FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of master's thesis

Title:	LPWAN Protocol Security Analysis Leveraging SDR Technology
Student:	Bc. Štěpán Koníček
Supervisor:	Ing. Jiří Dostál, Ph.D.
Study program:	Informatics
Branch / specialization:	Computer Security
Department:	Department of Information Security
Validity:	until the end of summer semester 2024/2025

Instructions

Low-Power Wide Area Network (LPWAN) protocols enhance the communication range of Internet of Things (IoT) devices while keeping minimal power consumption. With the increasing adoption of these networks by various industries, it is essential to ensure proper security measures. The goal of the thesis is to conduct a cybersecurity analysis of selected LPWAN protocols utilizing a Software Defined Radio (SDR) technology.

1. Get familiar with typical LPWAN protocols such as LoRa, NB-IoT, or Sigfox. Focus particularly on their security features and vulnerabilities.
2. Explore the capabilities of SDR in monitoring and analyzing LPWAN communications.
3. Based on the previous research, develop an SDR based sniffer tool for at least one of previously mentioned protocols.
4. The tool should be designed with the capability to be extended to other LPWAN protocols and further applications.
5. Test the tool in a real LPWAN environment and create documentation.

Electronically approved by prof. Ing. Róbert Lórencz, CSc. on 5 February 2024 in Prague.

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Master's thesis

LPWAN Protocol Security Analysis Leveraging SDR Technology

Bc. Štěpán Koníček

Department of Information Security
Supervisor: Ing. Jiří Dostál, PhD.

May 9, 2024

Acknowledgements

I would like to thank my supervisor, Ing. Jiří Dostál, Ph.D., for his guidance and advice on this thesis. I also want to express my gratitude to my girlfriend, Žanet, for her neverending support throughout my journey on FIT CTU.

Declaration

In Prague on May 9, 2024

Citation of this thesis

Koníček, Štěpán. *LPWAN Protocol Security Analysis Leveraging SDR Technology.* Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2024.

Abstract

This thesis introduces a framework LoRAttack designed to assess the security of LoRaWAN environment using Software Defined Radio (SDR) technology. It begins with a review of LPWAN protocols, focusing on LoRaWAN and highlighting its security features and vulnerabilities. LoRAttack includes features such as multi-channel sniffing for capturing LoRaWAN traffic with enabled channel hopping and session-based capture for structured data analysis. It also offers key derivation from handshake data for decryption and compatibility with Wireshark for seamless traffic analysis. Additionally, its replay module allows vulnerability testing by simulating specific attacks, supported by detailed attack guidance.

Keywords LoRaWAN, SDR, security testing, LPWAN

Abstrakt

Tato práce představuje framework LoRAttack určený k testování bezpečnosti prostředí s LoRaWAN za využití technologie softwarově definovaného rádia (SDR). Začíná analýzou protokolů LPWAN se zaměřením na LoRaWAN s důrazem na jeho bezpečnostní vlastnosti a zranitelnosti. LoRAttack nabízí funkce, jako je vícekanálové odposlouchávání pro zachycení provozu LoRaWAN s implementovaným channel-hoppingem a odposlech v jednotlivých relacích pro strukturovanou analýzu dat. Nabízí také odvození kryptografických klíčů z dat handshaku pro dešifrování a kompatibilitu s programem Wireshark pro snadnou analýzu provozu. Replay modul navíc umožňuje testování zranitelností pomocí simulace konkrétních útoků, které je podpořeno podrobným návodem.

Klíčová slova LoRaWAN, SDR, bezpečnostní testování, LPWAN

Contents

List of Figures

List of Tables

Introduction

With the integration of IoT devices into a wide array of sectors, including consumer products and critical infrastructure, the significance of LPWAN technology emerges, offering advantages in energy efficiency and long-distance communication. Ensuring the security of LPWAN networks is critical, given their pivotal role in transmitting sensitive data within essential systems.

The number of IoT devices is growing each year significantly. Machine-to-machine (M2M) gadgets, devices that communicate with each other directly, are an important subcategory of IoT devices which boosts the overall device count notably. In 2020, the number of active IoT device connections exceeded the number of non-IoT connections [1], and in 2021, the share of M2M gadgets in all IoT devices has been more than 20% [2]. IoT networks based on short-range wireless technology face several obstacles, such as poor scalability and non-robust environment, and cellular-based technology suffers from high deployment cost and network complexity [3]. These obstacles created the need for low-cost, long-range, and robust IoT technology with good network scalability, the Long-range Wide Area Network – LPWAN.

LPWAN is a wide range of network technologies that communicate small data packets over long distances wirelessly while maintaining minimal power consumption and low cost, which ideally meet the requirements of M2M applications [4]. Since 2013, when LPWAN emerged, many protocols have been introduced that vary in frequency band, power capabilities, or security measurements used, nevertheless, on average, LPWAN supports 40 km of coverage in rural areas and 10 km in urban areas [5]. Moreover, the cost of a single device is less than 5$ and maintenance including related network nodes cost less than 1$ on average [6]. Some of the most famous protocols include Long Range (LoRa), NB-IoT, Sigfox, LTE-M, Telensa, Ingenu, and others. The prevalence of LPWAN devices has increased from 2.5% to 14.4% of all IoT devices over the last five years since 2018, establishing its importance in the world of IoT [7].

With the increasing number of LPWAN devices and their possible incorporation into critical infrastructure, security and privacy have become important areas of focus for these protocols. Implementing adequate security measures may pose a significant challenge due to the specified design goals of LPWAN technology, which include low power consumption, affordability, and long-distance coverage. The authors of [8] have presented a comprehensive review of security and privacy issues and attacks that affect LPWAN. During the investigation, multiple security issues affecting the confidentiality, integrity, and availability were identified, along with the countermeasures provided by the LPWAN network developers. As evident, vendors of LPWAN make continuous efforts to integrate security updates to counter the potential attacks and dedicate significant resources to enhance product security, however, are their efforts sufficient? For instance, Slim Loukil and colleagues (2019) evaluated the enhancement of security in LoRaWAN by Semtech with the release of LoRaWAN v1.1 [9]. Nevertheless, many of the vulnerabilities addressed in this update persisted in certain compatibility situations, particularly when IoT nodes and LoRaWAN network servers operated on different protocol versions [9]. 8 out of 12 identified vulnerabilities concerned the availability either by lowering the battery life or abusing the radio frequency spectrum making availability a big security concern in LPWAN [9, p. 17].

This thesis will conduct a comprehensive examination of the features and security aspects chosen LPWAN protocols. It aims to pinpoint vulnerabilities and explore possibilities for the security testing of products utilizing these protocols.

Next, the Software Defined Radio - SDR technology will be analyzed for the purpose of conducting radio frequency testing of LPWAN protocols. Based on previous research, a suitable protocol will be selected considering its testing equipment, documentation, and tool accessibility.

Finally, a sniffer tool for the selected protocol utilizing a chosen SDR technology will be developed and tested in a real-life LPWAN environment.

Chapter 1

LPWAN Overview

This section will provide an overview of LPWAN technology, with a primary focus on LPWAN design objectives, features, network structure, and protocol classification. Additionally, an evaluation will be conducted on the advantages and disadvantages of LPWAN protocols, considering their suitability for diverse applications.

LPWAN, a term found in 2013, abbreviates low-power wide-area network. It refers to a set of network technologies intended for transmitting small data packets wirelessly over long distances at low transmission rates, using less power compared to mainstream network technologies such as WLANs or Bluetooth. The ability to transmit small data packets, consume minimal power, and provide broad signal coverage makes LPWAN suitable for IoT and M2M devices and applications [4].

1.1 Design Objectives

Initially, the design objectives of the LPWAN technology will be presented, along with the rationale behind their development.

- **Wide Coverage** – LPWAN technology offers extensive coverage for transmitting data across significant distances, making it suitable for use with remote standalone sensors, such as those commonly found in the agricultural sector [10]. The coverage of up to tens of kilometres is usually ensured by operation on sub-GHz band that helps with high propagation through obstacles [3, p. 4] and application of Ultra Narrow Band modulation, enabling communication over long distances with low transmit power [11].

- **Low Power Operation** – objective is essential for LPWAN applications as they are designed operate years on a single battery. To operate with low power, star-shaped networks are usually preferred over mesh as multi-hop communication is prevented and duty-cycling is implemented [12].

- **Low Cost** – is desired to develop the capability to connect a large number of nodes to the network [13]. Operating within an unlicensed band and utilizing simple and affordable hardware are essential for achieving cost-effectiveness.

- **Scalability** – Is together with low-cost essential for building large IoT networks. Increasing the number of connected nodes shall not disrupt the existing network; this is often accomplished by implementing adaptive channel selection. [3, p. 6]
- **Interference Management** – Utilizing the unlicensed band for free aligns with the need for cost-effectiveness, however, its performance is below standard because of issues with interference [3, p. 7]. Interference issues are remedied for example by additional base stations [14] or using smart antenna technology [15].
- **Quality of Service** Implementing LPWAN in projects with critical infrastructure and beyond requires ensuring high Quality of Service (QoS). Quality of Service (QoS) is not determined by a single factor; instead, it is influenced by a blend of throughput, latency, jitter (variation in latency), and error rate. Achieving a balance among these components is crucial for improving Quality of Service (QoS) [3, p. 7].

1.2 Licensed and Unlicensed Band

LPWAN technology employs both licensed and unlicensed bands, each with unique characteristics:

- **Licensed Band** – Reserved for specific entities by regulatory authorities. It guarantees exclusive use within a designated area, reducing interference, and ensuring reliable service quality. However, it involves significant costs for spectrum acquisition and regulatory compliance.
- **Unlicensed Band** – Open for use without license, facilitating easier and more cost-effective access to technologies and services. Although it promotes innovation and broad usage, it may suffer from congestion and interference due to the lack of exclusive rights. ISM band is a type of unlicensed band designated for industrial, scientific, and medical purposes without a license. It is globally recognized and used for various applications, offering the advantage of universal access, but with the challenge of managing interference among a wide range of devices [16].

Licensed LPWAN technologies, such as NB-IoT and LTE-M, are known for their effectiveness in terms of Quality of Service (QoS), reliability, latency, and range. On the other hand, unlicensed technologies like LoRa and SigFox stand out for their longer battery life, network capacity, and cost efficiency [17]. The taxonomy of LPWAN protocols based on their band usage can be seen in Figure 1.1.

1.3 Network Topologies and Architecture

The network structure of LPWAN technology typically comprises four categories of devices that interact across different OSI layers. It is important to

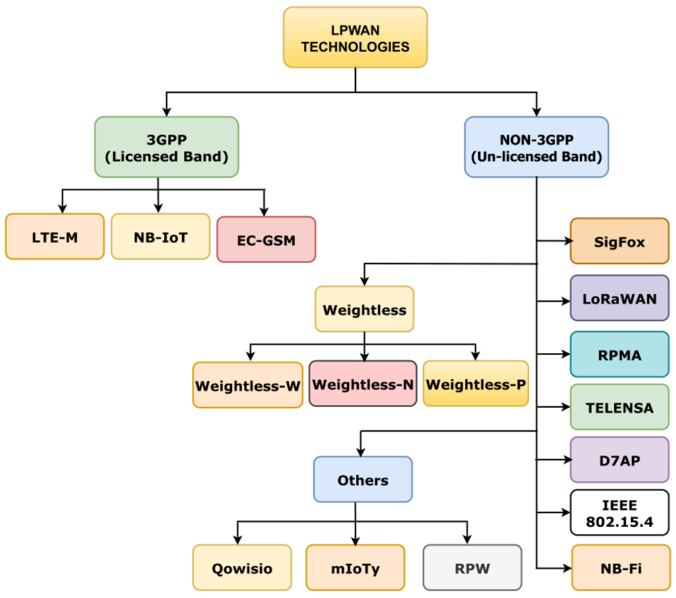


Figure 1.1: LPWAN protocol taxonomy [3]

mention that additional nodes, specific to the protocol, can be added to the network elements mentioned below.

- **End nodes** – End nodes are sensors or actuators collecting operational data. They transmit data to gateways, enabling remote monitoring and control with minimal energy consumption.
- **Gateway** – Gateways receive data from end nodes and forward it to the network server over IP networks. They convert data from the LPWAN protocol into IP packets, facilitating two-way communication between the end nodes and the network, including downlink messages for device control or configuration.
- **Network server** – The network server manages connections, including device authentication and data routing. It processes data received from gateways, ensuring secure transmission to the application server, and handles downlink communications back to end nodes via gateways. Examples of open-source LoRaWAN Network servers are for example The Things Network [18] or the ChirpStack [19].
- **Application server** – The application server is responsible for handling data related to particular applications. It accepts processed data from the network server, enabling data analysis or visualization, and communicates with the end nodes through downlink commands [3].

Find the typical LPWAN architecture of the network nodes shown in Figure 1.2. There are two most commonly used LPWAN network topologies, each with its characteristics and use cases:

- **Star topology** – In a star topology, all end nodes (such as sensors or IoT devices) are directly connected to a central gateway. This gateway then forwards the data to a network server, where it can be processed and analyzed. The star topology is simple and easy to deploy, making it one of the most common LPWAN topologies. Its direct connection approach minimizes the need for data routing, reducing latency and complexity. However, range and coverage are limited by the distance between the end nodes and the gateway, and the gateway can become a bottleneck or a single point of failure.

- **Mesh topology** – Mesh topology allows devices to connect with multiple other devices in the network, allowing data to hop from one device to another until it reaches its destination (either a gateway or another device). This topology can extend the range and coverage of the network beyond what is possible with a star topology, as data can be relayed by devices located between the source and the destination. Mesh networks are more resilient and can be reconfigured in the event of device failure, providing better reliability. However, they are more complex to manage and can consume more power due to the data relay function of the devices, and thus star topology is usually preferred over mesh. [3, p. 9]

It should be noted that cellular networks and other hybrid network types may not strictly adhere to the architecture outlined.

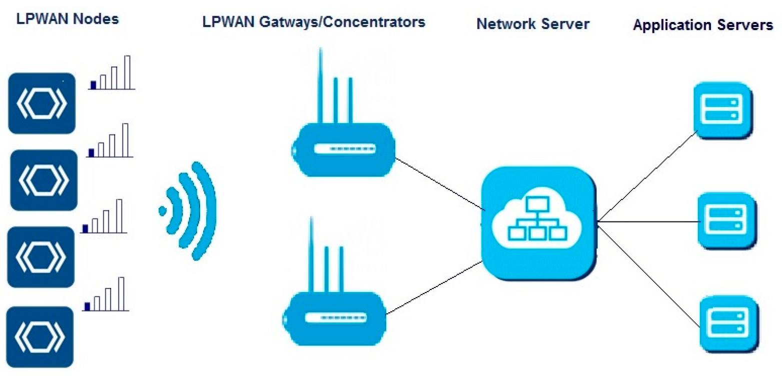


Figure 1.2: LPWAN network architecture [20]

1.4 LPWAN Protocol Suitability

LPWAN is a broad term encompassing various technologies with specific design goals. Each technology has its advantages and disadvantages, making it important to select the most suitable one based on the application requirements. Chilamkurthy, N.S., Pandey, O.J. et al. conducted a comparative analysis [3] to evaluate the most appropriate LPWAN technology for various applications. Key findings of their study include:

- **Energy Efficiency:** Cellular technologies generally consume more power than non-licensed ones, although technologies like Sigfox are notably energy-efficient, especially in sleep mode.
- **Cost of Implementation:** The complexity and size of the network affect costs, with licensed networks (e.g., LTE-M at $5 USD, NB-IoT at $3.3 USD and 3-5 EUR for a LoRa device [21][22]) incurring higher expenses due to subscription fees.
- **Coverage:** Licensed technologies offer better coverage, especially in rural and indoor environments. Unlicensed technology like Sigfox shows superior performance in certain contexts, despite overall lower data rates.
- **Scalability:** Unlicensed specifications allow for a larger number of nodes, while licensed technologies better support increased data loads. For example, D7AP and Weightless-P can support up to 2 million end nodes, while LTE-M accommodates around 80,00. [23].
- **Interference Management:** Both licensed and unlicensed technologies face challenges with jamming and interference. However, unlicensed networks generally have poorer interference management, although Sigfox is reported to outperform NB-IoT in some cases [24].
- **Quality of Service (QoS):** There is a trade-off between QoS and cost, with unlicensed networks suffering from higher latency due to sub-GHz frequencies, but offering lower costs.

In terms of suitability of the most well-known protocols, NB-IoT and LoRaWAN are recommended for noncritical applications, meeting LPWAN objectives at a lower cost, but with limitations in QoS. SigFox, Weightless-P, and D7AP are identified as optimal for specialized applications where they excel despite their limited application scope. For detailed information on the suitability of each technology for specific applications, refer to Table 8 in [3, p. 81953].

Chapter 2

LoRaWAN Security Analysis

This section explains how LoRaWAN technology works, including LoRa modulation, the functionality of the LoRaWAN protocol, and the security analysis of the protocol stack with identified vulnerabilities. It also evaluates the feasibility of exploiting vulnerabilities in terms of using SDR.

2.1 LoRa

Firstly, the difference between LoRa and LoRaWAN shall be demystified. Long-Range (LoRa) is a modulation technique designed for LPWAN developed by Semtech in 2014 [25]. It is based on Chirp-Spread-Spectrum modulation, where the frequency of the transmitted signal increases or decreases linearly. This process creates the "chirp" in the waterfall sink. Transmitting a signal in this way consumes more spectrum, and its energy is dispersed across it, making detection and jamming less feasible. CSS modulation enables LoRa to communicate over long distances with low power and increases resistance to interference, which is suitable for LPWAN [26].

LoRa implements the Spread Spectrum (SF) parameter that allows end nodes to optimize their modulation per their application. This parameter regulates the chirp rate and the speed of data transmission. A lower SF results in faster chirps, leading to a higher data transmission rate. A higher SF means a wider spread and, thus, better sensitivity and range, but at the expense of a lower data rate and longer transmission duration. SF can be set from SF7 to SF12, where higher numbers mean higher spread and lower communication speed. SF7 is appropriate for transmitting data over a distance of around 2 km, while SF12 is suitable for distances up to 8 km [28]. The LoRa chirps based on the SF parameter can be compared in Figure 2.1

In the ISO/OSI model, LoRa is placed in the physical layer, creating only a medium for transmitting bits. LoRa operates in unlicensed sub-GHz ISM bands (430/433/868/915MHz depending on location) with a fixed bandwidth channel of 125 kHz or 500 kHz (for uplink channels) and 500 kHz (for downlink channels) [28].

