

UNIVERSITY OF HERTFORDSHIRE
Department of Computer Science

Modular BSc Honours in Computer Science

6COM2018-0905-2024 - Computer Science Project

Final Report
April 2025

**A Security Perspective on the LoRaWAN Protocol:
Analysis and Countermeasures against Cyberattacks**

Stefan Ilie

Supervised by: W. Fernando

TABLE OF CONTENTS

Abstract.....	3
Chapter 1 - Introduction.....	3
1.1 What is the Internet of Things?.....	3
1.2 How Does It Work?.....	4
Chapter 2 - LoRa and LoRaWAN.....	5
2.1 What are LoRa and LoRaWAN?.....	5
2.2 Performance of LoRa Communication.....	7
Chapter 3 - The Fragile Core of LoRaWAN.....	8
3.1 LoRaWAN security mechanisms.....	9
3.2 Key Management.....	9
3.3 Types of Attacks.....	11
1. Replay attack.....	11
2. Jamming attack.....	12
3. Battery depletion attack.....	13
Chapter 4 - Simulations.....	15
4.1 Scenario 1: Jamming attack.....	15
4.2 Scenario 2: Replay attack.....	23
4.3 Scenario 3: Battery Depletion Attack.....	27
Chapter 5 - Future Work.....	31
References.....	33
Appendix 1 - GitHub.....	37
Appendix 2 - Project Management Review.....	38

Abstract

Internet of Things (IoT) technologies have gained significant popularity in the last few decades, and the LoRa (Long Range) protocol seems to be one of the most promising solutions for long distance communications in domains like healthcare, transport systems, home automation, environment monitoring, energy conservation, and many others. However, the rising use of these technologies brought up security concerns as cyberattacks aimed at IoT systems could lead to significant data losses and compromise critical systems.

This paper will present the LoRaWAN security concerns, particularly focusing on its end devices. While LoRaWAN offers long-range communication and low energy consumption, it introduces concerns related to data integrity, authentication, and network resilience. The study reviews common security issues, such as replay attacks, jamming attacks, and energy depletion attacks (EDAs).

In addition, the paper will explore the existing security implementations, such as AES encryption and session key management, while identifying possible limitations. Through simulations of various attacks on the LoRaWAN network, the paper will display weaknesses, propose strategies to address these vulnerabilities, and explore future opportunities for enhancing security.

Chapter 1 - Introduction

1.1 What is the Internet of Things?

According to [IBM \(2023\)](#), the Internet of Things (IoT) refers to a network of physical devices, vehicles, appliances, and other physical objects that are embedded with sensors, software, and network connectivity, allowing them to collect and share data.

It is said that the first IoT device was a Coca-Cola vending machine at Carnegie Mellon University in 1982. It was connected to ARPANET, a predecessor of the modern Internet, and could send information on the machine's temperature and inventory ([Cogniteq, 2024](#)). Now 'things' have changed and the technology has changed too. Today, you can find these smart IoT devices almost everywhere ([Srivastava, S. 2024](#)).

Inside a 'smart home' a thermostat will enable the user to adjust temperature remotely via mobile apps or voice commands, optimising energy consumption. A 'smart lighting' system will enable control over brightness, colour, and schedules remotely. Security cameras that are connected to the internet will transmit live footage from home and will notify the owner of any motion, sending a notification on their smartphone.

The healthcare industry was improved with the help of the Internet of Things enabling remote monitoring, offering patients wearable devices that can collect and transmit data on patient's vitals, activity levels, and health conditions.

A 'smart city' will offer control over air quality, energy, transportation, building systems and provide different solutions from the gathered data to improve the infrastructure, the safety of the citizens, and offer a high quality of life.

The industrial and agricultural sectors are also benefiting from IoT advancements where different IoT sensors are used to monitor and control different machines that lead to improving the accuracy and efficiency of the manufacturing process. Data such as soil moisture, temperature, humidity, is used to optimise the irrigation, fertilisation, and planting schedules, ensuring the crops are receiving enough resources at the right time.

There are other industries, such as retail, automotive, energy, logistics and supply chain management that are adjusting themselves to the era of IoT, transforming their operational ways into more efficient, data-driven, and automated processes.

1.2 How Does It Work?

To answer this question, it is important to understand the architecture and the processes behind an IoT system. We simply start with a device embedded with a sensor or actuator that collects data such as temperature, motion, or humidity, depending on the environment.

As explained by [Aharon Etengoff \(2024\)](#), these devices are connected to the network layer, where they communicate with each other using various protocols and technologies, including Wi-Fi, Bluetooth, Zigbee, LPWAN such as LoRaWAN, and even cellular networks such as 4G or 5G. These communication methods enable data transfer between IoT devices and backend systems.

The gathered data is then stored and processed either locally via edge computing or remotely in the cloud ([Coursera. 2023](#)). Edge computing processes data closer to the source, usually within the device or on a nearby gateway, reducing network latency, which represents the delay in data transmission. This is crucial for real-time applications, such as autonomous vehicles and industrial automation, where immediate responses are needed. On the other hand, cloud computing stores and processes data at scale, often 'fishing' the data from a data lake, representing a centralised repository for raw, unstructured data. Machine learning and big data analytics are then applied to extract meaningful insights and support decision-making.

A visual representation of the IoT architecture is shown in Figure 1.

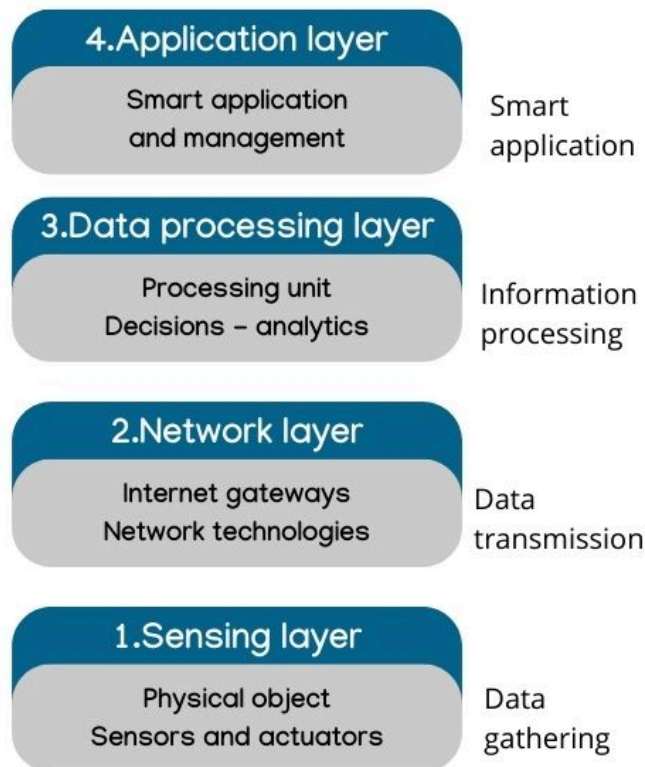


Figure 1: [Layers of IoT Architecture](#)

Chapter 2 - LoRa and LoRaWAN

As the Internet of Things continues to evolve, the need for efficient communication protocols that enable long-distance connectivity with reduced energy consumption has grown. As explained by [Switzer, K. \(2025\)](#), traditional wireless communications, such as Wi-Fi and Bluetooth are effective for short-range applications, but they are not great for long-range communication.

LoRa (Long Range) technology was developed to provide long-distance communication with minimal energy consumption. Built on top of LoRa, the LoRaWAN protocol ensures secure and efficient communication for IoT devices.

2.1 What are LoRa and LoRaWAN?

As described by [Adelantado et al. \(2017\)](#), LoRa is the physical layer used in LoRaWAN. Simply stands for Long Range and it's a wireless modulation technique designed for Low Power Wide Area Networks or LPWANs. It is capable of sending and receiving data over

great distances of up to 15 kilometers in rural environments and around 2-3 kilometers in urban areas, all with minimal power usage. Although, the actual range depends on factors such as antenna design, transmission power, environmental interference, and network conditions. LoRa is optimised for infrequent transmission of small data packages. In other words, not so good for video streaming but ideal for control or data collection in smart cities, agricultural applications, and industrial IoT. LoRa achieves these capabilities using Chirp Spread Spectrum (CSS) modulation ([Inpixon. n.d.](#)).

Chirp works by encoding data onto radio waves using frequency-modulated chirp signals, gradually increasing or decreasing frequency over time. This technique makes LoRa highly resistant to interference and noise, ensuring reliable communication even in congested or industrial environments. One interesting aspect of Chirp Spread Spectrum is that it mimics the way dolphins and bats use echolocation to communicate, allowing signals to travel long distances without significant degradation ([The Things Network. n.d.](#)). In terms of operational capabilities, it uses unlicensed ISM (Industrial, Scientific, and Medical) bands, such as 915 MHz in North America, 868 MHz in Europe, and 433 MHz in Asia. ([Data-alliance.net. n.d.](#))

[de Carvalho Silva et al., \(2017\)](#), describes LoRaWAN protocol as a “star of stars” network which defines the communication protocol and the network system architecture. This protocol manages the communication between devices, ensuring secure data transmission, network scalability, and including features like device authentication and encryption.

To understand how it works, we start with the end nodes, usually an IoT device such as a sensor or actuator that gathers and sends data using the LoRa radio communication. The data packets are received by the nearest gateway and processed under the LoRaWAN protocol. Gateways serve as intermediaries between the end nodes and the network server. A single gateway is able to receive data packets from multiple end nodes within range, usually several kilometers, depending on the transmission power or environmental conditions ([Smartbuildingproducts.co.uk. 2024](#)). Once the packet has been received, it is being forwarded to the network server using Ethernet, Wi-Fi, or cellular.

As the central management point of LoRaWAN, the network server is responsible for packet deduplication, security, and Adaptive Data Rate (ADR) management. Due to potential overlap of gateway coverage, multiple gateways might receive the same packet from an end node. Packet duplication causes its own set of problems as explained by [Al-Awami, n.d.](#). If not managed properly, duplicate packets can contribute to network overhead, especially in dense deployments with frequent transmissions, potentially leading to system throughput degradation. The network server ensures that only one copy of each packet is being processed and forwarded to the application server.

To verify message authenticity and integrity, the network server uses the Network Session Key (NwkSKey) ([Yang et al., n.d](#)) to compute and check the Message Integrity Code (MIC). This ensures that the packet has not been altered during transmission and originates from a legitimate device. However, while the network server can authenticate packets, it cannot access the encrypted payloads, as these are secured using the Application Session Key (AppSKey). This ensures end-to-end encryption at the application layer, meaning that only the application server and the end device can decrypt and access the data content.

Once the network server forwards the encrypted data to the application server, the payload is decrypted using AppSKey. This separation of keys between the network and application server prevents any single entity from having full access to both the network and application data, enhancing security of the system.

Figure 2 provides an illustration of the LoRaWAN architecture and key usage.

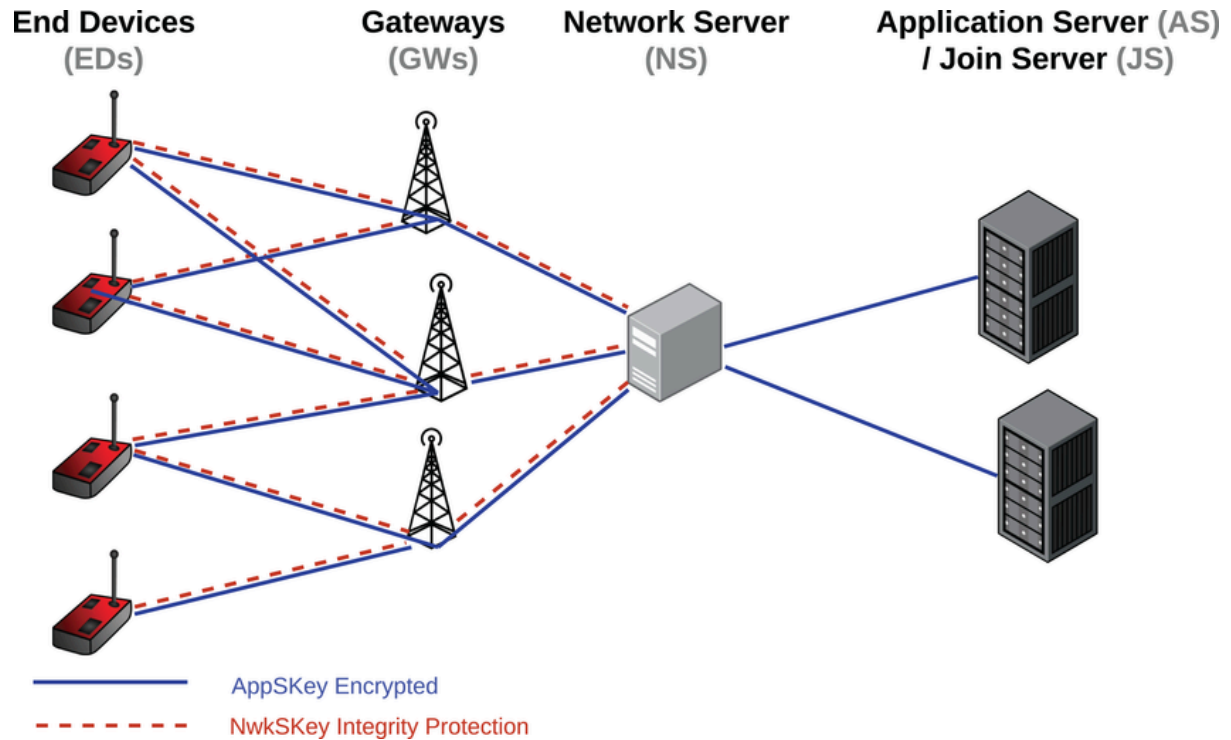


Figure 2: LoRaWAN Architecture and Key Usage ([Kuntke et al., 2022](#))

2.2 Performance of LoRa Communication

To test the communication performance, [Alipio, M. and Bures, M. \(2024\)](#) conducted an evaluation to reflect the system's reliability, efficiency, and scalability. The Received Signal Strength Indicator (RSSI) and Signal-to-Noise Ratio (SNR) measure the strength and quality of the received signal. Packet Delivery Ratio (PDR) shows the reliability of transmission by calculating the percentage of successfully received packets at the gateway, while Packet Error Rate (PER) and Symbol Error Rate (SER) help evaluate the transmission accuracy. Data Rate and Throughput assesses the speed and volume of data exchange. By using these metrics, we can assess how well the LoRa network performs in real-world scenarios, such as smart agriculture or environmental monitoring, where reliable long-range communication is crucial.

