

Since the proposed detection approach works only for devices with an intermittent advertising pattern, we evaluate the methods on those specific devices. We used the BLE-ARD dataset containing 9 devices, with 5 devices following the intermittent pattern[10]. In the dataset, we split each sample captured by the parallel monitoring probe by channel, because the advertising interval is defined in the standard [7] as the time between two advertising events (which include transmissions on all channels). This effectively triples the available samples. We then compute advertising deltas as the time between two consecutive advertisements and evaluate the methods on time series of these advertising deltas. To assess the results and allow for supervised learning, we manually labelled the data to identify the exact advertising delta that represent the connection.

For all methods, we determined the hyperparameter grid, then applied them to the data of each device, and finally evaluated their ability to identify the connections of these devices. Based on the results, we selected the universally best parameters for each method. In this configuration, we then compared the methods with each other, a which allowed us to select the most efficient method.

To assess the performance of selected methods for BLE connection detection, we calculate their F1 scores, considering each environment and device separately. The F1 score is chosen as the evaluation metric because it provides a balanced measure of a method's performance by combining precision and recall into a single value:

$$F1 = 2\frac{Precision * Recall}{Precision + Recall}$$

Method parameters

We utilise a grid search approach to identify the best parameters, defining the hyperparameter grid according to the dataset's characteristics and using the F1 score as the performance metric. The best parameters are determined from this grid, though they are not guaranteed to be universally optimal. We evaluate the performance metric across all samples to establish a device-independent method setting. This section outlines the hyperparameter grid definition for each method.

The threshold setting parameters are general for several methods, so we present them here. For the statistical based methods (ARIMA, GMM) the threshold is set by the *sigma* parameter, which is a multiplicator of the standard deviation σ. Smaller *sigma* values create stricter criteria and flag more data points as outliers, while larger *sigma* values create more lenient criteria. We test $sigma \in \{2, 3, 4\}$, as the parameter is directly tied to data percentiles and the range covers approximately 95th to 99.995th percentile. This range allows testing from common to broader thresholds to balance outlier detection and accuracy.

For the LOF and iForest methods, the contamination parameter assumes the same role. It defines the ratio of outliers in the sample, and the threshold value of the method metric is set accordingly during the fitting phase. In our dataset, there is one connection per minute, corresponding to a contamination rate of roughly 0.01. To ensure robustness, we test contamination rates an order of magnitude lower and higher than this baseline.

1. *Autoregressive integrated moving average*

[10] We consider Nuki to follow a persistent pattern based on the observations in Section 5.6.6

ARIMA model is defined by three parameters - p, d and q. Traditional approach is to find the parameter d by checking the stationarity and perform differencing until stationarity is achieved. As we have already differenciated the advertisement timestamps to deltas in dataset processing, we performed Augumented Dickey-Fuller test to check whether the series is stationary and confirmed stationarity for all single-channel samples. For this reason, we use $d = 0$, thus reducing our ARIMA model to ARMA.

In order to find optimal parameters p and q we calculated autocorrelation (ACF) and partial autocorrelation (PACF) functions for every device in the dataset and confirmed that the autocorrelation in the behaviour is negligible as there is no real pattern. This would suggest using $ARIMA(0, 0, 0)$ which would always predict the mean value of the training data. To verify this conclusion and generalise the method, we decided to test all combinations of $p, q \in \{0, 1, 2, 3\}$ where the upper value was determined to always contain at least one previous value from the same channel if run on the data with all channels included.

2. *Gaussian mixture model*

   For GMM, determining the correct number of components is crucial. The number of components should ideally correspond to the maximum multiple of missed advertisements so that each component identifies a multiple loss. To estimate the appropriate number, we test 2 components at the beginning, since it is common to miss one ad in our dataset. We then extend the testing up to 4 components, which accounts for up to 3 consecutive missed advertising packets.

3. *Local outlier factor*

   The local density deviation in LOF is calculated against the number of neighbouring data points specified by the neighbours parameter. It is crucial that the neighbours accurately represent the data, as poor representation can lead to inaccurate results. To evaluate the method, we began with a minimal value of 2 neighbours and employed an exponential scale with steps of power of 2. We tested values up to 128 neighbours, considering the mean number of data points in a sample was 158.

4. *Isolation forest*

   For the iForest method, the key parameter is the number of trees. For general datasets, using 100 trees usually provides an optimal balance between detection accuracy and computational efficiency. Since our data consists of only one feature, the trees differ only by their subsets. Given the low data diversity, with several clusters of condensed data, we test the number of trees ranging from 5 to 200, specifically at values of 5, 10, 25, 50, 75, 100, and 200.

5. *Support vector machines*

   Hyperparameters of Support Vector Machines include type of the kernel function and parameters for the respective functions. Each type of kernel function responsible for transforming the data as an initial step for SVM can be defined by different parameters. The tested kernel functions are defined in Table 5.7.

   While the linear kernel is suitable for linearly separable datasets, other kernels provide a support for non-linearly separable data. If the kernel is defined as polynomial functions, it can be further parameterised by the degree d of such function. Radial

Table 5.7: Kernel Functions for SVM

Kernel function	Definition
Linear	$\langle x, x' \rangle$
Radial Basis Function	$\exp(-\gamma \|\|x - x'\|\|^2)$
Polynomial Kernel	$(\langle \gamma \langle x, x' \rangle + r)^d$
Sigmoid Kernel	$\tanh(\gamma \langle x, x' \rangle + r)$

Basis Function and Sigmoid Kernel can be parameterised by a constant value r that determines a bias or constant shift of the decision boundary.

Kernel parameter γ (gamma) is important for the overall model performance as it influences on which data points is the model focused when estimating the decision boundary. The higher the value, the more influential is even a small distance between individual data points.

Parameter ν (nu) for the SVM model represents an upper bound of the fraction of outliers and lower bound of fraction of support vectors. Its role is similar to contamination parameter of other methods as higher values allow more outliers at the output and support vectors are defined by less data points.

6. *Multi-layer perceptron*

A Multi-layer perceptron (MLP) is a flexible model characterised by its architecture, defined by the number of hidden layers, the number of neurons in each layer, and the activation function used for each neuron. More hidden layers enable the network to learn complex patterns, but require more data for effective training. Given the simplicity of our data, we focus on straightforward architectures.

Table 5.8: Neuron activation functions

Activation function	Definition
Logistic	$f(x) = \frac{1}{1+\exp(-x)}$
Tanh	$f(x) = \tanh(x)$
ReLU	$f(x) = max(0, x)$

We tested three common non-linear activation functions: ReLU, Tanh, and Logistic, as defined in Table 5.8. For optimisation, we used the Adam method, influenced by the learning rate α, which affects convergence and the degree of weight adjustment during training. Table 5.9 presents our hyperparameter grid.

Table 5.9: Tested MLP parameters

Hyperparameter	Values
Hidden Layers	[5], [15], [100], [2,2], [2,2,2] [80,70,20]
Activation Function	ReLU, Tanh, Logistic
Learning rate	Constant (0.001), Adaptive

Fitting the model

We leverage the fact that the dataset contains five passive samples for each action. We use a 4:1 (or 80% train, 20% validate) split approach to divide the dataset into the training and validation data.

Because ARIMA, GMM, LOF, and OCSVM methods are fitted to the normal behaviour, we use only the passive samples for fitting. We still split the five passive samples to four training samples and reserve one for validation. This approach maximises the training set while retaining one control sample to evaluate model performance on previously unobserved passive device behaviour. We then include all the non-passive samples in the validation data.

On the other hand iForest and MLP methods require a representative sample of both the passive data and sample including a connection. For this reason we split every action into four training samples and one validation sample. This approach allows for a larger training set, required for MLP, and represents all the actions equally.

When using the split approach, a key challenge is selecting the validation data, which can introduce bias and affect the robustness of the results. To address this, we employ cross-validation, fitting the model separately for each split. This approach results in multiple fitted models, but it ensures that the results are independent of the specific fitting data, providing a more reliable and unbiased assessment of the model's performance.

Evaluation

After fitting the models, we have multiple fitted models available to work with. We evaluate all of these models separately for each environment and device by applying the fitted models to all the validation samples in that environment. Since we split the original samples by channel, we fit and compute metrics for each channel separately. To obtain a thorough and reliable results for each method, we average all the results obtained for a device within an environment.

We evaluate the models using precision, recall, and F1-score metrics. Precision measures the proportion of true positive detections among all positive detections, indicating the accuracy of positive predictions. Recall measures the proportion of true positive detections among all actual positives, reflecting the model's ability to identify all relevant instances. The F1-score is the harmonic mean of precision and recall, providing a balanced evaluation of the model's performance.

To perform a cross-method comparison, we compare only the metrics achieved using the best parameters we identified for each method. This ensures that our evaluation reflects the generalised best performance of each detection method.

5.7.4 Results

To identify the best parameters, a grid search was conducted over the hyperparameter grids defined in the previous section. The parameters were determined by averaging the F1-scores across all evaluated devices and both environments for each method. This approach ensures that the identified parameters, as presented in Table 5.10, are generalised. However, it should be noted that these parameters may not represent the best tuning for each individual device or specific environment. In Appendix B you can find the details of the measured scores per method.

Table 5.10: Best identified method parameters

Method	parameters
ARIMA	$p = 0, d = 0, q = 1$; $sigma = 4$
GMM	$components = 4$; $sigma = 4$
LOF	$neighbors = 14$; $contamination = 0.001$
OCSVM	$kernel = rbf, nu = 0.1$
iForest	$estimators = 25$; $contamination = 0.01$
MLP Classifier	$neurons = [80, 70, 20]$, $activation = relu, learning = constant$

When testing ARIMA parameters, we discovered that the p, d, and q parameters only marginally affect the results. The differences in outcomes are mostly influenced by the sigma multiplier. Similarly, for GMM, the best results were achieved with the highest sigma multiplier, but in this case the number of components also affected the outcome significantly.

LOF exhibited instability between environments. The best configuration in the shielded room was 47% better than the best configuration in the office, while the second-best configuration in the office performed subpar in the shielded room. Despite our high expectations for OCSVM, it displayed poor performance for the use case overall.

The iForest method generally achieved similar results to statistical methods. With the precision of 68%, it did not generate an excessive number of false positives. By far the best results were achieved by the MLP, which in some cases managed to correctly identify all connections. Moreover, while the best results were produced by the complex network, the simpler network with a single hidden layer with five neurons achieved very similar results (on average worse by 1.5%), making this method applicable even for low-performance devices.

Tables 5.11–5.15 describe the method characteristics for the evaluated devices. Each table presents precision, recall, and F1-score for both environments, allowing for comparison. Since the advertising profiles of devices differ (e.g., advertising interval, connection length), the performance of the methods is affected. These data provide insight into the applicability of each method for device monitoring.

When summarised for all devices, among all tested methods, the MLP Classifier performed the best by far, achieving a universally high F1-score of 90.9%.

Table 5.11: Bentech FP3 lock results

	Shielded room			Office		
Method	Precision	Recall	F1-score	Precision	Recall	F1-score
ARIMA	79.9%	100%	88.8%	54.1%	100%	69.8%
GMM	59.6%	100%	74.5%	74.5%	100%	84.8%
LOF	93.7%	100%	96.8%	68.9%	77.8%	73%
OCSVM	16.9%	100%	28.8%	18.5%	100%	31.2%
iForest	100%	73.3%	84%	94.5%	66.7%	77.4%
MLP	**100%**	**100%**	**100%**	**96.7%**	**100%**	**98.3%**

Table 5.12: Danalock V3 results

	Shielded room			Office		
Method	Precision	Recall	F1-score	Precision	Recall	F1-score
ARIMA	47.9%	100%	64.7%	46.5%	100%	63%
GMM	48.1%	100%	63.7%	50.4%	100%	66.7%
LOF	93.3%	97%	95%	0%	0%	0%
OCSVM	2.1%	100%	4.1%	7.1%	100%	13.3%
iForest	89%	100%	94%	80%	100%	88.9%
MLP	**97%**	**100%**	**98.4%**	**100%**	**100%**	**100%**

Table 5.13: Revogi Bluetooth LED bulb results

	Shielded room			Office		
Method	Precision	Recall	F1-score	Precision	Recall	F1-score
ARIMA	11.2%	100%	20.1%	54.6%	100%	70.3%
GMM	89.7%	100%	94.5%	68.3%	100%	80%
LOF	77.1%	70%	73%	33%	3%	6%
OCSVM	0.4%	100%	0.8%	0.5%	100%	1.1%
iForest	8.5%	100%	15.7%	9.4%	100%	17.2%
MLP	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**

5.7.5 Evaluation

Based on the results in the previous chapter, we can deduce that advertising patterns do not resemble typical time series, as random advertisement omissions disrupt traditional time series analysis approaches. Additionally, outlier detection algorithms (LOF, OCSVM, iForest) generally struggle with the non-uniformity of normal data, which clusters around advertising interval multiples due to omitted advertisements. However, the iForest algorithm stands out as an exception, demonstrating the ability to identify connections that are significantly different from the rest of the data.

Since there is no real pattern to be learned, ARIMA offers no added value for our case. Regardless of the ARIMA parameters, it will always predict the mean of the training data with some added jitter. Outlier detection based on residuals is thus reduced to the detection function applied to these residuals, and ARIMA can be substituted by a simple mean. This confirms the parameter determination from Section 5.7.3.

The results of the GMM method seem to improve with the number of Gaussian distributions and the sigma multiplier. We believe this is due to overfitting on the available

Table 5.14: Mi Temperature and Humidity Monitor 2 results

	Shielded room			Office		
Method	Precision	Recall	F1-score	Precision	Recall	F1-score
ARIMA	45%	100%	61.5%	61.6%	100%	75%
GMM	45.7%	100%	62.2%	65.6%	100%	79%
LOF	85%	100%	91.6%	61%	83%	70.2%
OCSVM	24%	100%	38.6%	25.9%	100%	41.1%
iForest	82.2%	46.7%	59.4%	88.9%	45.8%	58%
MLP	**93.4%**	**100%**	**96.3%**	**100%**	**100%**	**100%**

Table 5.15: BeeWi Motion Sensor results

	Shielded room			Office		
Method	Precision	Recall	F1-score	Precision	Recall	F1-score
ARIMA	51%	52%	44.3%	53%	33%	39.9%
GMM	60.5%	62%	60.8%	3.3%	99.6%	6.2%
LOF	33%	28.8%	30.5%	30.3%	11%	16.2%
OCSVM	2.5%	98.7%	4.9%	3%	100%	5.9%
iForest	66.7%	57.9%	60.8%	54.2%	33.4%	41.2%
MLP	**100%**	**51.5%**	**68%**	**90.5%**	**33.4%**	**48.5%**

data, given the rather small training set and data dispersion around the advertising interval multiples. Since the number of multi-omissions is random, the method might be improved by iterative supervised fitting.

The statistics-based methods generally yield good results, but to achieve truly usable outcomes, it seems necessary to develop a custom-made statistical method tailored to the specifics of Bluetooth advertising.

The only suitable method for outlier detection is iForest, which achieves similar results to statistics-based methods. The binary trees in iForest can help distinguish between an advertisement omission and a connection, but the construction of these trees would need to be fine-tuned for better results.

In general, to answer question *RQ3*, it cannot be said that the use of outlier detection methods significantly improves the identification of connections.

On the other hand, answering the research question *RQ4*, using AI did improve the achieved results by a margin. The MLP neural network achieved exceptionally good results, even with a small number of neurons and thus low computational demand. Initially, it seemed to be an overly complex approach for this problem, but experimental results showed that the data specificity causes issues for more straightforward approaches. This indicates that future developments in BLE connection detection should focus on simplistic neural networks.

5.8 Final reflections on BLE security monitoring

Our research focused on enabling security monitoring of Bluetooth devices in SOHO and shared economy environments. To achieve this, we introduced a novel principle that monitors advertisement channels to determine whether a device is in use. We conducted a thorough analysis of the behaviour of multiple devices, resulting in the creation of two datasets: one for simple verification and a larger one intended to support further development in this field. To optimise data collection, we designed a custom monitoring probe capable of simultaneously observing all three advertising channels, providing a comprehensive overview of all activity on them. Ultimately, we identified a detection method based on an MLP Classifier, which reliably identifies connections with an F1-score of 90%. By integrating the advertisement collector and the detector based on this method into the framework described in Chapter 4, we empower SOHO environments to independently monitor the connections of their Bluetooth devices.

To address the security monitoring problem, we formulated four research questions and systematically addressed each one. Our findings revealed that the proposed monitoring principle can indeed be effectively applied to the selected Bluetooth LE devices. Addition-

ally, we discovered that the standard Bluetooth controller yields high-quality data and that optimising hardware does not significantly enhance the collected data and thus detection efficiency. Despite our initial expectations, our hypothesis that outlier detection methods would deliver the desired level of precision and reliability was not supported by our research. However, we successfully tested the artificial intelligence method based on MLP, identifying it as a suitable detection method for our detector.

Our solution effectively identifies when a device has been in a connected state, but it does not offer additional details, such as the origin of the connection. We anticipate integrating our solution with other systems, such as correlating it with access logs or utilising behavioural analysis to provide intrusion detection. The primary limitation of our solution is inherent in its design, as it can only monitor devices that follow an intermittent advertising pattern.

To conclude this chapter, we have successfully found a solution to the Bluetooth monitoring problem we identified. We designed a solution, which can reliably identify the connections of any Bluetooth LE device following the intermittent pattern, which can be used for security monitoring. By plugging our solution into Security Information and Event Management (SIEM) systems, it can provide insights into operations with the device, which can be then cross-checked against the access logs provided by the vendor. Alternatively, it can be used as a data source for intrusion detection systems based on behavioural analysis, or included in simple home automation as a warning system that a device has been used (for example, outside defined hours).

Chapter 6

Designing security features for a proprietary network

This chapter is dedicated to designing security features for the proprietary IQRF wireless network. The IQRF network was created by the Czech company MICRORISC in 2004 for ultra-low-power and low-speed mesh communication between IoT devices and is now backed by nearly 40 worldwide patents [60]. The network is mainly used in industrial applications, including smart cities, buildings, and Industry 4.0. Its use has gradually expanded beyond Europe, with applications and distributors reaching from Argentina to Russia. In 2017, the IQRF Alliance [58] was formed to unite commercial and non-commercial members to develop the IQRF ecosystem. The alliance promotes knowledge exchange, provides support, and defines the IQRF interoperability standard.

Along with the formation of the alliance, a MICRORISC spin-off company called IQRF Tech was established to maintain the IQRF network. Until 2023, connectivity was exclusively provided through hardware IQRF transceiver (TR) modules with the network stack implemented as IQRF OS. To address the network's expanding use and the alliance's requirements, an „IQRF communication standard for wireless mesh networks“ is being developed to clearly define the protocols, rules, and procedures related to IQRF communication. The first preliminary documentation of this standard was released in August 2023, with version 0.96 publicly available [127] at the time of writing.

The author of this thesis played a central role in the development of the security mechanisms for this communication standard, the details of which are contained in this chapter. It's worth noting that the standard is still in development, and the features described here already surpass the available version 0.96, even though they have not yet been published. Since the standard is not yet finalised and the reference implementation is still in progress, a minimal implementation has been created to verify security operations and create testing vectors outlined in the standard. Section 6.3 briefly mentions this implementation, but since it is an internal tool, it has not been published. The security design was informed by a comprehensive investigation of existing IoT protocols described in Chapter 2 and was guided by the insightful security analysis presented in Chapter 3.

