**Chapter 1**

**Introduction**

* 1. **Context**

Lorawan (Low Power Wide Area Network) is a low power wireless communication technology designed to connect smart objects to sensor networks. This technology is based on the LoRa (Long Range) protocol and uses spread spectrum modulation to enable long range communication and low power consumption.

Lorawan networks are used in many fields, such as agriculture, industry, smart city, logistics and transportation. Sensors and devices connected to these networks can collect a large amount of data, such as temperature, humidity, location, energy consumption, etc., which can be analyzed to make informed decisions and optimize processes.

However, like any communication technology, Lorawan is susceptible to attack by malicious hackers. Attacks may aim at disrupting network operations, stealing sensitive data or injecting malicious data. Therefore, the security of Lorawan networks is a major concern for users of this technology.

In this context, the study of normal and attack traffic traces for Lorawan is essential to understand the network behavior and to identify potential security flaws. This study can help security experts to develop effective defense methods to protect Lorawan networks against malicious attacks.

* 1. **Goal**

The purpose of this thesis is to provide a simulation environment to study normal and attack traffic for LoRaWAN. On the basis of the simulation environment that abstracts component of a real LoRaWAN ( End device , Gateway , Network & Application Server ) we are able to generate, collect and analyze traces under normal operation and in the presence of a malicious gateway.

More specifically , the study of normal traffic under a simulated environment provides amongst many others :

* Cost effeciency : No extra hardware required to deploy, maintain and test a LoRaWAN network
* Realistic testing : The existing simulation environment simulates seamlessly most parameters and operating condition of real lorawan nodes

Secured testing : Performing security test on the simulated environment provides a controlled environment to produce reliable results and avoid potential signal interference and noise effect from a real environment .

Given the existing limited capability of the simulation environment that abstracts communication between the node simulator , gateway bridge and servers via semtech-udp protocol , the study provided is limited to generating UDP payloads containing realistic LoRa traces.

Consequently the malicious traffic generates is produced by abstracting gateway forwarder and end point devices traffic at transport layer level. The malicious traffic collected is correlated and analyzed against logs generated by node simulator and network server.

* 1. **Structure**

In chapter 2, we provide a description of LoRaWAN v1.0 , LoRa protocol , semtech-udp protocol . We also provide some background on general security features and dive in depth on aspects relevant to our work.

In chapter 3, we present a theoritical review and illustration of vulnerabilities and attacks specific to the LoRaWAN version of interest.

In chapter 4, we present the simulation environment, the architecture & its specifications. The attacks performed on the environment our then discussed.

In chapter 5, we explain our testing methodology and provide an analysis of the results obtained

Code and implementation design/choice is available at the following repository.

**Chapter 2**

**The LoRaWAN protocol stack**

In this section , we introduce the architecture of a LoRaWAN network , present the LoRa technology and provide a description of the network layer protocol used for communication between simulation components. The security features of LoRaWAN v1.0 are also reviewed.

**2.1. LoRaWAN Architecture**

Lorawan is a low-speed, long-range wireless communication protocol (LPWAN) that is specifically designed for low-power connected objects (IoT). The Lorawan architecture is divided into three main layers: the Physical Layer (PHY), the Data Link Layer (DLL) and the Network Layer.

PHY Layer: This layer is responsible for the transmission and reception of data on the radio channel. The radio technology used is called "chirp spread spectrum" (CSS) which allows to send signals on a large frequency band with minimum energy consumption. Lorawan uses free and regulated frequency bands, such as ISM (Industrial, Scientific and Medical) bands for data transmission.

Data link layer: This layer is responsible for encapsulating data in Lorawan frames before sending them on the network. It also manages headers, error detection and correction, as well as the fragmentation of data to send them on several packets.

Network layer: This layer is responsible for managing communications between nodes and Lorawan gateways. It allows to create networks of connected objects using a hierarchical routing system. Messages are sent from the nodes to the gateways, then forwarded to the Network Server which manages the network.

LoRaWAN networks are organized in a star-of-stars topology and composed of end-devices (motes) , gateways (concentrators) and a central network server.

- End-devices communicate with the Gateway via the PHY layer using LoRa over different frequency channels and data rates. The data rate ranges from 0.3 kbps to 50 kbps and can be managed along with RF by means of ADR scheme. This is in order to maximize battery power of end-devices & network capacity.

- Gateways receives uplink transmission from end-devices and perform frame encapsulation to forward the resulting payloads to network server. The transmission to towards the network server is performed via standard IP protocol.

- The LoRaWAN Network Server is a critical part of the LoRaWAN infrastructure that manages connectivity and communications between Internet of Things (IoT) nodes and downstream applications. The Network Server is responsible for configuring nodes, monitoring the status of the communication link, managing security, and managing data traffic between nodes and applications.