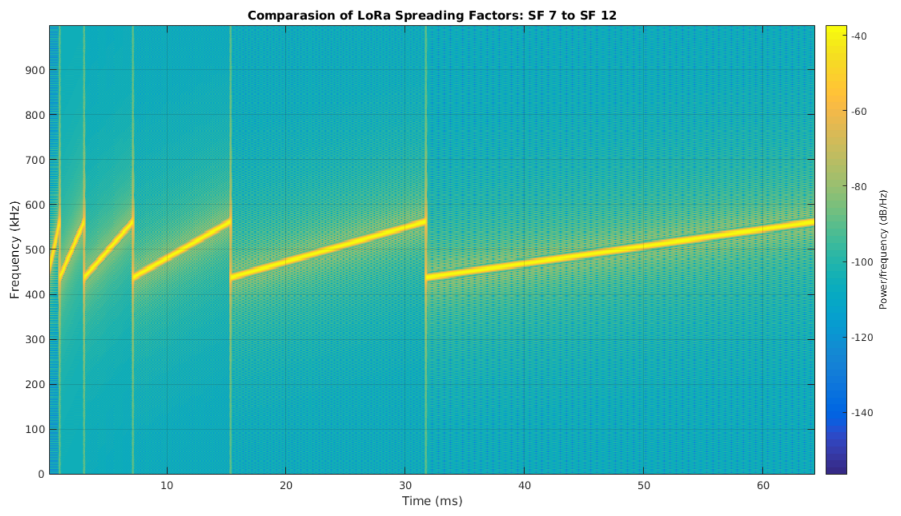


Figure 2.1: Chirps in LoRa modulation and Spreading Factor [27]

2.2 LoRaWAN

LoRaWAN is a communication protocol that leverages LoRa for modulation, enabling long-range, low-power wireless transmissions. It is widely applied across various industries for tasks such as smart city initiatives, environmental monitoring, and healthcare applications, facilitating efficient data collection and management in these sectors [9, p. 101827].

LoRa is a wireless communication technology representing the physical layer (PHY). It supports LoRaWAN, an open networking protocol that enables secure bidirectional communication. LoRaWAN protocol provides the Medium Access Control (MAC) layer in the network model. You can find the LoRaWAN technology stack in Figure 2.2. The technology stack is developed, standardized, and maintained by the LoRa Alliance, which also issues compatibility certificates for end-node products. [29]

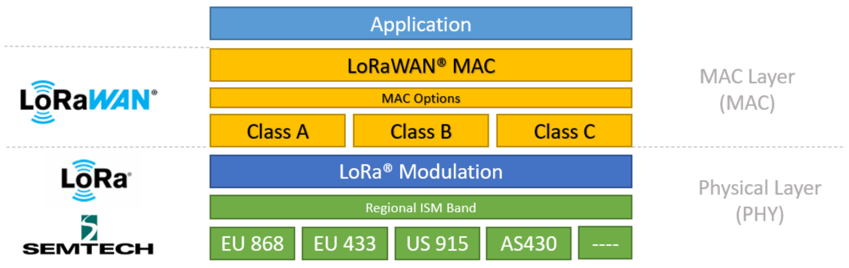


Figure 2.2: LoRaWAN network stack [28]

2.3 LoRaWAN network architecture

A standard LoRaWAN implementation consists of four or five network element types based on the LoRaWAN version. Those elements are defined in accordance with Semtech [28] as follows:

- **End Nodes** – are wireless sensors or actuators that connect to a LoRaWAN network via radio gateways using LoRa modulation. These nodes are assigned unique identifiers during manufacturing and are used to digitize physical conditions or environmental events. They are commonly battery-operated and are used for applications such as street lighting, wireless locks, and more.

- **Gateways** – receive RF messages from end devices and forward them to the LoRaWAN network server through an IP stack. Multiple gateways can serve the same sensor, reducing the packet error rate and battery overhead for mobile sensors. There is no fixation between each node and gateways; all gateways within the distance of the end node will serve it.

- **Network Servers** – manage network operation and security. Dynamically adjusts network parameters, handles device authentication, ensures message integrity, and manages secure connections using 128-bit AES encryption. While it facilitates the flow of control and data traffic, it does not access the content of the application data itself. Here is an overview of the key responsibilities of the NS:

  - **Device Address Checking:** Verifies the addresses of devices attempting to connect to the network to ensure they are authorized.
  - **Frame Authentication and Counter Management:** Authenticates every frame sent over the network and manages the frame counters to prevent replay attacks.
  - **Acknowledgements of Received Messages:** Sends acknowledgments for messages received from devices, ensuring reliable communication.
  - **Adaptive Data Rate (ADR):** Dynamically adjusts the data rate for devices using the ADR protocol to optimize the network efficiency and battery life of the device.
  - **MAC Layer Request Handling:** Responds to all MAC layer requests from devices, facilitating efficient device management.
  - **Application Payload Forwarding:** Forwards uplink application payloads to the appropriate application servers, ensuring that the data reach their intended destination.
  - **Downlink Payload Queuing:** Queues downlink payloads from any application server to any device, managing the delivery of data back to devices.
  - **End Node Onboarding:** Handles the exchange of Join-request and Join-accept messages between devices and the join server, securing device entry into the network.

- **Application Servers** – They are responsible for securely handling and interpreting sensor application data, as well as generating downlink payloads for connected end devices at the application layer.

- **Join Server** handles the over-the-air activation (OTAA) process for the end nodes that are onboarded on the LoRaWAN network. The join server is present only in LoRaWANv1.1, and in LoRaWANv1.0, the management of the OTAA process is handled by the Network Server. The process of OTAA is described in the following sections.

Figure 2.3 displays the LoRaWAN network architecture.

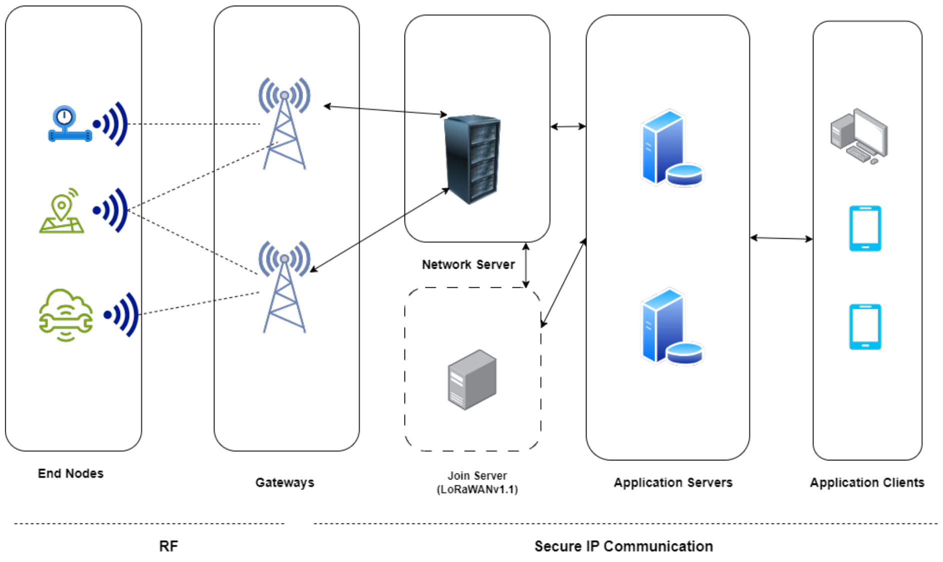


Figure 2.3: LoRaWAN Network Architecture

2.3.1 End Device Classes

End nodes can be classified into 3 subcategories:

- **Class A Devices:** End Devices (EDs) can send data whenever they choose. Gateways (GWs) can send data back to EDs during two specific reception windows, which are opened following an ED's uplink transmission.

- **Class B Devices:** Builds on Class A by adding scheduled receive windows for GWs to send data to EDs, in addition to the two receive windows provided in Class A.

- **Class C Devices:** Allows GWs to send data to EDs at any time, significantly increasing the opportunity for downlink communication. [9, p. 101827]

Class B devices must support Class A operation and Class C devices must support both Class A and Class B operation modes.

Class B devices initiate their reception time slots by synchronizing their clocks with the Gateways. This synchronization procedure, known as *beaconing*, relies on the regular transmission of beacon frames from the GW to the ED. The ED utilizes these beacon frames to synchronize its clock, allowing it to open its downlink window at the appropriate time [28]. However, since beacon frames do not incorporate authentication or integrity protection, they create a vulnerability that can be exploited by Class B LoRaWAN attacks, as detailed in the following sections.

2.4 LoRaWAN Security

This subsection explains the attack surface of LoRa and LoRaWAN according to the Confidentiality, Integrity, and Availability triad. Along with the attack surface, mitigations to the described threats provided by the protocol are presented. Moreover, variations in the creation of session context by different versions of LoRaWAN are outlined. Finally, possible vulnerabilities in the protocol are covered and their feasibility with the use of SDR is assessed. Kindly note that the assessment focuses primarily on security of end devices and confidentiality of data that they transmit as it is related to RF testing with SDR. Threats related to the deployment of the IP layer in the network are identified but not thoroughly discussed.

2.4.1 Threat Model

Network entities present in the LoRaWAN model were defined in the previous section: End Device, Gateway, Network Server, Application Server, and Join Server in case of LoRaWANv1.1.

2.4.1.1 Processes Identification

There are several processes that are happening within the LoRaWAN Network:

1. **Data Uplink Transfer:** Sending data from an end device to the application server through the network server and gateway.
2. **Data Downlink Transfer:** Sending data from the application server to an end device through the network server and gateway.
3. **End Device Onboarding:** Registering and setting up a new device to communicate with the LoRaWAN network.

2.4.1.2 Trust Boundaries

Trust boundaries in terms of data confidentiality refer to the points within a system where the level of trust in protecting sensitive data changes. These boundaries indicate where data transition from a more secure area to a less secure one or vice versa, highlighting potential areas of vulnerability where data might be exposed to unauthorized access or leakage.

The data collected by the end nodes transit between four main trust boundaries:

1. **End Nodes Trust Boundary:** There is a trust boundary between the end devices (nodes) and the rest of the network. The end devices collect or create information that is passed to the application servers. The information is not confidential for single devices within this trust boundary.
2. **Gateways Trust Boundary:** Gateways act as the link between end devices and the network server. Since they can be provided by any third-party service, gateways are not inherently trusted. Within this boundary of trust, the data being transmitted is confidential, and only the essential information required for communication between the network server and a specific end node should be available to them.
3. **Network Server Trust Boundary:** Communication between the end nodes and the application servers is managed by a network server, which also oversees the network itself. While the transferred data remain confidential, network servers have access to information about end devices, application metadata, and other data related to network management.
4. **Join Server Trust Boundary:** In LoRaWANv1.1, the Join Server is delegated to manage the end device onboarding process and thus it is considered a separate trust boundary.
5. **Application Server Trust Boundary:** Application servers process and manage data received from end devices. The transferred information is accessible by the application server as it is needed for further processing.

2.4.1.3 Attacker Model

Creating a threat model for a LoRaWAN environment involves understanding the various attacker profiles that could exploit the unique vulnerabilities at each trust boundary. Below are some attacker models tailored to the described trust boundaries, focusing on attackers with physical access to end devices and those that could perform RF attacks between gateways and end devices:

1. **Attacker with Physical Access to End Device:** This attacker directly interacts with the devices to extract data, clone them or manipulate their functionality. They might use physical tools or software to breach device security. This attacker profile is relevant as the end devices can be placed in an unmonitored environment.
2. **RF Attacker Between Gateway and End Device:** This type of attacker is present within the range of hearing distance between the gateway and the end device. They might use sophisticated radio equipment, such as SDR to eavesdrop, intercept, or disrupt RF signals.
3. **Attacker with Physical Access to Gateway:** The attacker has direct physical access to the gateway device. This attacker profile is relevant as community networks such as [30] are established and the Gateway trust boundary interacts with confidential data.

4. **Remote Attacker:** operates within the IP network and has the capability to target LoRaWAN components through IP-based attack vectors.

The graphic threat model is shown in Figure 2.4

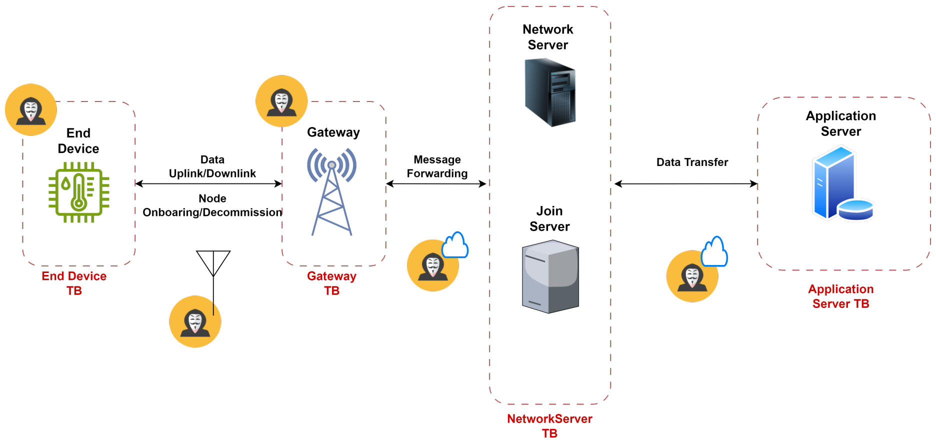


Figure 2.4: LoRaWAN Threat Model

2.4.1.4 Threat Identification

This subsection will identify threats with a focus on RF threats, building upon the previous threat model.

1. **Threats associated with the Physical Attacker Between ED and GW profile**

    - **Physical Exploitation of the End Device:** This involves direct tampering with the hardware of the End Device (ED) to access keys, firmware, or sensitive data stored in its memory. Attackers may also exploit debugging interfaces to manipulate the device or extract information.
    - **Firmware Modification:** Unauthorized changes to the ED's firmware can introduce malicious functionalities or create backdoors for persistent access and control.
    - **Side-Channel Attacks:** These exploit indirect information, such as power consumption or electromagnetic emissions from devices, to extract sensitive information like cryptographic keys, without direct access to the device's data or internal processes.

2. **Threats associated with the RF Attacker Between ED and GW profile**

    - **Eavesdropping:** Attackers capture the data transmitted between the End Device (ED) and the gateway, potentially accessing confidential information.

- **Jamming:** This disrupts the communication link between the ED and the gateway. The ED may incorrectly believe that the data was successfully transmitted or may be prevented from sending the data entirely.
- **Spoofing:** Malicious data is transmitted to the gateway as if they were from a legitimate ED, potentially leading to unauthorized actions or access.
- **Gateway or ED Overload:** By overwhelming the network with excessive traffic, attackers can cause a Denial of Service (DoS). This may particularly target the ED, leading to its malfunction.
- **Replay Attacks:** An attacker captures legitimate messages and replays them to cause confusion or unauthorized actions within the network.

3. **Threats associated with the Network Attacker Between the GW and Network/Join Server profile**

    - **Exploitation of the End Device Onboarding Process:** Attackers may intercept the onboarding process, causing the ED to join a rogue application or preventing it from joining the intended network.
    - **Man-in-the-Middle (MitM) Attacks:** These involve intercepting communication between devices, enabling the attacker to eavesdrop, tamper with data, spoof identities, or reduce the availability of the network.
    - **Rogue Device (Cloning):** Attackers clone an ED to replicate its identity and credentials, allowing unauthorized access and potential data breaches within the network.

4. **Threats associated with the Network Attacker Between the Network/Join Server and Application Server profile**

    - **Man-in-the-Middle (MitM) Attacks:** Similar to the attacks between the GW and network/join server. These involve intercepting communications to eavesdrop, alter data, impersonate devices or servers, and potentially disrupt the service's availability.

2.4.2 LoRaWAN Security Analysis

This section will provide a comprehensive analysis of LoRaWAN security. It will address the threats identified in the previous section and assess whether they have been sufficiently mitigated or not. The persisting vulnerabilities are identified, and their exploitation feasibility using SDR technology will be assessed.

2.4.2.1 Generation of Session Context

Firstly, the generation of the session context will be examined. The generation of session context is a process of preparing the context between the end device and the network/application servers to enable secure communication. It is important to distinguish between LoRaWAN version 1.0 and 1.1, as the newer version enables some additional security features.

Generation of session context can be performed in two ways [31]:

- **OTAA (Over-the-Air Activation)** – is the preferred and more secure method for connecting devices to a LoRaWAN network. In this method, each device is pre-configured with a unique device identifier (DevEUI), an application identifier (AppEUI), and an application key (AppKey). When a device attempts to join the network for the first time, it performs a join procedure that will be described below.
- **ABP (Activation by Personalization)** – is a simpler but less secure activation method. In ABP, devices are pre-configured with static session keys before deployment: a network session key (NwkSKey) and an application session key (AppSKey), along with a device address (DevAddr).

OTAA in LoRaWANv1.0
The End Device is pre-configured with the following elements:

- **DevEUI**: A globally unique IEEE EUI-64 identifier that serves as the device's unique identifier. This information is public.
- **AppEUI**: A globally unique IEEE EUI-64 identifier associated with the application to which the device is connecting to. This information is public.
- **AppKey**: A 128-bit AES key used for encrypting join accept messages and deriving session keys. This key is known only to the device and the application server. This key is considered a secret and shall never be transmitted over the network.

The context generation occurs as follows:

1. ED generates a Device Nonce, a 2-byte random number used once to ensure the uniqueness of each join request.
2. ED constructs a Join Request message containing its DevEUI, AppEUI, and the Device Nonce. This message is not encrypted. The Message Integrity Code (MIC) is computed and added to the Join Request message with the AppKey.
3. The Join Request is sent to the Network Server.
4. The Network Server verifies whether the given ED has permission to join the network based on its DevEUI and MIC validation.
5. The Network Server responds with Join Accept message containing:
   - **AppNonce** - random number generated by the NS (3 bytes)
   - **NetID** (Network ID)
   - **DevAddr** - End Device address assigned by the NS

- **Other security unrelated elements** (e.g. downlink message instructions)

The MIC is computed over all presented values using AppKey and appended to the Join Accept message for integrity protection. Further, the Join Accept is encrypted with AppKey using AES-Decrypt operation in ECB mode [32].