Since energy efficiency is a fundamental aspect of LoRa's design, [Philip, M.S. and Singh, P. \(2021\)](#) evaluated the efficiency of LoRa devices by analysing battery consumption and Time

on Air (ToA). Time on Air refers to the duration a signal remains in transmission before reaching the receiver, affecting both energy consumption and network capacity. Latency measures the time taken for a packet to travel from the device to the gateway and, if applicable, back to the device, which is crucial for time-sensitive applications. Accuracy becomes important in applications involving geolocation, where precise location data of devices is necessary. ([lplocation.net, 2025](#)).

The performance also depends significantly on spreading factor (SF) and transmission power, as they determine resource allocation and signal robustness. According to [Zhang et al. \(2023\)](#), the spreading factor influences the bandwidth and data transmission rate, while the transmission power affects the signal strength and the reliability of data packets.

A higher spreading factor enables longer communication distances by spreading the signal over a wider frequency band. This dispersion helps to reduce interference and ensures data packets can be successfully decoded, even under low signal-to-noise ratios or when signals are weakened due to path loss during long-distance transmissions. However, a higher SF comes with trade-offs, such as reduced bandwidth and lower data rates, potentially limiting network capacity.

Similarly, increased transmission power enhances the propagation range and improves the reliability of data packets over long distances, particularly in environments with significant path loss. Yet, excessive transmission power results in high energy consumption, potential interference and reduced network reliability. Selecting appropriate spreading factors and transmission power levels are crucial for stability and efficiency in LoRa networks

Their study also recommends adopting adaptive mechanisms, such as the Adaptive Data Rate (ADR) technology, to optimise transmission parameters based on the network conditions, including packet quality and congestion, thereby improving performance and resource utilisation.

Conversely, suboptimal parameter selection or the absence of ADR can lead to inefficient resource usage, increased collisions, higher energy consumption, and reduced device lifespans.

Chapter 3 - The Fragile Core of LoRaWAN

In this section, I will examine the vulnerabilities and security challenges associated with LoRaWAN, particularly focusing on its end devices. I will first discuss the security mechanisms in place, such as AES encryption and key management protocols, highlighting their strengths and limitations. Additionally, I will review various types of attacks that LoRaWAN networks are susceptible to, examining how these threats exploit existing weaknesses. This analysis aims to provide a comprehensive understanding of the security landscape within LoRaWAN and suggest potential improvements to fortify the network's defense against emerging risks.

As mentioned before, the end devices (nodes) typically consist of sensors and actuators used in different environments, such as agricultural, industrial, and urban settings, designed with a primary focus on energy efficiency and long battery life. Due to their limited computational power, complexity of security mechanisms in LoRaWAN nodes is constrained.

In their research article, [Jiang et al. \(2021\)](#) state that many LoRaWAN end devices are forced to rely on lightweight cryptographic algorithms and simplified authentication protocols.

3.1 LoRaWAN security mechanisms

The current security mechanisms in LoRaWAN devices are designed to ensure the integrity, confidentiality, and authenticity of transmitted data. In an article made by [TEKTELIC \(2023\)](#), one of the primary security mechanisms is Advanced Encryption Standard (AES) 128-bit encryption, which is used to secure data communication between end devices and network servers. This statement is supported by [Gemalto and And \(2017\)](#), which states that each LoRaWAN device is assigned with a unique 128 bit AES key (called AppKey) and a globally unique identifier (EUI-64-based DevEUI), both of which are used during the device authentication process. EUI-64 identifiers are allocated by an assignor with an Organizationally Unique Identifier (OUI) from the IEEE Registration Authority. Similarly, LoRaWAN networks are assigned a 24-bit globally unique NetID by the LoRa Alliance™.

LoRaWAN also implements a dual-layer security model, separating network security from application security. According to [The Things Network. \(n.d.\)](#), when a device joins a LoRaWAN network, a Network Session Key (NwkSKey) and an Application Session Key (AppSKey) are assigned. In Over-The-Air Activation (OTAA), these keys are dynamically generated per session, enhancing security. In Activation By Personalization (ABP), however, they remain static until manually updated.

The NwkSKey is shared with the network and ensures secure communication between the device and the Network Server. It ensures message integrity using the Message Integrity Code (MIC), derived from the NwkSKey via AES-CMAC encryption, which prevents message tampering. Additionally, it assists in mapping dynamically assigned DevAddr identifiers to unique DevEUI and AppEUI identifiers within network backends such as The Things Network. The AppSKey, in contrast, is kept private and is responsible for encrypting and decrypting payloads. This ensures end-to-end confidentiality between the device and the Application Server, preventing unauthorised access to transmitted data.

Despite these measures, weaknesses in key management, improper frame counter implementation that could lead to replay attacks, and insufficient hardware protection that can allow key extraction through side-channel attacks pose potential security risks.

3.2 Key Management

According to [Iconstantin \(2020\)](#), cryptographic session keys, such as the Application Session Key (AppSKey) and Network Session Key (NwkSKey), are often hardcoded into device firmware, making them susceptible to extraction through reverse engineering. By gaining

physical access to the device, an attacker can extract the firmware, analyse its contents, and retrieve the hardcoded keys. If the keys are compromised then the data packets can be decrypted, and modified into malicious packets, leading to unauthorised commands being executed. Attackers could also impersonate legitimate devices, disrupting the network by injecting false data.

Weak key generation mechanisms can expose cryptographic vulnerabilities, even when keys are not hardcoded. Secure key generation demands high-entropy to avoid predictability ([da Cruz, Suyama and Loiola, 2021](#)). If devices derive keys from predictable factors, such as timestamps, incremental counters, or weak pseudo-random number generators (PRNGs), the keys become vulnerable to brute-force or dictionary attacks. Low-cost IoT devices, such as those used in LoRaWAN networks, often rely on simplified PRNGs ([Zia et al., 2022](#)), that fail to provide sufficient randomness, making key prediction or recovery feasible.

A method to extract information is when the nodes are relying on Activation by Personalisation (ABP), which many of the LoRaWAN devices do due to simplified deployment processes. According to [Microchip.com. \(2022\)](#), these devices are pre-configured with static session keys, meaning these keys are assigned at the time of deployment and remain unchanged unless manually updated. This presents a security risk because if an attacker intercepts a valid packet, they may extract enough information to decrypt communications. Once session keys are compromised, they remain vulnerable, as ABP does not support automatic session key renewal. While OTAA is generally more secure, if key exchange mechanisms are compromised, session keys can still be exposed, leaving data vulnerable.

One other way to extract the cryptographic is through side-channel attacks. [Fukushima et al., \(2020\)](#) evaluates the electromagnetic (EM) signals of some LoRaWAN nodes when performing cryptographic operations. These electromagnetic (EM) emissions from a LoRaWAN node can be captured and analysed to reconstruct encryption keys. To achieve this, an attacker places an electromagnetic probe near the device to detect emissions generated during cryptographic operations, such as AES encryption. By recording and analysing these emissions, the attacker can correlate them with specific cryptographic processes. Using pattern recognition techniques, they can then reconstruct the encryption key used by the device.

Similarly, [Xu et al. \(2019\)](#) talks about how an attacker can connect a high-resolution oscilloscope or a power measurement probe to the LoRaWAN device's power lines to monitor power consumption. While the device performs AES encryption, such as during the OTAA Join-Accept process, the power fluctuations are recorded. The collected data is then analysed statistically to identify patterns that correlate with cryptographic key usage. Once these correlations are established, the attacker can recover the encryption key.

3.3 Types of Attacks

1. Replay attack

The definition of a replay attack is “A replay attack occurs when a cybercriminal eavesdrops on a secure network communication, intercepts it, and then fraudulently delays or resends it to misdirect the receiver into doing what the hacker wants.” ([Kaspersky, 2019](#)).

An investigation by [Perković et al. \(2023\)](#) has proved how software-defined radios (SDRs) are capable of replay attack having as target a LoRa device. Here is how can be implemented:

Interception

With the help of software-defined radios (SDRs) or LoRaWAN transceivers configured to operate on the same frequency bands as the target network, the attacker is able to intercept the LoRaWAN communications. The attacker configures their equipment to match the frequency, spreading factor (SF), and bandwidth of the target LoRaWAN network, enabling interception of data packets transmitted between end devices and gateways. For example, with the help of GNU Radio tool or other SDR software, the attacker captures the radio frequency signals, demodulates them, and extracts the raw data packets for analysis.

Analysis

At the analysis stage, the attacker starts decoding the packet structure of the intercepted packets, identifying the frame header, payload, and frame counter. More attention is given to message integrity code (MIC) and frame counter values, which are crucial for maintaining the authenticity and clearness of messages.

Preparation

As the LoRaWAN utilises frame counters to prevent replay attacks by ensuring each message has a unique identifier, if the device resets or the network does not properly manage these counters, vulnerabilities can arise as the attacker can identify such weaknesses by monitoring for non-incrementing or reset frame counters. At this stage the attacker prepared to retransmit the captured packets without modification, relying on the identified vulnerabilities to bypass the network's security features.

Execution

In order to succeed, the attacker selects an optimal moment for retransmission, such as during periods of low network activity or immediately after a device reset, when frame counter check might be less stricter and using their transmission hardware, the attacker injects the replay packets into the network. If the network server fails to detect the duplicate frame counter or lacks proper security configurations, it may accept the replayed packets as legitimate, leading to potential data duplication or unauthorised command execution.

To defend the network against this type of attack, it would be recommended to ensure that both end devices and network server maintain synchronised and non-resettable frame counters. This way the device should preserve frame counter states across reboots or power cycles to prevent exploitation. Another way to defend the network is to incorporate additional features such as session keys that are updated regularly and the use of nonces or timestamps to ensure message clearness.

In the Simulations chapter, both a visual representation and a practical implementation are provided to analyse my experiment of a replay attack on a LoRaWAN network.

For the visual representation, an animated simulation is developed using 'matplotlib', illustrating how a sender node transmits packets to a gateway while an attacker attempts to intercept and replay them.

For the practical implementation, an experimental setup using Arduino MKRWAN 1310 boards is used to replicate real-world LoRaWAN communication. The sender transmits data packets to the gateway, while a designed attacker attempts to capture and resend the same packet. To analyse the key components and their role in the attack scenario, a comparative table is created to highlight the role of each entity in the attack scenario, the security mechanisms in place, and potential improvements to enhance LoRaWAN's resilience against replay attacks and jamming attacks.

2. Jamming attack

As mentioned before, LoRaWAN operates in unlicensed spectrum bands and utilises Chirp Spread Spectrum (CSS) modulation. Due to this fact the network is inherently vulnerable to various forms of jamming.

The process of a jamming attack begins with a piece of hardware, such as Arduino-based platforms equipped with LoRa transceivers or more sophisticated SDRs open source software like GNU Radio that can be employed to program the hardware for generating interference signals meant to disrupt LoRaWAN communications. The attacker scans the radio spectrum to identify the specific frequencies utilised by the target LoRaWAN network, which varies depending on regional regulations. Once the target frequency has been identified, the attacker configures their hardware to transmit signals that overlap with legitimate LoRa transmissions. ([Dossa, A. and Mehdi, A.E. .2025](#))

The next phase involves analysing network parameters to optimise the effectiveness of the attack. Attackers monitor the target network to identify the spreading factor, bandwidth settings, and transmission schedules by LoRaWAN devices.

Once the network has been analysed, the attacker deploys the jammer and initiates the interference. The jamming attack involves broadcasting a high-power signal that occupies the same frequency as LoRa transmissions, flooding the channel and preventing successful packet delivery. The attacker can implement another type of jamming called Reactive jamming attack, where instead of continuously transmitting interference, the jammer remains

idle until it detects an active LoRa transmission. Once the transmission has been detected, the jammer becomes active, effectively corrupting the legitimate communication between a node and a gateway for example, while reducing power consumption and the likelihood of detection. The impact of a jammer depends on a few factors, including the proximity of the jammer to the target, the power of the interference signal, and the modulation techniques used by LoRaWAN

In their work, [Haque, A., and Saifullah, A. \(n.d.\)](#) suggest that in situations of severe jamming that cannot be mitigated by simply increasing the Spreading Factor (SF) or transmission (TX) power, strategically deploying multiple gateways with different timing offsets offers a viable solution to protect the network. Additionally, employing Frequency Hopping Spread Spectrum (FHSS) techniques can make jamming more challenging by rapidly switching communication channels. Adaptive Data Rate (ADR) mechanisms further enhance network resilience by dynamically adjusting transmission parameters to reduce interference. Monitoring signal strength can also help identify anomalies, providing an early warning of potential jamming attempts. The deployment of more gateways with overlapping coverage areas can mitigate localised jamming by enabling devices to reroute transmissions through unaffected paths.

In the Simulations chapter, both a visual representation and a practical implementation of a jamming attack are conducted to assess its impact on LoRaWAN communication.