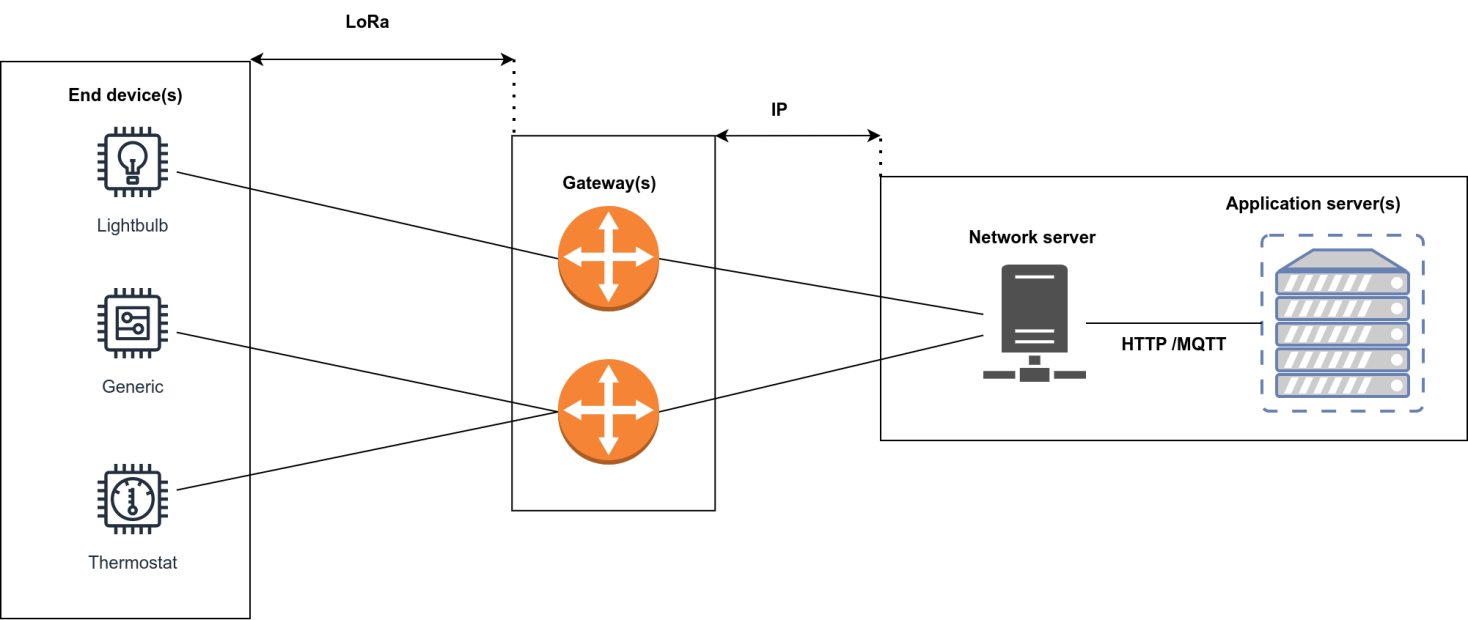


Figure 1. LoRaWAN Architecture

**2.2. LoRa Technology**

LoRaTM is a radio modulation technology used for long-range , low-power wireless communications in IoT networks.

LoRa uses spread spectrum frequency modulation (FSK) which allows for long range transmissions while maintaining relatively low power usage.

A LoRa network has 3 end-device classifications :

Class A: In this class , end-devices allow bidirectional communications whereby each uplink transmission is followed by two short downlink receive windows.Class A devices have the lowest power consumption of the three classes.

Class B : End devices in class B allow for more receive slots. The end-device opens a receive window at scheduled interval by receiving a sychronized beacon from the gateway. This permits the server to have knowledge of listening state of end-device.

Class C : End devices in class C have continuous receive windows hence has the highest power utilization but lowest latency for end-to-end communication.

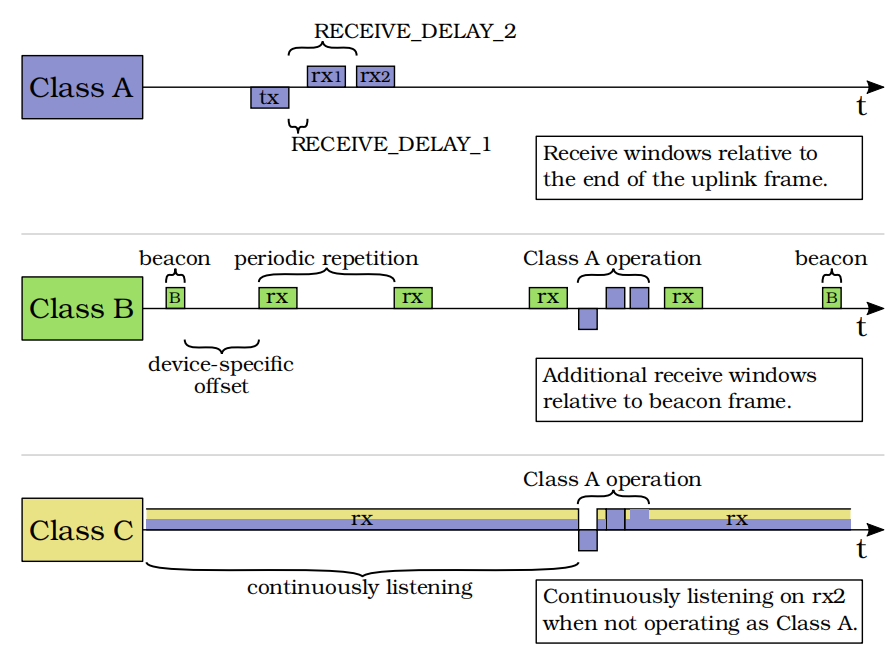


Figure 2. LoRaWAN Classes

**Decoding the LoRaWAN Protocol**

The LoRaWAN wireless protocol sits untop of LoRa physical layer.