6. The End Device (ED) and Network Server (NS) can now use identical values for DevNonce, AppNonce, and NetID to generate two distinct session security keys – the NwkSKey and the AppSKey by deriving from the AppKey. The AES128-Encrypt [32] function is used to derive these session keys. The processes to calculate both the AppSKey and the NwkSKey is done in the following way:

   NwkSKey=AES128_Encrypt(AppKey,0x01|AppNonce|NetID|DevNonce)
   AppSKey=AES128_Encrypt(AppKey,0x02|AppNonce|NetID|DevNonce)

7. The AppsKey is distributed to the Application Server over secure IP channel.

After successful OTAA activation, the communication shall begin with the generated context:

- **DevAddr** - a 4B address assigned by the Network Server to identify the activated device
- **NwkSKey** - the Network Session Key is used for encrypting MAC layer messages and MIC computation to provide confidentiality and integrity of messages used for secure communication
- **AppSKey** - the Application Session Key is used to encrypt and decrypt application data to provide end-to-end confidentiality between the End Device and the Application Server
- **FCntUp** - Additional frame counter incremented by the ED for each message sent. The NS must verify that new messages have this counter greater or equal that the previously received to prevent replay attacks in **uplink**.
- **FCntDown** - Additional frame counter incremented by the NS for each message sent. The ED must verify that new messages have this counter greater or equal that the previously received to prevent replay attacks in **downlink**.

ABP in LoRaWANv1.0

In case of Activation by Personalization in LoRaWANv1.0, the DevAddr, the NwSKey and the AppSKey are pre-configured in the device from the very beginning. The NwSKey and DevAddr is in possession of the Network Server and the AppSKey is in possession of Application Server. With this preconfigured setup, the identical can start in a similar way to the OTAA method.

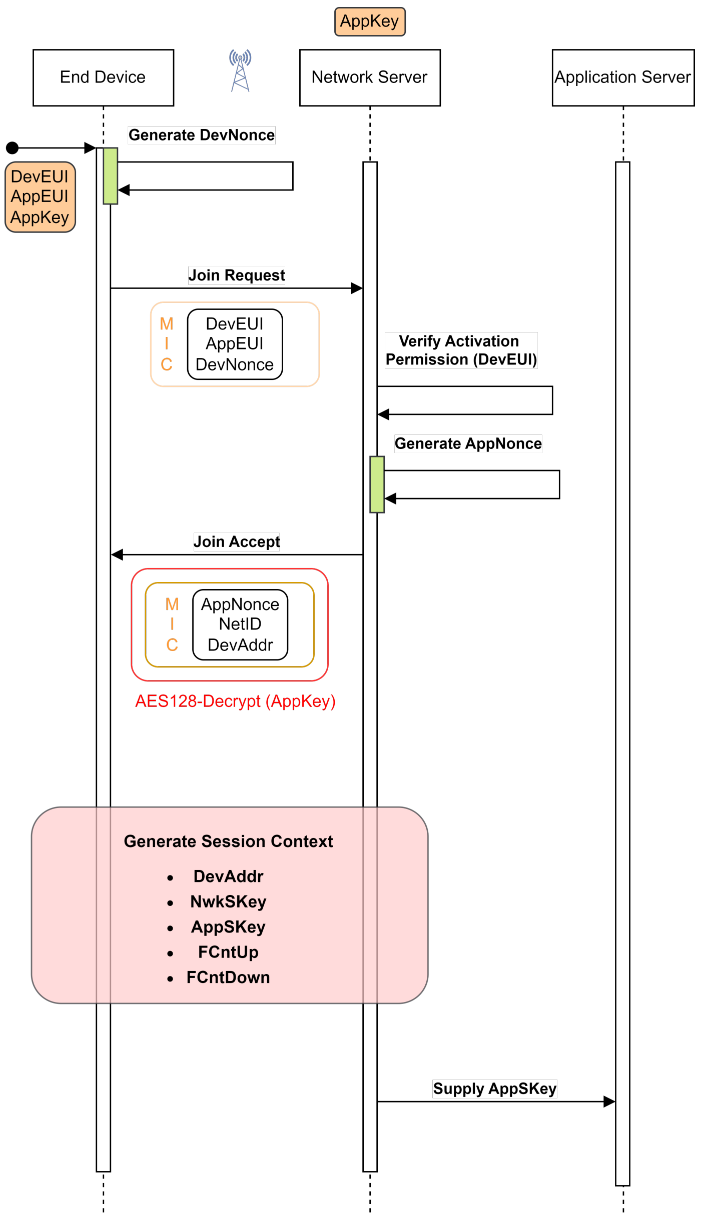


Figure 2.5: LoRaWANv1.0 Over-the-Air Activation

OTAA in LoRaWANv1.1

In LoRaWANv1.1, the Join Server is responsible for the device onboarding process.
The End Device is pre-configured with the following elements:

- **DevEUI**: A globally unique IEEE EUI-64 identifier that serves as the device's unique identifier. This information is public.
- **JoinEUI**: A globally unique IEEE EUI-64 identifier associated with the

Join Server that the ED shall use for activation. This information is public.

- **AppKey**: A 128-bit AES key used for deriving session key for secure communication with the Application Server. This key is known only to the device and the Application Server. This key is considered a secret and shall never be transmitted over the network.
- **NwkKey**: A 128-bit AES key used for deriving session keys for secure communication with the Network Server. This key is known only to the device and the application server. This key is considered a secret and shall never be transmitted over the network.

The context generation occurs as follows:

1. ED generates a Device Nonce, a 2-byte incremental counter used once to ensure the uniqueness of each join request.
2. ED constructs a Join Request message containing its DevEUI, JoinEUI, and the Device Nonce. This message is not encrypted. The Message Integrity Code (MIC) is computed using the *NwkKey* and added to the Join Request message.
3. The Join Request is sent to the Network Server.
4. Network Server passes the Join Request message to the Join Server identified by the JoinEUI.
5. The Join Server verifies whether the given ED has permission to join the network based on its DevEUI. If not, no response is provided back to the ED.
6. The Join Server responds with Join Accept message containing:
   - **JoinNonce** - incremental counter number generated by the JS (1 byte)
   - **NetID** - Network ID (3 bytes)
   - **DevAddr** - End Device address assigned by the JS (4 bytes)
   - **Other security unrelated elements** (e.g. downlink message instructions or RF channels) The MIC is computed over all presented values using NwkKey and appended to the Join Accept message for integrity protection.
     Further, Join Accept is encrypted with NwkKey using AES-Decrypt operation in ECB mode [32].
7. The End Device (ED) and Join Server (NS) can now use identical values for DevNonce, JoinNonce, and NetID to generate four distinct session security keys – the AppSKey, FNwkSIntKey, SNwkSIntKey and the NwkSEncKey by deriving from the AppKey.

   AppSKey=AES128_Encrypt(AppKey,0x02|JoinNonce|JoinEUI|DevNonce)
   FNwkSIntKey=AES128_Encrypt(NwkKey,0x01|JoinNonce|JoinEUI|DevNonce)

SNwkSIntKey=AES128_Encrypt(NwkKey,0x03|JoinNonce|JoinEUI|DevNonce)
NwkSEncKey=AES128_Encrypt(NwkKey,0x04|JoinNonce|JoinEUI|DevNonce)
[9, p. 101829].

Note that different keys are used for message integrity code calculation (FNwkSIntKey and SNwkSIntKey). This is due to the roaming feature in LoRaWANv1.1 that enables the devices to re-join the network within different network operators. Different network operators can use different gateways that are used to communicate with different network server. In this case, the ED communicates with the network operator's Forward Server that forwards the messages to the home-network Network Server through Serving Network Server.

After successful OTAA activation, the communication shall begin with the generated context:

- **DevAddr** - a 4B address assigned by the Join Server to identify the activated device.
- **FNwkSIntKey** - a Network Session Key for partial MIC computation of uplink data messages to secure integrity. With this key, the second 2 bytes of MIC are calculated.
- **SNwkSIntKey** - a Network Session Key for the partial MIC computation of uplink data messages and full MIC computation of downlink data messages to secure message integrity. With this key, the first two MIC bytes are calculated in case of the uplink message.
- **NwkSEncKey** - a Network Session Key for encryption and decryption of MAC commands of the uplink and downlink data messages to secure message confidentiality. This key is further distributed to the Network Server by the Join Server.
- **AppSKey** - a session key used by both the Application Server and the End Device to encrypt and decrypt the application data to secure end-to-end confidentiality. This key is further distributed to the Application Server by the Join Server.
- **FCntUp** - uplink frame counter (for both the network and application contexts) [33]
- **NFCntDown** - downlink frame counter for network context [33]
- **AFCntDown** - downlink frame counter for application context [33]

ABP in LoRaWANv1.1

In case of Activation by Personalization in LoRaWANv1.1, the DevAddr and the session key set (FNwkSIntKey, SNwkSIntKey, NwkSEncKey, and AppSKey) are pre-configured in the device from the very beginning. The DevAddr and network context related session keys (FNwkSIntKey, SNwkSIntKey, NwkSEncKey) is in possession of the Network Server and the AppSKey is in possession of Application Server. With this preconfigured setup, the identical can start in a similar way to the OTAA method.

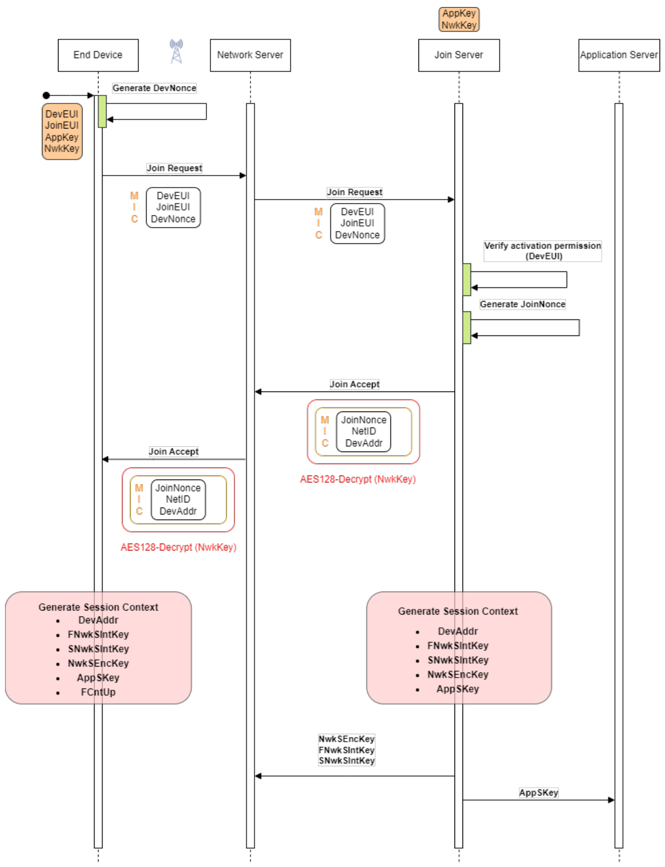


Figure 2.6: LoRaWANv1.1 Over-the-Air Activation

2.4.2.2 Secure Communication

This subsection describes how secure communication in LoRaWANv1.0 and LoRaWANv1.1 occurs with a generated session context as described above.

In LoRaWAN the fundamental security is provided by the security keys generated during the context generation. LoRaWANv1.0 utilizes two layers of security: one for the network (NwkSKey) and one for the application (AppSKey). Each layer uses its own set of keys to secure communication:

- **NwkSKey** is used for securing messages at the network level. It is used for insurance of MAC (Media Access Control) payloads and is responsible for ensuring the authenticity of the device in the network. This key is used for operations such as MIC (Message Integrity Code) generation and validation, ensuring that the message has not been tampered with during transit.

- **AppSKey** is used for encrypting and decrypting the application payload. This key ensures that the application data transmitted between the end device and the application server remains confidential and can only be accessed by the intended parties.

In addition to keys, there are several types of frame counters that are used for replay attacks prevention. For further information, kindly refer to the Generation of Session Context section.

Cryptography

- **AES-CTR for Encryption**: AES-CTR is used for encrypting the payload data. In CTR mode, AES encryption is applied to a counter value that is incremented for each block of data, generating a stream of keystream blocks. These blocks are then XORed with the plaintext blocks to produce the ciphertext, or vice versa for decryption. The AppSKey is used with AES-CTR for encrypting and decrypting application payloads. This mode is chosen for its efficiency and suitability for devices with limited processing capabilities, and because it allows for the encryption using only XOR operation.
- **AES-CMAC for Integrity and Authentication**: For generating the Message Integrity Code (MIC), AES-CMAC is utilized. AES-CMAC provides a way to verify the integrity of messages and authenticate the source, using the NwkSKey. It generates a fixed-size output (MIC) that is attached to each message, allowing the receiver to verify that the message has not been tampered with and that it comes from a legitimate source.

LoRaWAN 1.1 introduces several enhancements to the security architecture, aiming to address some of the vulnerabilities identified in version 1.0 and to provide a more secure and robust framework for device-to-network communication. These enhancements include changes to key management, the introduction of additional security keys, and modifications in the encryption and integrity protection mechanisms. The main enhancements are the following:

- **New Key Architecture:** Separate root keys for network (NwkKey) and application (AppKey) layers, enhancing data segregation and security.
- **Multiple Network Session Keys:** Introduction of FNwkSIntKey, SNwkSIntKey, and NwkSEncKey for nuanced security controls and specific operations within the network layer.
- **Improved Join Procedure:** A dedicated Join Server centralizes and secures the device join process, using encrypted messages for safer key exchanges.
- **Frame Counter Enhancements:** Implementation of a 32-bit frame counter to better protect against replay attacks.
- **Acknowledgment Checks:** Devices must receive acknowledgments for downlink messages, ensuring synchronization and enhancing replay attack protection.
- **MIC Enhancements:** Use of direction-specific and layer-specific keys for computing MICs, strengthening message integrity and authenticity.

- **MAC Layer Encryption:** Encryption of MAC commands within FRMPayload for added confidentiality.

LoRaWAN Message Structure

In LoRaWANv1.0 and LoRaWANv1.1, several message types can be sent over the network. Some of them have already been described in the Generation of Session context section. Find the overview of all message types in accordance with [34] below.

1. **Join-request:** An uplink message utilized in the over-the-air activation (OTAA) process, facilitating devices to initiate communication with the network.
2. **Join-accept:** A downlink message deployed during the OTAA procedure, whereby the network acknowledges and accepts a device's request to join.
3. **Data Up and Down (Unconfirmed/Confirmed):**
   - **Unconfirmed Data Up/Down:** These messages contain data frames transmitted either uplink or downlink, respectively, where no acknowledgment is required from the receiving end. They are used for scenarios where data delivery confirmation is not critical.
   - **Confirmed Data Up/Down:** In contrast, these messages involve data frames sent uplink or downlink, respectively, with a request for confirmation from the receiver. This mechanism ensures data integrity and delivery acknowledgment, essential for critical data transmissions.
4. **RFU (Reserved for Future Use)/Rejoin-request:** Initially reserved in v1.0. In v1.1, this category now contains uplink messages specifically for over-the-air activation (OTAA) Rejoin-requests, introduced in version 1.1 to enhance device reconnection procedures.
5. **Proprietary:** This category is designated for non-standard message formats that enable the implementation of custom protocols or features beyond the predefined LoRaWAN specifications.

The message structure of Join Request and Join Accept message types have already been described in previous section. The structure of Data messages is described below in accordance with [34] and [35].

Physical Layer

- **Preamble**: The initial part of the message, preparing the receiver for incoming data.
- **Physical Header (PHDR)**: Specifies the format and properties of the message.
- **Physical Header CRC (PHDR_CRC)**: A cyclic redundancy check for the PHDR to ensure its integrity.

- **Physical Payload (PHYPayload)**: The main content of the message.
    - For uplink messages, an additional **CRC field** is appended to ensure data integrity.

MAC Layer (Contained within PHYPayload)

- **MAC Header**: Indicates the message type and the MAC major version. All current LoRaWAN versions use a value of 0, corresponding to LoRaWAN release 1.x.
- **MAC Payload**: The core content within the MAC layer, varies based on message type.
- **Message Integrity Code (MIC)**: A 4-byte code for validating the message's integrity.
- **For join-request and join-accept messages**: Consists of application and device identifiers, plus a random number (DevNonce) for the join-request.
    - **AppEUI**: Unique identifier of an application.
    - **DevEUI**: Unique identifier of an End Device (ED).
    - **DevNonce**: A random number used during the join process.
- **For other message types**: Includes a frame header (FHDR) and may include optional fields such as the port field (Fport) and frame payload (FRMPayload).
    - **Frame Header (FHDR)**: Contains the device address (DevAddr), frame control (FCtrl), a 2-byte frame counter (FCnt), and up to 15 bytes of frame options (FOpts) for MAC commands.
    - **Optional Fields**:
        * **Fport**: Specifies the application port (0 for MAC commands, 1-223 for application data).
        * **FRMPayload**: Carries the application-specific data or MAC command.

Capacity Limits

- **MACPayload Maximum Length (M)**: Region-specific cap on the MACPayload size. For example, in Europe, it can be up to 250 bytes.
- **FRMPayload Maximum Size (N)**: Also region-specific, with a maximum of 242 bytes in Europe.

The following figures depict how secure communication of uplink and downlink messages occur in LoRaWANv1.0 and LoRaWANv1.1. The displayed messages contain both the application data for the Application Server or MAC commands for the Network Server.

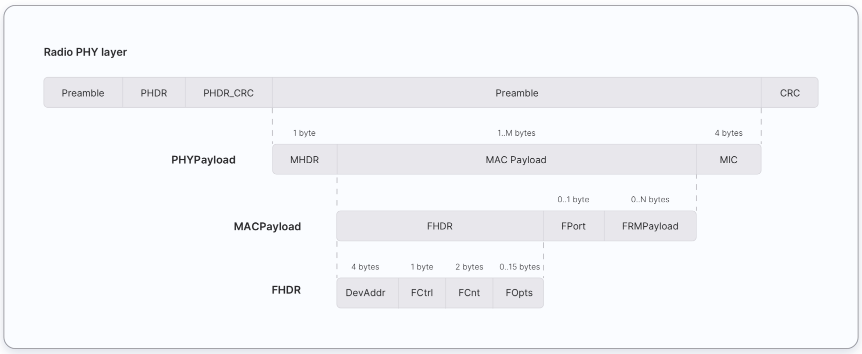


Figure 2.7: LoRaWAN Message Structure [34]

2.5 LoRaWAN Vulnerabilities

Even though the LoRaWAN protocol was created with a security by design mindset, it is not invulnerable to security risks. As technology progresses and becomes more universally adopted, it often uncovers certain flaws that might have been overlooked in the initial threat evaluations. These security gaps have been documented in multiple papers. This section is dedicated to systematically outlining these vulnerabilities, showing their characteristics, the possible effects they could have on the network, the specific LoRaWAN versions that are susceptible, and the preventive measures that have been integrated into later protocol versions to lessen these vulnerabilities.