We begin by describing the specifics of the IQRF network as it existed prior to the establishment of the standard, which we will refer to as „IQRF Legacy“ to distinguish it from the updated version. Following this, we characterise the standard and provide a narrative explanation of each component of the security architecture.

6.1 IQRF Legacy

IQRF is a wireless platform designed for static systems with a stable topology. The heart of the platform is the transceiver module (TR), a system-on-chip (SoC) that contains the entire IQRF stack as illustrated in Figure 6.1. It consists of a embedded radio module providing the physical layer and the IQRF Operating System (IQRF OS) which contains the network implementation [62]. Above the OS layer, there is an optional layer called *Direct Peripheral Access (DPA)*. DPA is a byte-oriented application layer protocol that provides abstraction for peripherals and OS services [81]. In IQRF Legacy, both the OS and DPA layers were distributed as binary libraries for the transceiver modules. While guides for their usage were provided [62, 81], no official standard was released.

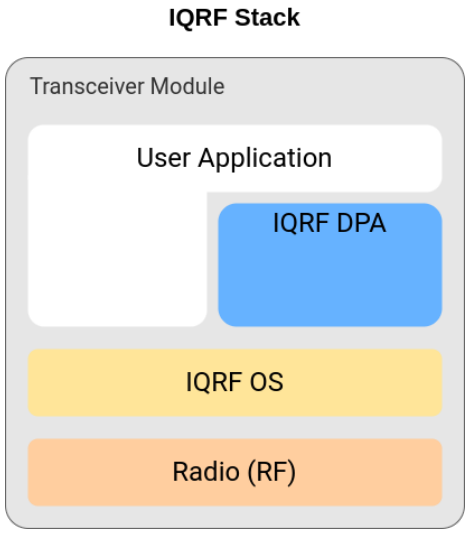

Figure 6.1: IQRF protocol stack

The IQRF OS guide [62] recognises two approaches for utilisation of the TR:

- *Non-networking* approach uses only two layers - IQRF OS and user application. The user application can access the radio module directly and is responsible for calling appropriate IQRF OS functions to ensure the required functionality.
- *Networking* approach utilises the *DPA* middleware and the user application is realised as a DPA Handler. The reason the mode is named „networking“ is that through DPA is provided access to the proprietary routing protocol *IQMESH*.

IQRF technology offers a data rate of 20 kb/s and range over 10 kilometers, which classifies it as a LoWPAN IoT network with long range, or NAN according to smart-grid terminology. IQRF operates within the ISM bands of 868 MHz, 916 MHz, and 433 MHz, which are further divided into multiple channels, each with a channel width of 100 kHz. Within each band, three specific channels are designated as service channels, which are used for network joining procedure (referred to as bonding) [62]. After bonding, the network communication is conducted on a separate operating channel, distinct from the service channels and uniformly used across the entire network. IQRF employs Time-Division Multiple Access (TDMA) for channel access, where each device is allocated a dedicated time slot for transmission. This method enhances the predictability and reliability of transmissions,

addressing the congestion and data request blockage issues associated with CSMA/CA used in IEEE 802.15.4 networks, as identified by Misic [82] and Francesco [39].

IQRF Legacy supports two connection topologies: a point-to-point link provided directly by IQRF OS and an IQMESH mesh network enabled by the DPA plugin. Each IQMESH network, illustrated in Figure 6.2, is managed by a single coordinator and can accommodate up to 239 nodes [61]. Additionally, a special mode is available through a provided DPA handler, designed for nodes with stringent battery-saving requirements, known as offline mode or beaming. In this mode, the device remains in a sleep state, waking up only periodically or upon an event to asynchronously and unidirectionally transmit data. Nodes with aggregation functions within range store this data and relay it to the network upon request [61].

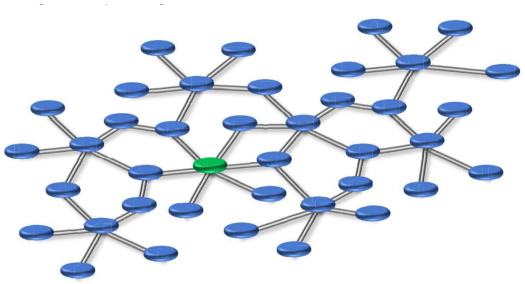

Figure 6.2: IQMESH Topology (coordinator is highlighted in green) [59]

In IQRF Legacy networks, each node can be assigned two 1-byte addresses. The first address is the logical address, which is assigned to the device during the bonding process and is used to identify the device within the network uniquely. The coordinator always has logical address 0 [62]. The second address is known as the Virtual Routing Number (VRN), which is used explicitly for routing within IQMESH networks. The VRN is assigned during the discovery process when the network's routing topology is established. This address is crucial as it determines the TDMA time slot in which the device is permitted to transmit data, ensuring organised and efficient communication within the network.

In IQRF IQMESH networks, communication predominantly operates synchronously, where the Coordinator initiates communication by sending a request to a Node, which then responds. A unique variation of this synchronous communication is the Fast Response Command (FRC), where the Coordinator gathers data from multiple Nodes simultaneously, processing their responses in a map-reduce fashion [61]. Additionally, IQRF supports several types of asynchronous communication. One of these is non-network communication, where a device that is not part of an IQMESH network broadcasts a message. Another form is custom communication, managed by a custom DPA Handler, allowing for specialised functions such as the beaming (offline mode) mentioned.

The IQRF security architecture is based on packet integrity verification using CRC-16 (Cyclic Redundancy Check) and multi-layer encryption using the industry-standard AES-128 symmetric block cypher [62] summarised in Table 6.1. Since CRC does not provide authentication, an additional consistency check routine is performed after the encryption/decryption of network packets. Encryption occurs in two layers, with the base layer being networking encryption, which is mandatory for all network packets. The second layer is optional user encryption, which provides end-to-end security.

Table 6.1: Overview of encryption in IQRF Legacy

	Encryption mode	**Encryption keys (derived from)**
Networking encryption	AES-CDC	Network key (Network password)
User encryption	AES-ECB	User key
Access encryption	AES-ECB	Access key (Access password) Individual Bonding Key One-Time Key

The networking encryption is automatically handled by the IQRF OS and uses a proprietary CDC (Cypher Data Chaining) algorithm [62]. The 128-bit key for this encryption is derived from a network password, a 192-bit vendor-supplied random number present in every TR. As this key needs to be shared by all nodes in the network, the coordinator's network password is used and transferred to the node during bonding (a network join procedure detailed later).

The optional user encryption can be handled either by the IQRF OS or externally. If the integrated encryption is chosen, the data is encrypted using AES-128 in ECB mode [62].

To be included in the IQMESH network, a node has to undergo a joining process known as bonding. This process occurs on the service channels and is responsible for supplying the node with crucial network details, including the network's operating channel, Network Identification Number (NID), network password, and the assignment of a logical address [61]. The data exchange during this procedure is secured by Access encryption, a specialised maintenance encryption that uses ECB mode. There are four distinct methods available to carry out the bonding procedure:

- **Local Bonding** method is used to connect devices within the direct range of the coordinator. This method uses an Access Password for security that must be configured on both the coordinator and the node. If a node requests to connect with the correct password, it receives the necessary information and is integrated into the network.
- **Smart Connect** enhances security by utilising an Individual Bonding Key (IBK) for encryption, which is a unique, fixed code supplied by the vendor. Before the bonding process begins, the coordinator must receive this IBK along with the module ID (MID) for identification purposes. The coordinator then initiates the process, and if the device is within the network's range, it connects to the network.
- **Autonetwork** allows multiple devices to join the network at once. All the devices have to be configured with the same Access Password, and the procedure is split into several steps. First, the devices are prebonded, meaning they receive the network details but not the logical address. Then, all the joining devices are provided with the logical address.
- **IQuip** is a special method utilising a dedicated bonding device - IQuip [63]. This device then performs the prebonding stage by receiving a One-Time Key (OTK) from the device by Near Field Communication (NFC) and supplying the network details protected by this key. The coordinator then can include the device in the network by providing the logical address in the same manner as in the Autonetwork method.

6.2 IQRF Communication Standard

The newly formed communication standard aims to provide a comprehensive description of IQRF processes, interfaces, and data structures, facilitating the implementation of interoperable wireless mesh applications based on IQRF technology. However, it is not merely a specification of the original processes; the IQRF network has undergone an evolution aimed at enhancing its reliability, usability, and security. Consequently, this new version is not compatible with IQRF Legacy. Among other improvements, the network's capabilities have been expanded, allowing it to support up to 1,024 devices in a single network.

The redesigned protocol stack, as depicted in Figure 6.3, is structured in a layered architecture, with each layer corresponding to the established ISO/OSI model. The thesis author took responsibility for the Security Services (SES) layer within this protocol stack, which is responsible for the security features of the network. This section details the design considerations and features integrated into this SES layer to ensure robust security measures throughout the communication process.

The primary responsibility of the SES layer is to ensure integrity and authenticity, maintain the payload confidentiality, and provide replay protection. Integrity and authenticity verify that the transmitted data has not been altered and comes from a legitimate source. Payload confidentiality ensures that the transmitted data remains private and is protected from unauthorised access. Replay protection is implemented to prevent malicious actors from reusing the transmitted data to deceive the system.

In the design process, we follow a structured approach to maximise security and minimise risks. We prefer established and well-known solutions that have been thoroughly tested and proven reliable. By reusing these solutions in the correct manner, we significantly reduced the potential for introducing vulnerabilities, as developing new security features from scratch is both complex and prone to errors. We thoroughly studied how other systems implement these security features, as detailed in Chapter 2. Additionally, we examined the vulnerabilities identified in the security features of other networks, as outlined in Chapter 3. Based on the knowledge we acquired from these studies, we designed a security architecture tailored for the new IQRF standard.

The AES-128 block cypher was chosen as the basis for cryptographic operations due to its status as an industry standard, recommended by organisations such as the Open Connectivity Foundation (OCF) [88]. It is already used in many IoT networks and remains robust, with no known practical attacks against it. We chose a 128-bit block size, as in IQRF Legacy, because it offers sufficient security while maintaining low resource requirements, making it energy-efficient and suitable for embedded devices with limited capabilities. Its wide availability and familiarity among developers make it a practical and reliable choice with broad support across hardware and software platforms.

To ensure both authentication and encryption of data frames simultaneously, we conducted an analysis of various Authenticated Encryption with Associated Data (AEAD) options. After careful consideration, we selected Counter with Cipher Block Chaining Message Authentication Code (CCM) as standardised by NIST SP 800-38c [40] and RFC 3610 [113], aligning with its established use in standards like IEEE 802.15.4 and Bluetooth. While GCM (Galois/Counter Mode) was also considered, it was ultimately not chosen due to its increased complexity and the frequent unavailability of readily prepared solutions. CCM was favoured because it has been extensively tested, offers better support, and ensures greater interoperability across different platforms and systems.

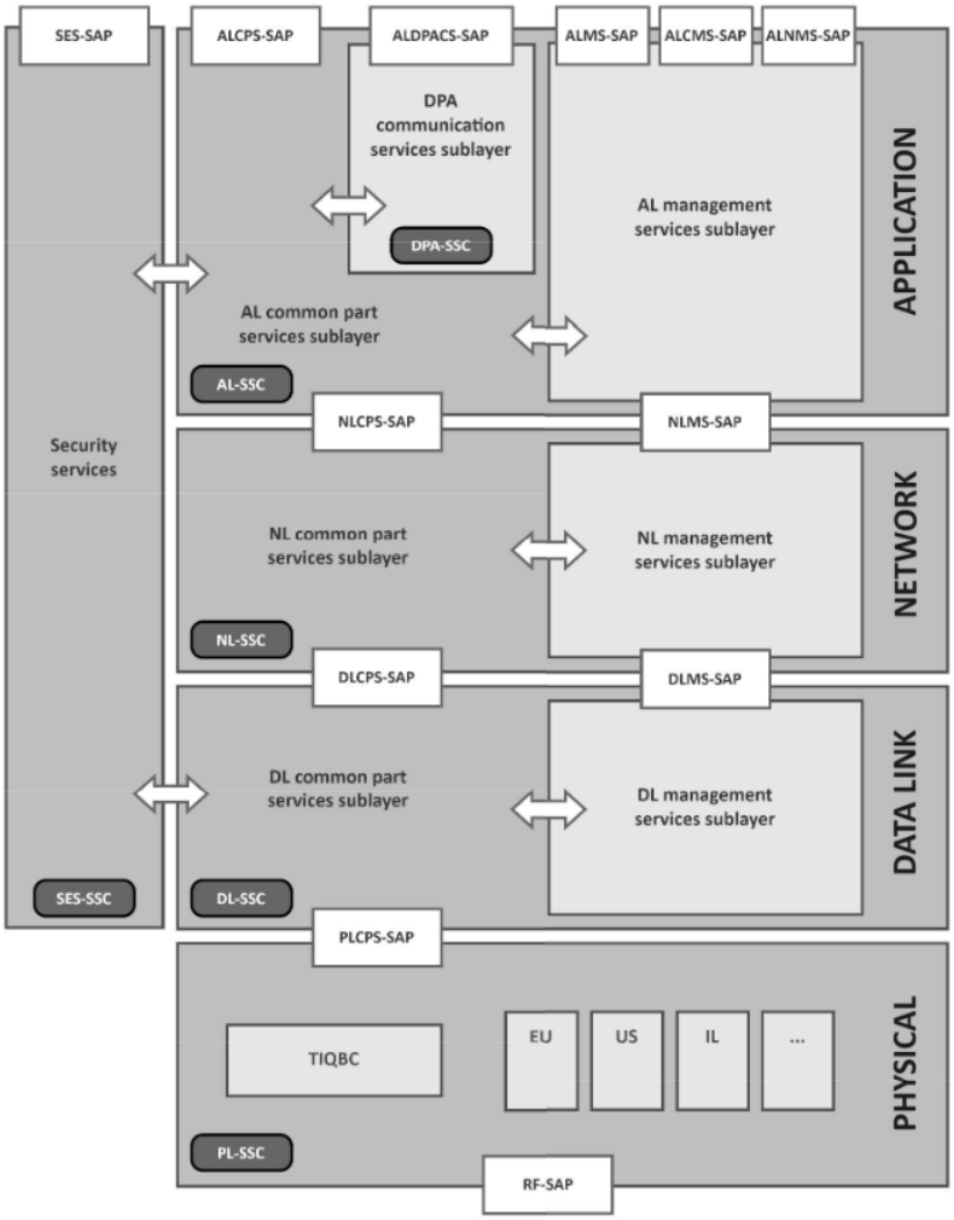

Figure 6.3: IQRF standard design architecture [59]

Before discussing the security features, it is necessary to understand some fundamentals about the frame structure. As shown in Figure 6.4, the frame is organised into several layers. The Data-Link layer consists of a header and a footer that contain the payload length (LEN), the frame counter (FCNT), and two authentication tags: the FAT and the RIT. The frame also contains a network header, which holds the network address (NADDR), and a routing header with information used for multi-hop transports. Depending on the frame type, not all layers may be present. Only the data link is required for non-network communication, while routed frames require all the headers mentioned.

Several key identifiers are used in IQRF networks. Although not found in data frames, the 48-bit Network ID (NID) uniquely identifies the network. Similar to IEEE MAC ad-

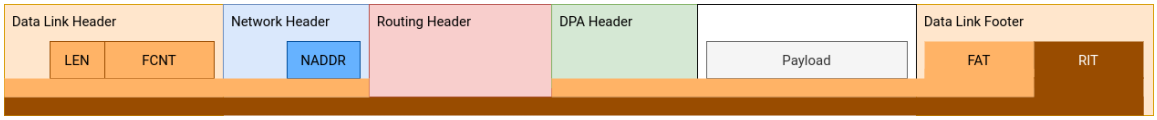

Figure 6.4: Overview of IQRF Frame structure.

dresses, each device can be uniquely identified using the 48-bit IQRF MAC Address (IMAC). However, the 10-bit network address (NTWADDR) is used for network addressing. This NTWADDR is an important part of the NADDR field found in the network header, which further contains flags for distinguishing direction (coordinator/node) and private communications.

The direction aligns with the fact that in IQRF networks, the communication mode has remained consistent with IQRF Legacy, continuing to predominantly use a synchronous request-response method. Private communication within the network is established as a secure link between the coordinator and a specific device, protected by an exclusive key unknown to other devices in the network. This approach is comparable to the trust center link keys used in Zigbee networks.

After this brief introduction to IQRF networks according to the emerging standard, we will take a closer look at the different security features that have been formulated.

6.2.1 Integrity

Integrity in the standard is ensured through a multi-layer approach incorporating 2-3 distinct layers of protection depending on the frame type. All frames include two essential layers: Time Quanta Bit Coding (TIQBC) data encoding and a Frame Authentication Tag (FAT). For routed frames, an additional layer is added with the Routing Integrity Tag (RIT), which safeguards the routing information. The traditional cyclic redundancy codes (CRC) used in IQRF Legacy have been replaced by a more robust CBC-MAC approach, enhancing authentication and ensuring compatibility with the CCM scheme.

Time Quanta Bit Coding

TIQBC [109] is a technique for encoding wireless transmissions at the physical layer. It works by dividing the bitstream into sequences of bits with identical values, then interpreting the length of each sequence as a distinct symbol. If a received pulse does not match the predefined symbols, this indicates an integrity breach, causing the decoder to fail and reject the frame.

Frame Authentication Tag

The second layer of integrity control is ensured by a 4-byte FAT, which is actually a CBC-MAC (Cipher Block Chaining Message Authentication Code) computed as part of the AEAD CCM scheme described in detail in the following Section 6.2.2. For unencrypted non-network frames, only the CBC-MAC portion of the AEAD CCM scheme is performed to calculate the FAT.

Since no secret key is exchanged between the communicating parties for unencrypted non-networked frames, the static integrity substitute key (SIK) defined in the standard is used. Since this key is publicly known, this reduces the authentication code to a simple

error detection tag that can detect transmission errors but does not prevent intentional frame modification.

Routing Integrity Tag

To increase the efficiency and reduce the energy footprint of routing, FAT is used only for those parts of the frame that remain unchanged during routing and are checked only by the end-recipient. Therefore, routed frames are supplemented with an additional 4-byte RIT tag to ensure the security of routing information. This tag is calculated as a CBC-MAC over the entire frame, including routing headers and encrypted data, thereby enhancing integrity assurance during transmission in multi-hop networks.

In order to calculate RIT, frame data are reordered and split into a 128-bit block sequence B, corresponding to the selected block size of the AES. In addition to reordering, NID is also inserted into the sequence binding the computation to the given network. The purpose of reordering is mainly to save resources during routing. In the reordered sequence, data that remains unchanged during the routing process is placed at the beginning. This allows it to be computed only once when verifying the RIT of the received message. Subsequently, this pre-calculated data can be reused to compute the outgoing RIT during transmission.

The procedure to calculate the RIT can be expressed as follows:

1. Lets have the reordered block sequence $B = B_0, B_1, \cdots, B_r$
2. Set $Y_0 = AES\text{-}128_{key}(B_0)$
3. For $i = 1 \ldots r$ do $Y_i = AES\text{-}128_{key}(B_i \oplus Y_{i-1})$
4. Set $RIT = MSB_{32}(Y_r)$

Where AES-128 denotes the selected block cypher, $\oplus$ denotes xor operation and MSB_{32} returns 32 most significant bits of its parameter.

For calculation of RIT, a specific key called Routing Integrity Key (RIK) is shared within the network.

The CBC-MAC is known to be vulnerable to variable-length messages, as malicious data could be appended to the message if the length is not known. This weakness is not applicable because data length is a part of the frame headers and is included in the calculation of RIT.