**The LoRaWAN Packet Format**

LoRaWAN defines the subsequent layers of the OSI model on top of LoRa (OSI layer 1). It mainly operates at the data link layer Medium Access Control (MAC) (OSI layer 2) and it includes some elements of the network layer (OSI layer 3). The network layer handles join accept-request and packet-forwarding for nodes nodes in a LoRaWAN network. The LoRaWAN packet format further divides the network layer into MAC and application layers.

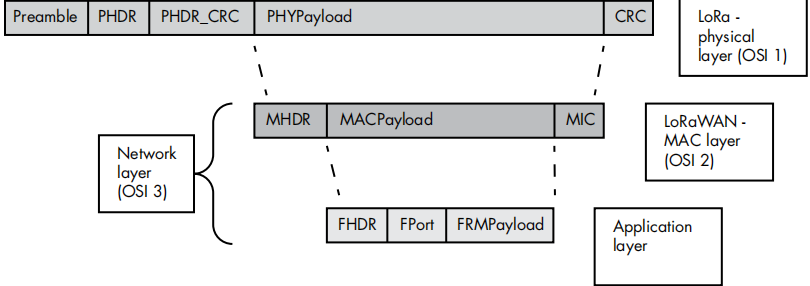


Figure 3. LoRaWAN Packet Format

**The LoRa physical layer** defines the radio interface, modulation scheme, and an optional CRC for error detection. It also carries the payload for the MAC layer. It has the following parts:

**Preamble** The radio preamble, which contains the synchronization function and defines the packet modulation scheme. The duration of the preamble is usually 12.25 Ts.

**PHDR** : The physical layer header, which contains information such as the payload length and whether the Physical Payload CRC is present.

**PHDR\_CRC** The CRC of the physical header (PHDR). The PHDR and PHDR\_CRC are 20 bits in total.

**PHYPayload** The physical layer payload, which contains the MAC frame.

**CRC** The optional 16-bit CRC of the PHYPayload. Messages sent from a network server to a node never contain this field for performance reasons.

**The LoRaWAN MAC** layer defines the LoRaWAN message type and the MIC, and it carries the payload for the application layer above. It has the following parts:

**MHDR** The MAC header (MHDR), which specifies the message type (MType) of the frame format and the version of the LoRaWAN specification used. The three-bit MType specifies which of the six different MAC message types we have: Join-Request, Join-Accept, unconfirmed data up/down, and confirmed data up/down. Up refers to data traveling from the node to the network server, and down indicates data traveling in the opposite direction.

**MACPayload** The MAC payload, which contains the application layer frame. For Join-Request (or Rejoin-Request) messages, the MAC payload has its own format and doesn’t carry the typical application layer payload.

**MIC** The four-byte MIC, which ensures data integrity and prevents message forgery. It’s calculated over all fields in the message (msg = MHDR | FHDR | FPort | FRMPayload) using the NwkSKey. Keep in mind that in the case of Join-Request and Join-Accept messages, we calculate the MIC

differently, because they’re a special type of MAC payload.

**The application layer** contains application-specific data and the end device address (DevAddr) that uniquely identifies the node within the current network. It has the following parts:

**FHDR** The frame header (FHDR), which contains the DevAddr, a frame control byte (FCtrl), a two-byte frame counter (FCnt), and zero to 15 bytes of frame options (FOpts). Note that FCnt increases every time a message is transmitted, and it’s used to prevent replay attacks.

**FPort** The frame port, used to determine whether the message contains only MAC commands (for example a Join-Request) or application specific data.

**FRMPayload** The actual data (for example, a sensor’s temperature value). These data are encrypted using the AppSKey.

**The Semtech UDP Packet Forwarder & Protocol**

A packet forwarder is a program running on the host of a real LoRa gateway and interfaces with the LoRa concentrator to pull and push packets, simultaneously interacting with the network server.

The testbed of our LoRaWAN simulation environment implements the Semtech UDP Packet Forwarder & Protocol.

The Semtech UDP Protocol is used for the communication between LoRaWAN gateways and network servers. It operates over UDP (User Datagram Protocol) and provides reliable data transmission over a wide area network[63] . It uses a specific packet structure and provides error detection and correction mechanisms to ensure the integrity of the transmitted data.

The Semtech UDP Protocol is an important part of the LoRaWAN stack, which includes several other layers such as the physical layer, data link layer, and application layer. It enables the efficient and secure transmission of data over long distances with low power consumption, making it well-suited for IoT applications that require long battery life and reliable connectivity.

Figures 4,5 & 6 [63] present the protocol construct for uplink and downlink communication between gateway and network server in a LoRaWAN network.