- **Jamming Attack**
  - **Description:** Jamming in LoRaWAN takes advantage of the fact that while Chirp Spread Spectrum (CSS) is inherently resistant to interference, two devices can still cause significant interference to each other if they transmit using the same frequency and spreading factor. This can be exploited by a jammer intentionally transmitting messages with these parameters, effectively creating a denial of service. For more information about jamming LoRa signals, see [36].
  - **Impact:** Results in a denial of service (DoS) in the radio frequency (RF) part of the communication and can facilitate other attack vectors when jamming is employed strategically.
  - **Affected Versions:** LoRaWAN 1.0 or 1.1.
  - **Countermeasures:** The primary mitigation strategy involves investigating abnormal transmission patterns indicative of jamming and employing anomaly detection systems.
- **Physical Attacks on End Devices/Gateways**
  - **Description:** This category of attacks involves physical tampering with the end devices (ED) or gateways (GW). Tactics include stealing, removing, or destroying the hardware; extracting sensitive security parameters; or altering the device's firmware.

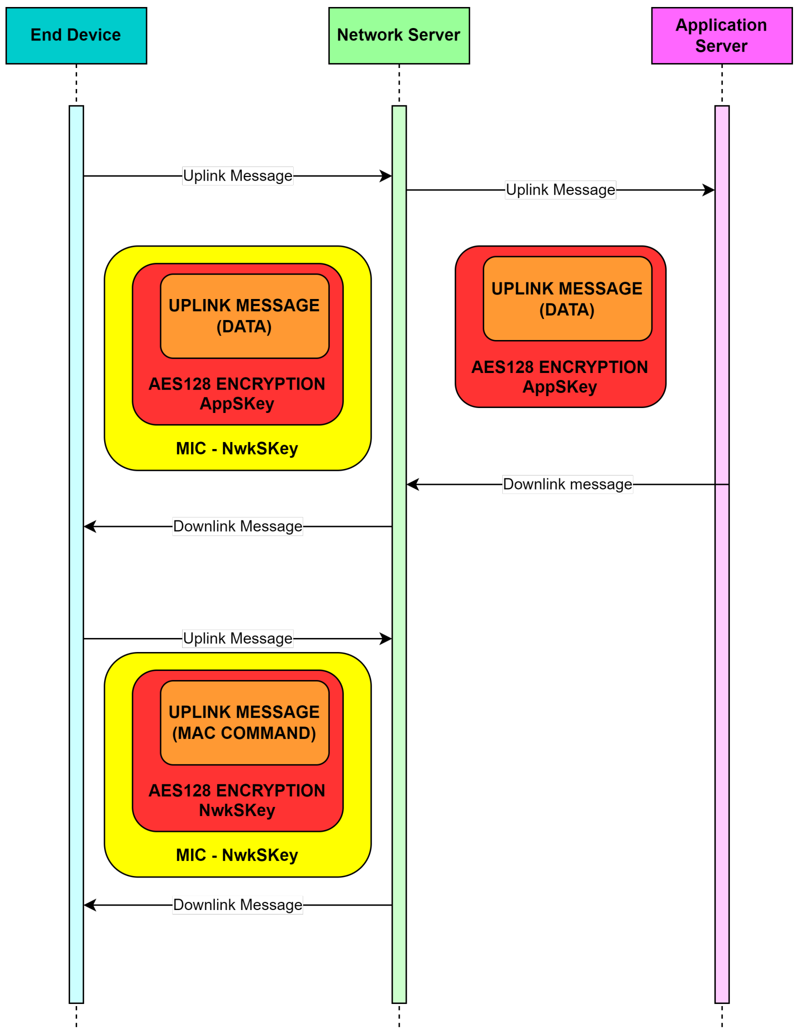


Figure 2.8: Communication in LoRaWANv1.0

- **Impact:** Compromises device availability and integrity and could potentially lead to the unauthorized extraction and misuse of confidential information or technology.
- **Affected Versions:** LoRaWAN 1.0 or 1.1.
- **Countermeasures:** Protective measures include the use of Hardware Security Modules (HSM), firmware encryption, and secure physical security protocols.

- **Lack of Device Decommissioning Process**
  - **Description:** LoRaWAN protocol does not define a standard procedure for the decommissioning of devices. This omission means that previously used device addresses and IDs might not be reassigned securely. This threat was described in [37].

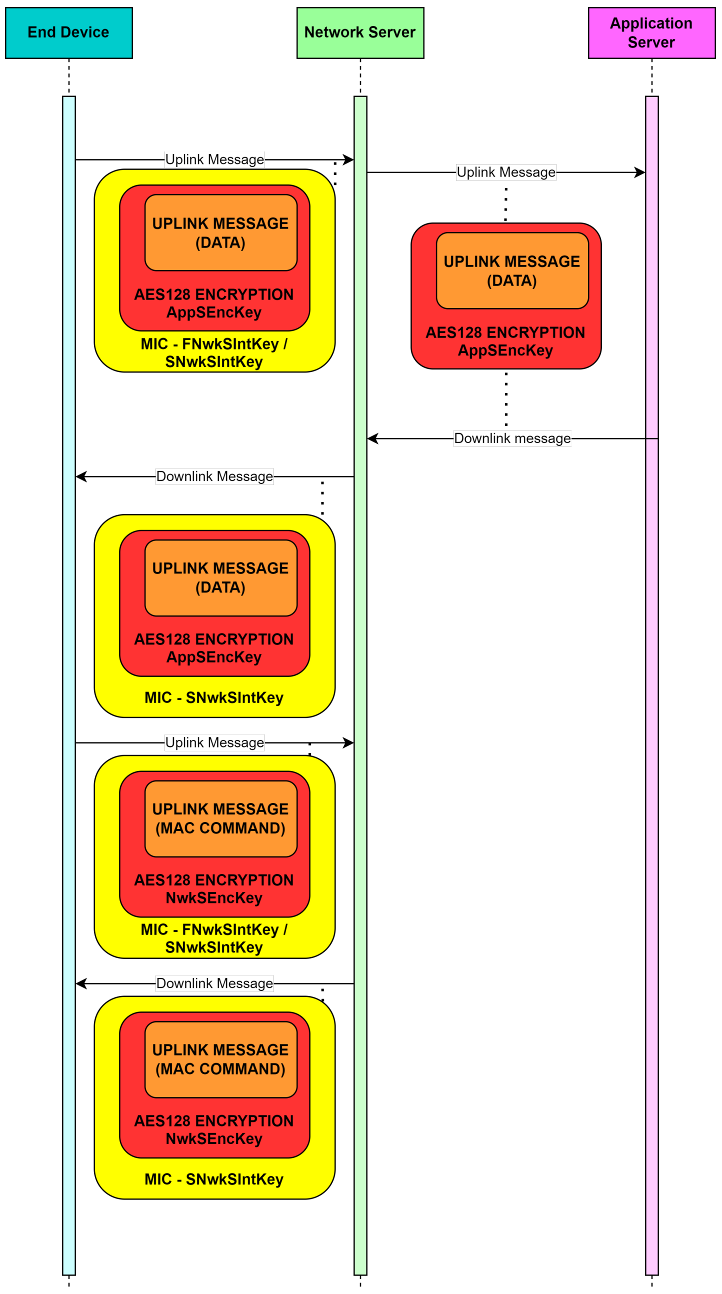

End Device
Network Server
Application Server
Uplink Message
UPLINK MESSAGE (DATA)
AES128 ENCRYPTION AppSEncKey
MIC - FNwkSIntKey / SNwkSIntKey
Uplink Message
UPLINK MESSAGE (DATA)
AES128 ENCRYPTION AppSEncKey
Downlink message
Downlink Message
UPLINK MESSAGE (DATA)
AES128 ENCRYPTION AppSEncKey
MIC - SNwkSIntKey
Uplink Message
UPLINK MESSAGE (MAC COMMAND)
AES128 ENCRYPTION NwkSEncKey
MIC - FNwkSIntKey / SNwkSIntKey
Downlink Message
UPLINK MESSAGE (MAC COMMAND)
AES128 ENCRYPTION NwkSEncKey
MIC - SNwkSIntKey

Figure 2.9: Communication in LoRaWANv1.1

- **Impact:** Potentially allows for the reuse of device addresses in a malicious manner if no custom implementation is provided.
- **Affected Versions:** LoRaWAN 1.0 or 1.1.
- **Countermeasures:** Implement a custom decommissioning process to manage device retirement securely.

- **Downlink Routing Vulnerability**
  - **Description:** When an end device sends an uplink message, it is broadcast to all gateways in range. However, downlink messages are sent to only one chosen gateway. If an attacker captures the uplink message and transmits it from an alternate location, the network server may select the attacker's gateway for the downlink route. This vulnerability is described in [38] and [39].
  - **Impact:** Can be exploited to create a denial of service on downlink communications or to hijack downlink traffic.
  - **Affected Versions:** LoRaWAN 1.0 or 1.1.
  - **Countermeasures:** Utilize advanced security measures to detect and mitigate relay attacks.
- **Missing End-to-End MIC Verification**
  - **Description:** The Message Integrity Code (MIC) is verified between the end device and the network server but not between the network server and the application server. This process relies on the transport layer security, which may not provide sufficient integrity checks. This threat is described in [37].
  - **Impact:** Leaves the system vulnerable to integrity corruption and undermines the security of communications between network and application servers.
  - **Affected Versions:** LoRaWAN 1.0 or 1.1.
  - **Countermeasures:** Implement end-to-end integrity checks beyond transport layer security.
- **Weak DevNonce Length**
  - **Description:** In LoRaWAN, the DevNonce is a short identifier intended to make each join request unique. It's only 2 bytes long, which inherently limits its range. The network server is responsible for remembering DevNonces to prevent join request replays. If it remembers too few, an attacker can reuse an old DevNonce, causing the network server to mistakenly initiate a new session. Conversely, if it remembers too many, it may reject legitimate requests from devices that have exhausted the DevNonce range, leading to a potential lockout situation. For further reference, see [40].
  - **Impact:** If the network server's DevNonce list is not managed correctly, attackers could cause a denial of service by submitting requests with previously used DevNonces. In addition, legitimate devices could be denied access if all possible DevNonces are used up and the server fails to forget old ones.
  - **Affected Versions:** LoRaWAN 1.0.
  - **Countermeasures:** LoRaWAN version 1.1 addresses this by implementing an incremental DevNonce, which reduces the likelihood of reuse

- **No AppNonce Reuse Check (Join Accept Spoofing)**
  - **Description:** In LoRaWAN's establishment of secure communication between the end device (ED) and the network server (NS) is the exchange of the JoinAccept message. This message includes an AppNonce, a nonce value used once during the setup to generate session keys. However, the protocol does not mandate the ED to verify if an AppNonce has been used before, leading to a vulnerability. If an attacker replays an old JoinAccept message, and the ED does not check for AppNonce reuse, the ED and NS may compute different encryption keys. While both believe a secure session has been established, the mismatched keys render them incapable of proper encryption and decryption of communications. For further reference, see [40].
  - **Impact:** The absence of AppNonce reuse checks can result in the ED and NS having desynchronized session keys, which effectively prevents any encrypted communication between them. This can lead to a denial of service, where legitimate messages are rejected or lost due to encryption key mismatches.
  - **Countermeasures:** To strengthen this aspect of the protocol, subsequent versions of LoRaWAN should ensure that both join requests and accepts are tied to a unique identifier, preventing the replay of a JoinAccept message. A mechanism to check and prevent AppNonce reuse would also be crucial, mitigating the risk of such replay attacks. Implementing these measures would ensure that each session setup is indeed unique and secure.
- **Frame Counter Reset without Re-Keying**
  - **Description:** LoRaWAN uses two frame counters, FCntUp for uplink and FCntDown for downlink messages, which are essential for maintaining the sequence of messages. These counters are also part of the input to the AES-CTR encryption algorithm that secures communications. However, if these counters are reset to zero without simultaneously changing the encryption keys, it creates a vulnerability. An attacker could take advantage of this by intercepting and storing messages with a high frame counter value, then replaying them after the counter reset. The network server might accept these replayed messages as valid if it cannot distinguish them from new messages. For further reference, see [40].
  - **Impact:** This vulnerability allows for two potential attacks. Firstly, an attacker can replay old messages, causing confusion and potentially incorrect actions as the server processes outdated or incorrect information. Secondly, a DoS condition arises for legitimate messages that now have a lower frame counter value than the replayed messages, leading the network server to reject them as out-of-sequence.
  - **Affected Versions:** LoRaWAN 1.0, particularly in scenarios where counters may reset, such as with Activation By Personalization (ABP) or counter overflows in Over-The-Air Activation (OTAA).

  - **Countermeasures:** A solution implemented in LoRaWAN 1.1 is to ensure that frame counters are stored in a way that they do not simply reset (e.g., in non-volatile memory), and a key renewal process is mandatory whenever counters are reset. This preventive measure ensures continuity in the sequence of frame counters and maintains the integrity of encrypted message streams.

- **ACK Spoofing Vulnerability**
  - **Description:** The network server (NS) sends an ACK message to confirm the receipt of a message from an end device. However, this ACK is not specific to the message it confirms. An attacker can exploit this by recording the ACK and using it to spoof the confirmation for another message. For this attack to be successful, the attacker must jam the gateway downlink to prevent the genuine ACK from reaching the end device. The recorded ACK can then be played back to the end device to falsely confirm a subsequent message, misleading it to believe the transmission was successful. This attack is described in detail in [41].
  - **Impact:** Since the end device thinks its message has been acknowledged, it will not retransmit, potentially leading to the loss of critical data. Additionally, if the attacker uses this technique selectively, it can induce a denial of service as legitimate messages may be denied due to the network server's belief that they have already been confirmed. Note that the captured ACK messages can be spoofed only once as it contains a frame counter that prevents the attack from being successful more than one time for each message.
  - **Affected Versions:** This issue primarily affects LoRaWAN version 1.0, where ACK messages are not tied to the frame count (FCnt) of the message being acknowledged.
  - **Countermeasures:** The vulnerability can be mitigated by implementing a mechanism to tie the ACK directly to the message it acknowledges, perhaps through an enhanced message integrity code (MIC) that includes the frame count. Updating to LoRaWAN version 1.1, where such measures are standard, is also a recommended countermeasure.

It has been established that various vulnerabilities exist within the LoRaWAN protocol. The subsequent sections will introduce detailed security testing procedures that have been developed to address these vulnerabilities.

Chapter 3

Software Defined Radio

In this section, Software Defined Radio (SDR) is introduced, explaining its functionalities and its potential uses. Attention is given to evaluating different SDR systems to determine which are the most suitable for the purposes of LPWAN security testing, taking into account factors such as functionality and compatibility with LPWAN protocols.

SDR represents a type of radio that is capable of tuning to any frequency band and supports various modulation and demodulation techniques and standards within a single device through the use of adaptable hardware and software. It offers versatile, upgradable, and durable radio solutions that cater to both military and civilian communication systems. [42]

3.1 SDR Functionality and Components

It is a technology that utilizes a standard hardware platform but relies on software to define its functionality. This software controls aspects such as frequency, bandwidth, modulation, and error correction. Essentially, SDR acts as a digital radio with a programmable core, allowing developers to create diverse applications through software, offering greater flexibility and adaptability compared to traditional fixed-function radios. [43]

The key SDR components as described in [44] are:

- **Antenna**: Captures radio frequency (RF) signals and converts them to electrical signals for processing. Selecting the right antenna is important for effective signal reception and transmission across various frequencies.

- **Radio Frontend**: This section processes the received RF signals using components like amplifiers, filters, mixers, and local oscillators. Its programmability allows adaptation to different frequencies and protocols, typically utilizing a superheterodyne architecture (radio receiver design that uses frequency mixing to convert a received signal to a fixed intermediate frequency (IF)) for tuning to the desired frequency.

- **Analog-to-Digital Converter (ADC) / Digital-to-Analog Converter (DAC)**: Converts analog RF signals into digital data via ADC, and digital to analog via DAC, preparing them for digital processing.
- **Digital Backend**: Here, the digital data undergoes channelization to break the spectrum into manageable frequency channels and sample rate conversion to adjust the digital signal's rate for further processing.
- **User-End Software**: Allows users to select frequency bands, bandwidths, filters, modulation schemes, and manage decryption processes, providing the interface for final user interaction.

3.2 SDR Comparison

SDRs vary in their specifications and capabilities, their cost, performance, and application range. The cost of an SDR often reflects its complexity and capabilities. Frequency range specifies the span of frequencies the device can handle. Bandwidth determines how much of the frequency spectrum can be processed at once, essential for handling high-data-rate communications. Sample rate measures how often the signal is sampled per second, critical for capturing higher frequency signals accurately. Finally, transmission capability distinguishes SDRs that can only receive signals from those that can also transmit.

Comparison summary of the most popular SDRs on the market based on [45] is presented on the following tables. For more detailed information about various SDR parameters, kindly visit [45].

Table 3.1: Specifications of Various SDRs

Model	**Frequency Range**	**Bandwidth**	**Sample Rate**
Generic RTL-SDR Dongles	25 MHz – 1.7 GHz	2.4 MHz	3.2 MSPS
RTL-SDR Dongle v3	24 MHz – 1.7 GHz	3.2 MHz	3.2 MSPS
HackRF	1 MHz – 6 GHz	20 MHz	20 MSPS
LimeSDR Mini v2	10 MHz – 3.5 GHz	40 MHz	30.72 MSPS
BladeRF x40	300 MHz – 3.8 GHz	40 MHz	40 MSPS
LimeSDR	100 kHz – 3.8 GHz	61.44 MHz	61.44 MSPS
Ettus B210	70 MHz – 6 GHz	61.44 MHz	61.44 MSPS

Table 3.2: Transmit Capability of Various SDRs

Model	**Transmit Capability**
Generic RTL-SDR Dongles	No
RTL-SDR Dongle v3	No
HackRF	Yes (Half-Duplex)
LimeSDR Mini v2	Yes (Full Duplex)
BladeRF x40	Yes (Full Duplex)
LimeSDR	Yes (Full Duplex)

Table 3.2 – continued from previous page

Model	Transmit Capability
Ettus B210	Yes (Full Duplex)

3.3 Tooling

Here is a brief overview of of common SDR tooling used by security testing teams including GNU Radio, GQRX, and Universal Radio Hacker.

GNU Radio
This is a free and open-source toolkit for building software radios. GNU Radio provides a flexible framework that includes signal processing blocks to implement software radios. It can be used with readily available low-cost external RF hardware to create software-defined radios, or without hardware in a simulation-like environment to test and develop signal processing algorithms [46].