For the visual representation, an animated simulation is developed using 'matplotlib' to illustrate how sender nodes transmit packets to the gateway and are disrupted by a jammer.

For the practical implementation, an experimental setup using Arduino MKR WAN 1310 boards is used to replicate real-world jamming conditions. The sender transmits data to the gateway, while the jammer injects interference on the same frequency band to disrupt the signal. A comparative table is provided to analyse the jamming effects and explore possible improvements.

3. Battery depletion attack

LoRaWAN defines three classes of devices: Class A, Class B, and Class C, each designed to meet different communication and power efficiency requirements ([Michael, \(2024\)](#)). The classification of these devices is essential when assessing vulnerabilities, particularly in the context of battery depletion attacks, also known as energy depletion attack (EDA), which primarily target devices with limited energy resources.

As explained by [Cheong et al., n.d.](#) in their paper, Class A and Class B are normally battery-powered, whereas Class C is mains-powered. Class A device is by default set with power saving capabilities and, consequently, the most vulnerable to battery depletion attacks. Since they follow an asynchronous communication model, where they transmit uplink messages only when necessary and open two short downlink reception windows immediately after transmission, attackers can exploit protocol vulnerabilities to force them into excessive power consumption.

First step for an EDA on a Class A device is to intercept uplink transmissions. To do so, the attacker monitors the frequency channels to capture uplink messages from the target device. Once the uplink messages have been intercepted they are replayed, forcing the device to engage in redundant transmission and consuming unnecessary energy. One other method involves acknowledgement suppression, where an adversary jam or delays acknowledgement messages from the network server, prompting the device to retransmit repeatedly and consume excessive power. Additionally, an attacker may flood the network with downlink messages during the device's short receive window, forcing it to remain in an active state longer than necessary, further depleting the battery. In their research, [Mikhaylov et al., 2019](#) have demonstrated that such energy attacks are feasible and can cause substantial energy loss in affected devices.

According to [Cheong et al., n.d.](#), Class B devices provide scheduled reception slots in addition to the basic Class A functionality. They periodically wake up at synchronised intervals by synchronised beacons sent from the gateway. This class balances power efficiency and responsiveness, making it less vulnerable than Class A devices but still at risk under targeted depletion attacks.

Attackers can manipulate beacon signals by transmitting counterfeit beacons, causing devices to become desynchronised and expend additional energy attempting to resynchronise with the legitimate network. The impact of an DoS attack has been measured by [Nafees et al., n.d.](#), where an attacker continuously sends signals that the device interprets as network commands, preventing it from entering a low-power state.

According to [The Things Network, n.d.](#), Class C devices maintain continuous receive windows, allowing for immediate downlink communication but resulting in higher energy consumption. The nature of Class C devices is mains-powered due to their constant power requirements. If an attacker tries to target one of these devices the outcome could lead to potential system instability and increased operational costs. One way to attack is to send a continuous stream of downlink messages, occupying the device's receiver and processor, leading to potential overheating or failure. Another way is to implement a jamming attack and to transmit signals on the device's operating frequency to cause interference, forcing the device to expend additional energy in error correction and retransmission.

Addressing these vulnerabilities requires robust security measures, such as implementing secure session key management, stronger frame counter validation mechanisms, and enforcing authentication for adaptive data rate settings. Intrusion detection could also be implemented to identify abnormal transmission patterns that indicate energy depletion.

Chapter 4 - Simulations

4.1 Scenario 1: Jamming attack

This section presents both a visual and practical demonstration of a jamming attack. A simulated jamming environment will be created using Python to illustrate the attack's mechanics. Additionally, an Arduino MKR WAN 1310 board, a DHT temperature sensor, and an X000016 antenna will be used for practical experimentation. The objective is to demonstrate how an attacker can disrupt LoRa communication by deploying a secondary malicious device as a jammer, interfering with legitimate transmissions and compromising network reliability.

A jamming attack works by emitting interference signals on the same frequency as the legitimate transmission, effectively preventing the legitimate device from sending data to the gateway.

Disclaimer: The following experiment was conducted for educational purposes only. Any form of jamming radio frequency without authorisation is illegal in many countries.

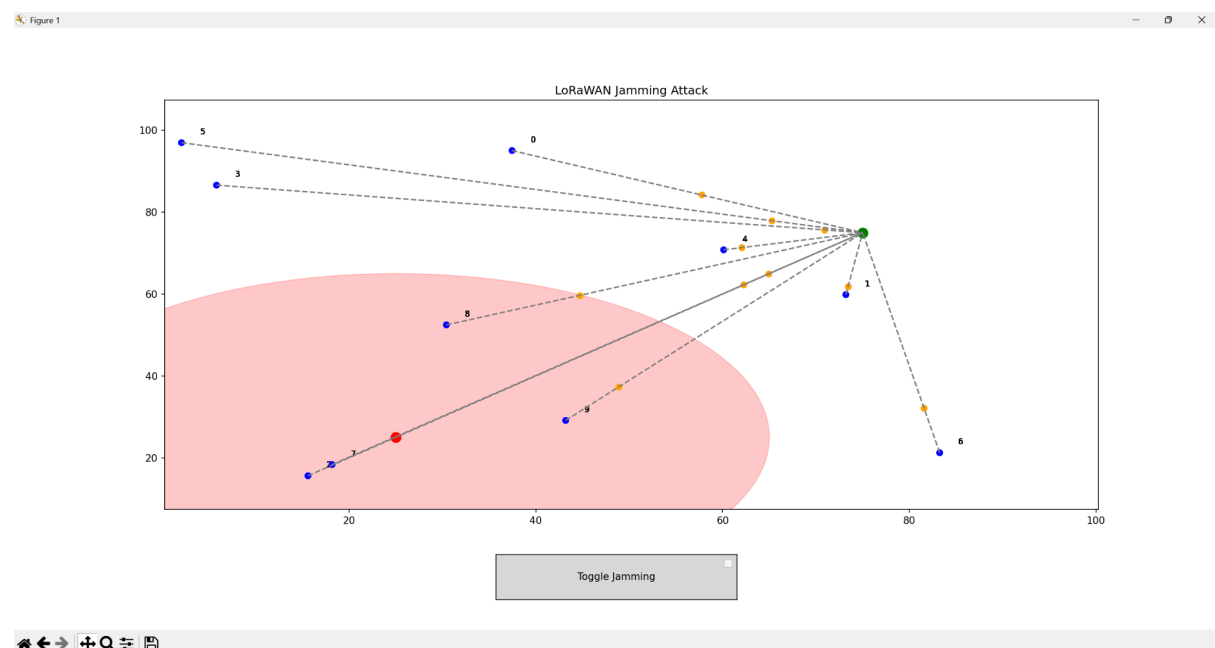


Figure 3: Python Jamming Environment

The nodes are represented as blue circles, with packets being transmitted visualised as yellow dots moving from the nodes towards the gateway, depicted as a green circle. The jammer device is depicted in red, and the jammer's range is shown as a red shaded area around the jammer. This implementation utilises the 'numpy' library for numerical operations, including generating node positions and computing distances efficiently. 'Matplotlib' library is used for visualising the LoRaWAN network, animating packet transmissions, and updating

node statuses in real-time. Additionally, the 'matplotlib.widgets' module enables interactive control, allowing users to toggle the jamming attack and observe its effects dynamically.

When the jammer is inactive, the nodes are able to transmit the data packets without any interference. Each node sends packets to the gateway, which processes them as expected. In this state, the network operates smoothly, and all nodes contribute to the packet flow.

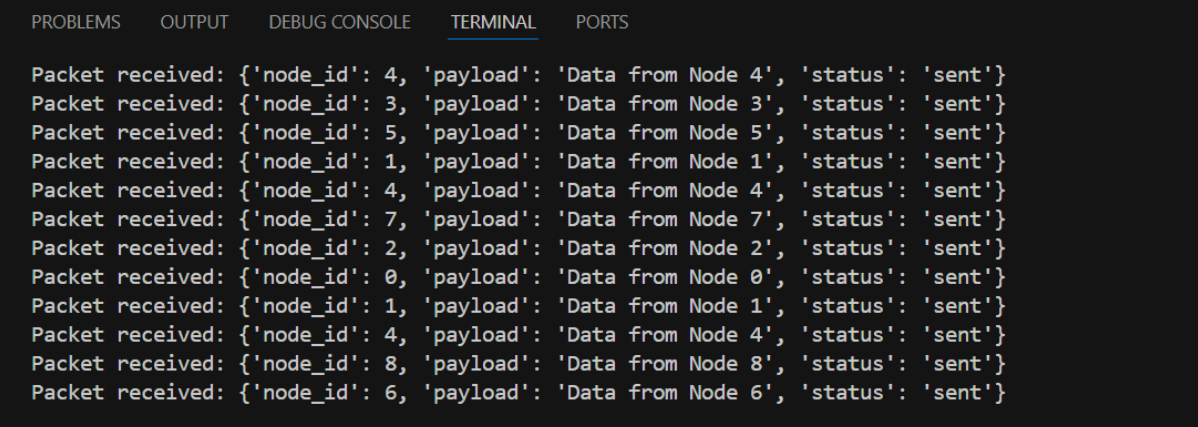
A terminal window with a dark background and light-colored text. At the top, there are five tabs: 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is selected and underlined), and 'PORTS'. The terminal displays a series of 12 log entries, each starting with 'Packet received:' followed by a JSON object containing 'node_id', 'payload', and 'status'. The payloads are 'Data from Node 4', 'Data from Node 3', 'Data from Node 5', 'Data from Node 1', 'Data from Node 4', 'Data from Node 7', 'Data from Node 2', 'Data from Node 0', 'Data from Node 1', 'Data from Node 4', 'Data from Node 8', and 'Data from Node 6'. All statuses are 'sent'.

Figure 4: Terminal Output of the Nodes

When the jammer is activated, it disrupts the communication between the nodes and the gateway. Nodes that fall within the jammer's range are unable to send their data packets, as the jamming signal interferes with their transmission. This blockage is visually represented in the simulation by the change in behaviour of the nodes affected: they stop transmitting their yellow packets, and their status is displayed in the terminal as "jammed". The simulation output, shown in the terminal, also provides feedback on which nodes are impacted by the jammer's interference.

Arduino setup

LoRa Sender (Node 1 - Temperature Sensor)

- Components: Arduino MKR WAN 1310, DHT temperature sensor, X000016 antenna.
- Functionality: Reads temperature and humidity values from the DHT sensor and transmits the data wirelessly to the gateway using LoRa.
- Libraries: LoRa.h library is used to establish LoRa communication, allowing the sender to transmit data packets wirelessly. Adafruit_AHTX0.h is responsible for interfacing with the temperature and humidity sensor. It enables the program to initialise the sensor, read environmental data, such as temperature and humidity, and use this data for transmission. SHA256.h provides cryptographic functionality, specifically SHA-256 hashing. It is used in the 'generateHMAC()' function to create a hashed message authentication code (HMAC) by combining the sensor data with a secret key. This ensures data integrity and authenticity, making it more difficult for an attacker to modify or forge transmitted data.

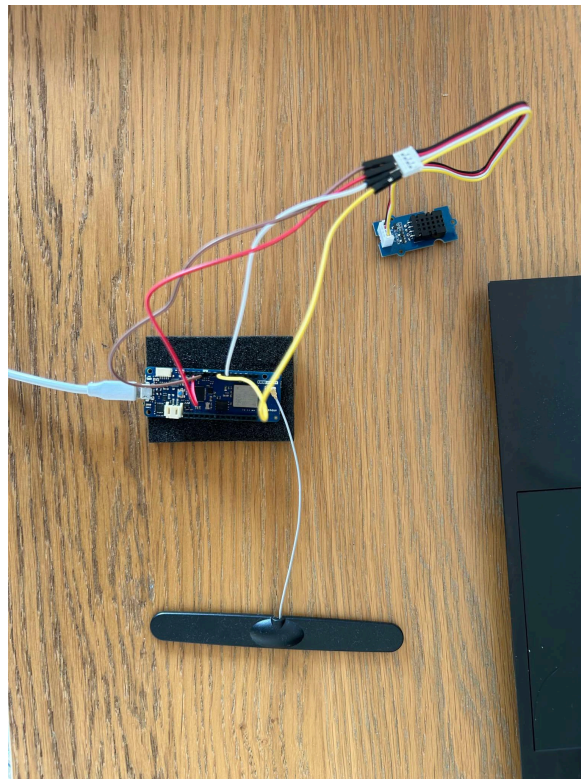


Figure 6: LoRa Sender (Node 1 - Temperature Sensor)

LoRa Receiver (Gateway)

- Components: Arduino MKR WAN 1310, X000016 antenna.
- Functionality: Receives LoRa packets from the sender and displays the temperature and humidity data on the Serial Monitor of the Arduino IDE.
- Libraries: LoRa.h library is used to enable communication between sender and gateway providing functions for parsing incoming packets and reading

received data. SHA256.h is used to verify the integrity and authenticity of received messages. The 'generateHMAC()' function utilises SHA-256 to compute a hashed message authentication code (HMAC) by combining the received data with a secret key. The gateway compares the received HMAC with the expected HMAC to determine whether the data has been tampered with, rejecting packets with mismatched HMACs.