**Upstream communication**

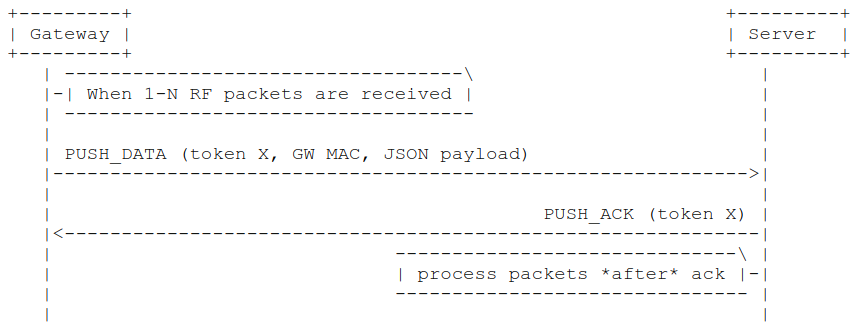


Figure 4. Upstream transmission RX

Upstream communication is made possible via 2 packet types :

**PUSH\_DATA** : The payload is composed of protocol version , random token , identifier = 0x00 , Gateway ID & a JSON Object .It forwards RF packets received from nodes to the Network Server.

**PUSH\_ACK** : The payload is composed of protocol version, random token (= PUSH\_DATA) and identifier = 0x01. It forward acknowledgment for the packets received by the Network Server.

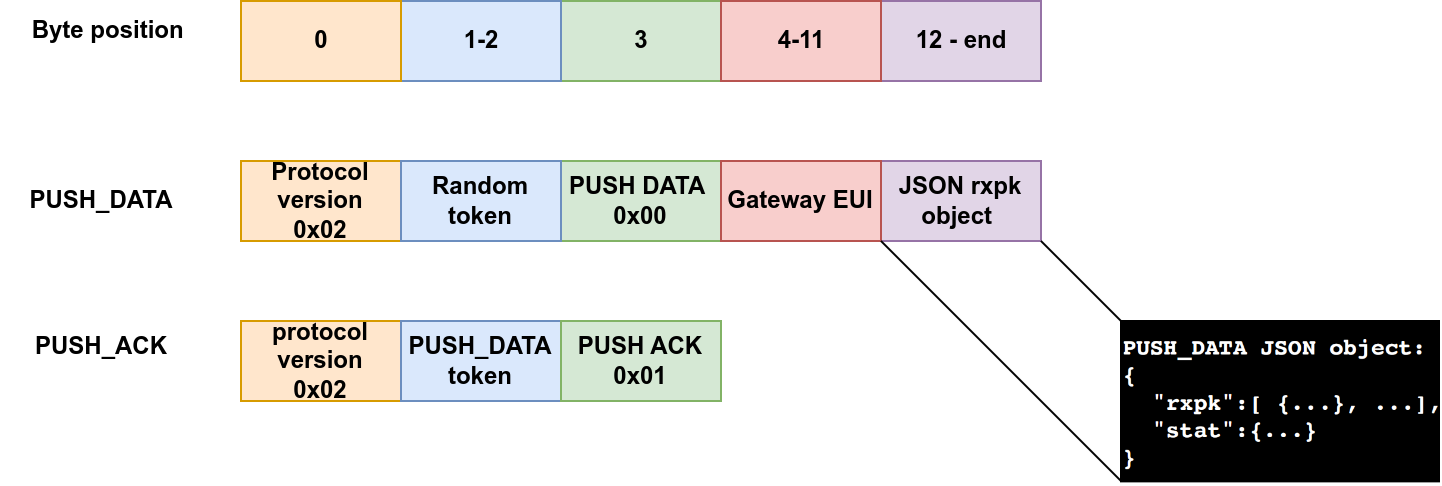


Figure 5. Upstream message formats

|  |  |  |
| --- | --- | --- |
| Name | Type | Function |
| datr | string/unsigned int | Data rate identifier |
| codr | string | ECC coding rate |
| rssi | signed float | RSSI in dBm |
| lsnr | signed float | Lora signal to noise ration in dB |
| size | unsigned int | RF packet payload size in bytes |
| data | string | Encoded RF packet payload in base64 |

Table 1. RXPK json object structure

**Downstream communication**

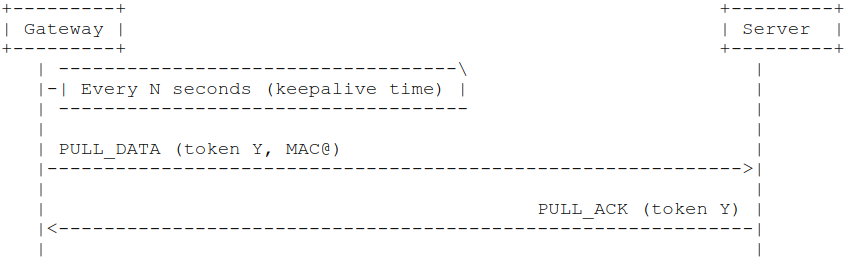


Figure 6. Keepalive transmission TX

**PULL\_DATA :** The payload is made up of protocol version , random token , identifier = 0x02 and gateway id. This packet type is used by the gateway to initiate and keep alive connection with the server.

**PULL\_ACK :** The payload is is made up of protocol version , random token & identifier = 0x04. This packet type is used by the server to confirm connectivity with connectivity.