GQRX
GQRX is a software-defined radio receiver powered by GNU Radio and the Qt graphical toolkit. It is highly user-friendly and supports many of the SDR hardware available on the market. GQRX offers features like a spectrum analyzer and an oscilloscope, and it is capable of receiving and demodulating a broad range of signals [47].

Universal Radio Hacker
URH is a tool for analyzing and manipulating signals. It allows users to record, analyze, and replay radio signals just with a few clicks. Ideal for reverse engineering of radio protocols, URH supports a wide range of common SDRs and is invaluable for both hobbyists and security researchers working with wireless protocols [48].

3.4 Requirements for LPWAN Security Testing

To effectively sniff LPWAN (Low Power Wide Area Network) protocols using Software-Defined Radios (SDRs), certain key specifications must be met:

- **Frequency Range:** Must cover the specific frequencies used by LPWAN technologies, such as 868 MHz for Europe and 915 MHz for North America, to capture relevant signals.
- **Bandwidth:** Should support at least a few kHz of bandwidth, suitable for handling narrowband LPWAN transmissions like LoRa, which typically uses 125 kHz to 500 kHz.
- **Modulation Capabilities:** Must be able to demodulate the specific modulation techniques used by LPWAN protocols, such as the proprietary spread spectrum technique used by LoRa.

- **Software Support:** Requires software capable of decoding the specific LPWAN protocol. This often means using specialized software tools that can demodulate and decode signals from technologies like LoRa, Sigfox, or NB-IoT. These tools may come from the SDR community or be developed internally. The software should provide features for signal analysis, protocol decoding, and potentially automated scripting to handle data collection and analysis tasks.

Software-Defined Radios (SDRs) are effective tools for security testing because they can handle many different wireless protocols. They are particularly useful for sniffing, which involves capturing wireless data traffic to identify potential security vulnerabilities. Additionally, many SDRs can transmit signals, making them capable of performing replay attacks. This feature allows security testers to capture legitimate signals and retransmit them to test how well a system can withstand such attacks.

Chapter 4

LoRaWAN Security Testing Framework

This section outlines a comprehensive security testing framework for LoRaWAN networks, designed to provide structured guidelines and scoping for systematic security testing across various components and interactions within LoRaWAN systems. The framework is intended to assist security professionals in systematically identifying and mitigating potential security vulnerabilities within LoRaWAN deployments.

The guidelines cover various scope areas and testing phases, such as initial reconnaissance and client engagement. This phase defines the boundaries of the security assessment and gathers essential information about the network's architecture. Another area is physical security testing, which assesses the integrity and resilience of hardware components, including end devices and gateways.

Note that while all scope areas are presented with guidelines, the framework focuses particularly on testing RF communication between end devices (ED) and gateways using Software Defined Radio (SDR) technology. These guidelines are based on previous security research and include methodologies for intercepting, analyzing, and potentially exploiting communication channels to assess their robustness against RF attacks.

This framework is structured to provide the comprehensive testing approach outline of the entire LoRaWAN system. It's worth mentioning, however, that the RF testing protocols are particularly exhaustive, whereas other scope areas aren't as extensively addressed due to their exclusion from the scope of this thesis.

4.1 Reconnaissance

Reconnaissance is an initial step in security testing, aimed at gathering key information before the start of the assessment or early in the process. This phase involves collecting a variety of details, either directly from the client—like the importance of system availability—or through more technical channels in cases of gray or white box testing, which include specifics on implementations, firmware files, and version information.

The following points outline the essential information to be gathered during the reconnaissance phase of the security testing process:

- **System Functionality**: What is the general purpose of the system? What functions and services does it provide?
- **Network Topology and Architecture**: Assess the physical placement of end devices and gateways within the LoRaWAN network. Determine whether the network utilizes third-party servers or if it relies on internally managed, proprietary systems. This detail is important for evaluating the trustworthiness of the network server and identifying potential security risks associated with external dependencies.
- **LoRaWAN Version Usage**: Which versions of LoRaWAN are used by the network server and end devices? Are different versions mixed, such as a server on version 1.1 while end devices are on 1.0, and vice versa? This information is important to assess whether certain attacks might be feasible.
- **Device Activation Methods**: Are devices activated using ABP (Activation By Personalization) or OTAA (Over-The-Air Activation)?
- **System Availability**: How critical is the system's availability? What are the implications if devices are stolen or destroyed and consequent Denial of Service?
- **Confidentiality of Technology**: How confidential is the technology or firmware? What are the risks if it is stolen and subjected to reverse engineering? How are devices physically secured against such threats?
- **Message Delivery Criticality**: How critical is the delivery of each message? If critical, are confirmed message deliveries used? Also if critical, focus more on ACK spoofing attack possibility.
- **Protocol Implementation**: Does the network use its own protocol implementation or a third-party implementation? Is there going to be a firmware sample available for the testing?

4.2 Physical Security Testing

Physical security testing focuses on protecting end devices and gateways in LoRaWAN networks. It evaluates whether the security measures are strong enough to prevent unauthorized access, tampering, or damage. The testing should take into account how a compromised device would affect network operations and the risk of exposing confidential technology through theft. These considerations guide the depth and breadth of the security measures implemented. Kindly note that this is a general evaluation of which tests should be performed; specific testing procedures will depend on the concrete devices in use.

- **Interface Security**: Analyze the security measures for any physical ports or interfaces that could be used to alter device configurations or extract data. This might include measures to seal off USB ports or other interfaces.
- **Debug Port Security**: Investigate the security of debug interfaces such as JTAG, or other debug interfaces. Ensure there are physical lockout mechanisms or settings that disable these ports unless specifically enabled via secure internal procedures. Evaluate if authentication mechanisms or encryption are used to protect access to debug commands and data.
- **Cryptographic Key Storage Security**: Assess the mechanisms for securing cryptographic keys, particularly the use of Hardware Security Modules (HSMs) to ensure keys are stored securely and are resilient to physical and logical attacks.
- **Tamper Detection**: Check for the presence of tamper detection technologies that can alert system administrators if physical interference occurs. This includes tamper switches or seals that trigger when unauthorized access is attempted. Evaluate the robustness of the physical casing to resist unauthorized opening or tampering. Assess whether the casings are tamper-evident and if they effectively protect against environmental factors such as water, dust, and extreme temperatures.
- **Access Control**: Review physical access controls to the devices, such as locks, security screws, or controlled access to the installation locations. Assess how these controls prevent unauthorized physical access. Evaluate the impact of stolen or destroyed devices.
- **Power Supply Security**: Evaluate the security of the power supply, especially for devices that are solar-powered or use batteries. Check for mechanisms that secure against power disruption or manipulation.

4.3 Radio-Frequency Testing

This section outlines detailed guidelines for RF testing of communications between end devices (ED) and gateways (GW) within LoRaWAN networks, utilizing Software Defined Radio (SDR) technology. Building on vulnerabilities identified in previous sections, a structured testing procedures designed to evaluate these vulnerabilities are presented. A specialized tool has been developed to support these tests, providing essential utilities such as sniffing, packet capturing, and replaying capabilities.

Throughout these guidelines, specific references will be made to this tool, guiding testers on how to effectively use features like the sniffer utility to capture specific messages. Detailed documentation of the tool and its functionalities will be provided in a subsequent section.

4.3.1 Jamming

- **Attack Description**: Jamming in LoRaWAN involves transmitting radio signals that interfere with the communication between end devices

(ED) and gateways (GW). LoRa uses Chirp Spread Spectrum (CSS), which is resistant to general interference, but effective jamming can occur if two devices transmit on the same frequency and spreading factor (SF). Selective jamming is possible by reading the header of transmitted messages to identify the target device address (DevAddr), allowing the jammer to transmit on the same frequency and SF, making it less detectable.

- **Preconditions**:
  - The attacker must have a device capable of emitting signals at the same frequency and SF as the target LoRaWAN communications.
  - The jamming device should be positioned within a functional range of the target end device or gateway to effectively disrupt communications. Note that **Jamming module of the LoRaAttack tool** can be utilized for testing. For configuration details, follow the next sections.
- **Attack Steps**:
  1. Configure the jamming device to emit noise or signals at the frequency and SF used by the target LoRaWAN device.
  2. For selective jamming, monitor transmissions first, identify packets with the targeted DevAddr, then transmit interference signals.
  3. Position the jamming device within effective range and activate it to disrupt the LoRaWAN communication, monitoring the impact on signal reception and transmission.
- **Impact Evaluation**:
  - Observe and record the effect on data transmission from the end device to the gateway during the jamming.
  - Assess how communication interference affects network performance, including packet loss and the need for data retransmissions.
  - Evaluate the impact based on the resources used and the level of criticality of the system availability recognized during the reconnaissance phase of the testing.
- **Mitigation**:
  - Based on the assessed impact and application, consider implementing channel hopping techniques or running the network devices in protected environment.

4.3.2 Downlink Routing Attack

- **Attack Description**: This attack exploits the way LoRaWAN handles uplink and downlink messages. When an end device (ED) sends uplink data, it broadcasts to all gateways within range. However, the downlink response from the network server (NS) is sent to only one gateway, selected from a database. An attacker can capture the uplink data and replay it near a different gateway. If the network server selects this second gateway for the downlink message, it can result in misrouted responses.

- **Preconditions**:
  - Access to at least two gateways involved, which are not within range of each other.
  - The attacker must have the capability to jam signals from the ED to the original gateway, can be done using an RF shielded box.
  - The attacker must be able to replay messages, can be done using LoRaAttack tool's replay module.
  - The network must utilize downlink messages as part of its communication protocol.
- **Attack Steps**:
  1. Allow the ED to send an uplink data message.
  2. Capture this message using the LoRaAttack tool's sniffer module while simultaneously preventing it from reaching the intended gateway, either by jamming or using an RF shielded box.
  3. Replay the captured message to the second gateway, which is out of range from the first.
  4. Monitor the network to see if the downlink messages are routed through the original gateway or the one where the attacker replayed the uplink message.
- **Impact Evaluation**:
  - Determine whether the downlink message is misrouted to the second gateway.
  - Evaluate the impact of temporary Denial of Service of downlink messages. Note that the attack might be difficult to discover as uplink data flow is not affected.
- **Mitigation**:
  - Implement a strategy to send downlink messages through multiple gateways that have previously been used by the end device, to reduce the risk of successful routing attacks.

4.3.3 Weak DevNonce Length Exploitation

- **Attack Description**: In LoRaWAN 1.0, the DevNonce, which is only 16 bits long and generated randomly, can create vulnerabilities due to its limited size. The network server (NS) keeps a record of previously used DevNonces to prevent replay attacks. If an already used DevNonce reappears in a Join Request and is not in the NS's memory, the NS will accept it, leading to a potential denial of service as the NS will have incorrect session keys assuming a handshake has been completed.

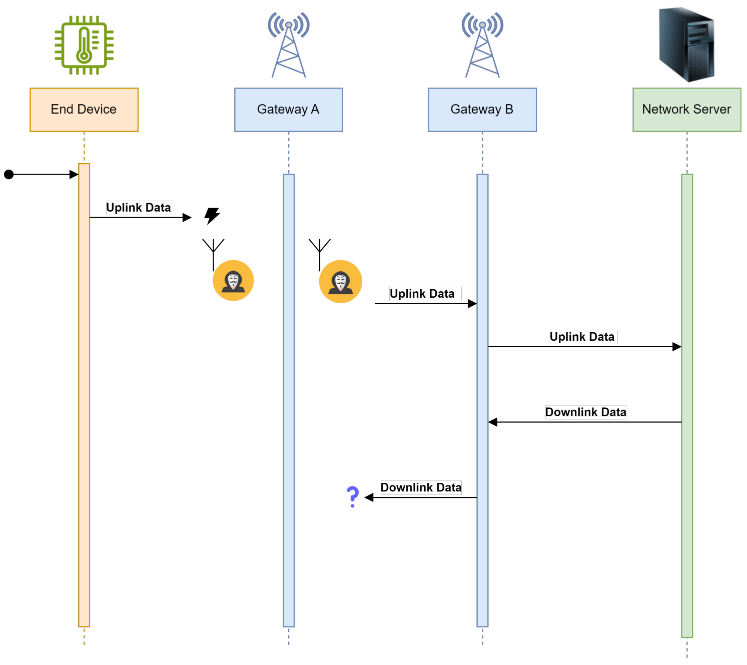


Figure 4.1: Downlink Routing Attack

- **Preconditions**:
  - A functional setup with LoRaWANv1.0 consisting of an end device (ED) and gateway (GW) must be in place.
  - Ability to reset the device or force regeneration of the session context, optional if happens frequently by design, depending on the device configuration.
- **Attack Steps**:
  1. Attempt to determine how many DevNonces the network server retains in its memory.
  2. Force the context regeneration and capture Join Request packets using the LoRaAttack tool's sniffer module.
  3. After a sufficient number of resets, which depends on the estimated number of remembered nonces, transmit another Join Request and prevent it from reaching the gateway using jamming or RF shielding.
  4. Replay a Join Request with a previously used but forgotten DevNonce.
  5. Observe if the network server accepts the Join Request and sends a Join Accept, indicating that the NS and ED believe a session has been established with mismatched keys.
- **Impact Evaluation**:

  - Evaluate the impact of the Denial of Service resulting from mismatched session keys based on the criticality of availability for the system.

- **Mitigation**:
  - Upgrade to a newer version of LoRaWAN, such as 1.1, where DevNonces are generated incrementally.

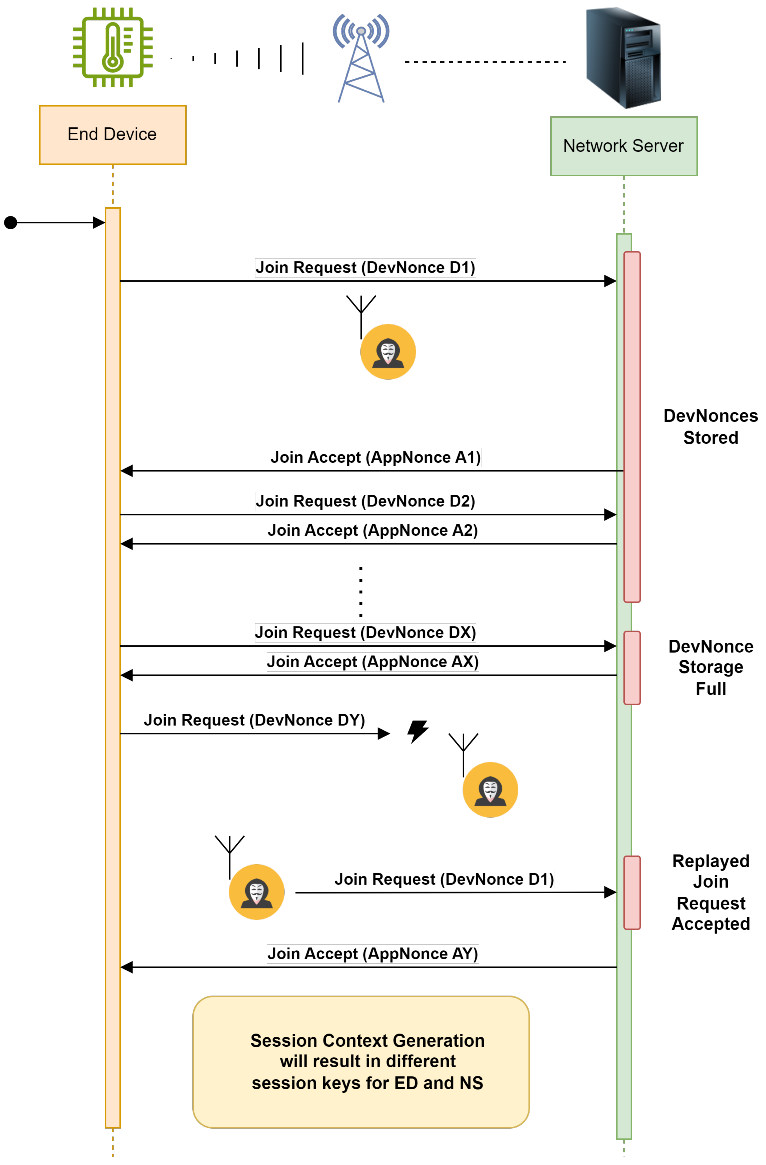

End Device
Network Server
Join Request (DevNonce D1)
DevNonces Stored
Join Accept (AppNonce A1)
Join Request (DevNonce D2)
Join Accept (AppNonce A2)
Join Request (DevNonce DX)
Join Accept (AppNonce AX)
DevNonce Storage Full
Join Request (DevNonce DY)
Join Request (DevNonce D1)
Replayed Join Request Accepted
Join Accept (AppNonce AY)
Session Context Generation will result in different session keys for ED and NS

Figure 4.2: Weak DevNonce Length Exploitation

4.3.4 Join Accept Spoofing

- **Attack Description**: The Join Accept Spoofing attack exploits vulnerabilities in the handling of Join Accept messages within LoRaWAN. If an attacker can replay an old Join Accept message, the legitimate end device (ED) may not verify if the AppNonce has been previously used and will proceed to compute the session keys. Consequently, both the network server (NS) and ED may think a session has been established, but they will actually have mismatched keys, leading to encryption and decryption failures.
- **Preconditions**:
  - A functional setup with LoRaWANv1.0 consisting of an end device (ED) and gateway (GW) must be in place.
  - Attacker has the capability to capture and replay Join Accept messages.
- **Attack Steps**:
  1. Sniff on the communication during the session generation process (can be done using LoRaAttack's sniffer module) and store a valid Join Accept message.
  2. Try resetting the end device or Wait for a new Join Request from the same ED.
  3. Jam the downlink communication or prevent the ED from receiving the Join Accept message using a RF shielding. Replay the old Join Accept message in response to the new Join Request.
  4. Monitor the network to see if the ED and NS proceed with the session setup, indicating they have accepted the replayed Join Accept.
- **Impact Evaluation**:
  - Determine if the network accepts the replayed Join Accept and attempts to establish communication using mismatched keys.
  - Evalueate the impact of Denial of Service caused by the disruption and the potential data loss due to encryption failures.
- **Mitigation**:
  - Upgrade the network to LoRaWANv1.1, where AppNonce reuse check is enforced and AppNonces are generated incrementally.

4.3.5 Frame Counter Reset without Re-Keying

- **Attack Description**: In LoRaWAN, the uplink (FCntUp) and downlink (FCntDown) frame counters are used for creating the keystream for AES-CTR encryption. If these counters are reset to zero without re-keying, the same ciphertexts previously sent will be produced. An attacker can exploit this by recording these ciphertexts and replaying them, which the network server (NS) will accept as legitimate messages due to the unchanged counter values.