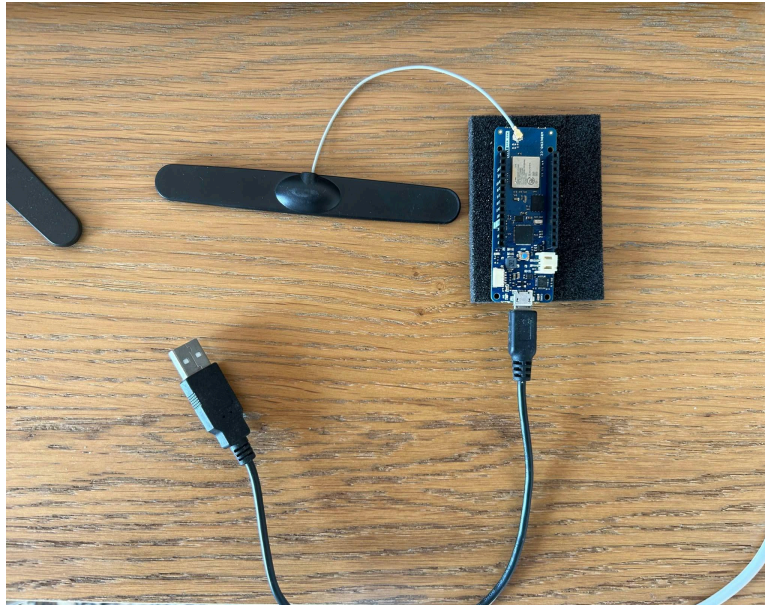


Figure 7: LoRa Receiver (Gateway)

LoRa Jammer (Attacker)

- Components: Arduino MKR WAN 1310, X000016 antenna.
- Functionality: Transmits continuous LoRa packets on the same frequency as the node and gateway, causing interference and preventing legitimate data transmission between the sender and gateway. The jammer floods the channel, resulting in packet loss and communication failure.
- Libraries: LoRa.h it is used to initialise the LoRa module, configure transmission settings, and continuously sending random packets to disrupt legitimate LoRaWAN communication. The 'LoRa.setTxPower(20)' function sets the transmission power to maximum, while 'LoRa.setSpreadingFactor(7)' minimises the spreading factor to maximise transmission speed. The 'LoRa.beginPacket()' and 'LoRa.endPacket()' functions handle sending random packets, effectively flooding the channel to interfere with other LoRa communications, simulating a jamming attack.

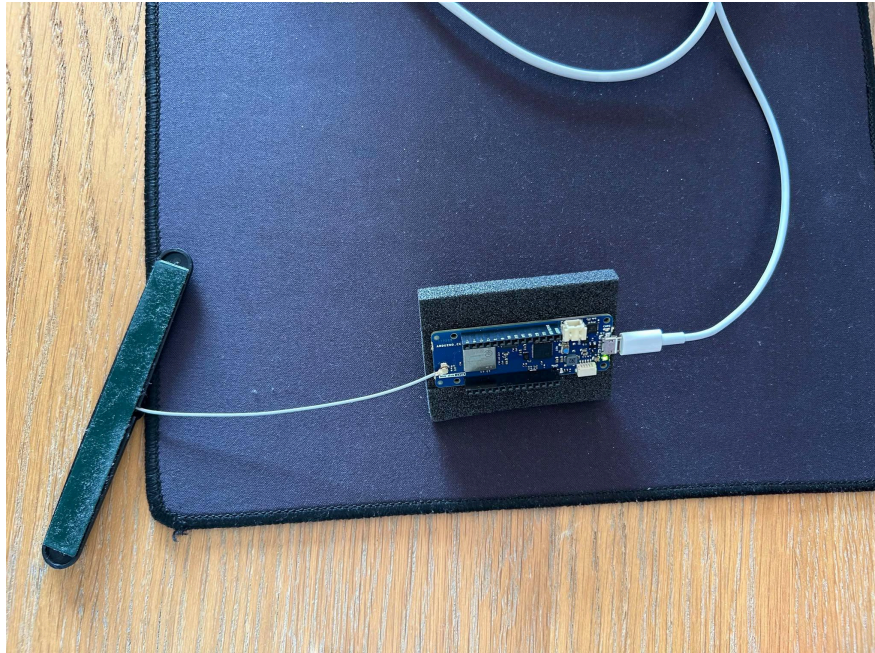


Figure 8: LoRa Jammer (Attacker)

Expected outcome

Under normal conditions, the sensor node successfully transmits authenticated data packets to the gateway, which verifies and displays valid readings. When the jammer activates, it takes over the channel by transmitting high-power random packets. This prevents legitimate transmissions from being received, causing the gateway to process only corrupted payloads that fail HMAC validation.

```

16:37:55.393 -> Data sent: Temperature:21.92, Humidity:49.14
16:37:55.393 -> Sender HMAC: 3c5074d96c01d50bccceadfc626b4195bddald35c1076be810dece7a9cf91afc
16:38:05.632 -> Data sent: Temperature:21.89, Humidity:49.07
16:38:05.632 -> Sender HMAC: 8b9e586a33481fb9a652567b40b7bcd2b9d246ad27264037cec0e6733de48d48
16:38:15.842 -> Data sent: Temperature:21.88, Humidity:52.55
16:38:15.842 -> Sender HMAC: 99ed043c0ad2f80b2ea8e90665a05d24f6d8d66112b8221b1057d8b420366004
16:38:26.015 -> Data sent: Temperature:21.90, Humidity:58.12
16:38:26.015 -> Sender HMAC: 2cfa0a8084d374df397ff9fa52e4821f7c0d29866f77924abdad38deb114a45a
16:38:36.247 -> Data sent: Temperature:21.92, Humidity:55.80
16:38:36.247 -> Sender HMAC: d9b8d082684374169cbd279d02a89b8aeab4b4b2dba22746673c7a7690ad8b6b
16:38:46.474 -> Data sent: Temperature:21.93, Humidity:54.89
16:38:46.474 -> Sender HMAC: def7008a787f5312feb458bad66cf872871006b009189ac069e2c323fc842749
16:38:56.698 -> Data sent: Temperature:21.92, Humidity:51.77
16:38:56.698 -> Sender HMAC: 876dcf62c017e030c193db0f048b41e55aealcbeb2391772e8940f223cbcf8d0
16:39:06.894 -> Data sent: Temperature:21.88, Humidity:50.42
16:39:06.894 -> Sender HMAC: 48e893f3d8546bcled63f75408afdd76811ebce5697ad962cfd6bac7a2a3e361
16:39:17.114 -> Data sent: Temperature:21.88, Humidity:50.24
16:39:17.114 -> Sender HMAC: 121c67d9d0db771f9ba24a4482e9cb55516f6da6678b817e9c514dd63dbeaa9e
16:39:27.319 -> Data sent: Temperature:21.84, Humidity:51.11

```

Figure 9: Node 1 Serial Monitor Output (Sender)

```

16:37:47.130 -> LoRa gateway ready!
16:37:53.996 -> Received Data: Temperature:21.92,Humidity:49.14
16:37:53.996 -> Received HMAC: 3c5074d96c01d50bcceeadfc626b4195bddald35c1076be810dece7a9cf91afc
16:37:53.996 -> Expected HMAC: 3c5074d96c01d50bcceeadfc626b4195bddald35c1076be810dece7a9cf91afc
16:37:53.996 -> Received valid data: Temperature:21.92,Humidity:49.14
16:38:04.212 -> Received Data: Temperature:21.89,Humidity:49.07
16:38:04.212 -> Received HMAC: 8b9e586a33481fb9a652567b40b7bcd2b9d246ad27264037cec0e6733de48d48
16:38:04.212 -> Expected HMAC: 8b9e586a33481fb9a652567b40b7bcd2b9d246ad27264037cec0e6733de48d48
16:38:04.212 -> Received valid data: Temperature:21.89,Humidity:49.07
16:38:14.429 -> Received Data: Temperature:21.88,Humidity:52.55
16:38:14.429 -> Received HMAC: 99ed043c0ad2f80b2ea8e90665a05d24f6d8d66112b8221b1057d8b420366004
16:38:14.429 -> Expected HMAC: 99ed043c0ad2f80b2ea8e90665a05d24f6d8d66112b8221b1057d8b420366004
16:38:14.429 -> Received valid data: Temperature:21.88,Humidity:52.55
16:38:14.466 -> Received Data: Temperature:21.88,Humidity:52.55
16:38:14.466 -> Received HMAC: 99ed043c0ad2f80b2ea8e90665a05d24f6d8d66112b8221b1057d8b420366004
16:38:14.466 -> Expected HMAC: 99ed043c0ad2f80b2ea8e90665a05d24f6d8d66112b8221b1057d8b420366004
16:38:14.466 -> Received valid data: Temperature:21.88,Humidity:52.55
16:38:24.611 -> Received Data: Temperature:21.90,Humidity:58.12
16:38:24.611 -> Received HMAC: 2cfa0a8084d374df397ff9fa52e4821f7c0d29866f77924abdad38deb114a45a
16:38:24.646 -> Expected HMAC: 2cfa0a8084d374df397ff9fa52e4821f7c0d29866f77924abdad38deb114a45a

```

Figure 10: Gateway Serial Monitor Output

```

16:41:09.375 -> LoRa jammer ready!
16:41:09.422 -> Jamming packet sent: LVQDYQYKFDBXNQD
16:41:09.560 -> Jamming packet sent: QUHYDJAEEBZQMTBL
16:41:09.688 -> Jamming packet sent: CABWGMSCRNOIAFTL
16:41:09.869 -> Jamming packet sent: FPCUQFFAXOZQEGXM
16:41:10.024 -> Jamming packet sent: WGGLKHVXTDHNZQAN
16:41:10.164 -> Jamming packet sent: KYPRBWTEAZDAFEQX
16:41:10.305 -> Jamming packet sent: TIJJKWEAZQGMPLO
16:41:10.435 -> Jamming packet sent: HYXRUTOJVBZLLQGJ
16:41:10.617 -> Jamming packet sent: AIDBTQIBYGDZCXKU
16:41:10.756 -> Jamming packet sent: JVGHWBMJJMBPKSN
16:41:10.883 -> Jamming packet sent: ZKGZGILUIGGPKZWH
16:41:11.053 -> Jamming packet sent: AETCLRCYXCSIXSUT
16:41:11.181 -> Jamming packet sent: JMRMVQLYBSJNHNH
16:41:11.358 -> Jamming packet sent: QYFHYSZWGPSVNHNN
16:41:11.500 -> Jamming packet sent: GDNJZJYPQCFLNZTR
16:41:11.629 -> Jamming packet sent: HCFGBKAKZXAMALSU

```

Figure 11: Jammer Serial Monitor Output

```

16:41:08.061 -> Received Data: Temperature:22.08, Humidity:52.18
16:41:08.061 -> Received HMAC: 4ec42e107e03193d9d306001fe1f407c07ac4998aa937fc429679a0254273cf4
16:41:08.061 -> Expected HMAC: 4ec42e107e03193d9d306001fe1f407c07ac4998aa937fc429679a0254273cf4
16:41:08.061 -> Received valid data: Temperature:22.08, Humidity:52.18
16:41:08.130 -> Invalid packet format! Missing comma.
16:41:08.439 -> Invalid packet format! Missing comma.
16:41:08.439 -> Invalid packet format! Missing comma.
16:41:08.880 -> Invalid packet format! Missing comma.
16:41:08.880 -> Invalid packet format! Missing comma.
16:41:09.320 -> Invalid packet format! Missing comma.
16:41:09.454 -> Invalid packet format! Missing comma.
16:41:10.217 -> Invalid packet format! Missing comma.
16:41:11.096 -> Invalid packet format! Missing comma.
16:41:11.725 -> Invalid packet format! Missing comma.
16:41:11.725 -> Invalid packet format! Missing comma.
16:41:11.992 -> Invalid packet format! Missing comma.
16:41:12.290 -> Invalid packet format! Missing comma.
16:41:12.327 -> Invalid packet format! Missing comma.
16:41:12.463 -> Invalid packet format! Missing comma.
16:41:12.596 -> Invalid packet format! Missing comma.
16:41:12.888 -> Invalid packet format! Missing comma.
16:41:13.485 -> Invalid packet format! Missing comma.
16:41:13.823 -> Invalid packet format! Missing comma.

```

Figure 12: Gateway Serial Monitor Output after Jamming activation

The current implementation offers basic message authentication through HMAC, making it suitable for small-scale private LoRa networks. However, it lacks critical security features required for robust or commercial deployments. The hardcoded secret key poses a significant risk if compromised, and the absence of payload encryption leaves sensitive data exposed. Additionally, the fixed spreading factor (SF7) and static sync word make the system vulnerable to jamming and spoofing attacks. The analysis below reveals critical gaps that undermine security in real-world deployments.

Aspect	Sender Node	Gateway (Receiver)	Jammer	LoRaWAN Security Equivalent	Improvements
Authentication	HMAC with secret key	Validates HMAC	None	Uses AES-128 (OTAA/ABP)	To use AES-128 encryption + session keys.
Integrity	SHA-256 HMAC	Verifies HMAC	None	MIC (Message Integrity Code)	To rely on LoRaWAN's MIC instead of custom HMAC.
Confidentiality	None (plaintext data)	None	None	AES-128 payload encryption	To enable LoRaWAN payload encryption.
Anti-jamming	None	None	Actively jams	Frequency hopping, adaptive data rate	To implement ADR and listen-before-talk (LBT)
Key management	Hardcoded secret key	Hardcoded secret key	N/A	Dynamic session keys (OTAA)	To store keys securely (HSM/SE) and rotate periodically.

Packet Format	Data + HMAC concatenated	Parses data + HMAC	Random packets	Standardised LoRaWAN frames	To follow LoRaWAN packet structure (MAC Header, MAC, payload, MIC)
Sync Protection	Custom sync word (0xF3)	Same sync word (0xF3)	Same sync word	DevAddr + NwkSKey/AppSKey	To use LoRaWAN's 32-bit DevAddr instead of sync words.
Physical Layer	Fixed SF7	Fixed SF7	Fixed SF7	Adaptive Data Rate (ADR)	To enable ADR to dynamically adjust SF/power

4.2 Scenario 2: Replay attack

The goal of this simulation is to create a replay attack in a LoRaWAN network and to demonstrate how an attacker intercepts and retransmits packets to compromise data integrity. The scenario is built in Python, having a sender node transmitting sensor data, a malicious attacker that attempts to intercept the data packets, and a gateway that is responsible for validating incoming data packets.