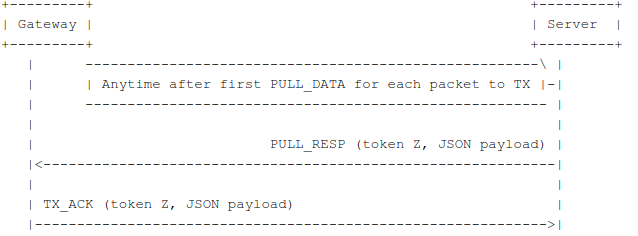


Figure 7. Downstream transmission TX

**PULL\_RESP :** That packet type is used by the server to send RF packets and associated metadata that will have to be emitted by the gateway.The packet type identifier is 0x03.

**TX\_ACK :** That packet type is used by the gateway to send a feedback to the server to inform if a downlink request has been accepted or rejected by the gateway.The datagram may optionnaly contain a JSON string to give more details on acknowledge. If no JSON is present (empty string), this means than no error occured. The packet type identifier is 0x05.

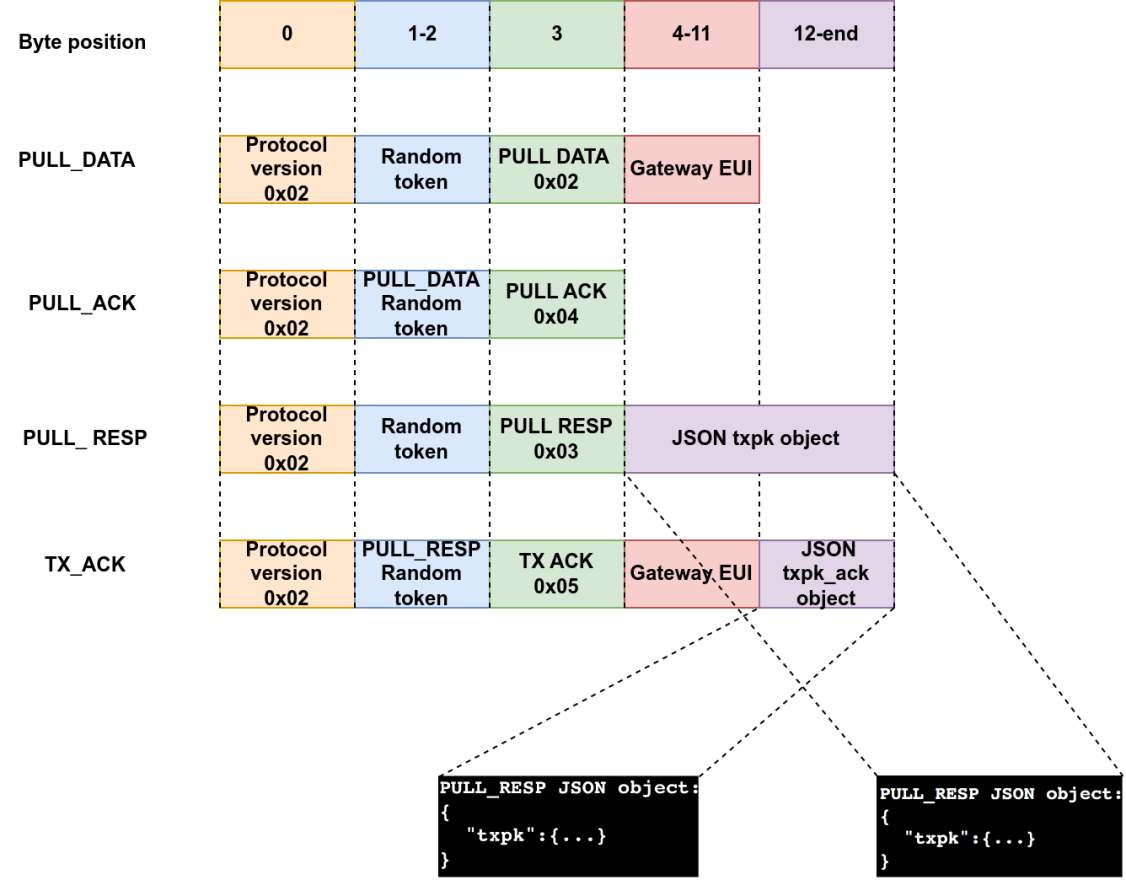
****

Figure 8.Downstream message formats

|  |  |  |
| --- | --- | --- |
| Name | Type | Function |
| codr | string | ECC coding rate k/n |
| fdev | unsigned int | FSK frequency deviation in Hz |
| ipol | bool | modu = LORA |
| prea | unsigned int | RF preamble size |
| size | unsigned int | RF packet payload size in bytes |
| data | string | Encoded RF packet payload in Base64 |
| ncrc | bool | No CRC generation by transmitter |

Table 2. TXPK json object structure

**LWN Simulator**

The LWN Simulator is a LoRaWAN mote simulator controlled via a web interface. It can be integrated to communicate with real LoRaWAN infrastructures or ad-hoc such as Chirpstack.