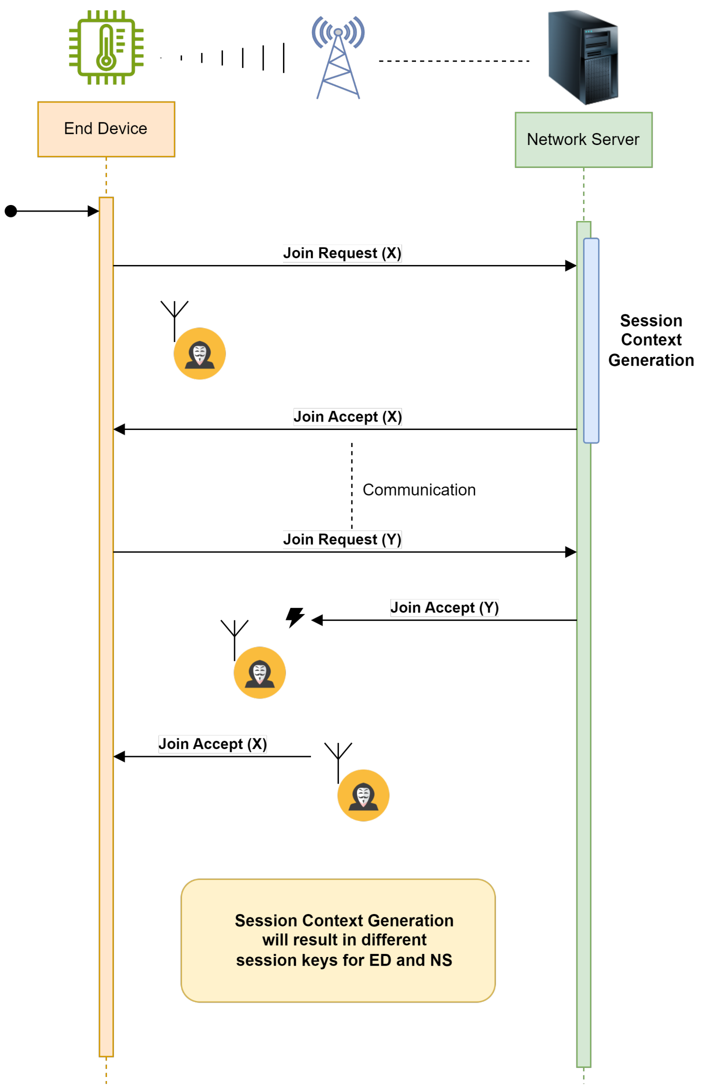


Figure 4.3: Join Accept Spoofing

- **Preconditions**:
  - A functional setup with LoRaWANv1.0 consisting of an end device (ED) and gateway (GW) must be in place.
- **Attack Steps**:
  1. Monitor the network to determine the current frame counter values, capture uplink packets using the LoRaAttack's sniffer module.
  2. Wait for or induce a situation where the counters are reset (in case of ABP activation, the ED can be reset, in case of OTAA, the attacker must wait for overflow).

3. Capture traffic before and after the counter reset, focusing on messages encrypted with the old counters.
4. Replay the captured messages after the reset to see if the NS accepts these as new, valid messages.
5. Replay the captured uplink packet with high frame counter number. The NS should then reject valid messages with lower counter value from the legitimate end device.

- **Impact Evaluation**:
  - **Message Spoofing**: Evaluate the impact of the old spoofed messages acceptance by the Network Server.
  - **Denial of Service (DoS)**: Evaluate the impact of temporary Denial-of-Service where the NS rejects legitimate messages due to the acceptance of a spoofed message with a higher counter.
- **Mitigation**:
  - Upgrade to LoRaWAN v1.1, where frame counters are required to be stored in non-volatile memory (NVM), preventing resets from affecting security.

4.3.6 ACK Spoofing

- **Attack Description**:
  - ACK Spoofing attack exploits the fact that ACK messages from the Network Server (NS) are not related to a specific uplink message. An attacker can capture these ACKs and use them to falsely confirm the receipt of different messages, disrupting communication by preventing the real ACKs from reaching the End Device (ED).
- **Preconditions**:
  - The attacker must be capable of intercepting ACK messages and obstructing the downlink to stop actual ACKs from reaching the ED. LoRaAttack sniffer and replayer modules can be utilized.
- **Attack Steps**:
  1. Sniff on a traffic between the ED and GW and capture an ACK reply to an uplink message using the LoRaAttack sniffer tool.
  2. Jam the communication to prevent the uplink packets and the genuine ACK from reaching the ED and vice versa.
  3. Replay the captured ACK at the time of the ED's next uplink transmission.
- **Impact Evaluation**:
  - Verify if the ED stops retransmissions, resulting in a disruption of service due to believing the message was confirmed. Evaluate the impact of ED believing that an uplink message was delivered successfuly.

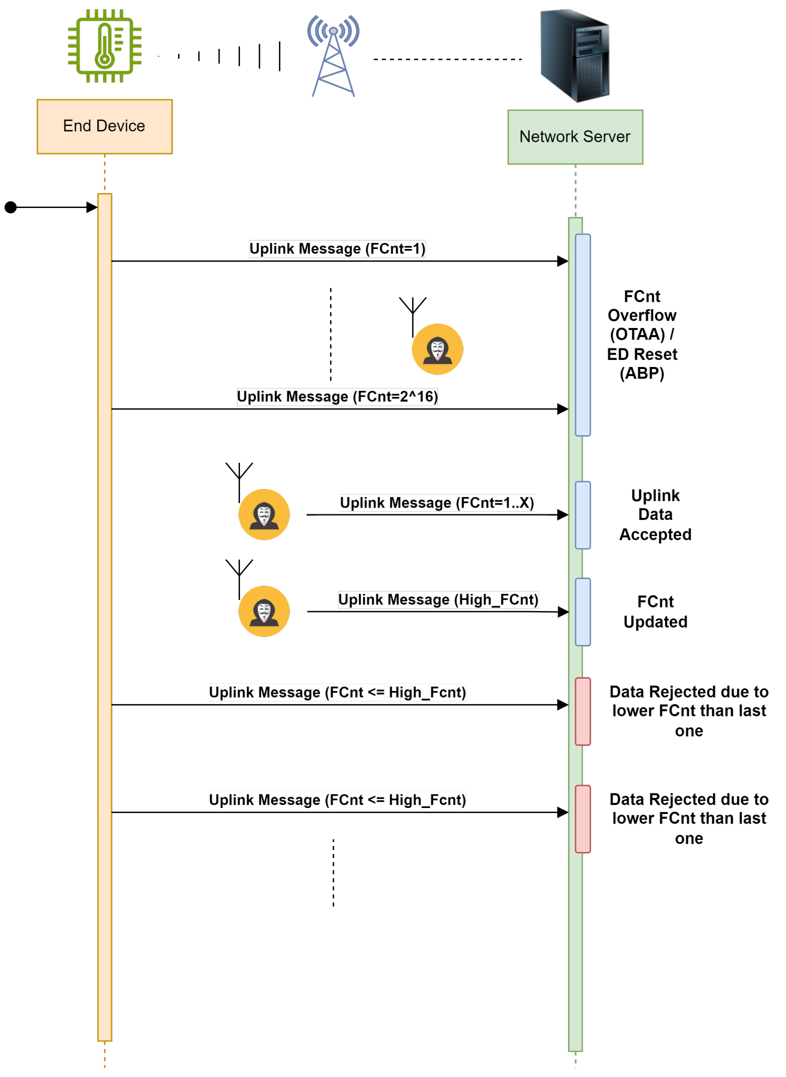

End Device
Network Server
Uplink Message (FCnt=1)
FCnt Overflow (OTAA) / ED Reset (ABP)
Uplink Message (FCnt=2^16)
Uplink Message (FCnt=1..X)
Uplink Data Accepted
Uplink Message (High_FCnt)
FCnt Updated
Uplink Message (FCnt <= High_Fcnt)
Data Rejected due to lower FCnt than last one
Uplink Message (FCnt <= High_Fcnt)
Data Rejected due to lower FCnt than last one

Figure 4.4: Frame Counter Reset without Re-Keying

- **Mitigation**:
  - Upgrade to LoRaWANv1.1, where message contents are included into the MIC calculation.

4.4 Network and Server Security Testing

This section addresses the network security testing of non-RF components within the LoRaWAN environment, emphasizing key aspects such as data confidentiality and data integrity. The primary focus is on ensuring that the network server maintains the confidentiality of sensitive data, particularly in scenarios where it might not be fully trustworthy, as in LoRaWAN 1.0 setups.

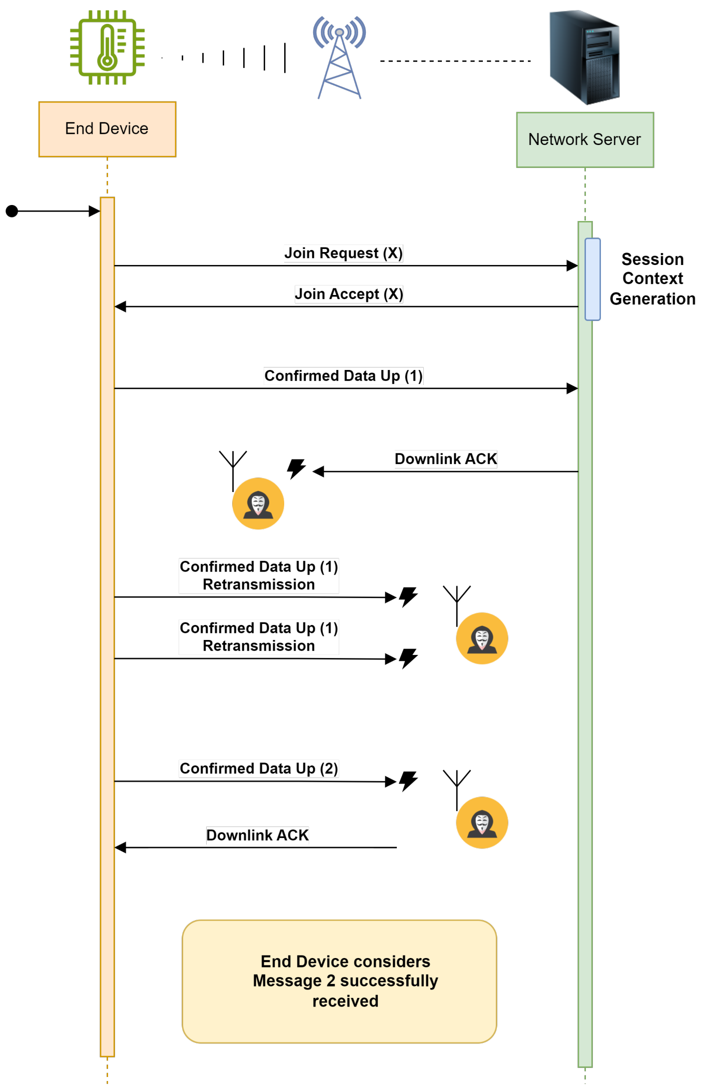

End Device
Network Server
Join Request (X)
Join Accept (X)
Session Context Generation
Confirmed Data Up (1)
Downlink ACK
Confirmed Data Up (1) Retransmission
Confirmed Data Up (1) Retransmission
Confirmed Data Up (2)
Downlink ACK
End Device considers Message 2 successfully received

Figure 4.5: ACK Spoofing

Additionally, this testing seeks to evaluate the integrity of data transmitted between the network server (NS) and the application server (AS), areas critical to preventing data manipulation and ensuring secure communication across the network.

- **Network Security Testing**: Conduct systematic penetration testing of the gateway, network server, and application server to identify and address vulnerabilities.
- **Data Confidentiality Evaluation**: For LoRaWAN 1.0 environments, assess the implications of the network server having access to decrypted data. This is due to the use of the same root key for deriving both the

AppSKey and NwkSKey, enabling the network server to access sensitive information.

- **Communication Security**:
  - **Gateway to Network Server (GW-NS)**: Evaluate the encryption and security protocols used in data transmission between gateways and the network server, focusing on maintaining data integrity and confidentiality.
  - **Network Server to Application Server (NS-AS)**: Examine the integrity protection measures between the network server and the application server. Given that the MIC is not verified between these components and relies on the transport layer security (e.g., TLS), assess:
    - Whether additional security measures are implemented beyond basic transport layer protection.
    - The enforcement and effectiveness of secure protocols such as TLS to safeguard data during transmission.

    Should the integrity protections be inadequate, the network might be vulnerable to bit-flipping attack, where packets communicated between the Network and Application Server are tampered without detection.

Chapter 5

LoRAttack Tool

A specialized tool has been developed to automate and simplify the process of LoRaWAN security testing, making it more accessible and efficient. This tool is designed to complement the testing guidelines presented in the previous sections, providing a practical means to apply those guidelines effectively.

Based on the previous research into the needs of security testing for LoRaWAN, the tool incorporates several the following features:

- **Multi-Channel Sniffing**: Uses Ettus USRP SDR to capture LoRaWAN traffic across various channels. A realtime packet decoding with automatic key derivation is also performed.
- **Session-Based Capture**: Efficiently organizes and stores captured data, including handshakes and communication exchanges, within designated sessions. This feature enhances the management and analysis of collected data, making it easier to track and review network interactions.
- **Key Derivation**: Generates cryptographic keys from captured handshake data, enabling the decryption of communications for a specific session.
- **Wireshark Compatibility**: Ensures that captured data is saved in PCAP files compatible with Wireshark. This compatibility allows security testers to utilize Wireshark's features to conduct in-depth analysis of the network traffic.
- **Vulnerability Testing**: Includes functionalities to replay specific payloads against the network, testing its response to known vulnerabilities and helping pinpoint areas where security improvements are needed. Experimental jamming feature is also implemented.

5.1 Tool architecture

The tool consists of several Python modules, each designed to facilitate different aspects of LoRaWAN security testing:

- **GUI Manager**: Provides a user interface that allows users to interact easily with the tool, managing the setup and execution of SDR operations.
- **Session Manager**: Manages testing sessions, organizing captured handshake data and communications within specific sessions for systematic analysis and retrieval.
- **Sniffer**: Captures LoRaWAN traffic across multiple channels using SDR, essential for monitoring network activity and identifying potential security threats.
- **Player**: Facilitates the replay of captured traffic to test network responses, particularly useful for simulating attacks and testing network vulnerabilities.
- **Analyzer**: Processes the demodulated data from the SDR, employing cryptographic tools for decryption and decoding, providing a thorough analysis of the security posture.
- **Crypto Tools**: Implements cryptographic functions to decrypt and analyze the security of network communications, supporting the other modules. This module is not directly accessible by the user.

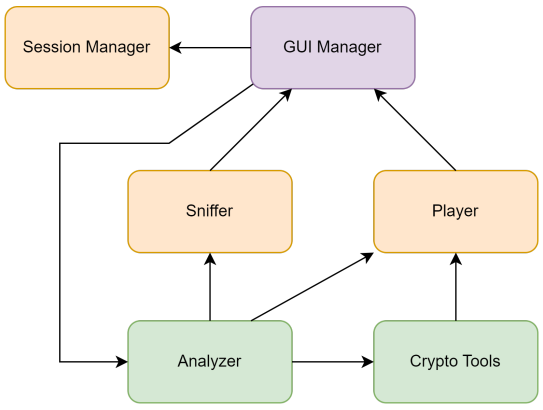


Figure 5.1: LoRAttack architecture

5.2 Implementation

This section contains specifics about implementation of each module including the issues encountered, third-party tooling used and highlights the tool's features.

5.2.1 GUI Manager

The GUI Manager module provides a user-friendly command-line interface that allows users to interact efficiently with the tool's various modules. It is achieved through the Urwid library [49], which supports the development of CLI-based interfaces, facilitating easy navigation and operation of the security testing features.

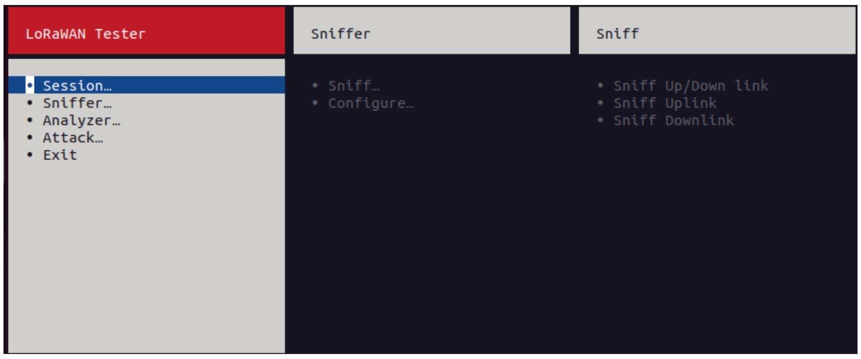


Figure 5.2: LoRAttack GUI

5.2.2 Session Manager

The Session Manager module manages session data for the security testing tool. Users can start new sessions or select from existing ones. It also allows other modules to access and update current session data, which is crucial for storing critical information like handshake details and cryptographic keys. This ensures a streamlined workflow for effectively organizing and conducting security analyses. The detailed session data description is present in 5.1

5.2.3 Sniffer

For the sniffer feature of the LoRaWAN security testing tool, the choice of Software Defined Radio (SDR) had to be made. The USRP (Universal Software Radio Peripheral) SDR by Ettus Research [50] was selected for its robust performance characteristics, which are well-suited for LPWAN testing. The USRP offers a broad frequency range, bandwidth, and high sample rate, making it ideal for capturing LoRaWAN signals.

The demodulation of LoRa signals is handled using the gr-lora project [51], which provides a GNU Radio implementation specifically for this purpose. This project was chosen as the foundation for demodulating LoRa signals due to its compatibility and efficiency in processing the unique characteristics of LoRa modulation. The tool allows user to run GNU Radio based sniffer scripts through the GUI. It also includes an option to configure the SDR parameters that are used in the GNU Radio blocks such as center frequency, gain, sample rate or bandwidth.