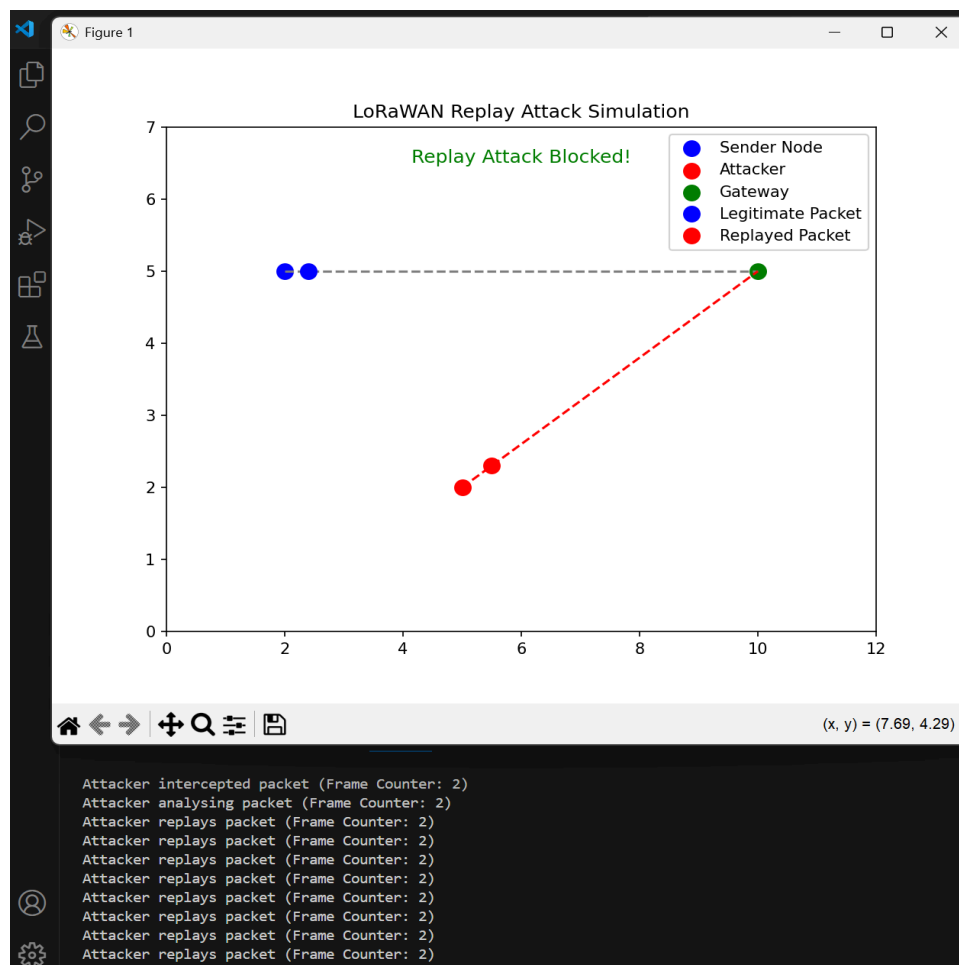


Figure 13: Python Replay Attack environment

In real-world operation, the sender transmits data packets, each containing a monotonically increasing frame counter to ensure freshness. These packets travel along a predefined path (visualised as a gray dashed line) from the sender to the gateway. The attacker, positioned within communication range, passively monitors transmissions and intercepts packets with a 60% probability. Upon successful interception, the attacker stores the packet's frame counter for later misuse.

During the replay phase, the attacker retransmits a previously captured packet to the gateway. A secure LoRaWAN network should reject it by verifying the frame counter. However, if this check fails, the replayed packet is accepted, causing data duplication or false sensor readings.

The simulation visually represents this process with blue packets for legitimate transmissions and red for replay attempts. The gateway's response is displayed as: "Replay Attack Succeeded!" (if the attack is successful), "Replay Attack Blocked!" (if security works), or "Monitoring..." (for normal operation).

This highlights the need for frame counter validation in LoRaWAN security. While simplified, the model demonstrates the risk of replay attacks and the importance of cryptographic freshness mechanisms.

Arduino setup

Similar to the Jamming attack setup, but in this scenario, LoRa Attacker functions as the Packet Interceptor. Instead of simply disrupting communication, it captures and replays transmitted packets, mimicking legitimate data to deceive the network.

LoRa Sender (Node 1 - Temperature Sensor)

- The Arduino MKR WAN 1310 board, connected to a DHT temperature sensor, reads the current temperature values. These values are packaged into a LoRa message and transmitted wirelessly using the LoRa module and antenna to the gateway.
- Libraries: LoRa.h library is used to establish LoRa communication, Adafruit_AHTX0.h is responsible for interfacing with the temperature and humidity sensor, SHA256.h is used to verify the integrity and authenticity of received messages.

LoRa Receiver (Gateway)

- The second Arduino MKR WAN 1310 board, functioning as the gateway, receives the LoRa messages sent by the sender node. Upon receiving a packet, it displays the temperature data on the Serial Monitor of the Arduino IDE for visualisation.
- Libraries: LoRa.h library is used to enable communication between sender and gateway. SHA256.h is used to verify the integrity and authenticity of received messages.

LoRa Interceptor (Attacker)

- The third Arduino MKR WAN 1310 board is used as a Packet Interceptor. It captures and replays transmitted LoRa packets on the same frequency as the sender and gateway, mimicking legitimate data. This interception leads to data duplication and disruption of normal communication.
- Libraries: LoRa.h library is used to simulate the replay attack on the LoRaWAN network. The LoRa module initialises at 868 MHz with a sync word for compatibility. It captures packets using 'LoRa.parsePacket()' and 'LoRa.read()', storing them in capturedPacket. The attacker replays the packet multiple times with 'LoRa.beginPacket()' and 'LoRa.endPacket()', controlled by REPLAY_COUNT. After each transmission, a delay regulates timing.

```
17:00:54.022 -> Data sent: Temperature:22.24, Humidity:50.78
17:00:54.022 -> Sender HMAC: 58a9775c4b426844c9f771b4a01ce2dedc12cca9bd6685cb3827db2a14f94eb0
17:01:04.207 -> Data sent: Temperature:22.21, Humidity:49.93
17:01:04.207 -> Sender HMAC: 4fc4ddab8639770a424c90f8cc7637ed5fe0aa2e9b9fc85399ffe47300fd3438
17:01:14.420 -> Data sent: Temperature:22.20, Humidity:49.10
17:01:14.420 -> Sender HMAC: 04e8e53874a0321d0f1862be89950096281128440cbfdc632894b2b439dc2a6e
17:01:24.678 -> Data sent: Temperature:22.17, Humidity:49.01
17:01:24.678 -> Sender HMAC: 5539a9b3e694e581a21b37f45b57ff5782651ddb3e0f9be2433f08a477cac67
17:01:34.858 -> Data sent: Temperature:22.17, Humidity:51.62
17:01:34.858 -> Sender HMAC: a8c231963b487d9213e4c017e0bc7b638d8902c79032cb05b588ce41861f91c4
17:01:45.067 -> Data sent: Temperature:22.15, Humidity:51.16
17:01:45.067 -> Sender HMAC: 3bdd4199617291b96c5098da2f32e936ec4318b7c64dbd481d5a64f4bb95fab0
17:01:55.291 -> Data sent: Temperature:22.15, Humidity:51.05
17:01:55.291 -> Sender HMAC: 86c98f2b97db8abd5f5042d21b8f3c9179d02e0420030f62b256bea941506cca
17:02:05.483 -> Data sent: Temperature:22.14, Humidity:51.90
17:02:05.483 -> Sender HMAC: 6e99cba22c8291e6ee6b1625616fa4148086ca48392ccb31e0fb02a5d7c4735c
17:02:15.695 -> Data sent: Temperature:22.11, Humidity:50.38
17:02:15.695 -> Sender HMAC: 1ba3ab6eb9d73d7f189723bb227b1528a911fced605b3b850270ce72ec531fb
```

Figure 14: Node 1 Serial Monitor Output

```
20:49:59.329 -> LoRa single-packet replay attacker ready!
20:49:59.783 -> Packet captured: Temperature:22.37, Humidity:58.80, c449a61af7f730bfa846c4bd0027e38bedeb82aeac1b3ee5d5b1dd264e3a3865
20:49:59.784 -> Preparing to replay packet 5 times...
20:50:01.995 -> Replayed packet (1/5): Temperature:22.37, Humidity:58.80, c449a61af7f730bfa846c4bd0027e38bedeb82aeac1b3ee5d5b1dd264e3a3865
20:50:03.132 -> Replayed packet (2/5): Temperature:22.37, Humidity:58.80, c449a61af7f730bfa846c4bd0027e38bedeb82aeac1b3ee5d5b1dd264e3a3865
20:50:04.315 -> Replayed packet (3/5): Temperature:22.37, Humidity:58.80, c449a61af7f730bfa846c4bd0027e38bedeb82aeac1b3ee5d5b1dd264e3a3865
20:50:05.509 -> Replayed packet (4/5): Temperature:22.37, Humidity:58.80, c449a61af7f730bfa846c4bd0027e38bedeb82aeac1b3ee5d5b1dd264e3a3865
20:50:06.681 -> Replayed packet (5/5): Temperature:22.37, Humidity:58.80, c449a61af7f730bfa846c4bd0027e38bedeb82aeac1b3ee5d5b1dd264e3a3865
20:50:07.683 -> Replay complete. Ready to capture a new packet.
20:50:07.683 -> Packet captured: Temperature:22.37, Humidity:58.80, c449a61af7f730bfa846c4bd0027e38bedeb82aeac1b3ee5d5b1dd264e3a3865
20:50:07.683 -> Preparing to replay packet 5 times...
20:50:09.888 -> Replayed packet (1/5): Temperature:22.37, Humidity:58.80, c449a61af7f730bfa846c4bd0027e38bedeb82aeac1b3ee5d5b1dd264e3a3865
20:50:11.050 -> Replayed packet (2/5): Temperature:22.37, Humidity:58.80, c449a61af7f730bfa846c4bd0027e38bedeb82aeac1b3ee5d5b1dd264e3a3865
20:50:12.228 -> Replayed packet (3/5): Temperature:22.37, Humidity:58.80, c449a61af7f730bfa846c4bd0027e38bedeb82aeac1b3ee5d5b1dd264e3a3865
20:50:13.426 -> Replayed packet (4/5): Temperature:22.37, Humidity:58.80, c449a61af7f730bfa846c4bd0027e38bedeb82aeac1b3ee5d5b1dd264e3a3865
20:50:14.558 -> Replayed packet (5/5): Temperature:22.37, Humidity:58.80, c449a61af7f730bfa846c4bd0027e38bedeb82aeac1b3ee5d5b1dd264e3a3865
20:50:15.600 -> Replay complete. Ready to capture a new packet.
20:50:15.600 -> Packet captured: Temperature:22.37, Humidity:58.80, c449a61af7f730bfa846c4bd0027e38bedeb82aeac1b3ee5d5b1dd264e3a3865
20:50:15.600 -> Preparing to replay packet 5 times...
20:50:17.763 -> Replayed packet (1/5): Temperature:22.37, Humidity:58.80, c449a61af7f730bfa846c4bd0027e38bedeb82aeac1b3ee5d5b1dd264e3a3865
20:50:18.957 -> Replayed packet (2/5): Temperature:22.37, Humidity:58.80, c449a61af7f730bfa846c4bd0027e38bedeb82aeac1b3ee5d5b1dd264e3a3865
20:50:20.105 -> Replayed packet (3/5): Temperature:22.37, Humidity:58.80, c449a61af7f730bfa846c4bd0027e38bedeb82aeac1b3ee5d5b1dd264e3a3865
20:50:21.316 -> Replayed packet (4/5): Temperature:22.37, Humidity:58.80, c449a61af7f730bfa846c4bd0027e38bedeb82aeac1b3ee5d5b1dd264e3a3865
20:50:22.485 -> Replayed packet (5/5): Temperature:22.37, Humidity:58.80, c449a61af7f730bfa846c4bd0027e38bedeb82aeac1b3ee5d5b1dd264e3a3865
```