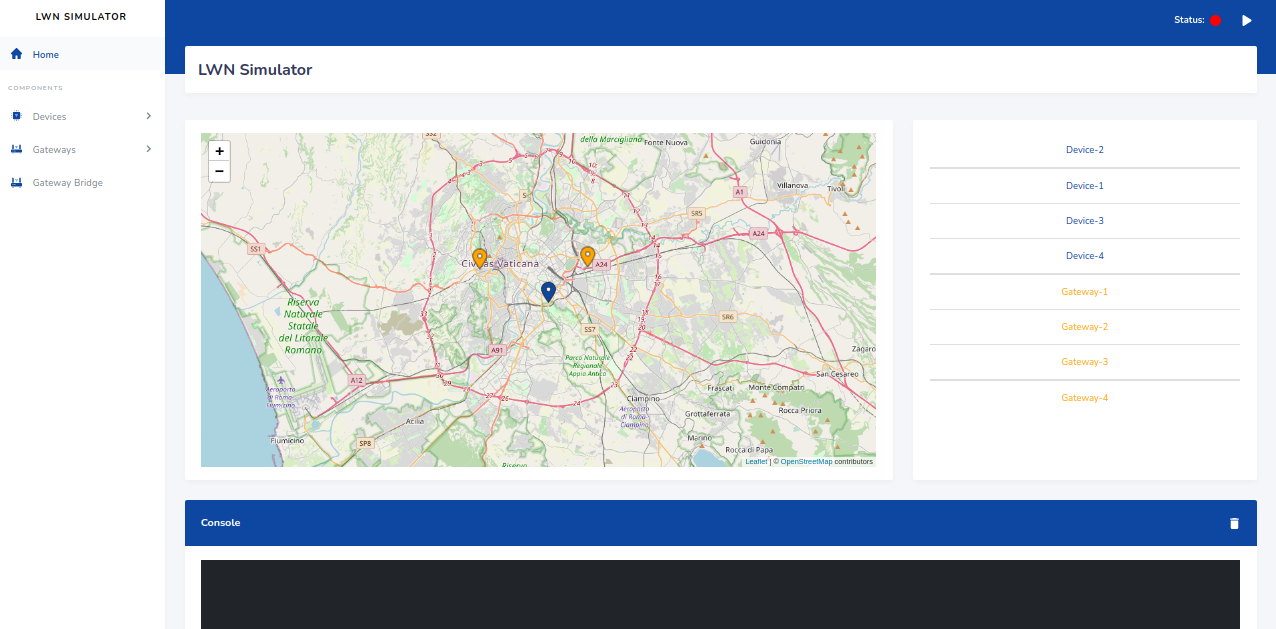


Figure 9. LWN Simulator

The simulator abstracts the following components together with specific characteristics :

### The device

* Based [specification LoRaWAN v1.0.3](https://lora-alliance.org/resource_hub/lorawan-specification-v1-0-3/);
* Supports all [LoRaWAN Regional Parameters v1.0.3](https://lora-alliance.org/resource_hub/lorawan-regional-parameters-v1-0-3reva/).
* Implements class A,C and partially even the B class;
* Implements ADR Algorithm;
* Sends periodically a frame that including some configurable payload;
* Supports MAC Command;
* Implements FPending procedure;
* It is possibile to interact with it in real-time;

### The forwarder

Receives the frames from devices, creates a RXPK object including them within and forwards to gateways.

### The gateway

There are two types of gateway:

* A virtual gateway that comunicates with a real gateway bridge (if it exists);
* A real gateway to which datagrams UDP are forwarded.

**Chirpstack**

ChirpStack provides open-source components for LoRaWAN networks. Together they form a ready-to-use solution including an user-friendly web-interface for device management and APIs for integration. The following components are provided:

[ChirpStack Gateway Bridge](https://www.chirpstack.io/gateway-bridge/): handles the communication with the LoRaWAN gateways

[ChirpStack Network Server](https://www.chirpstack.io/network-server/): a LoRaWAN Network Server implementation

[ChirpStack Application Server](https://www.chirpstack.io/application-server/): a LoRaWAN Application Server implementation



Figure 10. Chirpstack Architecture

**ChirpStack Gateway Bridge**

The [ChirpStack Gateway Bridge](https://www.chirpstack.io/gateway-bridge/) sits between the Packet Forwarder and MQTT broker. It transforms the Packet Forwarder format (like the [Semtech UDP Packet Forwarder protocol](https://github.com/Lora-net/packet_forwarder/blob/master/PROTOCOL.TXT)) into a data-format used by the ChirpStack components.

**ChirpStack Network Server**

The [ChirpStack Network Server](https://www.chirpstack.io/network-server/) is a LoRaWAN Network Server, responsible for managing the state of the network. It has knowledge of device activations on the network and is able to handle join-requests when devices want to join the network.

**ChirpStack Application Server**

The [ChirpStack Application Server](https://www.chirpstack.io/application-server/) is a LoRaWAN Application Server, compatible with the ChirpStack Network Server. It provides a web-interface and APIs for management of users, organizations, applications, gateways and devices.

**On the security in LoRaWAN 1.0**

Our analysis and implementation of normal and attack traffic is focused on OTAA procedure. In this subsection we elaborate on the existing security mechanisms that provide authentication, key exchange and data protection between entities in the simulated network.

**End-device Activation**

The LoRaWAN 1.0 specification defines two methods through which an end-device can connect and authenticate to a LoRaWAN network. The ABP ( Activation by Personalization) and OTAA ( Over The Air Activation). ABP relies on storing the session keys (NwkSkey & AppSkey ) and DevAddr directly into the end-device. OTAA on the other hand generates the 2 session keys without on both ends of the network. Figure 9 & 10 [12] present the message flow diagram for ABP and OTAA activation modes respectively.