During the development of the sniffer feature, several challenges were encountered, particularly in dealing with the nature of LoRaWAN communications:
Channelization
It was noted that common LoRaWAN implementations employ channel hop-

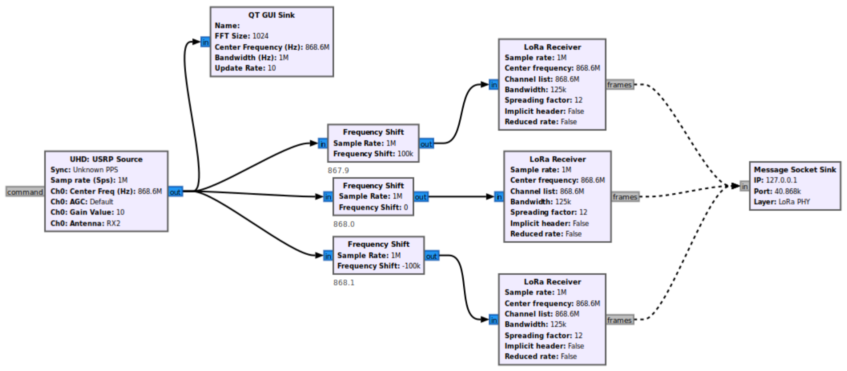
Field	**Description**
AppKey	Application Key used for encryption at the application layer.
NwkKey	Network Key used for securing network layer operations.
DevEUI	Device EUI (Extended Unique Identifier) in the join request.
AppEUI	Application EUI in the join request, identifies the application.
JoinEUI	Join EUI (formerly AppEUI) identifying the join server.
DevNonce	Device nonce, a random value used once during the join process.
AppNonce	Application nonce in the join accept, deprecated in LoRaWAN 1.1+.
JoinNonce	Join nonce, a random number used in the join accept message.
NetID	Network Identifier assigned to the end-device after a successful join.
DevAddr	Device address assigned to the end-device upon joining the network.
AppSKey	Application Session Key used for encrypting application payloads.
NwkSKey	Network Session Key used for encrypting network payloads (deprecated, replaced by FNwkSIntKey and SNwkSIntKey).
NwkSEncKey	Network Session Encryption Key used for encrypting certain MAC layer commands.
SNwkSIntKey	Serving Network Session Integrity Key used for integrity protection of network commands.
FNwkSIntKey	Forwarding Network Session Integrity Key used for operations across forwarding networks.

Table 5.1: Session data description

ping techniques to mitigate interference and potential jamming. This adaptive strategy enhances network resilience but requires the sniffer to monitor multiple channels simultaneously to ensure comprehensive coverage of all network traffic. This feature is not implemented in gr-lora project by default and thus had to be implemented by the tool itself.

To effectively manage multi-channel sniffing, a solution was implemented by tuning the Software Defined Radio (SDR) to a center frequency positioned in the middle of the channels being monitored. This approach allowed the SDR to cover all the necessary channels simultaneously. To accommodate the range of frequencies required for different channels, the Frequency Shift block in GNU Radio was used. This block adjusted the sniffed signal by shifting it to the designated center frequency, depending on the channel bandwidth.

After frequency adjustment, the shifted signal was then fed into the gr-lora receiver block, which is designed to demodulate LoRa signals. The processed signals were subsequently directed to a message socket sink for further analysis and handling. This setup ensured that signals from multiple channels could be simultaneously captured and processed efficiently, enhancing the tool's capability to monitor diverse network traffic effectively.



UHD: USRP Source
Sync: Unknown PPS
Samp rate (Sps): 1M
Ch0: Center Freq (Hz): 868.6M
Ch0: AGC: Default
Ch0: Gain Value: 10
Ch0: Antenna: RX2

QT GUI Sink
Name:
FFT Size: 1024
Center Frequency (Hz): 868.6M
Bandwidth (Hz): 1M
Update Rate: 10

Frequency Shift
Sample Rate: 1M
Frequency Shift: 100k
867.9

Frequency Shift
Sample Rate: 1M
Frequency Shift: 0
868.0

Frequency Shift
Sample Rate: 1M
Frequency Shift: -100k
868.1

LoRa Receiver
Sample rate: 1M
Center frequency: 868.6M
Channel list: 868.6M
Bandwidth: 125k
Spreading factor: 12
Implicit header: False
Reduced rate: False

LoRa Receiver
Sample rate: 1M
Center frequency: 868.6M
Channel list: 868.6M
Bandwidth: 125k
Spreading factor: 12
Implicit header: False
Reduced rate: False

LoRa Receiver
Sample rate: 1M
Center frequency: 868.6M
Channel list: 868.6M
Bandwidth: 125k
Spreading factor: 12
Implicit header: False
Reduced rate: False

Message Socket Sink
IP: 127.0.0.1
Port: 40.868k
Layer: LoRa PHY

Figure 5.3: GNU Radio Sniffer Snippet

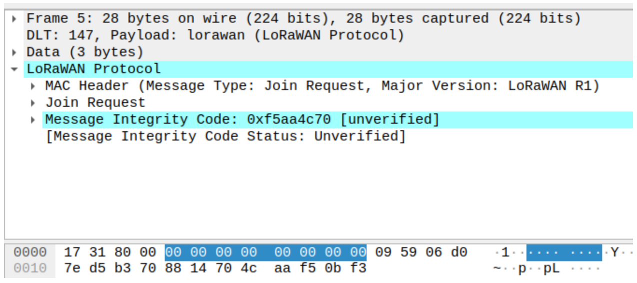
SF11 and SF12 demodulation
There were difficulties in demodulating SF 11 and SF 12 due to the need for Low Data Rate Optimization (LDRO) and soft decision decoding. LDRO is a setting within the LoRa physical layer that optimizes the receiver for low data rate communications, enhancing its sensitivity and range. To address this issue, an enhanced version of gr-lora by tapparelj [52], which supports these features, was utilized.

Downlink sniffing
To handle bidirectional sniffing in LoRaWAN, downlink communications invert the I and Q signals. Therefore, two gr-lora receiver blocks are used: one set to conjugate the downlink signal ('Yes') for the inverted signals, and another set to 'No' for uplink signals. This configuration ensures accurate decoding of both uplink and downlink transmissions.

Wireshark compatability
Although Wireshark supports LoRaWAN dissection, the captured packets are encapsulated in UDP, preventing direct analysis. To address this, the UDP header must be removed, and the Data Link Type (DLT) of the capture needs to be changed to 147, specific to LoRaWAN. The tool automates this process using the Bittwiste utility [53], which adjusts the packet headers and DLT settings to enable proper analysis in Wireshark.

5.2.4 Analyzer

The Analyzer module is designed for efficient packet processing and analysis within the security testing tool:



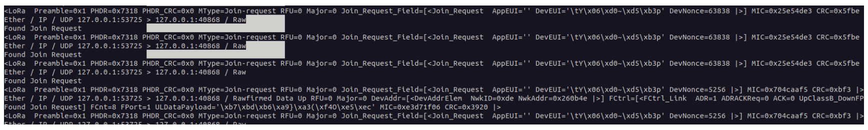
```
▸ Frame 5: 28 bytes on wire (224 bits), 28 bytes captured (224 bits)
  DLT: 147, Payload: lorawan (LoRaWAN Protocol)
▸ Data (3 bytes)
▾ LoRaWAN Protocol
  ▸ MAC Header (Message Type: Join Request, Major Version: LoRaWAN R1)
  ▸ Join Request
  ▸ Message Integrity Code: 0xf5aa4c70 [unverified]
    [Message Integrity Code Status: Unverified]

0000  17 31 80 00 00 00 00 00  00 00 00 00 09 59 06 d0   ·1·· ···· ····Y··
0010  7e d5 b3 70 88 14 70 4c  aa f5 0b f3               ~··p··pL ····
```

Figure 5.4: Modified LoRaWAN Packet analysis

- **Packet Decoding and Key Management:** This feature is responsible for decoding packets captured by the Sniffer. It also handles key cryptographic tasks; when it detects a handshake in the packet data, it saves critical handshake values such as nonces into the active session. If the configuration file contains root keys, the Analyzer works alongside the Crypto Tools module to generate session keys, which are then used to decrypt subsequent packets.
- **Retrospective Analysis:** Offers the capability to analyze previously captured LoRaWAN pcap files, enabling detailed examination of past network communications.

For packet dissection, the Analyzer module utilizes a Scapy dissector for LoRaWAN, developed by Penthertz [54]. This layer is used for breaking down the packet structures for in-depth analysis. Each packet, along with any session values decoded from these packets, such as network keys and device addresses, is systematically stored in the current session. This method ensures that all relevant data is retained and organized.



```
<LoRa  Preamble=0x1 PHDR=0x7318 PHDR_CRC=0x0 MType=Join-request RFU=0 Major=0 Join_Request_Field=[<Join_Request  AppEUI='' DevEUI='\tY\x06\xd0~\xd5\xb3p' DevNonce=63838 |>] MIC=0x25e54de3 CRC=0x5fbe
Ether / IP / UDP 127.0.0.1:53725 > 127.0.0.1:40868 / Raw
Found Join Request
<LoRa  Preamble=0x1 PHDR=0x7318 PHDR_CRC=0x0 MType=Join-request RFU=0 Major=0 Join_Request_Field=[<Join_Request  AppEUI='' DevEUI='\tY\x06\xd0~\xd5\xb3p' DevNonce=63838 |>] MIC=0x25e54de3 CRC=0x5fbe
Ether / IP / UDP 127.0.0.1:53725 > 127.0.0.1:40868 / Raw
Found Join Request
<LoRa  Preamble=0x1 PHDR=0x7318 PHDR_CRC=0x0 MType=Join-request RFU=0 Major=0 Join_Request_Field=[<Join_Request  AppEUI='' DevEUI='\tY\x06\xd0~\xd5\xb3p' DevNonce=63838 |>] MIC=0x25e54de3 CRC=0x5fbe
Ether / IP / UDP 127.0.0.1:53725 > 127.0.0.1:40868 / Raw
Found Join Request
<LoRa  Preamble=0x1 PHDR=0x7318 PHDR_CRC=0x0 MType=Join-request RFU=0 Major=0 Join_Request_Field=[<Join_Request  AppEUI='' DevEUI='\tY\x06\xd0~\xd5\xb3p' DevNonce=5256 |>] MIC=0x704caaf5 CRC=0xbf3 |>
Ether / IP / UDP 127.0.0.1:53725 > 127.0.0.1:40868 / Rawfirmed Data Up RFU=0 Major=0 DevAddr=[<DevAddrElem  NwkID=0xde NwkAddr=0x260b4e |>] FCtrl=[<FCtrl_Link  ADR=1 ADRACKReq=0 ACK=0 UpClassB_DownFP
Found Join Request] FCnt=8 FPort=1 ULDataPayload='\xb7\xbd\xb6\xa9}\xa3(\xf4O\xe5\xec' MIC=0xe3d71f06 CRC=0x3920 |>
<LoRa  Preamble=0x1 PHDR=0x7318 PHDR_CRC=0x0 MType=Join-request RFU=0 Major=0 Join_Request_Field=[<Join_Request  AppEUI='' DevEUI='\tY\x06\xd0~\xd5\xb3p' DevNonce=5256 |>] MIC=0x704caaf5 CRC=0xbf3 |>
Ether / IP / UDP 127.0.0.1:53725 > 127.0.0.1:40868 / Raw
```

Figure 5.5: Analysed LoRaWAN packets

5.2.5 Player

The Player module facilitates vulnerability and security testing in LoRaWAN networks by enabling the replay of previously captured packets or the crafting of new packets following LoRaWAN standards. It utilizes the gr-lora GNU Radio blocks by tapparelj [52] for packet transmission via a USRP device. Users can fine-tune transmission settings, including gain, center frequency, bandwidth, and spreading factor, directly from the module's configuration menu, ensuring precise control over testing scenarios.

- **Replay Features**:

  - **From PCAP**: Allows users to select a pre-captured PCAP file from which packets can be replayed to test how the network responds to previously observed traffic.
  - **Edit Replay Sequence**: Enables manual editing of the packet sequence within a chosen PCAP file before replaying, providing flexibility in creating tailored testing scenarios.

- **Crafting Packets**:
  - **Spoof Join Request**: Crafts a Join Request message using current session values (JoinEUI, DevEUI) and transmits it using SDR. Default values are used if necessary, with AppKey/NwkKey required for MIC computation.
  - **Spoof Join Accept**: Generates a Join Accept message based on current session data (NetID, DevAddr) and sends it via SDR, using defaults where needed. AppKey/NwkKey is required for both MIC computation and encryption.
  - **Spoof ACK Message**: Constructs an ACK message using session data (DevAddr, FCnt set to 0xFFF0 to enhance DoS chances) and transmits it using SDR, with NwkSKey needed for MIC computation.

5.3 Testing

To evaluate the developed tool, a real-world IoT application was set up using a LoRa-enabled device connected to a community LoRaWAN gateway. An USRP Software-Defined Radio was employed to assess the Radio-Frequency related features of the testing tool. For the details, please follow this section.

5.3.1 Laboratory setup

Firstly, a functional laboratory setup was established, consisting of an End Device, a community Gateway, and both Network and Application Servers. The attacker's station was equipped with a USRP Software-Defined Radio. Further details are provided below.

5.3.1.1 LoRaWAN Environment

For the LoRaWAN environment, The Things Network (TTN), which runs on The Things Stack—a widely recognized LoRaWAN Network Server essential for LoRaWAN solutions was used. Managed by The Things Industries, it securely manages applications, end devices, and gateways [55]. A key benefit of using The Things Network is its support for community gateways, enabling testing without owning a personal gateway and thus making the testing process more accessible and practical.

5.3.1.2 End Device and Gateway

- **End Device**: The chosen end device was the LilyGO TTGO LoRa32 T3_V1.6 [56], which is built on the ESP32 and includes an SF1276 LoRa module. Firmware for this device was implemented using arduino-lmic [57], a widely adopted LoRaWAN library ported for Arduino, ESP, and PlatformIO. This firmware choice was driven by its extensive use and compatibility, ensuring reliable performance across various development environments.

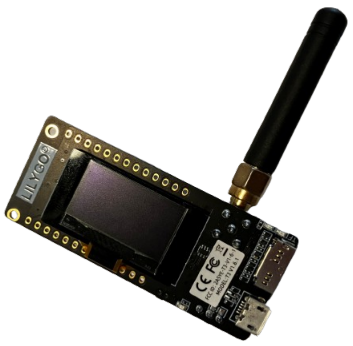


Figure 5.6: LilyGO TTGO LoRa32 T3_V1.6

- **Gateway**: Community gateways from The Things Network (TTN) were used for testing, with evaluations conducted in both an urban area in central Kladno, Czech Republic, and a rural area at Prinknash Abbey Park, Gloucester, UK [58]. This allowed for assessing the network's adaptability to different environmental conditions.

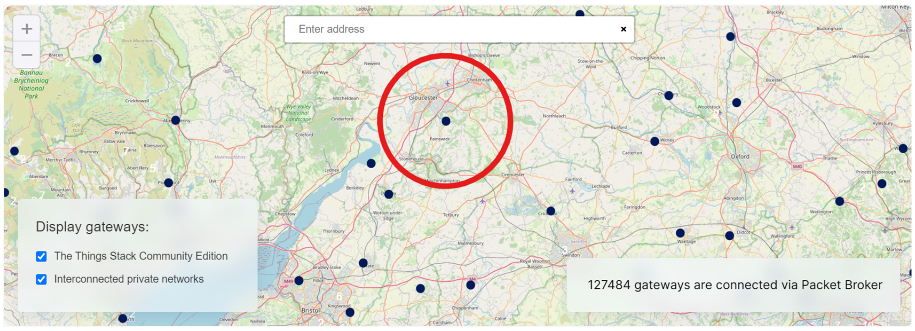

Enter address
Display gateways:
The Things Stack Community Edition
Interconnected private networks
127484 gateways are connected via Packet Broker

Figure 5.7: The Things Network community gateways map

5.3.1.3 Attacker equipment

For the attacker's equipment setup, Ettus B205mini and Ettus B210 Software-Defined Radios (SDRs) equipped with 868MHz antennas were utilized. These devices were chosen for their ability to effectively handle the specific frequency requirements of LoRaWAN operations, making them suitable for conducting detailed and effective security assessments.

Figure 5.8: Ettus USRP B205mini [59]

5.3.2 Scope limitations

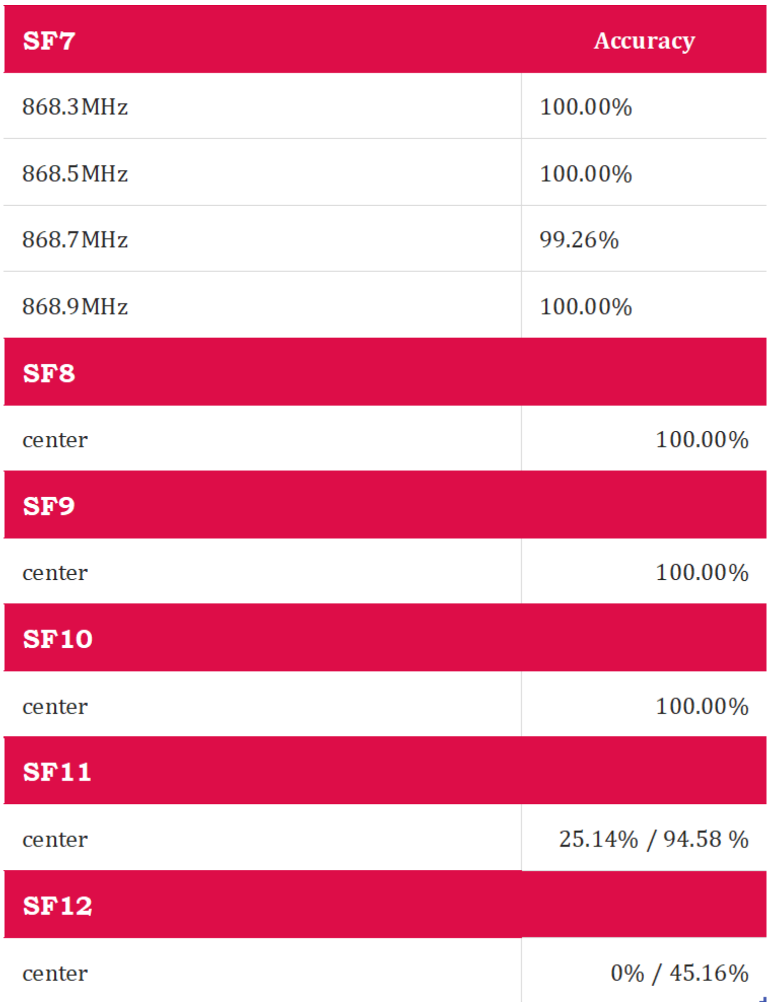
One of the limitations encountered during the project was not having a physical gateway in possession, which meant that testing had to be conducted within a distance of about 1-2 kilometers from the community gateway. This limitation affected the reliability of downlink signal sniffing. Although some packets were captured, the distance constraint prevented stable and consistent sniffing, thus limiting the effectiveness of downlink sniffing tests and complicating the execution of certain attack tests that depended on reliable packet replay.