6.2.2 Authenticated Encryption

As already mentioned, CCM mode was chosen for Authenticated Encryption with Associated Data due to its simplicity, low resource demands, and well-tested nature. CCM mode follows the authenticate-then-encrypt scheme, where data is first authenticated using a Message Authentication Code (MAC) tag calculated via the CBC-MAC method. Subsequently, both the data and the authentication tag are encrypted using the CTR mode. To enhance interoperability, we designed the process to align with RFC 3610 [113].

In order to use this mode, four parameters have to be specified:

- **Cypher key (K)**
- **Nonce (N)**

- **Message to authenticate and encrypt (m)**
- **Additional authenticated data (a)**

Because AEAD is used for network-level encryption, the Network Nommunication Key K is used by default. However, depending on the purpose of the communication frame, various other keys can be used. For example, the Individual Unicast Key designated for private communication. In addition, if non-network communication, such as the association process mentioned later, also requires encryption, a separate key may be used to secure this communication.

The primary purpose of a nonce N is to guarantee the uniqueness of authenticated messages and the keystream used for encryption. To achieve this, we derive the nonce from information found in the data-link and network headers. This uniqueness is specifically ensured by basing it on:

- **Network Identification Number (NID)** to differentiate between IQRF networks.
- **Network Address**, which is guaranteed to be different for every recipient in the network.
- **Frame Counter** which is updated for every message-flow in the network. To prevent overflow attacks, the keys are updated in the event of overflow as described in the following section.

  In non-network communication, the party initiating the communication must choose the FCNT carefully because, in this case, security is directly dependent on it as the only variable element.

After careful consideration, we chose to encrypt (m) only the payload, just like the approach used in IEEE 802.15.4, while keeping the headers in plaintext. We are confident that the integrity of the header data is sufficiently assured and have designed our security architecture to ensure that access to this information does not allow for any potential misuse.

Since the nonce N already encapsulates the data-link and network header information, and the routing header is excluded from the AEAD process, the additional authenticated data a include any remaining headers present in the frame. For non-encrypted, non-network communication, the payload itself is included in a, leaving the encrypted message m empty. This results in the computation focusing solely on the authentication component of CCM.

The procedure of the AEAD scheme used is described by the following two steps. First comes the authentication and then the encryption.

Authentication

The purpose of the authentication step is to calculate a MAC, which is then inserted into the FAT field to ensure the authentication of the transmitted message. Initially, a sequence of blocks B, as shown in Figure 6.6, is created. This sequence is then encrypted using CBC, and the tag is composed of the first 32 bits of the last encrypted block. The length of this sequence can vary from 2 to 6 blocks depending on the amount of transmitted data and the structure of the frame.

The first block consists of 1 byte of flags, as shown in Figure 6.5. For the IQRF case, the only variable is Adata, which indicates whether any unencrypted, authenticated A data

Flags							
7 reserved: 0	6 Adata	5 MAC: 0	4 MAC: 0	3 MAC: 1	2 LEN: 0	1 LEN: 0	0 LEN: 1

Figure 6.5: Flags byte of authentication step of CCM.

is present. This is followed by a 13-bit nonce N, described earlier, and the size of the encrypted data.

If additional headers are present, or if it is unencrypted communication and therefore data a is included, this data follows in the subsequent blocks. However, since the length of this data is unknown, its length is inserted before it, as illustrated in Figure 6.6. If data a is not aligned with the block size, it is padded with zeros to reach the block size.

Finally, the data intended for encryption, known as data m, is added. If this data is not aligned with the block size, it is also zero-padded to reach the block size.

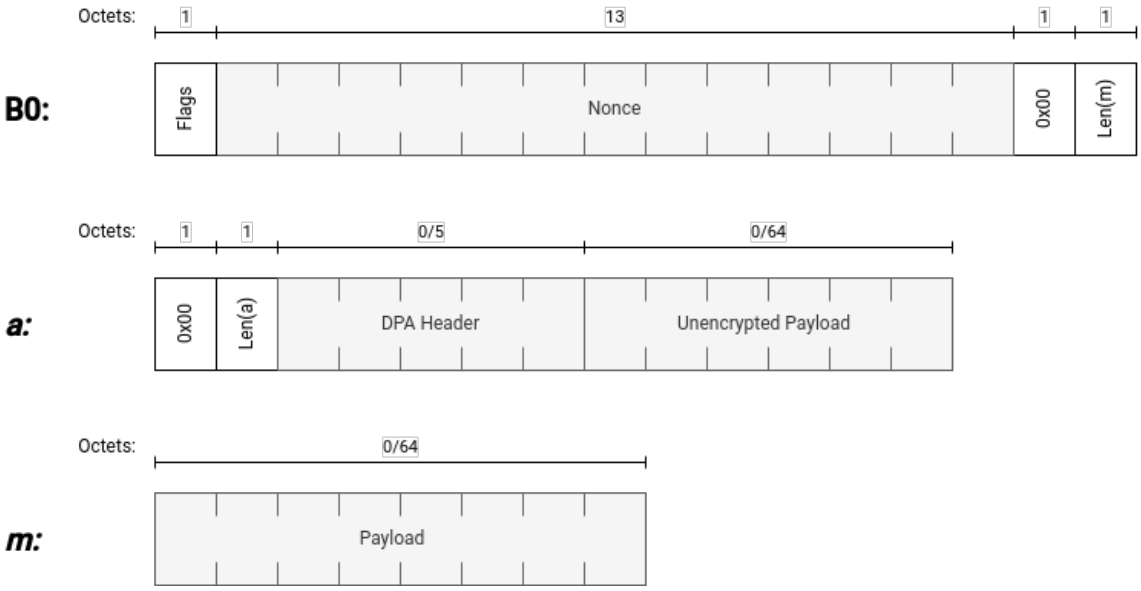

Figure 6.6: Blocks for authentication step of CCM.

The computation process of the FAT from this sequence can be expressed as follows:

1. Lets have the block sequence $B = B_0, B_1, \cdots, B_r$
2. Set $Y_0 = AES\text{-}128_{key}(B_0)$
3. For $i = 1 \ldots r$ do $Y_i = AES\text{-}128_{key}(B_i \oplus Y_{i-1})$
4. Set $FAT = MSB_{32}(Y_r)$

Where AES-128 denotes the selected block cypher, $\oplus$ denotes xor operation and MSB_{32} returns 32 most significant bits of its parameter.

Encryption

The CTR mode used in the CCM encryption scheme works by first defining a counter value that is incremented for each block of data. The counter blocks are encrypted to produce

a keystream, which is then XORed with the plaintext data to perform encryption. In the CCM scheme, both the data m and the MAC are encrypted using this method.

The counter is generated similarly to the first authentication block, B0, and comprises flags, a nonce N, and a block counter with a per-block increment. The flags solely indicate the number of octets in the length field; for IQRF, this is represented by the constant value `0x01`. The different definition of flags ensures that the counter block remains distinct from the B0 authentication block. The structure of the counter is illustrated in Figure 6.7.

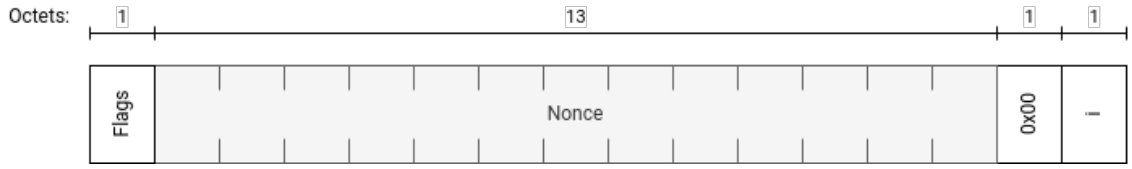

Figure 6.7: Counter structure for CTR encryption mode.

The encryption process of the payload m can be expressed as follows:

1. Create counter blocks $Ctr_0, Ctr_1, \ldots, Ctr_i$, where index denotes the block counter value and $i = \lceil len(m)/128 \rceil$.
2. For $j = 0 \ldots i$ do $S_j = AES\text{-}128_{key}(Ctr_j)$.
3. Set $S = S_1 \vee S_2 \vee \cdots \vee S_i$
4. Return $C = P \oplus MSB_{len(m)}(S)$, $C_{MAC} = MAC \oplus MSB_{32}(S_0)$

Where AES-128 denotes the selected block cypher, $\oplus$ denotes xor operation and MSB_x returns x most significant bits of its parameter.

In the resulting frame, substitute the occurence of payload m with the cyphertext C and unencrypted FAT with the corresponding encrypted C_{MAC}.

6.2.3 Security keys management

In IQRF networks, multiple keys are available for encryption and authentication to ensure secure communication. While some of these keys have been briefly mentioned in previous sections, this section provides a comprehensive overview of the various keys and details their specific use and management in the IQRF ecosystem.

We divide the keys according to the area of validity:

- *Globally valid keys* are defined in the standard and publicly available.
  - *Substitute Integrity Key (SIK)* is a predefined static key within the standard used to calculate the Frame Authentication Tag (FAT) for unencrypted, non-network frames and provide integrity control for devices without shared keys. It simplifies implementation by using the same calculation for authenticated and unauthenticated messages.
- *Network specific keys* are shared by the devices in a network.
  - *Network Access Key (NAK)* secures the process of a device joining a network by encrypting non-network frames involved in the association, thereby enabling the device to successfully connect.

  - *Base Network Communication Key (BNCK)* is shared with devices during association and acts as a secure root for deriving other network-specific keys.
  - *Network Communication Key (NCK)* is the primary network-layer authenticated encryption key, which is derived from the BNCK.
  - *Routing Integrity Key (RIK)* is used to calculate the Routing Integrity Tag (RIT), ensuring the authenticity and integrity of messages during routing within the network. It is also derived from BNCK.
  - *Beaming Communication Key (BCK)* secures offline communication instead of the NAK. Since offline communication follows a different message flow, it cannot accommodate the rotation of the NCK. Therefore, the separate BCK is derived from the BNCK to allow for different lifecycle of the key.

- *Device specific keys* are keys exchanged only between the device and the coordinator that serve as a trusted link.
  - *Device Joining Key (DJK)* can be used instead of NAK for networks which require higher security.
  - *Individual Unicast Key (IUK)* can be provided to a device by the coordinator, enabling private communication within the network.

The NCK, RIK and BCK are derived using the AES-128 encryption algorithm, with the BNCK as the encryption key. The block that undergoes encryption is constructed by distinct means using the Network Identification Number (NID) and, in case of NCK and RIK, the Rotation Index (RIDX). The RIDX, exchanged during the initial association and then kept secure without being transmitted again, acts as a counter for Frame Counter (FCNT) overflows. Whenever FCNT overflows, the NCK and RIK are regenerated using this process. This regeneration helps to mitigate the risk of replay attacks and prevents keystream reuse, addressing security vulnerabilities observed in LoRa networks.

To enable a device to generate keys and participate in network communication, it must first undergo an association process where it is authorised and provided with both an address and the BNCK. This process can be initiated either actively by the node or passively by the coordinator. During this process, NAK can be used to allow multiple devices to join simultaneously, while DJK is a device-specific method. If DJK is employed, it must be provided to the coordinator beforehand through an out-of-band method, along with the device's IMAC address. The coordinator then uses the knowledge of both the IMAC and DJK to authorise the device.

6.2.4 Message freshness

The data-link header includes a 4-byte frame counter (FCNT) that plays a crucial role in ensuring the freshness of messages and mitigating replay attacks. Each message flow is associated with a unique FCNT, which not only serves to distinguish individual messages but also contributes to the construction of the encryption nonce, as outlined in the previous section.

In IQRF networks, communication can occur in two distinct modes: online and offline. Online devices are always active, continuously receiving and processing incoming transmissions, and they play a crucial role in routing within the network. Communication in this mode typically follows a request-response pattern, where the coordinator sends a request,

and the node responds. In contrast, offline (beaming) devices spend most of their time in deep sleep, avoiding constant communication. These devices wake up only under specific conditions to transmit their data to nearby accumulators. Accumulators are special online devices, which then make this information available to the broader network. Due to the different operating modes, the handling of data freshness is tailored separately for each type, as described in the following sections.

Online communication

In online communication, the coordinator is responsible for maintaining the FCNT, which is shared among all devices in the network. Each message flow originates from the coordinator and must have the FCNT value incremented by one over the previous flow. The response from the end device belongs to the same message flow and, therefore, must use the same FCNT value.

To detect replayed messages, each device stores the FCNT of the last successfully received message. This enables the device to verify that any newly received message has a higher FCNT value than the previous one. This process is portrayed in Figure 6.8.

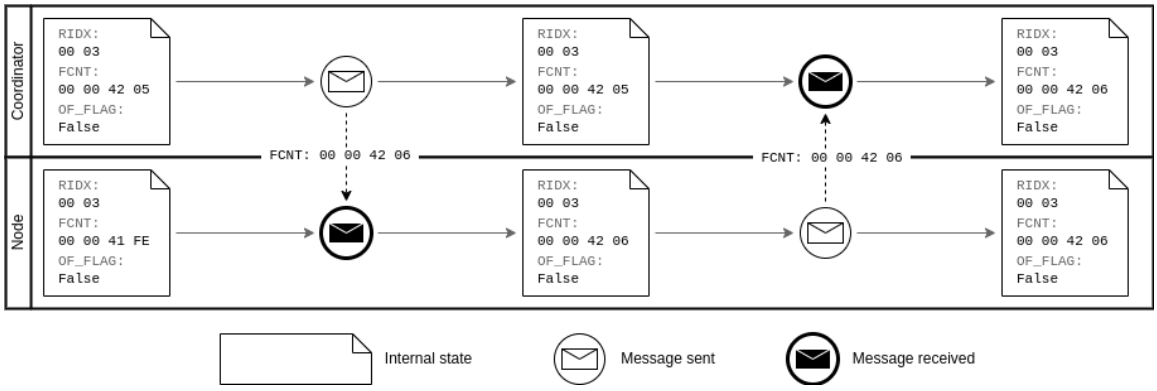

Figure 6.8: Replay protection in online mode.

To manage FCNT overflow, particularly since the counter is shared across the network, each device maintains an additional 16-bit Rotation Index (RIDX). This index, which is not transmitted in communication frames, tracks the overflow of the frame counter. It is initialised during device association and is incremented whenever an FCNT overflow is detected. The combination of RIDX and FCNT provides strong protection against replay attacks while minimising the overhead on both communication frames and device storage.

Offline communication

Since offline devices do not receive messages from other devices in the network, they cannot utilise the shared FCNT used by online devices. Instead, each offline device independently manages its own FCNT, incrementing its value by 1 with every new message sent. As a result, accumulator nodes need to maintain a separate FCNT for each offline device that is registered with them.

6.3 Security evaluation and implementation

The security features introduced for the new standard have greatly enhanced the security of all aspects of IQRF networks. By adopting well-known cryptographic schemes, we have not only achieved a higher level of assurance but also reduced the risk of implementation errors.

Specifically, the integrity check mechanism in IQRF Legacy, which previously relied on CRC-16, has been upgraded to authentication using CBC-MAC. This MAC offers a much stronger guarantee, although, in the case of non-network unencrypted frames, CBC-MAC is downgraded to integrity checking by the use of a publicly known key.

All network communication is now encrypted using authenticated encryption in standard CCM mode. This encryption also protects the association process, a significant improvement over the ECB mode previously used in IQRF Legacy.

In addition, we have introduced automatic key rotation to minimise the risk of keystream reuse and provide automatic recovery in the event of encryption key compromise. This rotation is a critical enhancement for maintaining long-term security integrity. Furthermore, the introduction of private communication with the network coordinator ensures a secure communication link even in the presence of an insider threat. This mechanism can also be used to deploy new keys in the event of a network security breach.

Lastly, the system now features a robust freshness assurance mechanism through the use of a four-byte frame counter. This counter ensures that each transmitted frame is unique, effectively reducing the risk of replay attacks.

In order to validate the security features proposed for IQRF standard, we created an implementation of the basic functionality. This implementation is used only for internal needs for now, but might be released as a supporting material to the standard in the future. We chose Scapy framework[1], which is a powerful interactive packet manipulation library written in Python. We added a custom IQRF layer to this library, which provides functionalities for both the generation and dissection of all IQRF network frames. To cover all possibilities, we implemented four layers in total: data link (`IQRF`), network (`IQRF_NHR`), routing (`IQRF_RTHR`) and DPA (`IQRF_DPA`). These layers correspond to the network layers depicted in Figure 6.4. You can refer to Figure 6.9 for an example of a generated routed frame.

```
>>> from scapy.contrib.iqrf import *
>>> configure(nid='80 0d e2 b8 64 5d', bnck='2c 1d 49 37 ec 8e d2 66 f0 dc 7b 08 26 f3 09 d6', ridx=1)
>>> frame = IQRF(fcnt=5) / IQRF_NHR(naddr=0x8003, asid=3) / IQRF_RTHR(rtvrn=3, rtslot=1, rthops=0, rtdid=1) / 'TEST'
>>> frame
<IQRF  ntwf=True routef=True fcnt=5 |<IQRF_NHR  naddr=32771 asid=3 |<IQRF_RTHR  rtvrn=0x3 rtslot=0x1 rtdid=0x1 rthops=0x0
>>> hexdump(frame)
0000  08 00 C0 04 05 00 00 00 03 80 03 00 03 01 01 00  ................
0010  00 00 00 00 C5 23 91 E9 F1 14 58 73 00 00 14 1D  .....#....Xs....
0020  19 BD                                            ..
>>>
```

Figure 6.9: Generated frame with the encrypted payload highlighted.

An implemented security context object manages security-related functions. This object is responsible for storing, and deriving security keys required by the IQRF protocol. It provides the Scapy layer with the necessary keys for integrating the protocol's encryption and decryption mechanisms, ensuring comprehensive testing of all security features.

[1] https://scapy.readthedocs.io/en/latest

The implementation serves multiple purposes. Firstly, it has allowed basic verification of the proposed security features and comparison with other implementations of the used schemes, ensuring compatibility. Secondly, it allowed for the generation of accurate testing vectors, which can be used by other developers and implementers to verify the compliance of their own IQRF protocol implementations with the standard, ensuring interoperability and adherence to security requirements across different implementations.

During the design of the security features, their use was verified both in terms of security and applicability with respect to the timing requirements of the network and the processing power of the TR module. At the time of writing, the standard is being checked and implemented. Implementation on the hardware in use will allow for subsequent proper validation.

Chapter 7

Privacy in IoT

The primary focus of this thesis has been the security of IoT. We have extensively explored this topic and have proposed a security monitoring approach for Bluetooth and security features of a real-world IoT network. The main goal of security is to prevent unauthorised misuse by third parties. However, closely related to this is the issue of privacy, which we believe is inherently connected. A privacy-respecting device ensures that personal data and metadata are not leaked to unauthorised users. While a secure IoT device can prevent adversaries from accessing or manipulating it, privacy concerns arise when vendors have the potential to access customers' private information. Therefore, we consider privacy an integral component of IoT security, and this chapter is dedicated to exploring this topic.

Privacy is a growing concern in IoT devices primarily because these devices handle large amounts of sensitive data [10], often stored in cloud services, as their use becomes more widespread. Many users underestimate the potential implications of the data they share, often overlooking the inferences that can be made from the collected big data [122]. This leaves the users vulnerable to privacy breaches [10], where adversaries can exploit this information to conduct efficient surveillance, including real-time tracking and monitoring of user activities through their connected IoT devices [91]. This is especially dangerous in wide-area networks capable of monitoring the movement of devices over a large area, such as LoRa, or in interconnected local area networks, such as the Eduroam-like approach for IoT networks.