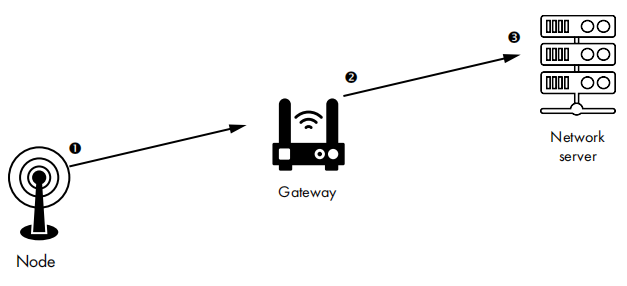


Figure 11. ABP message flow

The OTAA process is described in details in the following paragraphs.

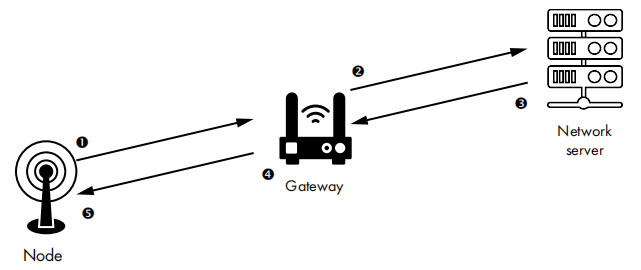


Figure 12. OTAA message flow

**Join procedure**

OTAA mode is implemented via a join procedure through which end-devices emits a join request messages towards the network server which replies with a join accept response. The join accept message contains the necessary session keys to protect data confidentiality & integrity. The NwkSkey ( Network Session Key) and AppSkey (Application Session Key) are derived from a device specific AppKey to encrypt.

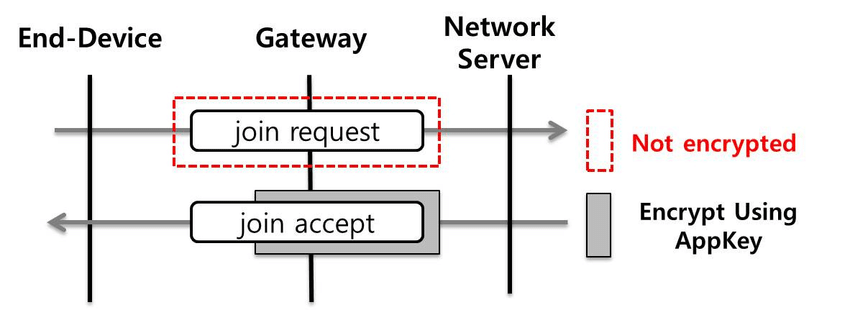


Figure 13. Join procedure[15]

**join-request message**

The join request message is composed of 3 parameters which are the AppEUI , DevEUI and a random Nonce of 2 octets. It is generated by the end device and forwarded to the gateway , then network server unencrypted.

To prevent replay attacks , the network server keeps track of a certain number of DevNonce values previously used by the end-device and ignores join request with any of these values from a specific end device.

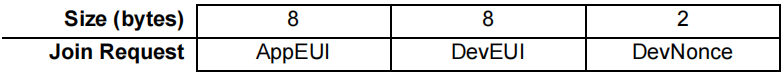


Figure 14. Join request

The join request even if not encrypted is integrity protected by a MIC (Message Integrity Code ) . It is calculated using AES−CMAC using the below operation:

*cmac* = aes128\_cmac(AppKey, MHDR | AppEUI | DevEUI | DevNonce)

AES-CMAC is a message authentication code (MAC) algorithm that provides data integrity and authenticity for messages. It is based on the AES block cipher and the CMAC mode of operation.

**join-accept message**

Upon reception of the join request message , the network server can reject or accept the request. If the request is accepted the server replies with join accept message.

The join-accept message contains the following fields : an application nonce (AppNonce) of 3 octets, a network identifier (NetID), an end-device address (DevAddr), a delay between TX and RX (RxDelay) and an optional list of channel frequencies (CFList) for the network the end-device is joining.

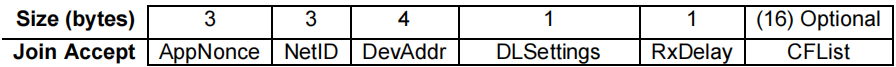


Figure 15. Join accept

**Securing the session**

The **AppNonce** is a random value or some form of unique ID provided by the network server and used by the end-device to derive the two session keys **NwkSKey** and **AppSKey** as follows:

NwkSKey = aes128\_encrypt(AppKey, 0x01 | AppNonce | NetID | DevNonce | pad16)

AppSKey = aes128\_encrypt(AppKey, 0x02 | AppNonce | NetID | DevNonce | pad16)

**Protecting message integrity**

The MIC value for a join-accept message is calculated as follows:

*cmac* = aes128\_cmac(AppKey, MHDR | AppNonce | NetID | DevAddr | RFU | RxDelay | CFList)

MIC = *cmac*[0..3]

**Protecting the payload**

The join-accept message itself is encrypted with the **AppKey** as follows:

aes128\_decrypt(AppKey, AppNonce | NetID | DevAddr | RFU | RxDelay | CFList | MIC)

**Key Generation & Management**

A LoRaWAN network implementing OTAA procedure typically handles 2 type of keys : A root key known as AppKey , unique for each end device in a LoRaWAN network. 2 Session keys : NwKSKey & AppSKey.