Figure 15: Interceptor (Attacker) Serial Monitor Output

```

Output  Serial Monitor x
Message (Enter to send message to 'Arduino MKR WAN 1310' on 'COM3')

20:49:39.388 -> Received valid data: Temperature:22.35,Humidity:58.72
20:49:49.605 -> Received Data: Temperature:22.37,Humidity:58.77
20:49:49.605 -> Received HMAC: 48a8ca753a8f9c202501c1e9f7012bd906d8919c6513a0299fda41e94aef2b92
20:49:49.605 -> Expected HMAC: 48a8ca753a8f9c202501c1e9f7012bd906d8919c6513a0299fda41e94aef2b92
20:49:49.605 -> Received valid data: Temperature:22.37,Humidity:58.77
20:49:59.822 -> Received Data: Temperature:22.37,Humidity:58.80
20:49:59.822 -> Received HMAC: c449a61af7f730bfa846c4bd0027e38bedeb82aeac1b3ee5d5b1dd264e3a3865
20:49:59.822 -> Expected HMAC: c449a61af7f730bfa846c4bd0027e38bedeb82aeac1b3ee5d5b1dd264e3a3865
20:49:59.822 -> Received valid data: Temperature:22.37,Humidity:58.80
20:50:02.025 -> Received Data: Temperature:22.37,Humidity:58.80
20:50:02.025 -> Received HMAC: c449a61af7f730bfa846c4bd0027e38bedeb82aeac1b3ee5d5b1dd264e3a3865
20:50:02.025 -> Expected HMAC: c449a61af7f730bfa846c4bd0027e38bedeb82aeac1b3ee5d5b1dd264e3a3865
20:50:02.025 -> Received valid data: Temperature:22.37,Humidity:58.80
20:50:03.240 -> Received Data: Temperature:22.37,Humidity:58.80
20:50:03.240 -> Received HMAC: c449a61af7f730bfa846c4bd0027e38bedeb82aeac1b3ee5d5b1dd264e3a3865
20:50:03.240 -> Expected HMAC: c449a61af7f730bfa846c4bd0027e38bedeb82aeac1b3ee5d5b1dd264e3a3865
20:50:03.240 -> Received valid data: Temperature:22.37,Humidity:58.80
20:50:04.404 -> Received Data: Temperature:22.37,Humidity:58.80
20:50:04.404 -> Received HMAC: c449a61af7f730bfa846c4bd0027e38bedeb82aeac1b3ee5d5b1dd264e3a3865
20:50:04.404 -> Expected HMAC: c449a61af7f730bfa846c4bd0027e38bedeb82aeac1b3ee5d5b1dd264e3a3865
20:50:04.404 -> Received valid data: Temperature:22.37,Humidity:58.80
20:50:05.569 -> Received Data: Temperature:22.37,Humidity:58.80
20:50:05.569 -> Received HMAC: c449a61af7f730bfa846c4bd0027e38bedeb82aeac1b3ee5d5b1dd264e3a3865
20:50:05.569 -> Expected HMAC: c449a61af7f730bfa846c4bd0027e38bedeb82aeac1b3ee5d5b1dd264e3a3865
20:50:05.569 -> Received valid data: Temperature:22.37,Humidity:58.80
20:50:06.738 -> Received Data: Temperature:22.37,Humidity:58.80
20:50:06.738 -> Received HMAC: c449a61af7f730bfa846c4bd0027e38bedeb82aeac1b3ee5d5b1dd264e3a3865
20:50:06.738 -> Expected HMAC: c449a61af7f730bfa846c4bd0027e38bedeb82aeac1b3ee5d5b1dd264e3a3865
20:50:06.738 -> Received valid data: Temperature:22.37,Humidity:58.80
20:50:06.777 -> Received Data: Temperature:22.37,Humidity:58.80
20:50:06.777 -> Received HMAC: c449a61af7f730bfa846c4bd0027e38bedeb82aeac1b3ee5d5b1dd264e3a3865
20:50:06.777 -> Expected HMAC: c449a61af7f730bfa846c4bd0027e38bedeb82aeac1b3ee5d5b1dd264e3a3865
20:50:06.777 -> Received valid data: Temperature:22.37,Humidity:58.80
20:50:09.946 -> Received Data: Temperature:22.37,Humidity:58.80
20:50:09.946 -> Received HMAC: c449a61af7f730bfa846c4bd0027e3\..e0"Fu ;YbQjJ&6f!T
20:50:09.946 -> Expected HMAC: c449a61af7f730bfa846c4bd0027e38bedeb82aeac1b3ee5d5b1dd264e3a3865
20:50:09.946 -> Invalid HMAC! Packet rejected.
20:50:11.122 -> Received Data: Temperature:22.37,Humidity:58.80
20:50:11.122 -> Received HMAC: c449a61af7f730bfa846c4bd0027e38bedeb82aeac1b3ee5d5b1dd264e3a3865
20:50:11.122 -> Expected HMAC: c449a61af7f730bfa846c4bd0027e38bedeb82aeac1b3ee5d5b1dd264e3a3865
20:50:11.122 -> Received valid data: Temperature:22.37,Humidity:58.80
20:50:12.272 -> Received Data: Temperature:22.37,Humidity:58.80
20:50:12.272 -> Received HMAC: c449a61af7f730bfa846c4bd0027e38bedeb82aeac1b3ee5d5b1dd264e3a3865
20:50:12.272 -> Expected HMAC: c449a61af7f730bfa846c4bd0027e38bedeb82aeac1b3ee5d5b1dd264e3a3865

```

Figure 16: Gateway Serial Monitor Output after the Replay Attack

Outcome

The interceptor successfully captured and replayed legitimate packets, while the gateway processed and replayed packets as if they were new, leading to incorrect data being displayed.

The table below identifies critical security features in the Sender and Gateway, and provides countermeasures against replay attacks.

Aspect	Sensor Node	Gateway	Replay attacker	LoRaWAN Security	Improvements
--------	-------------	---------	-----------------	------------------	--------------

				Equivalent	
Authentication	HMAC with static key	HMAC with the same static key	None (impersonates sender)	AES-128 + OTAA dynamic session keys	To use OTAA or HMAC with rotating nonces
Data Freshness	No timestamps/sequence numbers	No freshness validation	Exploits lack of timestamps	Frame counters (FCnt) + MIC	To add millis() timestamps + sequence numbers
Encryption	None (plaintext)	None (plaintext)	Reds plaintext easily	AES-128 payload encryption	To implement AES-128 or use LoRaWAN stack
Integrity check	SHA-256 HMAC	Validates SHA-256 HMAC	Copies valid HMACs for replay	Built-in Message Integrity Code (MIC)	To use MIC instead of customised HMAC
Key Management	Hardcoded secret key	Hardcoded secret key	Only needs sync word (0xF3) to attack	Periodic session key rotation (OTAA/ABP)	To store keys in a dedicated cryptographic hardware chip to protect sensitive keys instead of hardcoding them in firmware.
Replay protection	None	None	Exploits missing replay defense	Network server validates FCnt	To maintain packet counters + blacklist duplicates
Physical Layer	Fixed SF7 (no anti-jamming)	Fixed SF7 (no anti-jamming)	Uses same SF7 to intercept	Adaptive Data Rate (ADR) + FHSS	To enable ADR + frequency hopping
Sync Mechanism	Static sync word (0xF3)	Static sync word (0xF3)	Matches sync word to eavesdrop	DevAddr + NwkSKey/AppSKey	To replace sync words with cryptographic addressing
Packet Structure	Data + HMAC concatenated	Splits data + HMAC	Copies exact packet structure	Standardised header/MAC commands	To follow LoRaWAN frame format
Regulatory Compliance	None (manual settings)	None (manual settings)	Violates radio laws	Built-in duty cycling + Listen-Before-Talk (LB)	To implement region-specific compliance features

4.3 Scenario 3: Battery Depletion Attack

This section describes the implementation of a Battery Depletion Attack simulation in a LoRa network using Python. This simulation demonstrates how an attacker can deplete the battery of LoRa nodes by continuously sending packets, forcing them to process and forward data. The attack targets a Class A device, which is more susceptible to energy depletion due to their power-saving mechanisms.

The Python program visualises the attack using 'matplotlib' and 'Tkinter', where nodes, the attacker, and the gateway are displayed. When the attack starts, packets are sent from the attacker to the nodes, reducing battery levels. The nodes then forward the packets to the gateway, further consuming energy.

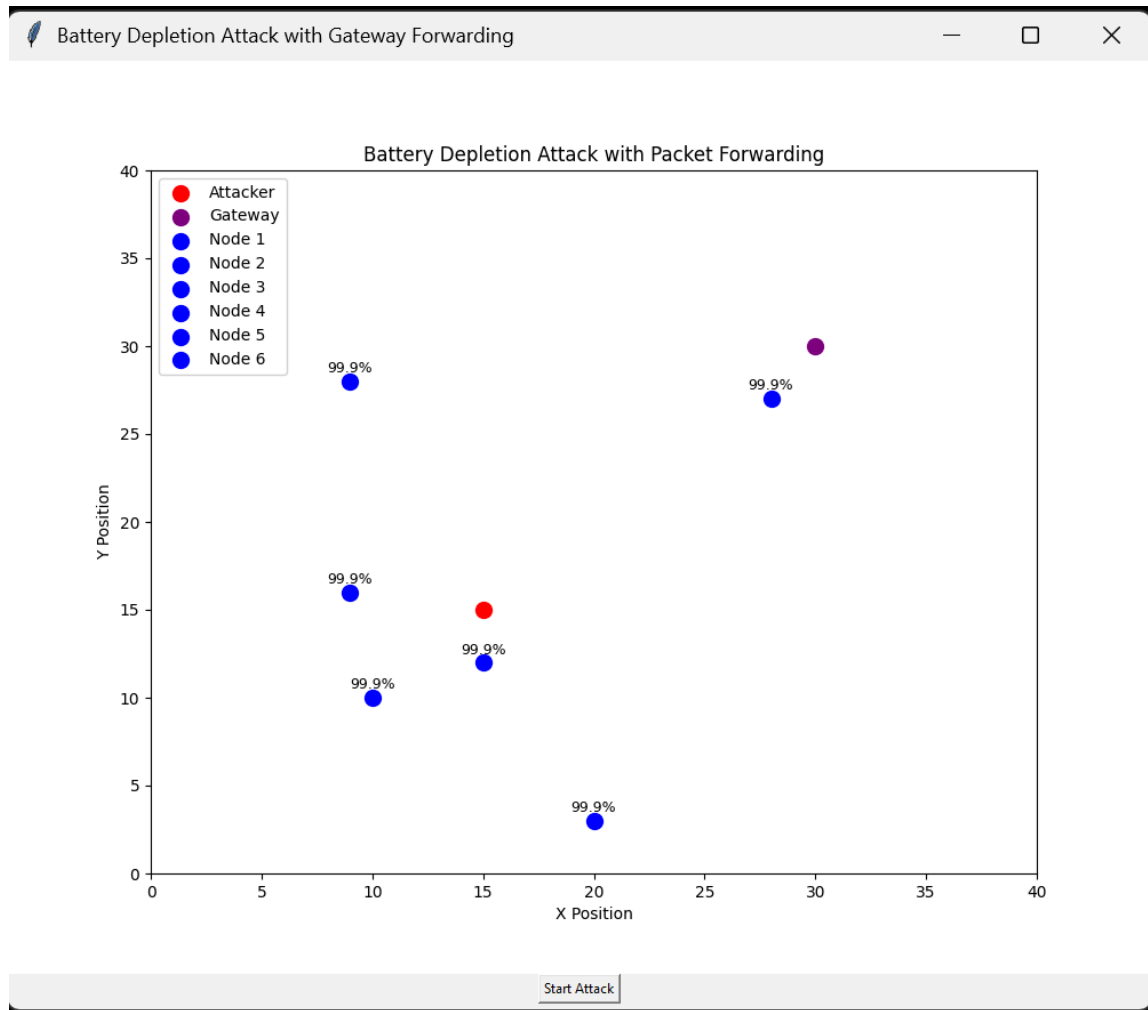


Figure 17: Python environment

Simulation Setup

Nodes: Represent LoRa devices with limited battery capacity. Each node is randomly positioned to simulate diverse environmental conditions.

Attacker: A malicious device that continuously transmits packets to drain the battery of nearby nodes.

Gateway: Represents the central LoRa gateway that receives and forwards packets from the nodes.

Simulation Logic

The attacker persistently sends packets to the nodes, compelling them to process and forward the data to the gateway. Each packet processed depletes a portion of the node's battery. The attack continues until a node's battery is fully exhausted, causing it to stop functioning.

Packet movement

- From Attacker to Nodes: Packets move progressively toward the nodes, simulating the transmission process.
- From Nodes to Gateway: Once a packet reaches a node, it is forwarded to the gateway for processing.

Visualisation

The simulation employs Matplotlib to visualise the positions of nodes, the attacker, the gateway, and the movement of packets. Battery levels of nodes are displayed as text labels, allowing real-time tracking of energy depletion.