There are multiple ways of how to address privacy in IoT. The Bluetooth SIG approached this problem by introducing Privacy Extension, which uses different device addresses over time to prevent data linking and device tracing. Perera et al. [91] identifies general privacy concerns of IoT solutions and proposes set of legal methods including standardisation and regulation to counter them. One of the concerns is identifiability based on fingerprinting for which they suggest using anonymity technology such as Tor. Yao et al. [118] proposed a protocol for anonymous sensor data collection that prevents cloud servers that compute statistics from these sensors from linking the data to participants.

We will address this important area briefly in this chapter to ensure that this important area is covered in our work. It is not our intention to provide comprehensive coverage of this area, as we consider it beyond the scope of our work given the extent of the issue. Instead, our goal is to highlight the problem, present intermediate results, and suggest potential directions for further research.

In this work, we focused on a security solution for use in small and home offices (SOHO) based on a framework that collects information from a local gateway. Therefore, our initial research question revolved around such gateways. We were particularly interested in

assessing how privacy is upheld in commercially available devices, leading us to define the following research question addressed in Section 7.1:

> **RQ5**: *How do the communication patterns of consumer-oriented IoT gateways differ, and is it possible to detect privacy leaks through analysing these communications?*

The second area we focused on is the use of anonymisation strategies in the IoT domain. As previously mentioned, Perera et al. [91] proposed the use of anonymisation networks to prevent the identifiability of devices. Accordingly, and given the heterogeneous nature of the IoT environment, in the Section 7.2, we answer the following research question:

> **RQ6**: *How can anonymisation be approached within various network environments used in IoT?*

By examining both of these research areas, our goal is to offer a thorough overview of privacy in the home environment, complementing our work on security. Furthermore, we will provide insight into how anonymisation can be integrated into IoT networks, thereby expanding the potential for their implementation and exploration.

7.1 IoT gateways communication analysis

This section is dedicated to analysing the communication of IoT gateways to address research question *RQ5*. Most commercially available gateways are closed-source, so their inner workings are not transparent to the public or those not involved in their production. This raises concerns about privacy, as these gateways often connect to vendor clouds to provide extended services. With security measures such as encryption and limited cloud access in place, it is unclear how data flows and is handled. Therefore, we aim to analyse the communication patterns of these gateways to deepen our understanding of the amount of data transmitted, as well as its forms and destinations.

For this analysis, we selected four popular gateways and placed them in an identical laboratory environment for examination. We collected both passive traffic data and the communication when the gateways were actively used, as detailed below. The collected data was cleaned and analysed to identify different behavioural patterns in gateway traffic flows.

7.1.1 Previous research of gateways privacy

Smart homes are especially sensitive as any privacy leakage directly affects people and their habits (compared to industrial IoT). Amar et al. [12] decided to assess the IoT behaviour of an average household. Their setup included four endpoint devices, four gateways, three general-purpose devices, and the necessary network elements. They monitored the internal LAN for 22 days and analysed the captured traffic for fingerprinting and privacy and security risk assessment.

Junges et al. [66] proved that an adversary monitoring traffic between an IoT gateway and its control cloud might be able to infer the user actions. While they do not focus on device identification, they note that some actions may be exclusive to a particular device. They made several assumptions, such as that the payload of the same command does not change significantly or that if a user sends multiple actions to multiple devices, it is sent as

a single list of actions. Yoshigoe et al. [119] confirmed their research on a concrete solution – Samsung SmartThings. They captured traffic between the IoT gateway and the cloud server and successfully identified patterns associated with different devices on the network. To prevent this privacy breach, they recommend using synthetic traffic and a VPN to hide the communication patterns.

Bai et al. [20] demonstrated that machine learning can be used to classify devices into categories based on their network traffic, enabling automated device type identification. While most works create categories based on known device types and train their classifiers accordingly, Cvitić et al. [35] attempt a different approach where categories are based on traffic flow features. They base the classes on the coefficient of variation of the received and sent data, which in effect classifies the devices according to the predictability of their behaviour. Their method of classification might be more appropriate for anomaly detection.

Although research generally revolves around identifying and monitoring devices and their activities for privacy reasons, we decided to focus on gateways as a key element of a smart home. Given that a gateway can provide an adversary with substantial insight into the internal network and events, it presents a tempting target. Like Junges [66], we focused on the communication between the gateway and the public network to observe the properties of gateway traffic theoretically available to an attacker monitoring the communication from the outside. We included similar gateways to Amar [12] and used similar traffic capture methodology to compare results and verify their revelations.

7.1.2 Traffic evaluation setup

In order to effectively observe the gateway communication, we have established a unified laboratory environment that is designed for simplicity and repeatability. This environment, illustrated in Figure 7.1, includes one IoT gateway under evaluation, an IoT device with its necessary operational components, a smartphone with the control application, and a router that provides connectivity and allows for traffic capture on its ports. Each gateway is evaluated separately in the same controlled environment to ensure that the generated traffic can be easily compared, and to prevent interference between individual gateways and measurements.

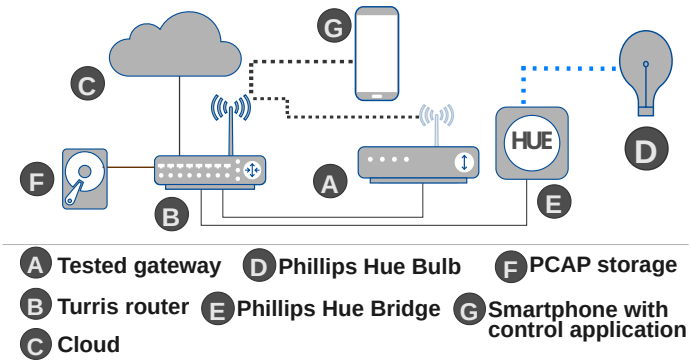

Figure 7.1: Testing environment setup.

We chose the Linux-based Turris MOX as a router, because the system allowed us to perform the capture without a need for a dedicated tap device. The captured data was stored on an external disk in pcap files, and the gateway under test always had a fixed IP

address to streamline the analysis process. Each gateway was set up and controlled using its specific app on an Android smartphone.

Our model does not represent the real-world network, but allows for better recreation and minimises the deviations caused by the communication in the LAN between different nodes. Since the goal of our experiment is to observe the behaviour of the gateway, we consider this setup to be sufficient.

Gateways selection

To gain a useful insight into the current home-IoT landscape, we selected a subset of consumer-oriented IoT gateways, primarily based on their availability and popularity. The full list of selection criteria is as follows:

- Market availability to the non-technical public
- Popularity of the gateway in smart home setups [111, 41]
- Compatibility with the same end devices across the selection (for comparability reasons)

Based on these criteria, we selected four representatives - Aeotec Smart Home Hub, Amazon Echo, Google Nest Mini and Home Assistant software running on Raspberry Pi. Three gateways are complete off-the-shelf solutions with closed-source architecture. Since the internal functionality is concealed, the traffic analysis can provide useful information about how the device operates. Home Assistant was selected for comparison as a popular open-source alternative.

To connect the gateways in our setup, we used the Ethernet connection when available (Aeotec, HA). For gateways without wired connectivity, Echo and Nest, we used WiFi connection of our router.

Selection of IoT devices

During the selection process, we tested a number of IoT devices from different classes, but compatibility with all the gateways turned out to be a problem. In the end, due to its vast compatibility, we chose a Philips Hue Light Bulb connected via a Hue Bridge. Philips Hue is a popular smart lighting ecosystem, which is targeting smart home installations since 2012 and became de facto standard in home lighting automation.

7.1.3 Traffic analysis methodology

To achieve comparable results, the methodology of capturing and analysis process of this work was based on the one of Amar et al. [12].

Data collection was performed for each of the gateways separately. To reduce any unwanted traffic in the network, no other device except the IoT gateway itself, Turris MOX router, Philips Hue light bulb and Philips Hue Bridge was connected to the LAN during the data capturing.

Traffic was captured by *tcpdump* program in two different operation modes for each gateway (active and passive), producing two data sets for each gateway.

- *Passive data capture* was collected during a span of one week, with a maximal deviation of less than $\pm$ 1 hour. During this time, no outer influence in the LAN had been

introduced. All outbound traffic captured in this mode is solemnly generated by IoT gateway and other network devices without any interference by the user.

- *Active data capture* consisted of sets of the same number of actions performed on the Philips Hue light bulb. This means repeatedly switching on and off the lights using a smartphone application meant to control the given gateway.

The captured files were then filtered to contain only the desired gateway traffic and analysed with Zeek[1]. Resulting logs contained metadata about traffic streams, DNS traffic, transferred files, TLS connections and statistics.

Data were manually inspected, and essential information was extracted from them. This includes resources regarding DNS records and queries, NTP behaviour, features of UDP streams, ICMP communications, TCP/TLS documents transferred via HTTP streams, etc. Zeek logs were regularly cross-checked with the content of the pcap files. We searched for interesting traffic behaviour, observable patterns, intriguing nuances of the traffic, cryptographic information and other significant properties of communication.

7.1.4 Analysis results

During the total duration of traffic capturing of 28 days, we captured more than 4080 MiB in 6 579 433 datagrams of raw data. These data were then filtered to contain only the traffic of the tested IoT gateway without LAN overhead such as ARP packets. Another source of noise appeared in the captures of WiFi connected gateways, as some external devices utilised the connection thus generating traffic invaluable for our experiments. Detailed features of the dataset are in Table 7.1.

Gateway	Capture mode	Raw Size	Filtered Size	Raw Packets	Filtered Packets
Aeotec	Active	58.8 KiB	51.0 KiB	302	268
Aeotec	Passive	139.6 MiB	93.7 MiB	701 448	525 089
Echo	Active	305.8 KiB	134.4 KiB	595	319
Echo	Passive	1.1 GiB	221.6 MiB	1 454 195	525 888
Nest	Active	49.6 KiB	24.1 KiB	383	172
Nest	Passive	2.7 GiB	117.0 MiB	3 407 511	507 148
HA	Active	19.3 KiB	29.6 KiB	289	194
HA	Passive	141.3 MiB	83.5 MiB	1 014 335	719 611

Table 7.1: Dataset properties

Figure 7.2 shows the total amount of data transmitted by the protocols of the transport layer. The most sent/received data were detected on the Amazon Echo gateway, which leads from the Google Nest in second place by almost a 100 % more bytes. Home Assistant leads in the IPv6 communication, while Google Nest sent or received the most UDP traffic.

Figure 7.3 also demonstrates the total amount of transmitted bytes, this time on the application layer. All gateways used encryption for cloud communication, but while most of them utilised TLS 1.2 for this purpose, Google Nest was using in-house developed QUIC protocol [50].

As you may notice, the total traffic volumes between Figures 7.2 and 7.3 differ. This is caused by counting only TLS payload bytes for the total volume on application layer, while transport layer includes TLS overhead.

[1] https://docs.zeek.org/en/master/index.html

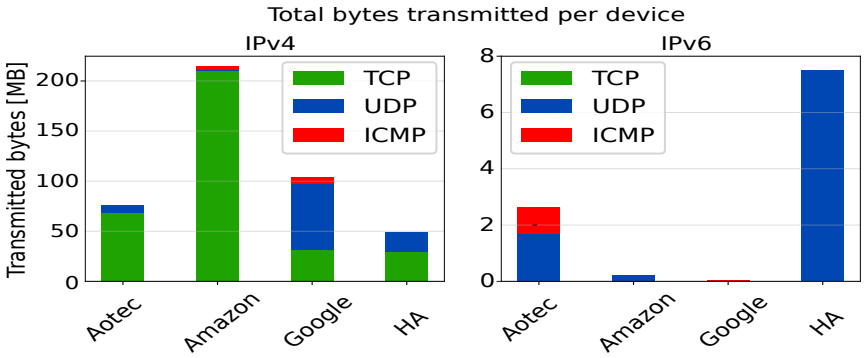

Figure 7.2: Total amount of transferred data (Transport layer)

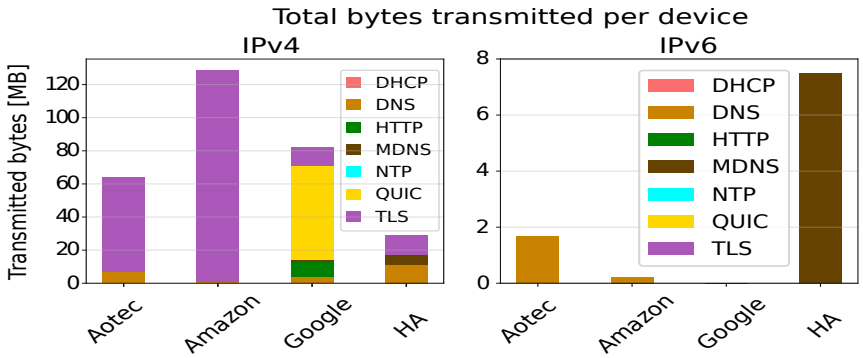

Figure 7.3: Total amount of transferred data (Application layer)

DNS

All gateways used DNS protocol extensively. Home Assistant transmitted most DNS data bytewise.

Amazon Echo showed the least DNS traffic, however it led in the number of total DNS queries for different endpoint (23 distinct locations). Most of these (2014, max interval of 5 minutes between them) queried for the d3p8zr0ffa9t17.cloudfront.net. All DNS queries from AE can be seen in Figure 7.4.

For Aeotec hub, DNS formed 6.2 % of total bytes. Aeotec sent 39 429 DNS requests, all asking only for three queries. These were: api.smartthings.com, fw-update2.smartthings.com and dc-eu01-euwest1.connect.smartthings.com.

Google Nest sent a total of 33 887 DNS queries, 20 for IPv4, 1 PTR and 5 MDNS queries. Google Nest used Google's own DNS servers with the 8.8.8.8 IP address. Other devices used the address given to them via DHCP by a router. DHCP given address was used by Google Nest only when queries towards 8.8.8.8 failed. IP addresses and domain names overlap— server hosts multiple services on different domain names.

The Home Assistant sent DNS queries to six endpoints - second-lowest number after Aeotec. It queries cognito-idp.us-east-1.amazonaws.com (used for Amazon authentication[2]), presumably in order to get to cloud.nabucasa.com, which provides cloud solutions for Home Assistant and is served by Amazon AWS. DNS queries for www.home-assistant.io and analytics-api.home-assistant.io were answered with the same range of IP addresses.

[2] Amazon Cognito lets you easily add user sign-up and authentication to your mobile and web apps. [13]

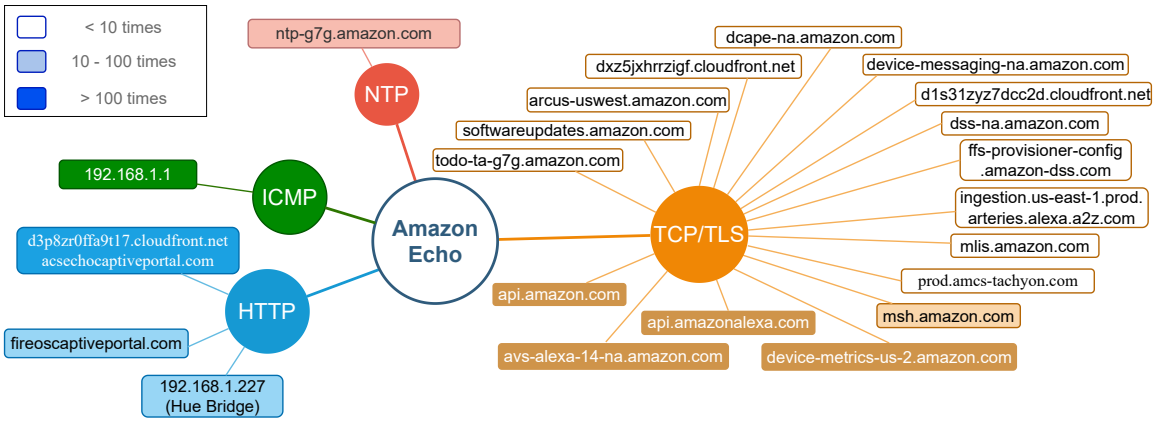

Figure 7.4: Map of Amazon Echo connection endpoints.

The overlap is demonstrated in Figure 7.5. These queries ran periodically, every 24 hours. An intriguing observation, not experienced in other gateways, was scanning the local area network using the PTR DNS record. The Home Assistant sent batches of 254 PTR queries for the entire network with a /24 prefix once a hour.

Figure 7.5: Multiple services sharing an endpoint

All gateways except Amazon Echo demonstrated use of IPv6 protocol. While Google Nest and Home Assistant received replies to their request, the Aeotec hub never got an answer for IPv6 DNS query.

TLS and QUIC

The vast majority of all communication was encrypted using TLS, while Google Nest also used QUIC protocol.

All devices used TLS 1.2, Google Nest and Amazon Echo utilised TLS 1.3 as well. Because of this, we were not able to extract exact content of communications.

55.1 % of Aeotec's communication was TLS. Most connections were held against api.smartthings.com (14 446). The connection towards dc-eu01-euwest1.connect.smartthings.com endpoint ran twice. The first stream lasted 6.5 days into the capture period. The second stream followed immediately after. The streams were idle most of the time, with only TCP keep-alive packets being sent. All endpoints used by Aeotec hub were hosted on Amazon's AWS servers.

Amazon Echo gateway participated in a total of 17 distinct TCP/TLS streams. One was directed towards the d1s31zyz7dcc2d.cloudfront.net. It lasted 42 seconds, and transmitted 105 MB inbound. We assume, with confidence, that the gateway pulled a software

update using this stream. We detected a DNS request and a TLS stream to softwareupdates.amazon.com just before this connection.

Connections to the device-metrics-us-2.amazon.com endpoint ran 1204 times. The endpoint's name suggests the sending of metrics data. Stream, which repeated most often (3 047 times), communicated with api.amazonalexa.com. These streams were short, with an average length of 0.37 seconds. They used multiple IP addresses, presumably for load balancing. All conversations are encrypted using TLS 1.3 or TLS 1.2. All TLS 1.3 conversations had client hello PDU padded to 583 TLS bytes.

Google Nest was the only device which used the QUIC protocol. The QUIC streams were directed to the www.google.com, www.googleapis.com, clients[3-4].google.com, fcm.googleapis.com, play.googleapis.com and tools.google.com endpoints. All QUIC streams were protected using TLS 1.3 with AES-GCM authentication and Curve25519 key exchange. All handshake packets were padded to the same length (1350B of UDP payload).

The Google Nest used the TCP/TLS protocol alongside QUIC. While the average stream duration, when using the QUIC protocol, was less than 1 second, communications via TLS were much longer—2 113.5 seconds on average.

The Home Assistant connected to github.com 112 times, always with two TLS streams starting simultaneously. Conversation with the cloud.nabucasa.com ran during the whole week, divided into two distinct TCP/TLS streams. Second stream started only 20 s after the first finished and manifested vastly different behaviour.

HTTP

Due to the vast use of encrypted communication, there was not observed many data transmitted via plain HTTP.

The most HTTP bytes comes from Google Nest. It did not carry user data, but it even if is is only a simple connectivity check, with 20 second interval between these check, it makes a great amount of bytes transmitted.

The Amazon Echo communicated with the local Philips Hue bridge, d3p8zr0ffa9t17.cloudfront.net, acsechocaptiveportal.com and fireoscaptiveportal.com using plain HTTP. The status code of all responses from the last-mentioned endpoint always arrived in two packets, with HTTP codes 204 and 400.