5.3.3 Testing results

The sniffer was tested for multi-channel sniffing, data demodulation accuracy, and functionality across spreading factors from SF7 to SF12. Simultaneously, the transmitter's accuracy was assessed by running its flow graph and checking if the signals were correctly received by the LilyGO device equipped with receiver firmware. These tests ensured that the tool could reliably handle and decode LoRaWAN communications, as well as accurately transmit signals for effective security evaluations.

The multi-channel functionality of the sniffer was tested by transmitting 100 payloads consisting of characters ”a-z” and numbers ”1-9” across four channels: 868.3, 868.5, 868.7, and 868.9 MHz. For spreading factors SF8 through SF12, the transmission was centered on a single frequency to test the robustness of reception at different data rates. Sample rate of the USRP was set to 1Msps. Specifically, for spreading factors 11 and 12, the testing was supplemented with the enhanced gr-lora from tapparelj, as mentioned earlier, to accommodate the lack of low data rate optimization. The effectiveness of these tests was quantified by calculating the percentage of correctly demodulated bytes for each scenario, with the results presented in the corresponding table.

The replay functionality was thoroughly tested and achieved 100% accuracy in terms of packet demodulation by a second device. The replay module's utility for conducting jamming attacks was also evaluated. Although LoRa modu-



SF7	Accuracy
868.3MHz	100.00%
868.5MHz	100.00%
868.7MHz	99.26%
868.9MHz	100.00%
SF8	
center	100.00%
SF9	
center	100.00%
SF10	
center	100.00%
SF11	
center	25.14% / 94.58 %
SF12	
center	0% / 45.16%

Figure 5.9: Sniffer testing results

lation is generally resistant to interference, there is a potential for jamming when two devices share the same spreading factor, frequency, and channel. To test this, the USRP was configured to transmit at a high gain of 70 dB, using the same parameters as the transmitting LoRaWAN device placed next to it. Despite these conditions, no disruption occurred, and all transmitted messages were successfully delivered to the TTN application server, indicating robust resistance to interference under the tested settings.

Conclusion

The goal of this thesis was to create a security testing framework for LoRaWAN networks utilizing SDR technology. The framework shall be based on comprehensive research of LPWAN protocol security and tested with real-world LoRaWAN envirnoment.

An analysis of various LPWAN protocols was conducted, detailing their design objectives and use-case applicability, laying a foundation for understanding their broader context. Security analysis of the LoRaWAN protocol followed, highlighting its security features, cryptographic measures, and potential threats. This analysis prepared the ground for practical security enhancements.

The thesis further explored Software Defined Radio (SDR) technology for its utility in sniffing and intercepting LPWAN communications, underscoring its use in security testing.

Building on this research, a security testing framework for the LoRaWAN protocol was proposed. This framework includes an SDR-based tool – LoRAttack – that features capabilities such as multi-channel sniffing, session-based packet capture, automated packet processing, and decryption. Furthermore, it incorporates a replay module complemented by detailed guidance for vulnerability testing of known issues in the LoRaWAN protocol. The modular design of this tool ensures easy extendability, allowing the addition of new functionalities or adaptation to other protocols in the future.

The practical application of this tool was tested in a real-world LoRaWAN implementation using community gateways. This setup effectively simulated an actual LoRaWAN IoT environment, providing insight into the tool's effectiveness.

To conclude, the developed LoRAttack framework is fully functional in terms of defined requirements and utilizable for practical security testing of LoRaWAN networks. Given the complexity of security testing, numerous potential enhancements exist. The proposed future work includes addition of fuzzing module to expand testing capabilities and supporting more types of SDRs to increase its versatility.

Bibliography

[1] IoT Analytics. Internet of Things (IoT) and non-IoT active device connections worldwide from 2010 to 2025 (in billions). Nov. 2020, accessed: 2024-02-24. Available from: `https://www.statista.com/statistics/1101442/iot-number-of-connected-devices-worldwide/`

[2] Cisco Systems. Number of Machine-to-Machine (M2M) connections worldwide from 2014 to 2021 (in billions). Feb. 2017, accessed: 2024-02-24. Available from: `https://www.statista.com/statistics/487280/global-m2m-connections/`

[3] Chilamkurthy, N. S.; Pandey, O. J.; et al. Low-Power Wide-Area Networks: A Broad Overview of Its Different Aspects. *IEEE Access*, volume 10, 2022: pp. 81926–81959, doi:10.1109/ACCESS.2022.3196182.

[4] Arduino. Low Power Wide Area Networks 101. Feb. 2024, accessed: 2024-02-24. Available from: `https://docs.arduino.cc/learn/communication/low-power-wide-area-networks-101/`

[5] Centenaro, M.; Vangelista, L.; et al. Long-Range Communications in Unlicensed Bands: the Rising Stars in the IoT and Smart City Scenarios. *IEEE Wireless Communications*, volume 23, 10 2015, doi:10.1109/MWC.2016.7721743.

[6] Chilamkurthy, N.; Pandey, O.; et al. Low-Power Wide-Area Networks: A Broad Overview of Its Different Aspects. *IEEE Access*, volume 10, 01 2022: pp. 1–1, doi:10.1109/ACCESS.2022.3196182.

[7] Cisco. Cisco Annual Internet Report (2018–2023) White Paper. `https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html`, 2020, [Online; accessed YYYY-MM-DD].

[8] Adefemi, O.; Ouahada, K.; et al. A Survey on the Security of Low Power Wide Area Networks: Threats, Challenges, and Potential Solutions. *Sensors*, volume 20, 10 2020: p. 5800, doi:10.3390/s20205800.

[9] Loukil, S.; Fourati, L. C.; et al. Investigation on Security Risk of LoRaWAN: Compatibility Scenarios. *IEEE Access*, volume 10, 2022: pp. 101825–101843, doi:10.1109/ACCESS.2022.3208171.

[10] Soy, H. Coverage Analysis of LoRa and NB-IoT Technologies on LPWAN-Based Agricultural Vehicle Tracking Application. *Sensors*, volume 23, no. 21, 2023, ISSN 1424-8220. Available from: `https://www.mdpi.com/1424-8220/23/21/8859`

[11] Chen, M.; Miao, Y.; et al. Narrow Band Internet of Things. *IEEE Access*, volume 5, 2017: pp. 20557–20577, doi:10.1109/ACCESS.2017.2751586.

[12] Peruzzi, G.; Pozzebon, A. A Review of Energy Harvesting Techniques for Low Power Wide Area Networks (LPWANs). *Energies*, volume 13, 07 2020, doi:10.3390/en13133433.

[13] Mekki, K.; Bajic, E.; et al. A comparative study of LPWAN technologies for large-scale IoT deployment. volume 5, 03 2019: pp. 1–7, doi:10.1016/j.icte.2017.12.005.

[14] Mo, Y.; GOURSAUD, C.; et al. Uplink Multiple Base Stations Diversity for UNB based IoT networks. 09 2018, pp. 1–4, doi:10.1109/CAMA.2018.8530530.

[15] Alexiou, A.; Haardt, M. Smart antenna technologies for future wireless systems: trends and challenges. *IEEE Communications Magazine*, volume 42, no. 9, 2004: pp. 90–97, doi:10.1109/MCOM.2004.1336725.

[16] Trenton Systems. Licensed vs Unlicensed Spectrum. `https://www.trentonsystems.com/en-us/resource-hub/blog/licensed-vs-unlicensed-spectrum`, 2024, accessed: 26 February 2024.

[17] Ikpehai, A.; Adebisi, B.; et al. Low-power wide area network technologies for Internet-of-Things: A comparative review. *IEEE Internet of Things Journal*, volume 6, no. 2, Apr 2019: pp. 2225–2240.

[18] The Things Industries Documentation. `https://www.thethingsindustries.com/docs/`, accessed: YYYY-MM-DD.

[19] ChirpStack Documentation. `https://www.chirpstack.io/`, accessed: YYYY-MM-DD.

[20] Chaudhari, B. S.; Zennaro, M.; et al. LPWAN Technologies: Emerging Application Characteristics, Requirements, and Design Considerations. *Future Internet*, volume 12, no. 3, 2020, ISSN 1999-5903, doi:10.3390/fi12030046. Available from: `https://www.mdpi.com/1999-5903/12/3/46`

[21] Tikhvinskiy, V.; Bochechka, G.; et al. Comparative analysis of QoS management and technical requirements in 3GPP standards for cellular IoT technologies. *J. Telecommun. Inf. Technol.*, volume 2, no. 2018, Jul 2018: pp. 41–47.

[22] Mekki, K.; Bajic, E.; et al. Overview of cellular LPWAN technologies for IoT deployment: Sigfox, LoRaWAN, and NB-IoT. In *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops (PerCom Workshops)*, Mar 2018, pp. 197–202.

[23] Lauridsen, M.; Nguyen, H.; et al. Coverage Comparison of GPRS, NB-IoT, LoRa, and SigFox in a 7800 km^2 Area. In *2017 IEEE 85th Vehicular Technology Conference (VTC Spring)*, 2017, pp. 1–5, doi:10.1109/VTCSpring.2017.8108182.

[24] Sinha, R.; Yiqiao, W.; et al. A survey on LPWA technology: LoRa and NB-IoT. *ICT Express*, volume 3, 03 2017, doi:10.1016/j.icte.2017.03.004.

[25] Semtech Corporation. LoRa - Long Range Wireless RF Technology. https://www.semtech.com/lora, 2024, [Accessed: 2023-08-01].

[26] Gbadoubissa, J. E. Z.; Ari, A. A. A.; et al. M-Ary Direct Modulation Chirp Spread Spectrum for Spectrally Efficient Communications. *Information*, volume 14, no. 6, 2023: p. 323, doi:10.3390/info14060323, hAL Id: hal-04126099.

[27] Sakshama Ghoslya. LoRa spreading factors comparison. https://www.sghoslya.com/p/lora-is-chirp-spread-spectrum.html, 2017, accessed: 2023-08-02.

[28] Semtech Corporation. LoRa and LoRaWAN Technical Papers and Guides. https://lora-developers.semtech.com/documentation/tech-papers-and-guides/lora-and-lorawan/, 2023, accessed: 2023-11-16.

[29] LoRa Alliance. About LoRaWAN. https://lora-alliance.org/about-lorawan/, 2023, accessed: 2024-03-01.

[30] The Things Network. Community - The Things Network. https://www.thethingsnetwork.org/community, 2024, accessed: 2024-03-03.

[31] Semtech Corporation. LoRaWAN Device Activation. https://lora-developers.semtech.com/documentation/tech-papers-and-guides/lorawan-device-activation/device-activation/, 2023, accessed: 2024-03-14.

[32] LoRa Alliance. LoRaWAN Specification Version 1.0.3. Technical report, LoRa Alliance, Nov. 2020, [Online]. Available: https://lora-alliance.org/wp-content/uploads/2020/11/lorawan1.0.3.pdf. Available from: https://lora-alliance.org/wp-content/uploads/2020/11/lorawan1.0.3.pdf

[33] Virk, H. Introducing LoRaWAN 1.1 support. https://os.mbed.com/blog/entry/Introducing-LoRaWAN-11-support/, 2019, accessed: 2024-03-17.

[34] The Things Network. LoRaWAN Message Types. https://www.thethingsnetwork.org/docs/lorawan/message-types/, 2023, accessed: 2024-03-30.

[35] Loukil, S.; Fourati, L.; et al. Analysis of LoRaWAN 1.0 and 1.1 Protocols Security Mechanisms. volume 22, 05 2018: p. 3717, doi:10.3390/s221103717.

[36] Aras, E.; Small, N.; et al. Selective Jamming of LoRaWAN using Commodity Hardware. MobiQuitous 2017, New York, NY, USA: Association for Computing Machinery, 2017, ISBN 9781450353687, p. 363–372, doi: 10.1145/3144457.3144478. Available from: `https://doi.org/10.1145/3144457.3144478`

[37] Butun, I.; Pereira, N.; et al. Security Risk Analysis of LoRaWAN and Future Directions. *Future Internet*, volume 11, no. 1, 2019, ISSN 1999-5903, doi:10.3390/fi11010003. Available from: `https://www.mdpi.com/1999-5903/11/1/3`

[38] van Es, E.; Vranken, H.; et al. Denial-of-Service Attacks on LoRaWAN. ARES '18, New York, NY, USA: Association for Computing Machinery, 2018, ISBN 9781450364485, doi:10.1145/3230833.3232804. Available from: `https://doi.org/10.1145/3230833.3232804`

[39] Aras, E.; Ramachandran, G. S.; et al. Exploring the Security Vulnerabilities of LoRa. In *2017 3rd IEEE International Conference on Cybernetics (CYBCONF)*, 2017, pp. 1–6, doi:10.1109/CYBConf.2017.7985777.

[40] Dönmez, T. C.; Nigussie, E. Security of LoRaWAN v1.1 in Backward Compatibility Scenarios. *Procedia Computer Science*, volume 134, 2018: pp. 51–58, ISSN 1877-0509, doi:https://doi.org/10.1016/j.procs.2018.07.143, the 15th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2018) / The 13th International Conference on Future Networks and Communications (FNC-2018) / Affiliated Workshops. Available from: `https://www.sciencedirect.com/science/article/pii/S1877050918311062`

[41] Yang, X.; Karampatzakis, E.; et al. Security Vulnerabilities in LoRaWAN. In *2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI)*, 2018, pp. 129–140, doi: 10.1109/IoTDI.2018.00022.

[42] Sinha, D.; Verma, A.; et al. Software defined radio: Operation, challenges and possible solutions. 01 2016, pp. 1–5, doi:10.1109/ISCO.2016.7727079.

[43] Page, R. What are the Components of Software Defined Radio and its Applications. Available from: `https://www.rfpage.com/what-are-the-components-of-software-defined-radio-and-its-applications/`

[44] Cadence. Software Defined Radio. `https://resources.pcb.cadence.com/blog/2023-software-defined-radio`, 2023, accessed: 2024-02-26.

[45] Radio Society of Sri Lanka. Popular SDRs. `https://rssl.lk/2023/07/18/popular-sdrs/`, 2023, accessed: 2024-04-26.

[46] GNU Radio Project. About GNU Radio. `https://www.gnuradio.org/about/`, 2023, accessed: 2024-03-26.

[47] GQRX. GQRX Software Defined Radio Receiver. `https://www.gqrx.dk/`, 2023, accessed: 2024-03-26.

[48] Pohl, J. Universal Radio Hacker. https://github.com/jopohl/urh, 2023, accessed: 2024-03-26.

[49] Urwid. Urwid - A Console User Interface Library for Python. Accessed: 2023, online; accessed 20-April-2023. Available from: https://urwid.org/index.html

[50] Research, E. Ettus Research - Leaders in Software Defined Radio (SDR). Accessed: 2023, online; accessed 20-April-2023. Available from: https://www.ettus.com/

[51] rpp0. gr-lora - GNU Radio OOT module for LoRa modulation and demodulation. Accessed: 2023, online; accessed 20-April-2023. Available from: https://github.com/rpp0/gr-lora

[52] Tapparel, J. gr-lora_sdr - A GNU Radio module for LoRa SDR. Accessed: 2023, online; accessed 20-April-2023. Available from: https://github.com/tapparelj/gr-lora_sdr

[53] Bittwist. Bittwist - A Simple yet Powerful Libpcap-based Ethernet Packet Generator. Accessed: 2023, online; accessed 20-April-2023. Available from: https://bittwist.sourceforge.io/

[54] Penthertz. LoRa_Craft: Scapy Dissector for LoRaWAN. Accessed: 2024, online; accessed 20-April-2023. Available from: https://github.com/PentHertz/LoRa_Craft/blob/master/layers/loraphy2wan.py

[55] The Things Network. The Things Network. Accessed: 2024, online; accessed 20-April-2024. Available from: https://www.thethingsnetwork.org/

[56] LILYGO. LILYGO TTGO LoRa32 T3_V1.6 868Mhz 0.96" SMA WiFi Module. Accessed: 2023, online; accessed 20-April-2023. Available from: https://www.lilygo.cc/products/lora3

[57] MCCI Catena. arduino-lmic - LoRaMAC-in-C library, fully compliant with the LoRaWAN specification for Arduino. Accessed: 2023, online; accessed 20-April-2023. Available from: https://github.com/mcci-catena/arduino-lmic

[58] The Things Network. The Things Network Map. Accessed: 2023, online; accessed 20-April-2023. Available from: https://www.thethingsnetwork.org/map

[59] Ettus Research. USRP B205mini-i. Accessed: 2023, online; accessed 20-April-2023. Available from: https://www.ettus.com/all-products/usrp-b205mini-i/

Appendix A

Acronyms

IoT	Internet of Things
M2M	Machine-to-Machine
LPWAN	Low Power Wide Area Network
LoRa	Long Range
NB-IoT	Narrowband IoT
LTE-M	Long Term Evolution for Machines
SDR	Software Defined Radio
QoS	Quality of Service
ISM	Industrial, Scientific, and Medical (band)
CSS	Chirp Spread Spectrum
SF	Spread Factor
MAC	Medium Access Control
AES	Advanced Encryption Standard
ECB	Electronic Codebook
MIC	Message Integrity Code
OTAA	Over-the-Air Activation
ABP	Activation by Personalization
DevEUI	Device Extended Unique Identifier
AppEUI	Application Extended Unique Identifier
AppKey	Application Key
NwkKey	Network Key
NwkSKey	Network Session Key

AppSKey Application Session Key

FNwkSIntKey Forwarding Network Session Integrity Key

SNwkSIntKey Serving Network Session Integrity Key

NwkSEncKey Network Session Encryption Key

DevNonce Device Nonce

AppNonce Application Nonce

JoinEUI Join Extended Unique Identifier

NetID Network Identifier

DevAddr Device Address

HSM Hardware Security Module

DoS Denial of Service

ACK Acknowledgment

RF Radio Frequency

NS Network Server

GW Gateway

ED End Device

JS Join Server

IP Internet Protocol

USRP Universal Software Radio Peripheral

PCAP Packet Capture

GUI Graphical User Interface

TTN The Things Network

LDRO Low Data Rate Optimization

UDP User Datagram Protocol

DLT Data Link Type

JTAG Joint Test Action Group (a debugging interface)

NVM Non-Volatile Memory

TLS Transport Layer Security

Appendix **B**

Contents of attachments

`README.txt`	the file with enclosed contents description
`lorattack`	the LoRAttack tool repository
`LoRa`	directory with LilyGo firmware for sniffer measurements
`measurements`	measurements data and evaluation script
`thesis`	the directory of LaTeX source codes of the thesis
`simple_lmic`	directory with the LilyGo firmware for LoRaWAN integration
`thesis.pdf`	the thesis text in PDF format