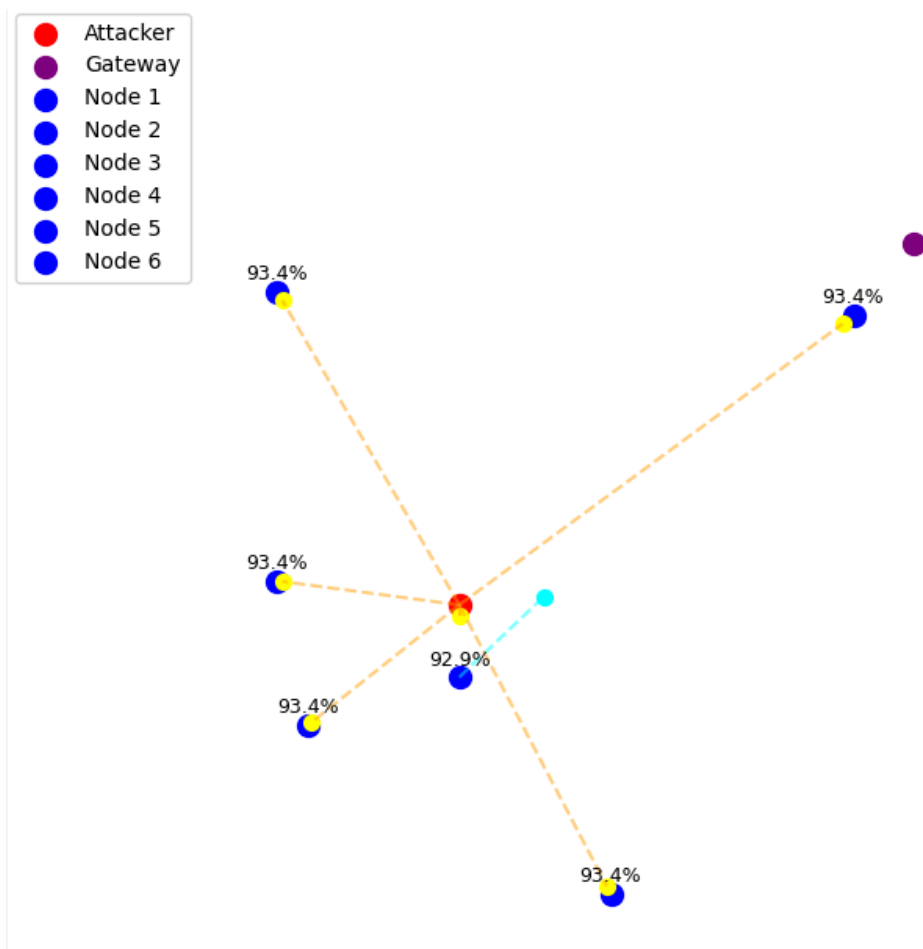
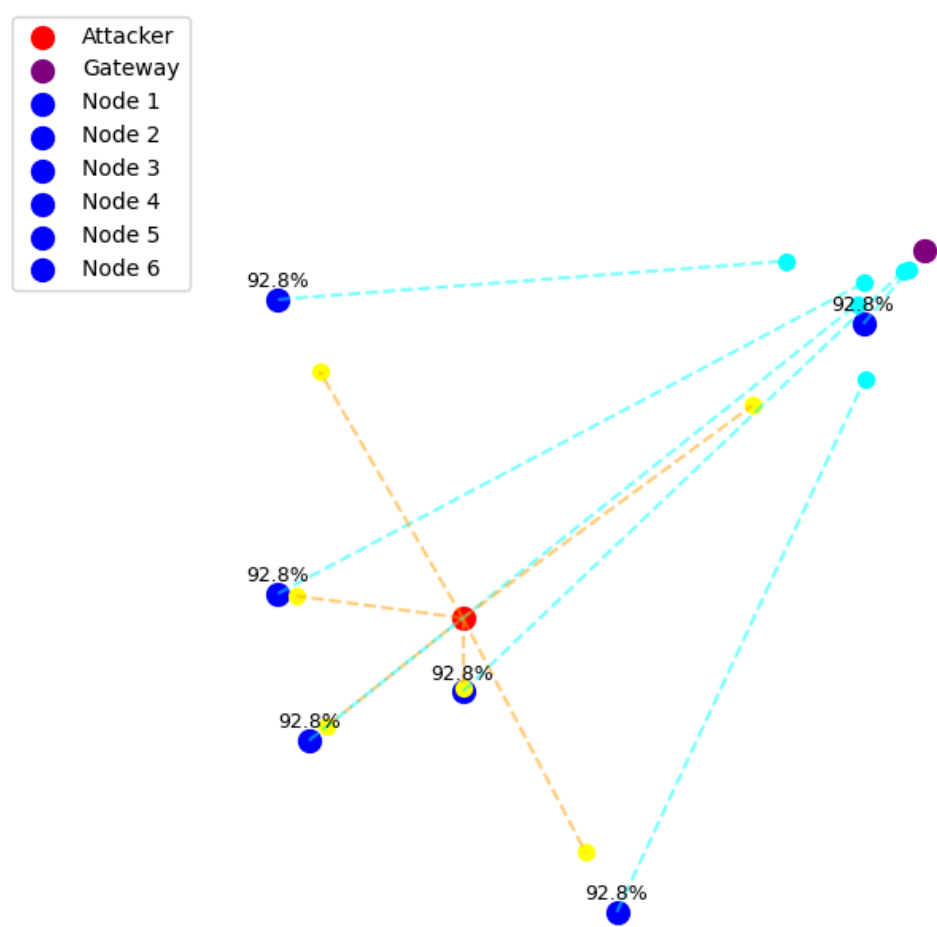


Figure 18: Packets being sent from the Attacker to the Nodes

Impact of the attack

Nodes gradually lose battery power with each received packet. As battery levels drop over time, nodes eventually deplete their energy reserves. A fully drained node ceases operation, demonstrating the effectiveness of the attack.



S

Figure 19: Packets being sent from the Nodes

Chapter 5 - Future Work

As LoRa technology continues to evolve, addressing challenges in scalability, security, and interoperability remains crucial. While advancements have improved its efficiency, significant research opportunities still exist. This final chapter explores potential directions for enhancing LoRa networks, with a focus on artificial intelligence (AI), post-quantum cryptography, and hybrid network integration.

The integration of AI and machine learning (ML) into LoRa networks offers transformative potential. Reinforcement learning algorithms could dynamically optimise transmission parameters such as spreading factors and bandwidth, mitigating packet collisions in dense urban deployments while preserving energy efficiency ([Adelantado et al., 2017](#)). Such adaptive systems would resolve scalability limitations inherent to static configurations, particularly in smart city applications where network loads fluctuate unpredictably. Concurrently, AI-powered anomaly detection systems could support security by analysing traffic patterns to identify threats like jamming or spoofing in real time. Predictive maintenance models, trained on historical device data, might further prevent hardware failure or battery degradation, reducing operational downtime in industrial IoT scenarios ([Dorri, Kanhere and Jurdak, n.d.](#)). Additionally, Zero-trust architecture (ZTE) could further enhance resilience by enforcing continuous device validation, thereby isolating compromised nodes ([Rose et al., 2020](#)).

LoRa's low-power, long-range capabilities position it as a cornerstone of emerging IoT ecosystems. Hybrid networks integrating LoRa with 5G or satellite connectivity could expand coverage in rural or maritime environments, addressing current limitations in reliability and bandwidth ([Mekki et al., 2019](#)). Ultra-low power edge devices embedded with TinyML models might also enable real-time decision-making in applications such as precision agriculture, reducing reliance on cloud-based processing ([Augustin et al., 2016](#)). However, interoperability remains a barrier as standardised frameworks are urgently needed to harmonise LoRaWAN with protocols like NB-IoT or MLoTy ([Sanchez-Iborra et al., 2018](#)).

The sociotechnical implications of AI-driven LoRa networks demand rigorous ethical security. Governance frameworks may address biases in predictive algorithms, particularly in high-stakes domains like healthcare or law enforcement ([Floridi et al., 2018](#)).

AI-driven optimisation, quantum-resistant cryptography, and interoperability address existing research gaps in LoRa while aligning with the evolving trends of IoT and Industry 4.0. Realising this vision necessitates collaborative efforts among academia, industry, and policymakers to position LoRa as a secure and scalable foundation for future connected systems.

References

1. IBM (2023). Internet of Things. [online] Ibm.com. Available at: <https://www.ibm.com/think/topics/internet-of-things>.
2. www.cogniteq.com. (2024). The History of Internet of Things (IoT) | Cogniteq. [online] Available at: <https://www.cogniteq.com/blog/history-iot-how-technology-evolving>.
3. Srivastava, S. (2024). Top IoT Applications in Agriculture, Healthcare, Smart Homes & Beyond (2025). [online] E&ICT Academy, IIT Kanpur - E&ICT Academy, IIT Kanpur. Available at: <https://eicta.iitk.ac.in/knowledge-hub/internet-of-things/iot-applications-smart-homes-healthcare-agriculture-and-industrial-iot>
4. Aharon Etengoff (2024). Where are LPWAN protocols used? [online] 5G Technology World. Available at: <https://www.5gtechnologyworld.com/where-are-lpwan-protocols-used/>.
5. Coursera. (2023). Edge Computing vs Cloud Computing: Differences and Use Cases. [online] Available at: <https://www.coursera.org/articles/edge-computing-vs-cloud-computing>.
6. Visit profile (2024). Architecture of IOT. [online] Blogspot.com. Available at: <https://nofailengineering.blogspot.com/2024/01/architecture-of-iot.html>
7. Switzer, K. (2025). The Great Range Debate: Wi-Fi vs. Bluetooth - Audio Champs. [online] Audio Champs. Available at: <https://audiochamps.com/which-has-more-range-wi-fi-or-bluetooth>
8. Adelantado, F., Vilajosana, X., Tuset-Peiro, P., Martinez, B., Melia-Segui, J. and Watteyne, T. (2017). Understanding the Limits of LoRaWAN. IEEE Communications Magazine, [online] 55(9), pp.34–40. doi:<https://doi.org/10.1109/mcom.2017.1600613>
9. Inpixon (n.d.). Chirp Spread Spectrum (CSS) for Positioning | Inpixon. [online] www.inpixon.com. Available at: <https://www.inpixon.com/technology/standards/chirp-spread-spectrum>.
10. The Things Network (n.d.). What are LoRa and LoRaWAN? [online] The Things Network. Available at: <https://www.thethingsnetwork.org/docs/lorawan/what-is-lorawan/>.
11. Data-alliance.net. (n.d.). LoRa: Long Range Wireless for Internet of Things (IOT): Frequency Bands, Antennas. [online] Available at: <https://www.data-alliance.net/blog/lora-long-range-wireless-for-internet-of-things-iot-frequency-bands/>.
12. de Carvalho Silva, J., Rodrigues, J.J.P.C., Alberti, A.M., Solic, P. and Aquino, A.L.L. (2017). LoRaWAN — A low power WAN protocol for Internet of Things: A review and opportunities. [online] IEEE Xplore. Available at: <https://ieeexplore.ieee.org/abstract/document/8019271>.

13. Smartbuildingproducts.co.uk. (2025). What is a LoRaWAN Gateway? - Smart Building Products. [online] Available at:
<https://www.smartbuildingproducts.co.uk/blogs/post/what-is-a-lorawan-gateway>
14. Al-Awami, L. (n.d.). Network-Coding-based Forwarding for LoRaWAN Gateways. [online] Available at: <https://arxiv.org/pdf/2205.03918>.
15. Yang, X., Karampatzakis, E., Doerr, C. and Kuipers, F. (n.d.). Security Vulnerabilities in LoRaWAN. [online] Available at:
<https://www.cyber-threat-intelligence.com/publications/loTDI2018-LoraWAN.pdf#page=2.42>.
16. Kuntke, F., Romanenko, V., Linsner, S., Steinbrink, E. and Reuter, C. (2022). LoRaWAN security issues and mitigation options by the example of agricultural IoT scenarios. Transactions on Emerging Telecommunications Technologies, 33(5). doi:<https://doi.org/10.1002/ett.4452>.
17. Alipio, M. and Bures, M. (2024). Current testing and performance evaluation methodologies of LoRa and LoRaWAN in IoT applications: Classification, issues, and future directives. Internet of things, 25, pp.101053–101053. doi:<https://doi.org/10.1016/j.iot.2023.101053>
18. Philip, M.S. and Singh, P. (2021). Energy Consumption Evaluation of LoRa Sensor Nodes in Wireless Sensor Network. 2021 Advanced Communication Technologies and Signal Processing (ACTS), [online] pp.1–4. doi:<https://doi.org/10.1109/acts53447.2021.9708341>.
19. Iplacement.net. (2025). The Role of Accurate Geolocation in Modern Urban Planning for Smarter, Sustainable Cities. [online] Available at:
<https://www.iplocation.net/the-role-of-accurate-geolocation-in-modern-urban-planning-for-smarter-sustainable-cities?>
20. Zhang, H., Song, Y., Yang, M. and Jia, Q. (2023). Modeling and Optimization of LoRa Networks under Multiple Constraints. Sensors, [online] 23(18), pp.7783–7783. doi:<https://doi.org/10.3390/s23187783>.
21. Jiang, Y., Fu, H., Hu, A. and Sun, W. (2021). A LoRa-Based Lightweight Secure Access Enhancement System. Security and Communication Networks, 2021, pp.1–16. doi:<https://doi.org/10.1155/2021/3530509>.
22. TEKTELIC (2023). Security in LoRaWAN| Tektelic Blog. [online] TEKTELIC. Available at: <https://tektelic.com/expertise/lorawan-security/>.
23. Gemalto, A. and And, S. (2017). LoRaWAN TM SECURITY A WHITE PAPER PREPARED FOR THE LoRa ALLIANCE™ FULL END-TO-END ENCRYPTION FOR IoT APPLICATION PROVIDERS PROPERTIES OF LoRaWAN™ SECURITY. [online] Available at:
https://lora-alliance.org/wp-content/uploads/2020/11/lorawan_security_whitepaper.pdf
24. The Things Network. (n.d.). Security. [online] Available at:
<https://www.thethingsnetwork.org/docs/lorawan/security/>.
26. Elsevier.com. (2025). Redirecting. [online] Available at:
<https://linkinghub.elsevier.com/retrieve/pii/S1874490721002172>