For Home Assistant, there were 2889 HTTP streams to Cloudflare hosted endpoints without registered domain names.

ICMP

The ICMP was actively used by Amazon Echo and Google Nest for testing connectivity towards various endpoints. Amazon Echo sent periodical bursts of 10 ICMP echo messages towards the LAN's default gateway in 5 minute intervals.

Google Nest checked periodically for connectivity to the default gateway and Google's DNS servers. These ping messages were sent in batches of two requests towards each endpoint. Nest also tested connectivity to prg03s13-in-f14.1e100.net, which is the address of Google's servers in Prague, Czechia.

The Aeotec and Home Assistant did not use ICMP messages for connectivity testing. They only used ICMPv6 Neighbour advertisement and Neighbour solicitation messages.

NTP

Aeotec hub was the only gateway which did not utilise NTP for time synchronisation. Amazon Echo used an NTP for time synchronization. NTP conversations occurred 31 times, with the maximal interval between data stream being 6 hours. Google Nest reached towards the time[0-4].google.com endpoints for its time keeping. A maximum of 20 minutes passed between two NTP synchronization streams. Home Assistant used Cloudflare NTP servers to get current time information. The endpoint, to which it connected 295 times (constant interval of 00:34:08), was time.cloudflare.com.

Active mode

During the active mode of capturing, only Aeotec hub and Home Assistant reported generation of traffic linked to switching the lightbulb on or off.

For Aeotec hub, one continuous TLS stream was detected during the live capturing, which could have been sending commands toward the cloud. Here, the sizes of frames are main determinants. There are two possible patterns of packet streams, which could had been sending the data towards the cloud at the dc-eu01-euwest1.connect.smartthings.com endpoint. Both are shown in Table 7.2. In the first case, it is assumed that there are six packets sent for one operation. The second scenario assumes that not all packets in the stream are directly connected to the light bulb operations.

Scenario 1		Scenario 2	
Direction	Size	Direction	Size
out	113 B	out	113 B
out	490 B	out	490 B
in	113 B	in	113 B
in	318 B	in	318 B
out	113 B		
out	490 B		

Table 7.2: Aeotec's active communication

The Home Assistant did not communicate with the cloud directly when operating the lights. However, a conversation between a Home Assistant and a controlling smartphone was observed. Twenty packets with TCP data and PSH flag, either 79 B or 80 B in length, were sent from the phone. There was no difference between turning the lights on or off. Fifty-seven packets with TCP ACK and TCP PSH flags were sent back to the smartphone. These responses were either 2, 3 or 4 packets long. The lengths of these packets varied between 13 B and 105 B of TCP payload. The first response packet usually arrived 50-60 ms after the request, with the next packet sent from the phone after the next ∼20 ms. For the fourth packet, the time it was sent depended on the type of action performed. While for the „lights out“ action it was sent immediately, for the „lights on“ action it was sent with a delay of approximately one second.

7.1.5 Comparison with results of other studies

From the Amazon Echo's data, we deduced that traffic in this paper and Amar's paper differ, yet they share similarities in other areas. The cause for this could be using a different version of the device, different settings, or simply the time difference between the two studies during which Amazon may have changed internal processes.

The communication of Amazon Echo in this research was mainly composed of TLS traffic (see Figure 7.3), while Amar showed that their Echo also communicated via ICMP and other protocols. Our Echo did not send as many DNS requests, and especially it did not send any requests for DNS resource records such as www.example.net. Also, NTP connections happened less often.

Assumption of Junges et al. [66] that gateways send data after command execution is not always correct, as the results of this paper manifest. The Amazon Echo and Google Nest Mini do not indicate that traffic is generated when operating a light bulb. Therefore, the tool created by Junges et al. would be unusable for these gateways.

7.1.6 Gateways communication insights

By conducting our experiment, we gained invaluable insight into the operation of today's IoT gateways and can answer the research question *RQ5*. We have proven the general expectation that commercial gateways generate more traffic and telemetry than the open source equivalents. While the most concise commercial gateway, Aeotec, generates twice as much traffic as the open source Home Assistant, Amazon Echo generates as much as eight times more.

The extensive use of encryption prevents us (as well as any adversary) from directly inspecting the content of data streams. While encryption enhances confidence in the confidentiality of the transmitted data, it also prevents us from auditing the specific data being collected by vendors. We cannot decisively detect privacy leaks, we can only conclude that the volume of data collected is significant.

The analysis of the traffic exhibited some interesting patterns, which might not raise legitimate safety concerns, but their uniqueness can be used for gateway fingerprinting. Some of these patterns are shown only in LANs, but their need is questionable and should be discussed and users be aware off.

An example is periodic scanning of the LAN for UPnP capable devices instead of scanning on demand. This might provide the user with faster response in device discovery, because the devices are already looked up, but can also provide the vendor with great insight into all devices that appeared in the user network. Another bad practise we could observe was sending UDP multicast packets with non-standard TTL.

While we could not prove with certainty whether home gateways respect customer privacy, we did show that commercial gateways in particular send a lot of data even when idle. Moreover, this data exhibits patterns that could identify the presence of a particular IoT gateway in the network. One option to hide these patterns is to use an anonymity network, which we discuss in the next section.

7.2 Anonymisation strategies

The second area we focus on within the scope of privacy is anonymisation. In this section, we address research question RQ6 by discussing various methods that can ensure anonymisation in IoT networks. Our strategy for device anonymisation relies on leveraging anonymity networks. An anonymity network comprises nodes and communication protocols that withhold certain information from potential adversaries, thereby hindering their ability to determine the communicating parties.

There are multiple issues to address when trying to implement efficient anonymity of IoT devices, such as the number of connected devices, their diversity and constraints. The

ever-increasing number of devices combined with multiple communication technologies used for the last mile and huge differences between various IoT devices make implementing anonymisation for those devices very challenging. Furthermore, IoT devices require low energy consumption, which results in limited computational power and constrains the ability to perform resource-intensive operations, such as cryptographic processes.

To address those issues, we propose possible ways of achieving anonymity in the domain of IoT networks. Given that traditional anonymity networks are typically implemented over IP, we first differentiate the networks based on their underlying technology, as discussed in Section 2.2, into IP and Non-IP networks. Due to the diverse nature of Non-IP networks, we focus on two specific examples: LoRaWAN as a representative of LPWAN and Zigbee as a representative of LR-WPAN, both of which were thoroughly described in Chapter 2. For each of these IoT networks, we propose specific methods to ensure privacy through anonymisation, detailing which pieces of information are concealed from which entities.

7.2.1 General approaches for anonymisation

Achieving anonymity on the internet (for IP devices) is typically done through anonymity networks. There are various types of these networks, with most relying on a method called mixing. Mixing achieves anonymity by altering the order and delaying the messages received by the device responsible for mixing, known as a mix. These mixes can be further organised within a network structure called a mixnet. Beyond mixing, other methods exist, such as reading shared memory anonymously by accessing more than just the intended portion. By employing these strategies, it becomes exceedingly difficult for an attacker to determine who has been communicating with whom.

There are two ways to approach anonymisation. One option is to use existing anonymisation networks to direct the traffic of IoT devices through them. Alternatively, a new anonymity network tailored to the specific needs of IoT devices can be developed. Developing such a network could address the energy and computational limitations of IoT devices, as well as their specific properties, to provide better or cheaper anonymity. For instance, the large number of IoT devices could enhance anonymity by expanding the anonymity set. However, a solution tailored for IoT would need to account for the network's heterogeneity; failing to do so could significantly reduce the number of participating nodes. Additionally, deploying such a network would be challenging due to the limited capacity of these devices to handle updates.

Important information when evaluating an anonymity network is its threat model, the anonymity set size, whether we need to trust parts of the anonymity network itself or the network's latency. The most widespread anonymity network is Tor, which is a circuit-based mixnet. Tor enables the creation of a circuit composed of multiple relays by employing several layers of encryption on data, including routing information. Each relay in the circuit decrypts one layer at a time and then uses the decrypted routing information to forward the message to the next relay. In this process, each relay only knows the identities of the previous and the next relay. Although some (mainly probabilistic) attacks exist, this method is generally considered secure against non-global adversaries. For such adversaries, a traffic confirmation attack could be possible due to Tor's low-latency nature. Importantly, Tor users do not need to trust all relays; it is enough that some relays in the circuit are trustworthy. While there are other existing anonymity networks (with potentially better anonymity levels), a crucial metric influencing anonymity level is the anonymity set site, which means how many users the adversary can pinpoint as possible origins of the message.

Thus, a large user base is important, and for this reason, we consider Tor to be a real option for anonymisation in the context of the proposed techniques of this work.

It is worth noting that anonymity networks protect us from disclosing our identity and location on network level. Nevertheless, it is still possible to get deanonymised based on the data we send – if we, for example, send our exact location (based on GPS) as part of the data, we have effectively disclosed our location despite using an anonymity network.

7.2.2 Anonymisation of IP Devices

IoT devices with implemented IP stack interact with the network in the same manner as every other IP device. As anonymity networks for IP devices are already developed and in use, we can reuse these solutions in the domain of IoT as well. If the device is powerful enough, a client executing anonymisation can be run directly on the device, otherwise it can run on the gateway between the device and the Internet. Those two approaches are described in the following sections and depicted in Figure 7.6

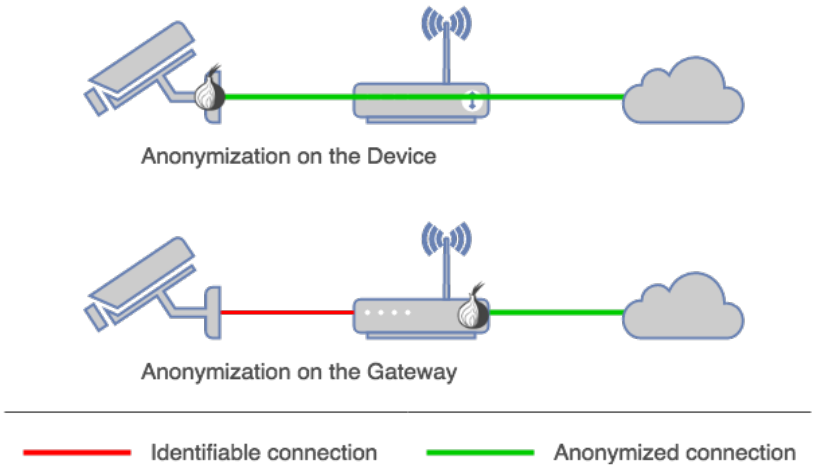

Figure 7.6: Different Anonymisation approaches for IP networks

Anonymisation on the Device

If the device can run its own anonymity network client, it is possible to implement such client as a part of its software with possible hardware support, e.g. for encryption operations. This approach has a great advantage of all the data sent from the device being already anonymised. However, this does not seem to be necessary in most cases as our primary goal is not achieving anonymisation against adversary who already has access/control of our LAN, but anonymise the traffic inside our network against adversaries who has not yet gained the access into it.

Another argument against embedding the anonymisation into a device is the lack of generality. We would have to implement support for anonymisation network into each device, either as a part of its firmware or as an additional module. We would also need to make sure the device indeed uses only the anonymised connection and does not disclose its identity and/or location by other means. Example of such revelation could be downloading

firmware update directly (not through anonymity network) or disclosing the information in the unencrypted data.

Anonymisation on the Gateway

There are also devices with IP network interface but not powerful enough to run an anonymity network client, perhaps because of energy consumption and computational power requirements of cryptographical operations. Such devices are, however, usually connected to the Internet through an IP gateway that is connected to the Internet. This gateway can also serve as an "anonymisation box" — a gateway that sends all the traffic directed to the Internet through an anonymity network client, while anonymising the communication. Every device or even every connection (in the case of TCP) could have its own anonymous identity ("circuit", in case of Tor).

There already exist such anonymisation gateways for the Tor network [55]. These devices can be reused in the case of IoT devices with IP network interface – this would be transparent as the gateway would not even know whether the device is IoT-based and would just anonymise its communication as if it would be a regular IP device (e.g. a computer). This solution is advantageous because there are implementations of such gateways already present, and mainly because this would not require any change to the IoT devices themselves, thus being much more universal and scalable.

Concerns, however, are that firstly the data travel unanonymised through LAN and secondly this approach, without any further improvements, only works for gateways we trust. We can usually trust the gateway that is fully under our control, however, this is a big issue for public gateways. And indeed, we also need to have public anonymisation gateways as there are IoT devices that are expected to move, for example wearables. Creating a protocol that would ensure anonymisation by an untrusted gateway (or at least detection of its malicious behaviour) is a problem not resolved yet.

Another problem of this approach is that the gateway must be used to connect to the Internet by the IoT device. Not all devices will be connected through an anonymisation gateway. If the device is not connected through it, it is not anonymous and it only has limited means of detecting whether it is being anonymised or not. Creating a protocol that would ensure the device is being anonymised — either through communication with the gateway or through some heuristics, like hop count — is an open problem.

7.2.3 Anonymisation of Non-IP Devices

More challenging task is to anonymise devices with network interfaces other than IP. The most spread type of non-IP devices that may benefit of anonymisation are IoT devices and their networks, so we will look deeper into the two networks we have chosen: LoRaWAN and Zigbee. Each of these need a different approach to anonymisation and it is also crucial to define *who* we are anonymising *against whom*.

Proposed Anonymisation for LoRaWAN Devices

For the purpose of LoRaWAN, we consider "anonymisation" as making the server unaware of *whose* device has sent data from *where* or *whose* device has received data. That means the Server only knows it has received data, the data were processed by some application and possibly that application has sent some instruction to another unknown device.

The part where we deny the information about *the owner* of the device seems straightforward. If the user registers his devices and the applications anonymously, it seems impossible to identify him. It is, however, still possible to assign multiple instances of communication to the same anonymous identity. This seems to be unavoidable because devices need to be authenticated in the LoRaWAN network.

A much harder challenge (and a problem to be resolved) is hiding the device's *location*. Each gateway informs the server of its own location. Therefore, when the Server receives some data through a certain gateway, it can easily conclude that the device is somewhere within its transmission range from the gateway. It seems like an easy task to use own gateway that would not send its location or fake it, therefore hiding the device's location as well. There are, however, at least two ways the device's location may be discovered (depicted in Figure 7.7):

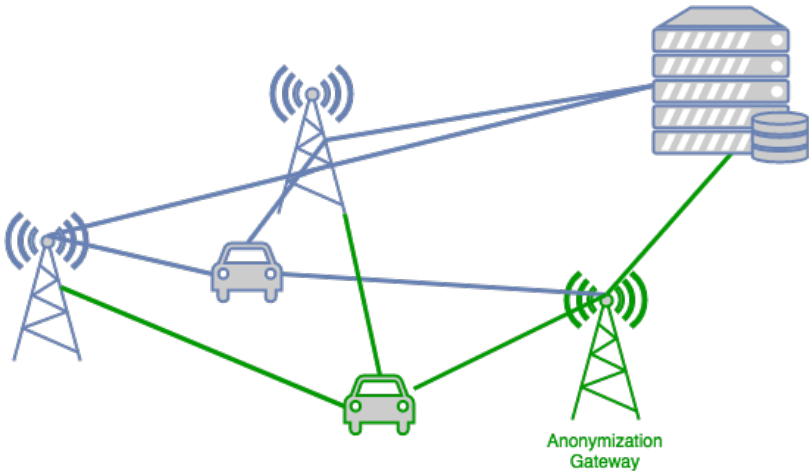

Figure 7.7: Anonymisation gateway in LoRaWAN ecosystem

- Another gateway receives the broadcast from the device (green path in Figure 7.7). This Gateway would also relay the data, however, this time with correct location information. We are not aware of any way of secure unicast to the gateway. We would need to change the protocol for communication between the gateway and the device – so, we would need to make changes to both of them and violate the standard to create a secure encrypted channel with our trusted gateway only.

  The encryption is essential to ensure that gateways not under our control do not understand and relay the messages not intended for them, because such gateways cannot be trusted to follow our modifications. For this reason not only payload, but even headers have to be encrypted.

- Our gateway receives broadcast from another device not under our control (blue line in Figure 7.7). This device's broadcast is also received by at least one other gateway. If both gateways transmit this data to the server as expected, the server can infer the location of the gateway because it knows that it is somewhere near the other gateway whose location the server knows. This would disclose location of all the devices using

our gateway including our devices. A fix to this seems to be straightforward – our gateway should not relay traffic from any other device than ours.

The first point seems problematic mainly due to required changes to the device, the gateway and protocol used by these, and therefore needs further investigation. Also note that one of the most advertised LoRaWAN's advantages is the possibility of device localisation.

Concealing the Location of LoRaWAN Gateways

Another approach to anonymity in LoRaWAN can be achieved by hiding the location of the gateway itself. A typical use case is a user who wants to use the devices that use LoRaWAN but his area is not covered by LoRaWAN yet. He therefore decides to deploy his own Gateway. He, however, does not want the server to know the location of the area in question.

As the communication between the gateway and the Server is routed through the Internet, which is the IP network, we could use one of the solutions shown in Section 7.2.2. However, while this approach would hide the Gateway's IP address, the location may be revealed the same way as mentioned in Section 7.2.3 (through other gateways receiving the same message). After all, part of anonymising the Device's location seems to be anonymising locations of all the Gateways it uses.

Proposed Anonymisation of Zigbee Devices

For the purposes of Zigbee, anonymisation is defined as follows: Let's have a device in a Zigbee network that communicates with another party on the Internet. If the device is anonymised, it is not possible to identify the owner or the location of the device.

As every device in Zigbee networks has to route all its data sent to the Internet through a gateway, we can use this gateway for running an anonymity network client. This way the gateway would actually anonymise the End Device by anonymising itself, as the IP packet originates from the Gateway. If the device itself does not leak any location or identity related information as a part of the data sent, this seems to be secure.

One concern is whether this architecture can not be compromised in the same way as described for LoRaWAN in Section 7.2.3. We conjecture that it cannot happen, as the Zigbee is protocol intended for a localised area and no other gateway should even be within the transmit range of the device. Even if such a gateway would be in a transmit range, the data is encrypted an different networks do not share any keys. Even though the Routers and Coordinator relay the data to other devices (so there may be other Routers or a Coordinator on a way between the communicating End Device and the gateway) it does not seem to be a problem as long as the adversary's device is not part of the user's topology.

Thus we have effectively reduced the task of anonymisation in Zigbee to the same task as anonymisation of IP Devices in Section 7.2.2 – because the Gateway is an IP device as any other – we can also opt for not changing the Zigbee Gateway itself but rather use a cascade consisting of Zigbee Gateway and Anonymisation Gateway similar to an architecture described in Section 7.2.2.

Anonymisation in Public Networks

With high proliferation of IoT devices used on the go such as wearables and sensors on cars, and the expected expansion of smart cities, there is urgent need for anonymisation. In this

environment, public gateways are expected to emerge for IoT devices (such as public WiFi access points) through which those devices would send their data.

If the privacy will not be a part of the network protocols used, privacy could be ensured by the public gateways. Because such gateways are managed by some third party, they cannot be trusted and it would be a very advantageous property if a device would be able to discover whether it is actually anonymised. Protocol to achieve such feature is a future work.

7.3 Concluding remarks to the privacy

In this chapter, we have outlined the problem of securing privacy in IoT networks. In particular, we focused on two areas that were characterised by different research questions.

First, in *RQ5*, we addressed the issue of collecting private information on consumer-oriented gateways. We conducted a thorough analysis of four available gateways, where we monitored their traffic and investigated whether we could detect privacy leaks within them. Based on the collected data, we can conclude that the volume of data collected is significant. However, since all the gateways utilise encryption for all their traffic and it is not possible to easily inspect the transmitted data in closed-source gateways, we cannot draw specific conclusions about the privacy of the transmitted data.