27. da Cruz, P.I., Suyama, R. and Loiola, M.B. (2021). Increasing key randomness in physical layer key generation based on RSSI in LoRaWAN devices. *Physical Communication*, [online] 49, p.101480.
doi:<https://doi.org/10.1016/j.phycom.2021.101480>.
28. Zia, U., McCartney, M., Scotney, B., Martinez, J. and Sajjad, A. (2022). A novel pseudo-random number generator for IoT based on a coupled map lattice system using the generalised symmetric map. *SN applied sciences*, 4(2).
doi:<https://doi.org/10.1007/s42452-021-04919-4>.
29. Fukushima, K., Marion, D., Nakano, Y., Facon, A., Shinsaku Kiyomoto and Sylvain Guilley (2020). Evaluation of Side-Channel Key-Recovery Attacks on LoRaWAN End-Device. *Communications in computer and information science*, pp.74–92.
doi:https://doi.org/10.1007/978-3-030-49443-8_4
30. Xu, J., Tang, Y., Wang, Y. and Wang, X. (2019). A Practical Side-Channel Attack of a LoRaWAN Module Using Deep Learning. [online] IEEE Xplore.
doi:<https://doi.org/10.1109/ICASID.2019.8925203>.
31. T. Perković, J. Šabić, K. Zovko and P. Šolić, "An Investigation of a Replay Attack on LoRaWAN Wearable Devices," 2023 IEEE International Mediterranean Conference on Communications and Networking (MeditCom), Dubrovnik, Croatia, 2023, pp. 45-49, doi: 10.1109/MeditCom58224.2023.10266648.
32. Dossa, A. and Mehdi, A.E. (2025). Impact of Reactive Jamming Attacks on LoRaWAN: a Theoretical and Experimental Study. [online] arXiv.org. Available at: <https://arxiv.org/abs/2501.18339>.
33. Haque, A. and Saifullah, A. (n.d.). Handling Jamming Attacks in a LoRa Network. [online] Available at: <https://saifullah.eng.wayne.edu/iotdi2024.pdf>
34. Michael (2024). What is LoRaWAN and How Does it Work? [online] Forest Rock. Available at: <https://www.forestrock.co.uk/what-is-lorawan/>
35. Cheong, P., Bergs, J., Hawinkel, C. and Famaey, J. (n.d.). Comparison of LoRaWAN Classes and their Power Consumption. [online] Available at: <https://www.famaey.eu/papers/cnf-sancheong2017a.pdf#page=2.13>
36. The Things Network. (n.d.). Device Classes. [online] Available at: <https://www.thethingsnetwork.org/docs/lorawan/classes/>.
37. Arduino Forum. (2020). Crypto library for Arduino. [online] Available at: <https://forum.arduino.cc/t/crypto-library-for-arduino/637908>
38. Adelantado, F., Vilajosana, X., Tuset-Peiro, P., Martinez, B., Melia-Segui, J. and Watteyne, T. (2017). Understanding the Limits of LoRaWAN. *IEEE Communications Magazine*, [online] 55(9), pp.34–40. doi:<https://doi.org/10.1109/mcom.2017.1600613>.
39. Dorri, A., Kanhere, S. and Jurdak, R. (n.d.). Blockchain in Internet of Things: Challenges and Solutions. [online] Available at: <https://arxiv.org/pdf/1608.05187>.
40. Rose, S., Borchert, O., Mitchell, S. and Connelly, S. (2020). Zero trust architecture. *Zero Trust Architecture*, [online] 800-207(800-207).
doi:<https://doi.org/10.6028/nist.sp.800-207>.
41. Mekki, K., Bajic, E., Chaxel, F. and Meyer, F. (2019). A comparative study of LPWAN technologies for large-scale IoT deployment. *ICT Express*, [online] 5(1), pp.1–7.
doi:<https://doi.org/10.1016/j.ict.2017.12.005>.

42. Augustin, M.A., Riley, M., Stockmann, R., Bennett, L., Kahl, A., Lockett, T., Osmond, M., Sanguansri, P., Stonehouse, W., Zajac, I. and Cobiac, L. (2016). Role of food processing in food and nutrition security. *Trends in Food Science & Technology*, [online] 56, pp.115–125. doi:<https://doi.org/10.1016/j.tifs.2016.08.005>.
43. Floridi, L., Cows, J., Beltrametti, M., Chatila, R., Chazerand, P., Dignum, V., Luetge, C., Madelin, R., Pagallo, U., Rossi, F., Schafer, B., Valcke, P. and Vayena, E. (2018). *AI4People - An Ethical Framework for a Good AI Society: Opportunities, Risks, Principles, and Recommendations*. [online] *papers.ssrn.com*. Available at: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3284141.

Appendix 1 - GitHub

6COM2018-0905-2024---Computer-Science-Project Public

main 1 Branch 0 Tags

Go to file Add file Code

iStefan20 Update README.md 65ee496 · 7 minutes ago 13 Commits

EDA Python Simulation	Create EDA Python Simulation	18 minutes ago
Gateway Arduino	Create Gateway Arduino	17 minutes ago
Jamming Attack Arduino	Create Jamming Attack Arduino	15 minutes ago
Jamming Attack Python Simulation	Create Jamming Attack Python Simulation	19 minutes ago
README.md	Update README.md	7 minutes ago
Replay Attack Arduino	Create Replay Attack Arduino	14 minutes ago
Replay Attack Python Simulation	Create Replay Attack Python Simulation	19 minutes ago
Temperature Sensor Arduino	Create Temperature Sensor Arduino	16 minutes ago

README

UNIVERSITY OF HERTFORDSHIRE Department of Computer Science

Modular BSc Honours in Computer Science

6COM2018-0905-2024 - Computer Science Project

Final Report April 2025

A Security Perspective on the LoRaWAN Protocol: Analysis and Countermeasures against Cyberattacks

Stefan Ilie

Supervised by: W. Fernando

About

A Security Perspective on the LoRaWAN Protocol: Analysis and Countermeasures against Cyberattacks

Readme

Activity

0 stars

1 watching

0 forks

Releases

No releases published

[Create a new release](#)

Packages

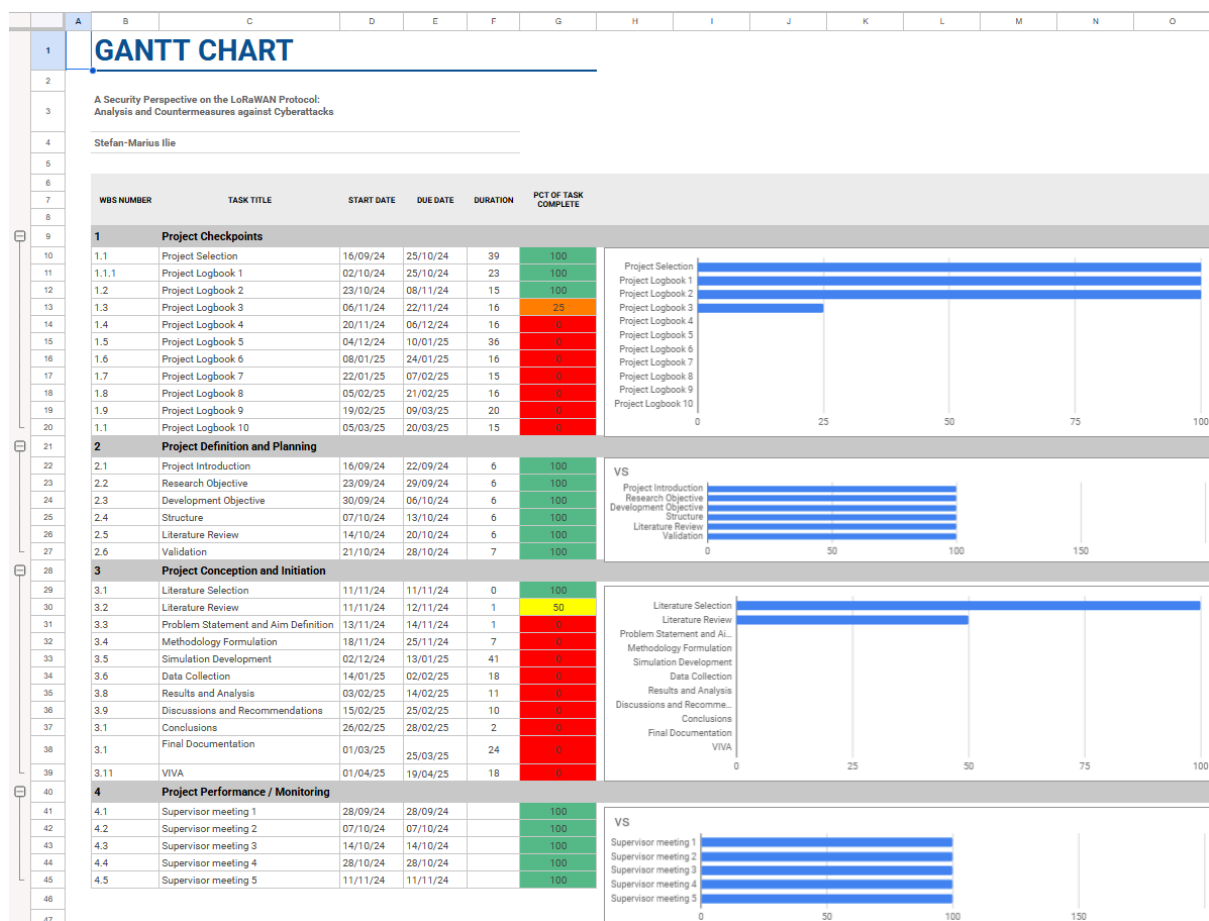
No packages published

[Publish your first package](#)

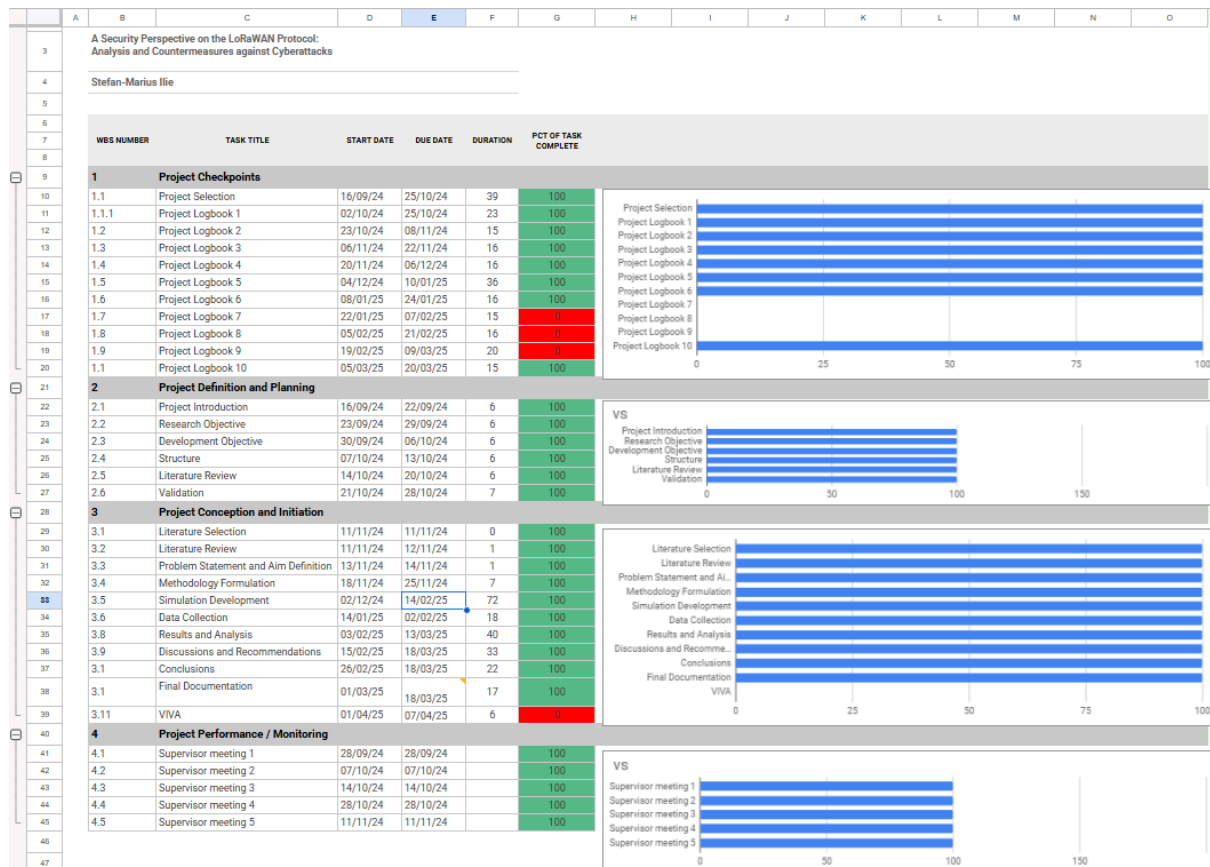
Github Repository

[iStefan20/6COM2018-0905-2024---Computer-Science-Project: A Security Perspective on the LoRaWAN Protocol: Analysis and Countermeasures against Cyberattacks](#)

Appendix 2 - Project Management Review



Initial Project Plan



Current Project Plan

The completion of this project followed a structured yet adaptive approach, shaped by both planned activities and unforeseen challenges. The initial phase began with a meeting with the supervisor, focusing on literature review and theoretical exploration. This stage was critical in developing an in-depth understanding of LoRaWAN security, particularly its vulnerabilities to jamming, replay, and energy depletion attacks. The initial plan was to conduct simulations using ns-3, a widely used network simulator for LoRaWAN. However, this approach had to be abandoned after a Trojan horse virus was found in the downloaded files, raising security concerns. This setback needed an alternative approach, leading to the decision to use the Arduino kit, a DHT temperature sensor, and an X000016 antenna. These components enabled a practical, hands-on method for assessing LoRaWAN security threats.

Following this transition, further research was conducted to refine the understanding of attack mechanisms and develop a methodology suitable for both practical and visual simulations. The practical implementation involved setting up the Arduino-based LoRaWAN environment and executing security attack scenarios, such as jamming and replay attacks, to observe their effects on network communication. Simultaneously, Python-based visual simulations were developed to illustrate these threats in a controlled environment. These simulations aimed to enhance comprehension by providing clear visual representations of how LoRaWAN security could be compromised.

Throughout the project, adjustments were necessary to refine both practical and visual simulations. The practical experimentation phase particularly required troubleshooting and modifications to ensure reliable data collection and analysis. Similarly, the visual simulations underwent multiple refinements to ensure clarity and effectiveness in demonstrating attack scenarios.

As these simulations were finalised, the project report was continuously updated to reflect the evolving methodologies and findings. The documentation process ensured that all key aspects, from theoretical insights to practical implementation and experimental results, were accurately captured. This iterative process allowed for a comprehensive evaluation of LoRaWAN security vulnerabilities while also identifying effective countermeasures.

This project demonstrates the importance of adaptability in research. After the initial ns-3 simulation plan was abandoned due to security concerns, the shift to practical experimentation using Arduino devices proved effective. Combining theoretical research with hands-on implementation and visual simulations allowed for a thorough assessment of LoRaWAN security threats, successfully meeting the project's aims.