Nonetheless, we were able to observe interesting patterns that allow for the fingerprinting of these gateways. This fingerprinting could potentially be exploited by an attacker to identify a gateway located within the network and possibly target it with an exploit. However, we did not pursue this profiling further, leaving it for future research. Additionally, we observed concerning behaviour in some of the gateways, such as the periodic scanning of devices within the network.

We then focused on the possibilities of anonymisation in IoT networks. In relation to research question *RQ6*, we briefly summarised the anonymity networks used in IP networks. For IoT-specific networks, we divided the problem into WAN and PAN networks. For WANs, we proposed methods to anonymise both the devices connected to these networks and the gateways forming the network, while also discussing potential issues that could arise. In the case of PANs, we equated the privacy problem to that of IP devices and highlighted the challenges that arise from using public networks for internet connections.

Chapter 8

Conclusions

In this thesis, our focus has been on the security of wireless IoT networks, which is a critical issue considering the increasing integration of IoT into vital aspects of life, including industry, smart cities, and our homes. The interconnectivity of these networks spans both private and public spaces, linking them with the global internet to enhance convenience and automation. However, this interconnectivity also brings significant challenges in terms of security and privacy. Our research emphasizes the importance of addressing these challenges to ensure that the continued expansion of IoT technologies can be achieved in a safe and secure manner, safeguarding both users and their data.

We began by thoroughly describing the environment and included a detailed analysis of selected networks and their security issues, as identified by researchers. Due to the challenges in applying security patches to deployed IoT devices, we believe that security monitoring has the potential to enhance security. Therefore, we focused on intrusion detection and proposed a multi-network monitoring solution based on the NEMEA framework. As part of this monitoring solution, we closely examined Bluetooth monitoring, addressing a non-trivial issue that previously lacked a clear solution. In addition to security monitoring, which can identify intrusions in existing installations, it is crucial to develop these wireless networks to enhance their security in the future. Consequently, we selected a specific IoT network and designed security features for its new version to ensure it is resilient against the identified issues within IoT environments. Finally, since we consider privacy to be a complementary aspect of security, we also conducted an investigation into this area.

As part of the mentioned monitoring solution, we introduced a complete development cycle for a monitoring method for Bluetooth LE. We presented a novel method based on indirect connection detection, which we consider to be the core of this dissertation, successfully addressing a previously unsolved problem. We established this concept within technical constraints, implemented it as a proof-of-concept, and validated it through experimentation. We then improved the quality of this approach to a usable level, ultimately choosing an MLP Classifier for detection. This classifier achieved an F1-score of 90% across the tested devices. This score indicates that we can reliably detect usage of Bluetooth devices with intermittent advertising pattern. By connecting it with systems like Security Information and Event Management (SIEM) or higher-level methods such as behavioural analysis, we can achieve a robust intrusion detection system. Through this work, we have expanded knowledge in the field and found a cost-effective way to perform basic monitoring of these devices, which is practical for real-world use.

As previously outlined, in addition to our primary research activity, we also briefly addressed privacy-related issues. We specifically looked into two aspects: monitoring the

communication of home IoT gateways for privacy leaks and exploring anonymisation possibilities in IoT networks. We analysed a total of four commercially available gateways, and although we were unable to examine the content of the transmitted data due to encryption, we compared the volume and patterns of the data. Our analysis showed that a significant amount of data is collected, and the patterns we observed appear distinct enough to be exploited by a sophisticated attacker during the reconnaissance phase of an attack to target specific gateways. Additionally, we noticed concerning behaviour in some gateways, such as the periodic scanning of devices within the network. From an anonymisation perspective, we summarised achieving anonymity in traditional IP networks and categorised IoT networks into infrastructure WAN networks and local PAN networks. We proposed anonymisation strategies and highlighted unresolved issues in both types.

We supplemented our research activities with a practical outcome by developing a new standard for a globally used IoT network. We enhanced the existing IQRF network by designing security features for its new standard, which represents an evolution of the network. These security features were designed with the vulnerabilities found in various IoT networks in mind and significantly improved the security of all aspects of the upcoming IQRF networks. Given the widespread use of the IQRF network, this work has made a significant contribution to enhancing the security of wireless IoT communication.

In conclusion, we thoroughly examined the security of IoT networks. Our main contribution was the development of a new principle for monitoring the security of Bluetooth devices. Additionally, we demonstrated how to secure a proprietary network by defining the security features of a new standard for such networks. Finally, we discussed privacy, outlined the scope of the issue, and suggested potential directions for future research.

Bibliography

[1] *Bluetooth Technology Overview* Bluetooth® Technology Website. Available at: https://www.bluetooth.com/learn-about-bluetooth/tech-overview/. [Online; Accessed 24.7.2024].

[2] *Comparison of Bluetooth BR/EDR and Bluetooth LE Specifications* The MathWorks, Inc. Available at: https://www.mathworks.com/help/bluetooth/gs/comparison-of-bluetooth-bredr-and-bluetooth-le.html. [Online; Accessed 7.8.2024].

[3] *Developer Study Guide: Bluetooth® Low Energy Security* Bluetooth® Technology Website. Available at: https://www.bluetooth.com/bluetooth-resources/le-security-study-guide/. [Online; Accessed 10.8.2024].

[4] *Bluetooth Adaptive Frequency Hopping on a R&S CMW* [Online; Accessed 25.7.2024]. Rohde & Schwarz, Dec 2016. Available at: http://www.rohde-schwarz.com/appnote/1C108.

[5] Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA relevance). *Official Journal of the European Union*, 2016-05-04, L 119, p. 1–88.

[6] IEEE Standard for Low-Rate Wireless Networks. *IEEE Std 802.15.4-2020 (Revision of IEEE Std 802.15.4-2015)*, 2020, p. 1–800.

[7] *Bluetooth Core Specification* Bluetooth Special Interest Group. Jan. 2023. Available at: https://www.bluetooth.com/specifications/specs/core-specification/. Rev. 5.4.

[8] *What is Zigbee? Things Must to Know Before Developing Zigbee Products in 2024* DusunIoT. Dec. 2023. Available at: https://www.dusuniot.com/blog/what-is-zigbee/. [Online; Accessed 5.8.2024].

[9] *The Bluetooth® Low Energy Primer* Bluetooth® Technology Website. March 2024. Available at: https://www.bluetooth.com/wp-content/uploads/2022/05/the-bluetooth-le-primer-v1.2.0.pdf.

[10] Abomhara, M. and Køien, G. M. Security and privacy in the Internet of Things: Current status and open issues. In: *2014 International Conference on Privacy and Security in Mobile Systems (PRISMS)*. 2014, p. 1–8.

[11] Afaneh, M. *Bluetooth Addresses & Privacy in Bluetooth Low Energy* Novel Bits. Apr 2020. Available at: https://novelbits.io/bluetooth-address-privacy-ble/. [Online; Accessed 7.8.2024].

[12] Amar, Y.; Haddadi, H.; Mortier, R.; Brown, T.; Colley, J. et al. *An Analysis of Home IoT Network Traffic and Behaviour*. March 2018. DOI: 10.48550/ARXIV.1803.05368. https://arxiv.org/abs/1803.05368.

[13] Amazon Web Services, Inc.. *Amazon CloudFront* online. 2022. [cit. 2021-03-11]. https://aws.amazon.com/cognito/.

[14] Amin, Y. M. and Abdel Hamid, A. T. Classification and analysis of IEEE 802.15.4 PHY layer attacks. In: *2016 International Conference on Selected Topics in Mobile Wireless Networking (MoWNeT)*. April 2016, p. 1–8.

[15] Antonioli, D.; Tippenhauer, N. O. and Rasmussen, K. Low Entropy Key Negotiation Attacks on Bluetooth and Bluetooth Low Energy. *IACR Cryptol. ePrint Arch.*, 2019, vol. 2019, p. 933.

[16] Anupriya, K.; Jerrin, Y. and Jubin, S. E. A Review on IoT protocols for long distance and low power. *Engineering Science and Technology: An International Journal (ESTIJ)*, 2015, vol. 5, no. 6. ISSN 2250-3498.

[17] Aras, E.; Ramachandran, G. S.; Lawrence, P. and Hughes, D. Exploring the Security Vulnerabilities of LoRa. In: *2017 3rd IEEE International Conference on Cybernetics (CYBCONF)*. June 2017, p. 1–6.

[18] Arroyo, J. G. del; Bindewald, J.; Graham, S. and Rice, M. Enabling Bluetooth Low Energy auditing through synchronized tracking of multiple connections. *International Journal of Critical Infrastructure Protection*. Elsevier, 2017, vol. 18, p. 58–70.

[19] Arroyo, J. G. del; Bindewald, J. and Ramsey, B. Securing Bluetooth Low Energy Enabled Industrial Monitors. In: Academic Conferences International Limited. *International Conference on Cyber Warfare and Security*. 2017, p. 167–176.

[20] Bai, L.; Yao, L.; Kanhere, S. S.; Wang, X. and Yang, Z. Automatic Device Classification from Network Traffic Streams of Internet of Things. In: *2018 IEEE 43rd Conference on Local Computer Networks (LCN)*. 2018, p. 1–9.

[21] Bertolaud, A.; Delclef, J.; Delport, V.; Duffy, P.; Dyduch, F. et al. *LoRaWAN® Specification v1.1*. LoRa Alliance, Oct 2017. Available at: https://resources.lora-alliance.org/technical-specifications/lorawan-specification-v1-1.

[22] Blanco, R.; Malagón, P.; Briongos, S. and Moya, J. M. Anomaly Detection Using Gaussian Mixture Probability Model to Implement Intrusion Detection System. In: *Hybrid Artificial Intelligent Systems*. Cham: Springer International Publishing, 2019, p. 648–659. ISBN 978-3-030-29859-3.

[23] Bluetooth Special Interest Group. *Bluetooth Core Specification*. Dec. 2019. Available at: https://www.bluetooth.com/specifications/specs/core-specification/. Rev. 5.2.

[24] Bormann, C.; Ersue, M. and Keränen, A. *Terminology for Constrained-Node Networks* RFC 7228. RFC Editor, may 2014. Available at: https://doi.org/10.17487/RFC7228.

[25] Boukerche, A.; Zheng, L. and Alfandi, O. Outlier Detection: Methods, Models, and Classification. *ACM Comput. Surv.* New York, NY, USA: Association for Computing Machinery, jun 2020, vol. 53, no. 3. ISSN 0360-0300. Available at: https://doi.org/10.1145/3381028.

[26] Bräuer, S.; Zubow, A.; Zehl, S.; Roshandel, M. and Mashhadi Sohi, S. On practical selective jamming of Bluetooth Low Energy advertising. In: *2016 IEEE Conference on Standards for Communications and Networking (CSCN)*. 2016, p. 1–6.

[27] Butun, I.; Morgera, S. D. and Sankar, R. A Survey of Intrusion Detection Systems in Wireless Sensor Networks. *IEEE Communications Surveys Tutorials*, First 2014, vol. 16, no. 1, p. 266–282. ISSN 1553-877X.

[28] Cai, H.; Fang, Y.; Huang, J.; Yuan, M. and Xu, Z. *BLEGuard: Hybrid Detection Mechanism for Spoofing Attacks in Bluetooth Low Energy Networks*. April 2024.

[29] Cao, X.; Shila, D. M.; Cheng, Y.; Yang, Z.; Zhou, Y. et al. Ghost-in-ZigBee: Energy Depletion Attack on ZigBee-Based Wireless Networks. *IEEE Internet of Things Journal*, Oct 2016, vol. 3, no. 5, p. 816–829. ISSN 2327-4662.

[30] Cayre, R.; Nicomette, V.; Auriol, G.; Kaaniche, M. and Francillon, A. OASIS: An intrusion detection system embedded in bluetooth low energy controllers. In: ACM, ed. *ASIACCS 2024, 19th ACM Asia Conference on Computer and Communications Security, 1-5 July 2024, Singapore, Singapore*. Singapore: [b.n.], 2024. ACM, 2024. This is the author's version of the work. It is posted here by permission of ACM for your personal use. Not for redistribution. The definitive version was published in ASIACCS 2024, 19th ACM Asia Conference on Computer and Communications Security, 1-5 July 2024, Singapore, Singapore.

[31] Cejka, T.; Bartos, V.; Svepes, M.; Rosa, Z. and Kubatova, H. NEMEA: A framework for network traffic analysis. In: *2016 12th International Conference on Network and Service Management (CNSM)*. 2016, p. 195–201.

[32] Centenaro, M.; Costa, C. E.; Granelli, F.; Sacchi, C. and Vangelista, L. A Survey on Technologies, Standards and Open Challenges in Satellite IoT. *IEEE Communications Surveys & Tutorials*, 2021, vol. 23, no. 3, p. 1693–1720.

[33] Chen, K.; Zhang, S.; Li, Z.; Zhang, Y.; Deng, Q. et al. Internet-of-things security and vulnerabilities: Taxonomy, challenges, and practice. *Journal of Hardware and Systems Security*, june 2018, vol. 2, no. 2, p. 97–110.

[34] Costin, A.; Zaddach, J.; Francillon, A. and Balzarotti, D. A large-scale analysis of the security of embedded firmwares. In: *Proceedings of the 23rd USENIX Conference on Security Symposium*. USA: USENIX Association, 2014, p. 95–110. SEC'14. ISBN 9781931971157.

[35] Cvitić, I.; Perakovič, D.; Periša, M. and Botica, M. Definition of the IoT Device Classes Based on Network Traffic Flow Features. In: Springer Nature Switzerland AG. *4th EAI International Conference on Management of Manufacturing Systems*. 1st ed. January 2020, p. 1–17. ISBN 978-3-030-34271-5.

[36] Cäsar, M.; Pawelke, T.; Steffan, J. and Terhorst, G. A survey on Bluetooth Low Energy security and privacy. *Computer Networks*, 2022, vol. 205, p. 108712. ISSN 1389-1286. Available at: https://www.sciencedirect.com/science/article/pii/S1389128621005697.

[37] Danalock. *Danalock V3 Smart Lock*. Available at: https://danalock.com/products/danalock-v3-smart-lock. [Accessed 27-08-2024].

[38] De Donno, M.; Dragoni, N.; Giaretta, A. and Spognardi, A. DDoS-Capable IoT Malwares: Comparative Analysis and Mirai Investigation. *Security and Communication Networks*, 2018, vol. 2018, no. 1, p. 7178164. Available at: https://onlinelibrary.wiley.com/doi/abs/10.1155/2018/7178164.

[39] Di Francesco, M.; Anastasi, G.; Conti, M.; Das, S. K. and Neri, V. Reliability and Energy-Efficiency in IEEE 802.15.4/ZigBee Sensor Networks: An Adaptive and Cross-Layer Approach. *IEEE Journal on Selected Areas in Communications*, 2011, vol. 29, no. 8, p. 1508–1524.

[40] Dworkin, M. J. *Recommendation for block cipher modes of operation :: the CCM mode for authentication and confidentiality*. 2007. Available at: http://dx.doi.org/10.6028/NIST.SP.800-38C.

[41] Dykes, D. P. *Aeotec Builds Upon its Industry Leading Portfolio of Smart Home Products Through Agreement with SmartThings* online. 2020. Available at: https://www.accesswire.com/620526/Aeotec-Builds-Upon-its-Industry-Leading-Portfolio-of-Smart-Home-Products-Through-Agreement-with-SmartThings. [cit. 2021-12-23].

[42] Espressif, Systems. *ESP32WROOM32E*. 2023. Available at: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32e_esp32-wroom-32ue_datasheet_en.pdf. [Accessed 10-05-2024].

[43] Fan, X.; Susan, F.; Long, W. and Li, S. *Security Analysis of Zigbee*. 2017. Available at: https://courses.csail.mit.edu/6.857/2017/project/17.pdf.

[44] Felch, R. *Understanding Zigbee and Wireless Mesh Networking* Black Hills Information Security. Aug. 2021. Available at: https://www.blackhillsinfosec.com/understanding-zigbee-and-wireless-mesh-networking/. [Online; Accessed 6.8.2024].

[45] Fouladi, B. and Ghanoun, S. Security evaluation of the Z-Wave wireless protocol. *Black hat USA*. Black Hat, 2013, vol. 24, p. 1–2.

[46] Garbelini, M. E.; Wang, C.; Chattopadhyay, S.; Sumei, S. and Kurniawan, E. SweynTooth: Unleashing Mayhem over Bluetooth Low Energy. In: *2020 USENIX Annual Technical Conference (USENIX ATC 20)*. USENIX Association, July 2020, p. 911–925. ISBN 978-1-939133-14-4. Available at: https://www.usenix.org/conference/atc20/presentation/garbelini.

[47] Gascón, D. *Security in 802.15.4 and ZigBee networks*. 2009. Available at: http://www.libelium.com/security-802-15-4-zigbee.

[48] Gemalto, Actility and Semtech. *LoRaWAN™ Security: Full End-to-End Encryption for IoT Application Providers*. LoRa Alliance, Nov 2020. Available at: https://lora-alliance.org/wp-content/uploads/2020/11/lorawan_security_whitepaper.pdf.

[49] Hamby, M. F. and Robinson, N. *Defensive Strategies for the Internet of Things Sensors Using Bluetooth Low Energy*. 2020. Dissertation. Capitol Technology University. ISBN 979-8-5825-5881-1.

[50] Hamilton, R.; Iyengar, J.; Swett, I. and Wilk, A. *QUIC: A UDP-Based Secure and Reliable Transport for HTTP/2*. Internet-Draft. Internet Engineering Task Force, July 2016. Available at: https://datatracker.ietf.org/doc/html/draft-hamilton-early-deployment-quic-00.

[51] Haxhibeqiri, J.; De Poorter, E.; Moerman, I. and Hoebeke, J. A Survey of LoRaWAN for IoT: From Technology to Application. *Sensors*, 2018, vol. 18, no. 11. ISSN 1424-8220. Available at: https://www.mdpi.com/1424-8220/18/11/3995.

[52] Heinrich, A.; Stute, M. and Hollick, M. BTLEmap: Nmap for bluetooth low energy. In: *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. 2020, p. 331–333.

[53] Hellebrandt, L.; Hujňák, O.; Hanáček, P. and Homoliak, I. Survey of Privacy Enabling Strategies in IoT Networks. In: *Proceedings of the 2017 International Conference on Computer Science and Artificial Intelligence*. 2017, p. 216–221. CSAI '17. ISBN 9781450353922.

[54] Heredia, R. *LTE Cat-M1 explained: Pros and cons of LTE-M for IOT devices*. Zipit, Mar 2023. Available at: https://www.zipitwireless.com/blog/lte-cat-m1-explained-pros-and-cons-of-lte-m-for-iot-devices.

[55] Herrmann, D.; Lindemann, J.; Zimmer, E. and Federrath, H. Anonymity Online for Everyone: What is missing for zero-effort privacy on the Internet? In: Springer. *International Workshop on Open Problems in Network Security*. 2015, p. 82–94.

[56] Hessel, F.; Almon, L. and Hollick, M. LoRaWAN Security: An Evolvable Survey on Vulnerabilities, Attacks and their Systematic Mitigation. *ACM Trans. Sen. Netw.* New York, NY, USA: Association for Computing Machinery, mar 2023, vol. 18, no. 4. ISSN 1550-4859. Available at: https://doi.org/10.1145/3561973.

[57] Hujňák, O.; Malinka, K. and Hanáček, P. Indirect Bluetooth Low Energy Connection Detection. In: *2023 International Conference on Information Networking (ICOIN)*. 2023, p. 328–333.

[58] IQRF Alliance. *The Alliance*. Available at: https://www.iqrfalliance.org/alliance. [Online; Accessed 19.8.2024].

[59] IQRF Standards Association. *IQRF Communication Standard Specification - BRIEF*. Available at: https://iqrf-standards-association.s29.cdn-upgates.com/e/e6650968d3eb40-iqrf-css-240424-brief.pdf. [Online; Accessed 15.8.2024].

[60] IQRF Tech. *Proven*. Available at: https://www.iqrf.org/technology/proven. [Online; Accessed 19.8.2024].

[61] IQRF Tech. *IQMESH Network Deployment: Technical Guide*. 2023.

[62] IQRF Tech. *IQRF OS: User's Guide*. 2023. Version 4.06G.

[63] IQRF Tech. *IQuip IQD-NFC-01: User's guide*. 2023.

[64] Jahangeer, A.; Bazai, S. U.; Aslam, S.; Marjan, S.; Anas, M. et al. A Review on the Security of IoT Networks: From Network Layer's Perspective. *IEEE Access*, 2023, vol. 11, p. 71073–71087.

[65] Jasek, S. Gattacking Bluetooth Smart Devices. In: *Black hat USA conference*. 2016.

[66] Junges, P.-M.; François, J. and Festor, O. Passive Inference of User Actions through IoT Gateway Encrypted Traffic Analysis. In: *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. 2019, p. 7–12. ISBN: 978-3-903176-15-7.

[67] Khanam, S.; Ahmedy, I. B.; Idna Idris, M. Y.; Jaward, M. H. and Bin Md Sabri, A. Q. A Survey of Security Challenges, Attacks Taxonomy and Advanced Countermeasures in the Internet of Things. *IEEE Access*, 2020, vol. 8, p. 219709–219743.

[68] Khraisat, A.; Gondal, I.; Vamplew, P. and Kamruzzaman, J. Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, Jul 2019, vol. 2, no. 1, p. 20. ISSN 2523-3246. Available at: https://doi.org/10.1186/s42400-019-0038-7.

[69] Krejčí, R.; Hujňák, O. and Švepeš, M. Security survey of the IoT wireless protocols. In: *2017 25th Telecommunication Forum (TELFOR)*. 2017, p. 1–4.

[70] Kruse, R.; Mostaghim, S.; Borgelt, C.; Braune, C. and Steinbrecher, M. *Computational Intelligence: A Methodological Introduction*. Springer International Publishing, 2022. ISBN 9783030422271. Available at: http://dx.doi.org/10.1007/978-3-030-42227-1.

[71] Kuzlu, M.; Pipattanasomporn, M. and Rahman, S. Communication network requirements for major smart grid applications in HAN, NAN and WAN. *Computer Networks*, 2014, vol. 67, p. 74–88. ISSN 1389-1286. Available at: https://www.sciencedirect.com/science/article/pii/S1389128614001431.

[72] Lacava, A.; Zottola, V.; Bonaldo, A.; Cuomo, F. and Basagni, S. Securing Bluetooth Low Energy networking: An overview of security procedures and threats. *Computer Networks*, 2022, vol. 211, p. 108953. ISSN 1389-1286. Available at: https://www.sciencedirect.com/science/article/pii/S1389128622001335.

[73] Lee, J.; Hwang, D.; Park, J. and Kim, K.-H. Risk analysis and countermeasure for bit-flipping attack in LoRaWAN. In: *2017 International Conference on Information Networking (ICOIN)*. January 2017, p. 549–551.

[74] Liu, F. T.; Ting, K. M. and Zhou, Z.-H. Isolation Forest. In: *2008 Eighth IEEE International Conference on Data Mining*. 2008, p. 413–422.

[75] Lonzetta, A. M.; Cope, P.; Campbell, J.; Mohd, B. J. and Hayajneh, T. Security vulnerabilities in Bluetooth technology as used in IoT. *Journal of Sensor and Actuator Networks*. Multidisciplinary Digital Publishing Institute, 2018, vol. 7, no. 3, p. 28.

[76] Lv, Y. An adaptive real-time outlier detection algorithm based on ARMA model for radar's health monitoring. In: *2015 IEEE AUTOTESTCON*. 2015, p. 108–114.

[77] Maleki, A.; Nguyen, H. H.; Bedeer, E. and Barton, R. A Tutorial on Chirp Spread Spectrum Modulation for LoRaWAN: Basics and Key Advances. *IEEE Open Journal of the Communications Society*, 2024, p. 1–1.

[78] Marjani, M.; Nasaruddin, F.; Gani, A.; Karim, A.; Hashem, I. A. T. et al. Big IoT Data Analytics: Architecture, Opportunities, and Open Research Challenges. *IEEE Access*, 2017, vol. 5, p. 5247–5261.

[79] Markus, B.; Hans Peter, K.; Raymond, N. and Jorg, S. LOF: Identifying Density-Based Local Outliers. *ACM SIGMOD 2000 Int. Conf. On Management of Data*, 2000.

[80] Melamed, T. An active man-in-the-middle attack on bluetooth smart devices. *Safety and Security Studies*. WIT Press, 2018, vol. 15, p. 2018.

[81] MICRORISC. *IQRF DPA Framework: User's Guide*. 2024. Version 4.31.

[82] Mišić, J.; Shafi, S. and Mišić, V. B. Performance limitations of the MAC layer in 802.15.4 low rate WPAN. *Computer Communications*, 2006, vol. 29, no. 13, p. 2534–2541. ISSN 0140-3664. Available at: https://www.sciencedirect.com/science/article/pii/S014036640600051X. Wirelsess Senson Networks and Wired/Wireless Internet Communications.

[83] Moghaddam, S. S. and Hellinga, B. Algorithm for detecting outliers in Bluetooth data in real time. *Transportation Research Record*. SAGE Publications Sage CA: Los Angeles, CA, 2014, vol. 2442, no. 1, p. 129–139.

[84] Müller, C.; Armknecht, F.; Benenson, Z. and Morgner, P. On the security of the ZigBee Light Link touchlink commissioning procedure. In: *Sicherheit*. 2016.

[85] Nguyen, V.-L.; Lin, P.-C. and Hwang, R.-H. Energy Depletion Attacks in Low Power Wireless Networks. *IEEE Access*, 2019, vol. 7, p. 51915–51932.

[86] Nikoukar, A.; Abboud, M.; Samadi, B.; Güneş, M. and Dezfouli, B. Empirical analysis and modeling of Bluetooth low-energy (BLE) advertisement channels. In: *2018 17th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*. 2018, p. 1–6.

[87] Nikoukar, A.; Raza, S.; Poole, A.; Güneş, M. and Dezfouli, B. Low-Power Wireless for the Internet of Things: Standards and Applications. *IEEE Access*, 2018, vol. 6, p. 67893–67926.

[88] Open Connectivity Foundation. *OCF Security Specification*. November 2023. Available at: https://openconnectivity.org/specs/OCF_Security_Specification.pdf. Version 2.2.7.

[89] Ossmann, M. and Spill, D. Building an all-channel bluetooth monitor. In: *ShmooCon*. 2009.

[90] Padgette, J.; Bahr, J.; Batra, M.; Holtmann, M.; Smithbey, R. et al. *Guide to Bluetooth Security*. SP 800-121. NIST, 2017.

[91] Perera, C.; Ranjan, R.; Wang, L.; Khan, S. U. and Zomaya, A. Y. Big data privacy in the internet of things era. *IT Professional*. IEEE, 2015, vol. 17, no. 3, p. 32–39.

[92] Rajan, A.; Jithish, J. and Sankaran, S. Sybil attack in IOT: Modelling and defenses. In: *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. 2017, p. 2323–2327.

[93] Ratasuk, R.; Mangalvedhe, N.; Zhang, Y.; Robert, M. and Koskinen, J.-P. Overview of narrowband IoT in LTE Rel-13. In: *2016 IEEE Conference on Standards for Communications and Networking (CSCN)*. 2016, p. 1–7.

[94] Ray, A.; Raj, V.; Oriol, M.; Monot, A. and Obermeier, S. Bluetooth low energy devices security testing framework. In: IEEE. *2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST)*. 2018, p. 384–393.

[95] Ray, P. A survey on Internet of Things architectures. *Journal of King Saud University - Computer and Information Sciences*, 2016. ISSN 1319-1578. Available at: http://www.sciencedirect.com/science/article/pii/S1319157816300799.

[96] Reynolds, D. A. et al. Gaussian mixture models. *Encyclopedia of biometrics*. Berlin, Springer, 2009, vol. 741, 659-663.

[97] Ronen, E.; Shamir, A.; Weingarten, A. O. and O'Flynn, C. IoT Goes Nuclear: Creating a ZigBee Chain Reaction. In: *2017 IEEE Symposium on Security and Privacy (SP)*. May 2017, p. 195–212.

[98] Rose, A. and Ramsey, B. Picking Bluetooth low energy locks from a quarter mile away. *DEF CON*, 2016, vol. 24.

[99] Roth, J. M. *A Time Series Approach to Removing Outlying Data Points from Bluetooth Vehicle Speed Data*. 2010. Master's thesis. University of Akron.

[100] Rudresh, V. *ZigBee Security: Basics (Part 2)* Kudelski Security. Nov. 2017. Available at: https://research.kudelskisecurity.com/2017/11/08/zigbee-security-basics-part-2/. [Online; Accessed 7.8.2024].

[101] Ryan, M. Bluetooth: With Low Energy Comes Low Security. In: *Proceedings of the 7th USENIX Conference on Offensive Technologies*. Berkeley, CA, USA: USENIX Association, 2013, p. 4–4. WOOT'13. Available at: https://www.usenix.org/system/files/conference/woot13/woot13-ryan.pdf.

[102] Samara, M. A.; Bennis, I.; Abouaissa, A. and Lorenz, P. A survey of outlier detection techniques in IoT: Review and classification. *Journal of Sensor and Actuator Networks*. MDPI, 2022, vol. 11, no. 1, p. 4.

[103] Schölkopf, B.; Williamson, R. C.; Smola, A.; Shawe Taylor, J. and Platt, J. Support vector method for novelty detection. *Advances in neural information processing systems*, 1999, vol. 12.

[104] Seri, B. and Vishnepolsky, G. *The dangers of Bluetooth implementations: Unveiling zero day vulnerabilities and security flaws in modern Bluetooth stacks*. Armis, 2017. Available at: https://info.armis.com/rs/645-PDC-047/images/BlueBorne%20Technical%20White%20Paper_20171130.pdf.

[105] Sharma, S.; Sharma, S. and Athaiya, A. Activation functions in neural networks. *Towards Data Sci*, 2017, vol. 6, no. 12, p. 310–316.

[106] Shumway, R. H. and Stoffer, D. S. ARIMA Models. In: *Time Series Analysis and Its Applications: With R Examples*. Cham: Springer International Publishing, 2017, p. 75–163. ISBN 978-3-319-52452-8. Available at: https://doi.org/10.1007/978-3-319-52452-8_3.

[107] Sinha, S. *State of IoT 2023: Number of connected IoT devices growing 16% to 16.7 billion globally*. May 2023. Available at: https://iot-analytics.com/number-connected-iot-devices/.

[108] Soukup, D.; Hujňák, O.; Štefunko, S.; Krejčí, R. and Grešák, E. Security Framework for IoT and Fog Computing Networks. In: *2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*. 2019, p. 87–92.

[109] Šulc, V.; Kuchta, R.; Kadlec, J. and Kuchtová, Z. A Time Quanta Bit coding method. *Wireless Networks*, Jan 2020, vol. 26, no. 1, p. 325–332. ISSN 1572-8196. Available at: https://doi.org/10.1007/s11276-018-1814-0.

[110] Sun, J.; Wang, Z.; Wang, H. and Zhang, X. Research on Routing Protocols Based on ZigBee Network. In: *Third International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP 2007)*. 2007, vol. 1, p. 639–642.

[111] The Business Research Company. *Smart Speakers Global Market Report 2022*. ReportLinker, March 2022.

[112] Tomasin, S.; Zulian, S. and Vangelista, L. Security Analysis of LoRaWAN Join Procedure for Internet of Things Networks. In: *2017 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*. March 2017, p. 1–6.

[113] Whiting, D.; Housley, R. and Ferguson, N. *Counter with CBC-MAC (CCM)* RFC 3610. RFC Editor, september 2003. Available at: https://doi.org/10.17487/RFC3610.

[114] Wu, J.; Nan, Y.; Kumar, V.; Payer, M. and Xu, D. {BlueShield}: Detecting Spoofing Attacks in Bluetooth Low Energy Networks. In: *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*. 2020, p. 397–411.

[115] Wu, J.; Nan, Y.; Kumar, V.; Payer, M. and Xu, D. BlueShield: Detecting Spoofing Attacks in Bluetooth Low Energy Networks. In: *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*. 2020, p. 397–411.

[116] Yang, X. *LoRaWAN: Vulnerability Analysis and Practical Expoitation*. The Netherlands, 2017. Master's thesis. Delft University of Technology. Available at: https://repository.tudelft.nl/islandora/object/uuid:87730790-6166-4424-9d82-8fe815733f1e?collection=education.

[117] Yao, F.; Ding, Y.; Hong, S. and Yang, S.-H. A survey on evolved LoRa-based communication technologies for emerging internet of things applications. *International Journal of Network Dynamics and Intelligence*, 2022, p. 4–19.

[118] Yao, Y.; Yang, L. T. and Xiong, N. N. Anonymity-based privacy-preserving data reporting for participatory sensing. *IEEE Internet of Things Journal*. IEEE, 2015, vol. 2, no. 5, p. 381–390.

[119] Yoshigoe, K.; Dai, W.; Abramson, M. and Jacobs, A. Overcoming invasion of privacy in smart home environment with synthetic packet injection. In: IEEE. *TRON Symposium (TRONSHOW), 2015*. 2015, p. 1–7.

[120] Yu, T.; Sekar, V.; Seshan, S.; Agarwal, Y. and Xu, C. Handling a trillion (unfixable) flaws on a billion devices: Rethinking network security for the Internet-of-Things. In: *Proceedings of the 14th ACM Workshop on Hot Topics in Networks*. New York, NY, USA: Association for Computing Machinery, 2015. ISBN 9781450340472.

[121] Zbořil, J.; Hujňák, O. and Malinka, K. IoT Gateways Network Communication Analysis. In: *2023 International Conference on Information Networking (ICOIN)*. 2023, p. 334–339.

[122] Zheng, S.; Apthorpe, N.; Chetty, M. and Feamster, N. User Perceptions of Smart Home IoT Privacy. *Proc. ACM Hum.-Comput. Interact.* New York, NY, USA: Association for Computing Machinery, nov 2018, vol. 2, CSCW. Available at: https://doi.org/10.1145/3274469.

[123] ZigBee Alliance. *ZigBee Specification*. 2015. Available at: https://zigbeealliance.org/wp-content/uploads/2019/11/docs-05-3474-21-0csg-zigbee-specification.pdf. [Online; Accessed 31.7.2024].

[124] Zohourian, A.; Dadkhah, S.; Neto, E. C. P.; Mahdikhani, H.; Danso, P. K. et al. IoT Zigbee device security: A comprehensive review. *Internet of Things*, 2023,

vol. 22, p. 100791. ISSN 2542-6605. Available at: https://www.sciencedirect.com/science/article/pii/S2542660523001142.

[125] Zuniga, J. C. and Ponsard, B. Sigfox system description. *LPWAN@ IETF97, Nov. 14th*, 2016, vol. 25, p. 14.

[126] Čejka, T.; Bartoš, V.; Švepeš, M.; Rosa, Z. and Kubátová, H. NEMEA: a framework for network traffic analysis. In: IEEE. *2016 12th International Conference on Network and Service Management (CNSM)*. 2016, p. 195–201.

[127] Šulc, V.; Hujňák, O.; Plecháč, P. and Poš, J. *IQRF Communication Standard for wireless mesh networks*. IQRF Standards Association, 2024.

Appendix A

Time synchronisation in proposed Parallel monitoring probe

This appendix provides an explanation of how we address the lack of built-in real-time clocks in the ESP32 chips used for collectors in our monitoring probe. To keep track of time, we make use of the internal FreeRTOS tick counter on the ESP32, which measures a relative timestamp in milliseconds since the device's startup. When the ESP32 is initialised, it sends a message containing the current value of this counter. The central monitor calculates the ESP32's start time based on the time of arrival of this message and the counter value, allowing it to convert these relative timestamps into real-time values. Although this initialisation process may cause a one-time shift in time, it ensures consistency for all subsequent timestamps. Upon receiving each subsequent message from the ESP32, the central monitor adjusts the relative timestamp to real-time by taking into account the previously stored start time of the device.

In order to synchronise the start of the ESPs, we use the Data Terminal Ready (DTR) signal. This signal triggers all the ESPs to reboot, effectively resetting their internal tick counters. Our approach focuses on sending the DTR signal to all devices in the shortest possible time, ensuring a near-simultaneous start across the ESPs. Any minor variations in the start times are captured by the aforementioned start message generated after device initialisation, which includes the time elapsed since boot. To prevent the devices from desynchronising, it is necessary to ensure that the brownout detector does not cause a subsequent reset.

To verify the time synchronisation accuracy of the three ESP chips involved in our study, we conducted five 10-minute measurement sessions. Each chip was programmed with firmware that sent messages containing an internal tick counter value every 20 ms, a time interval chosen to match the minimal time between Advertising events in a low-duty cycle. We recorded both the actual elapsed time and the tick counter for each message. To assess the clock skew, we performed statistical analysis, including p-values, linear regression, and confidence intervals. The visualisations in Figures A.1, A.2, and A.3 confirm that the clock skew is approximately 2 ms over the 10-minute measurement period. This level of precision is acceptable for our study, given that we are focusing on 1-minute samples, where the minimum advertising interval for low-duty cycles is 20 ms, and a 10 ms random anti-collision delay is added to each advertising event.

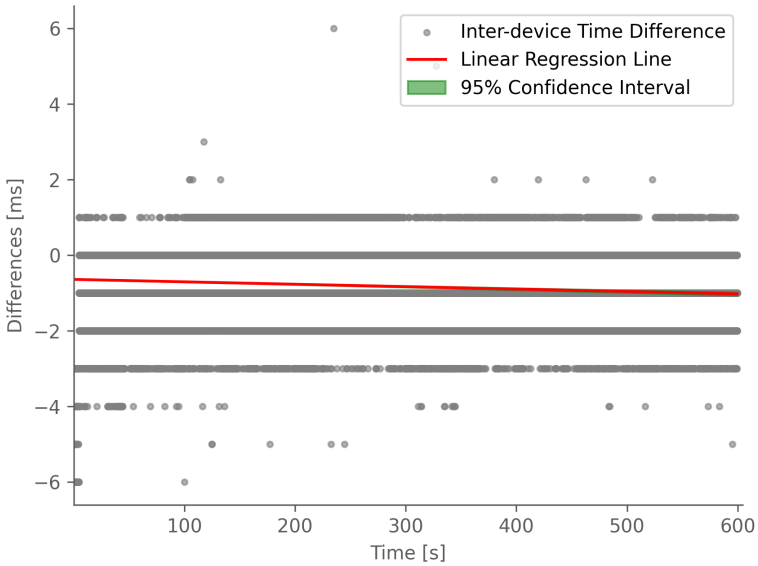

Figure A.1: Clock skew of ESP1

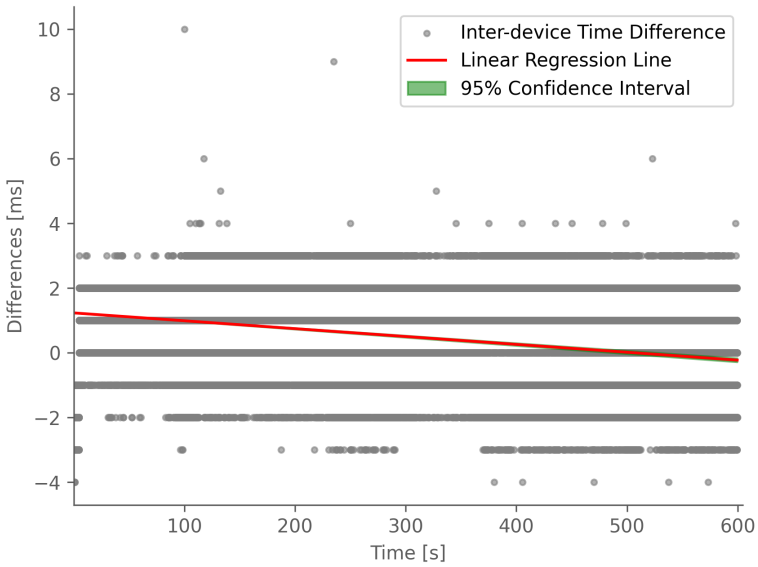

Figure A.2: Clock skew of ESP2

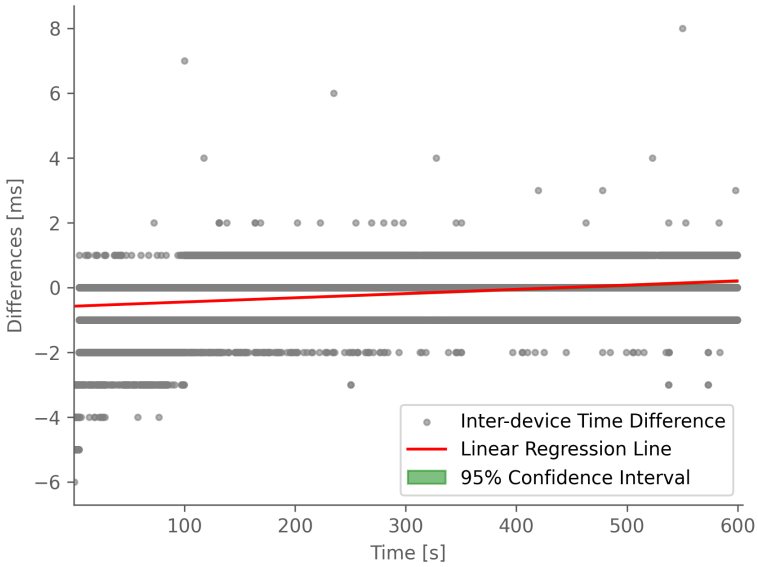

Figure A.3: Clock skew of ESP3

Appendix B

Results of the tested detection methods for BLE monitoring

In this appendix we state an overview of results achieved for detection methods examined in Section 5.7. For every method we provide the full hyperparameter grid indexed by *Configuration*. Then results measured in both environments – *shielded room* and *office* are displayed, sorted in descending order of F1-Score.

B.1 Autoregressive integrated moving average (ARIMA)

Configuration	p	d	q	*sigma*
1	0	0	0	3
2	0	0	1	3
3	0	0	2	3
4	0	0	3	3
5	1	0	0	3
6	1	0	1	3
7	1	0	2	3
8	1	0	3	3
9	2	0	0	3
10	2	0	1	3
11	2	0	2	3
12	2	0	3	3
13	3	0	0	3
14	3	0	1	3
15	3	0	2	3
16	3	0	3	3
17	0	0	1	2
19	0	0	1	4
20	0	0	1	5
21	0	0	1	6

Table B.1: Hyperparameter grid for ARIMA

Configuration	Accuracy	Precision	Recall	Specificity	F1-Score
1	0,997573	0,941818	0,900000	0,998496	0,901587
2	0,997573	0,941818	0,900000	0,998496	0,901587
8	0,997573	0,941818	0,900000	0,998496	0,901587
7	0,997573	0,941818	0,900000	0,998496	0,901587
6	0,997573	0,941818	0,900000	0,998496	0,901587
10	0,997573	0,941818	0,900000	0,998496	0,901587
4	0,997573	0,941818	0,900000	0,998496	0,901587
16	0,997573	0,941818	0,900000	0,998496	0,901587
15	0,997573	0,941818	0,900000	0,998496	0,901587
14	0,997573	0,941818	0,900000	0,998496	0,901587
13	0,997573	0,941818	0,900000	0,998496	0,901587
12	0,997573	0,941818	0,900000	0,998496	0,901587
11	0,997573	0,941818	0,900000	0,998496	0,901587
19	0,996266	0,931818	0,850000	0,998496	0,873810
3	0,996805	0,852929	0,900000	0,997721	0,873517
5	0,996037	0,818741	0,900000	0,996946	0,855210
9	0,996037	0,818741	0,900000	0,996946	0,855210
17	0,996037	0,818741	0,900000	0,996946	0,855210
20	0,993958	0,950000	0,695000	0,999329	0,783333
21	0,992133	0,933333	0,540000	0,999329	0,663532

Table B.2: Results of ARIMA method in the shielded room

Configuration	Accuracy	Precision	Recall	Specificity	F1-Score
1	0,998071	1,000000	0,866667	1,000000	0,900000
3	0,998071	1,000000	0,866667	1,000000	0,900000
9	0,998071	1,000000	0,866667	1,000000	0,900000
8	0,998071	1,000000	0,866667	1,000000	0,900000
7	0,998071	1,000000	0,866667	1,000000	0,900000
6	0,998071	1,000000	0,866667	1,000000	0,900000
5	0,998071	1,000000	0,866667	1,000000	0,900000
10	0,998071	1,000000	0,866667	1,000000	0,900000
4	0,998071	1,000000	0,866667	1,000000	0,900000
2	0,998071	1,000000	0,866667	1,000000	0,900000
16	0,998071	1,000000	0,866667	1,000000	0,900000
15	0,998071	1,000000	0,866667	1,000000	0,900000
14	0,998071	1,000000	0,866667	1,000000	0,900000
13	0,998071	1,000000	0,866667	1,000000	0,900000
12	0,998071	1,000000	0,866667	1,000000	0,900000
11	0,998071	1,000000	0,866667	1,000000	0,900000
17	0,997769	0,923810	0,866667	0,999694	0,872178
19	0,996010	1,000000	0,772222	1,000000	0,846429
20	0,994346	1,000000	0,680556	1,000000	0,776923
21	0,992203	1,000000	0,605556	1,000000	0,703077

Table B.3: Results of ARIMA method in the office

B.2 Gaussian mixture model (GMM)

Configuration	*components*	*sigma*
1	1	2
2	2	2
3	3	2
4	4	2
5	1	3
6	2	3
7	3	3
8	4	3
9	1	4
10	2	4
11	3	4
12	4	4

Table B.4: Hyperparameter grid for GMM

Configuration	Accuracy	Precision	Recall	Specificity	F1-Score
9	0,997573	0,941818	0,900000	0,998496	0,901587
5	0,997333	0,920000	0,900000	0,998254	0,888889
1	0,988501	0,666023	0,900000	0,989339	0,732935
2	0,969723	0,614226	0,665000	0,975606	0,606611
6	0,979054	0,800000	0,440000	0,987919	0,539927
10	0,989184	0,800000	0,350000	0,999329	0,421871
3	0,959287	0,249818	0,335000	0,967713	0,189599
4	0,963858	0,126832	0,305000	0,972522	0,147868
12	0,977383	0,263297	0,110000	0,988347	0,109899
8	0,970463	0,197120	0,130000	0,981131	0,101226
11	0,981785	0,243478	0,110000	0,993292	0,079654
7	0,966770	0,236200	0,170000	0,977766	0,062011

Table B.5: Results of GMM method in the shielded room

Configuration	Accuracy	Precision	Recall	Specificity	F1-Score
9	0,997840	0,890476	0,866667	0,999768	0,828455
5	0,995995	0,761905	0,866667	0,997907	0,757604
1	0,988859	0,618494	0,866667	0,990666	0,696848
2	0,973177	0,535708	0,388889	0,982663	0,432853
6	0,977951	0,660000	0,345000	0,988438	0,420000
10	0,985965	0,400000	0,193333	0,998255	0,248916
3	0,969111	0,493808	0,231111	0,978509	0,242586
4	0,972933	0,514726	0,225000	0,981915	0,242247
11	0,983193	0,428571	0,137778	0,995412	0,174068
8	0,985713	0,400000	0,098889	0,996719	0,153119
7	0,983129	0,183333	0,109444	0,994293	0,122540
12	0,987534	0,333333	0,072222	0,998947	0,112727

Table B.6: Results of GMM method in the office

B.3 Local outlier factor (LOF)

Configuration	*neighbours*	*contamination*
1	2	0.01
2	4	0.01
3	8	0.01
4	16	0.01
5	32	0.01
6	64	0.01
7	128	0.01
9	2	0.1
10	4	0.1
11	8	0.1
12	16	0.1
13	32	0.1
14	64	0.1
15	128	0.1
17	2	0.001
18	4	0.001
19	8	0.001
20	16	0.001
21	32	0.001
22	64	0.001
23	128	0.001

Table B.7: Hyperparameter grid for LOF

Configuration	Accuracy	Precision	Recall	Specificity	F1-Score
6	0,987532	0,496225	0,504394	0,995854	0,401345
9	0,962197	0,275288	0,400303	0,970167	0,301783
11	0,946776	0,194087	0,577273	0,954360	0,269866
13	0,944348	0,156657	0,686061	0,947492	0,248886
7	0,985820	0,419524	0,157727	0,995301	0,222044
14	0,941637	0,134256	0,672727	0,945021	0,213127
10	0,954096	0,172569	0,316970	0,963608	0,208364
3	0,981407	0,178524	0,271515	0,993312	0,205326
15	0,932962	0,125221	0,831515	0,933925	0,205281
12	0,936396	0,121138	0,654545	0,943149	0,198769
1	0,981416	0,206414	0,185455	0,993388	0,193613
2	0,982693	0,174261	0,192121	0,994547	0,178847
4	0,983940	0,130204	0,302424	0,995668	0,153473
17	0,986130	0,326599	0,122727	0,998695	0,144665
21	0,986831	0,431717	0,116061	0,999071	0,143427
22	0,987817	0,457143	0,085000	0,999855	0,126809
5	0,984844	0,271985	0,252727	0,996319	0,126616
18	0,986720	0,222533	0,099394	0,999393	0,103818
23	0,987605	0,366667	0,047727	0,999559	0,083657
19	0,986007	0,141667	0,025455	0,998709	0,040041
20	0,986363	0,000000	0,000000	0,999266	0,000000

Table B.8: Results of LOF method in the shielded room

Configuration	Accuracy	Precision	Recall	Specificity	F1-Score
9	0,964623	0,255401	0,487963	0,970143	0,322885
11	0,944462	0,155720	0,725000	0,948919	0,254118
12	0,944699	0,151195	0,731481	0,946589	0,245967
10	0,958955	0,165623	0,481852	0,967149	0,242805
7	0,986883	0,370099	0,307963	0,996164	0,239707
13	0,943913	0,142680	0,808889	0,945812	0,236806
6	0,986627	0,260598	0,318333	0,996811	0,188898
1	0,981882	0,165691	0,175926	0,993916	0,165107
14	0,928532	0,097190	0,665185	0,932364	0,162290
15	0,921358	0,093531	0,597407	0,924947	0,152881
2	0,978285	0,121060	0,205556	0,990570	0,146356
5	0,984449	0,303550	0,259444	0,996086	0,127996
23	0,986828	0,340000	0,074259	0,998399	0,115696
4	0,982024	0,104548	0,238889	0,994236	0,110987
3	0,979479	0,079922	0,210000	0,992071	0,103869
22	0,987841	0,266667	0,055000	0,999722	0,090370
18	0,984143	0,133516	0,084444	0,997033	0,086486
17	0,984823	0,214478	0,073333	0,997796	0,076640
21	0,987735	0,266667	0,046111	0,999745	0,076481
20	0,985359	0,133333	0,022222	0,998258	0,037908
19	0,985539	0,133333	0,013333	0,998579	0,024020

Table B.9: Results of LOF method in the office

B.4 Isolation forest (iForest)

Configuration	*trees*	*contamination*
1	5	0.01
2	10	0.01
3	25	0.01
4	50	0.01
5	75	0.01
6	100	0.01
7	5	0.1
8	10	0.1
9	25	0.1
10	50	0.1
11	75	0.1
12	100	0.1
13	5	0.001
14	10	0.001
15	25	0.001
16	50	0.001
17	75	0.001
18	100	0.001
19	200	0.1
20	200	0.01
21	200	0.001

Table B.10: Hyperparameter grid for iForest

Configuration	Accuracy	Precision	Recall	Specificity	F1-Score
6	0,992200	0,697637	0,755758	0,997118	0,628490
3	0,991862	0,692825	0,755758	0,996801	0,627815
20	0,991904	0,691658	0,742424	0,997022	0,616041
5	0,991579	0,677861	0,736364	0,996889	0,604398
4	0,991029	0,666652	0,723030	0,996648	0,594222
2	0,990902	0,679169	0,688030	0,997101	0,586472
1	0,991095	0,687234	0,656364	0,997280	0,573371
21	0,988868	0,902564	0,318333	0,999970	0,382958
17	0,988650	0,782828	0,291667	0,999985	0,346669
16	0,988601	0,849495	0,288333	0,999985	0,345413
18	0,988532	0,780952	0,281667	0,999980	0,331416
15	0,988393	0,866667	0,258333	1,000000	0,314625
14	0,988384	0,800000	0,221061	1,000000	0,285257
9	0,924472	0,140456	1,000000	0,923501	0,233888
11	0,923580	0,138917	1,000000	0,922588	0,232010
10	0,923187	0,138953	1,000000	0,922193	0,231974
12	0,923520	0,138883	1,000000	0,922528	0,231811
19	0,923111	0,137828	1,000000	0,922109	0,230509
7	0,923849	0,137720	0,993333	0,922919	0,230389
8	0,923349	0,137806	0,986667	0,922491	0,229942
13	0,987736	0,533333	0,116667	0,999973	0,178969

Table B.11: Results of iForest method in the shielded room

Configuration	Accuracy	Precision	Recall	Specificity	F1-Score
2	0,990835	0,674344	0,677037	0,997111	0,572691
3	0,990690	0,653830	0,691667	0,996661	0,565358
4	0,990843	0,603394	0,697222	0,996969	0,561384
1	0,990885	0,682048	0,649815	0,997360	0,557890
6	0,990704	0,583942	0,704630	0,996719	0,555451
5	0,990262	0,555446	0,679630	0,996995	0,533260
20	0,990041	0,540927	0,679630	0,996772	0,524717
17	0,987924	0,680342	0,297407	0,999944	0,331215
21	0,987482	0,736508	0,296296	0,999438	0,326857
18	0,987778	0,680952	0,281111	0,999944	0,308612
16	0,987202	0,683516	0,267778	0,999459	0,289911
15	0,987590	0,633100	0,245926	0,999969	0,285084
14	0,986926	0,517749	0,198889	0,999500	0,237755
19	0,917389	0,130750	0,982222	0,916554	0,220871
8	0,916209	0,129766	0,982222	0,915368	0,219169
9	0,916168	0,128608	0,977778	0,915384	0,217374
10	0,916907	0,128401	0,977778	0,916118	0,217372
12	0,915196	0,127753	0,991111	0,914186	0,217093
11	0,915259	0,127831	0,986667	0,914325	0,216876
7	0,916165	0,127907	0,968889	0,915512	0,216351
13	0,987441	0,503704	0,160926	0,999979	0,207974

Table B.12: Results of iForest method in the office

B.5 One-class support vector machines (OCSVM)

Configuration	kernel	*degree*	ν	γ
1	linear			
2	sigmoid			
3	rbf		0.5	scale
4	poly	3		
5	poly	4		
6	poly	5		
7	rbf		0.1	scale
8	rbf		0.5	scale
9	rbf		0.9	scale
10	rbf		0.5	auto

Table B.13: Hyperparameter grid for OCSVM

Configuration	Accuracy	Precision	Recall	Specificity	F1-Score
7	0,811744	0,066514	0,980000	0,809624	0,120526
3	0,539495	0,025174	0,940000	0,534686	0,048595
8	0,539495	0,025174	0,940000	0,534686	0,048595
9	0,381864	0,019210	1,000000	0,373649	0,037491
10	0,143628	0,014933	1,000000	0,134014	0,029170
5	0,637288	0,001521	0,080000	0,643654	0,002985
4	0,446142	0,001203	0,080000	0,451030	0,002370
6	0,413721	0,001103	0,080000	0,418549	0,002177
2	0,522682	0,001090	0,080000	0,528303	0,002151
1	0,253335	0,001006	0,080000	0,256375	0,001988

Table B.14: Results of OCSVM method in the shielded room

Configuration	Accuracy	Precision	Recall	Specificity	F1-Score
7	0,905093	0,131123	0,960000	0,904622	0,217487
3	0,447814	0,024401	0,906667	0,442461	0,046814
8	0,447814	0,024401	0,906667	0,442461	0,046814
10	0,249084	0,019421	1,000000	0,239841	0,037708
9	0,141554	0,013731	0,920000	0,131170	0,026864
6	0,476025	0,003579	0,120000	0,480691	0,006950
4	0,470345	0,003564	0,120000	0,474835	0,006923
5	0,439543	0,003448	0,120000	0,444122	0,006704
2	0,451759	0,003371	0,120000	0,456189	0,006557
1	0,359253	0,003249	0,120000	0,362879	0,006327

Table B.15: Results of OCSVM method in the office

B.6 Multi-layer perceptron (MLP)

Configuration	*neurons*	*activation function*
1	[100]	ReLU
2	[100]	Logistic
3	[5]	ReLU
4	[15]	ReLU
5	[2, 2]	ReLU
6	[2, 2, 2]	ReLU
7	[80, 70, 20]	ReLU
8	[80, 70, 20]	Tanh

Table B.16: Hyperparameter grid for MLP

Configuration	Accuracy	Precision	Recall	Specificity	F1-Score
8	0,998572	0,983492	0,903030	0,999526	0,926914
7	0,998545	0,980606	0,903030	0,999499	0,925366
1	0,998480	0,980606	0,896970	0,999499	0,919974
3	0,998376	0,974545	0,896970	0,999393	0,916799
4	0,998206	0,969495	0,890909	0,999287	0,907802
5	0,553429	0,451318	0,927273	0,544342	0,471319
2	0,992163	0,533333	0,393333	1,000000	0,430769
6	0,987184	0,066667	0,013333	1,000000	0,022222

Table B.17: Results of MLP method in the shielded room

Configuration	Accuracy	Precision	Recall	Specificity	F1-Score
7	0,997816	0,974286	0,866667	0,999756	0,893461
1	0,997879	0,993333	0,862222	0,999886	0,891228
8	0,997880	0,993333	0,862222	0,999886	0,891228
3	0,997450	0,985926	0,848889	0,999644	0,870932
4	0,997427	0,993333	0,831111	0,999886	0,850086
5	0,491673	0,412739	0,946667	0,482119	0,424773
2	0,990794	0,600000	0,330556	1,000000	0,391225
6	0,986737	0,000000	0,000000	1,000000	0,000000

Table B.18: Results of MLP method in the office