The AppKey (Application Key) is a 128-bit AES key shared between the device and the application server, used to encrypt and authenticate the communication between the device and the network. According to the LoRaWAN specification this key is unique per end device.

The NwKSKey ( Network Session Key ) is a 128-bit key whose function is to encrypt and authenticate network level transmissions. This key is generated by the network server during the OTAA process using the AppKey.

The AppSKey (Application Session Key) is a 128-bit AES key generated by the application server and provided to the device during the OTAA process. This key is used to establish a secure session between the device and the application server and is used to encrypt and authenticate all data transmitted between the device and the application server.

The 2 sessions keys above are generated from the root AppKey using Device Nonce ( DevNonce) and Application Nonce ( AppNonce) through the below encryption operations :

NwkSKey = aes128\_encrypt(AppKey, 0x01 | AppNonce | NetID | DevNonce | pad16)

AppSKey = aes128\_encrypt(AppKey, 0x02 | AppNonce | NetID | DevNonce | pad16)

When an end device is turned off or reset it generates new pair of Nonces with Network Server , hence different set of session keys. This prevents replay attacks on the network.

Some advance techniques such as side channel attack can recover the keys by observing variations in electromagnetic emissions during AES encryption process.

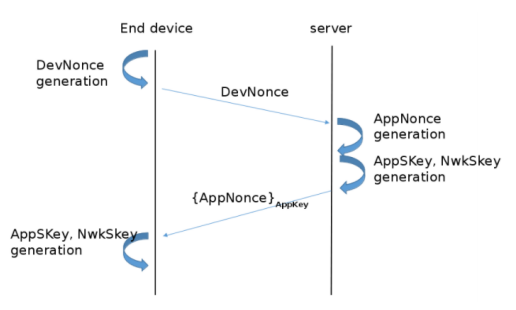


Figure 16. OTAA session keys generation & exchange[7]

**Counter Management**

In LoRaWAN OTAA, the network assigns an initial FCnt and MIC to the device during the join procedure. The device then increments the FCnt for each subsequent message sent to the network. The network checks the FCnt and MIC of each message it receives from the device to ensure that the message has not been tampered with and is sent in the correct order.

To prevent the FCnt from reaching its maximum value and to avoid the possibility of a replay attack, the device and the network must periodically reset the FCnt to its initial value. This process is called frame counter synchronization, and it is typically performed during a rejoin procedure or at predetermined intervals. By managing the FCnt and MIC counters, LoRaWAN OTAA ensures the integrity and confidentiality of communication between the device and the network.

**Message Acknowledgment**

n LoRaWAN, message acknowledgment is used to confirm the successful delivery of an uplink transmission from a device to the network. This is important for ensuring reliable communication between the device and the network, especially in scenarios where the device may be located in a remote or hard-to-reach area.

To request acknowledgment for a message, the device sets the ACK bit in the message header before transmitting the message to the network. When the network receives the message, it sends a confirmation message (ACK) back to the device, indicating that the message was received successfully. If the network is unable to receive the message correctly, it will not send an ACK message.

The ACK message sent by the network contains the same sequence number (FCnt) as the original message, allowing the device to confirm that the correct message was received by the network. If the device does not receive an ACK message within a certain timeframe, it can assume that the message was not received and retransmit the message.

**Chapter 3**

**Threats and vulnerability analysis of Join procedure**

**A review of OTAA replay attack scenarios in LoRaWAN 1.0**

Like in any other conventional communication network , LoRaWAN may be vulnerable to replay attacks on the nodes (end-devices) and network server. In order to successfully perform this attack , a malicious device must obtain a man in the middle position between the nodes and network server. Ideally the malicious device should be a rogue gateway capable of capturing uplink and downlink LoRa traffic on the air interface and later replaying.

The studies by Gildas & Ferreiras [4], Van Es & Co [10], Seungjae [15], Zulian [23], Lifchitz [39], and Florian & Al [59] provide valuable insights into the security vulnerabilities related to OTAA replay attacks in LoRaWAN 1.0.

Gildas & Ferreiras [4] conducted an extensive security analysis of LoRaWAN 1.0 join messages and identified several protocol weaknesses that could be exploited to perform replay and decrypt attacks, as well as denial of service attacks against a node and network server.

Van Es & Co [10] illustrated denial of service attacks against LoRaWAN end-devices by creating a formal model of important sections of the protocol v1.0.2 on Coloured Petri Nets. The study identified three vulnerabilities: join accept replay vulnerability, downlink routing vulnerability, and beaconing vulnerability.

Seungjae [15] performed a join request replay attack scenario and proposed XOR masking on join request messages as a mitigation strategy.

Zulian [23] evaluated the security level of the join process from the perspective of RSSI and DevNonce generation.

Lifchitz [39] presented attack scenarios due to protocol and implementation weaknesses in LoRaWAN infrastructures. The study described in detail the attacks targeting join requests and reported that the Semtech gateway and its protocol were vulnerable to spoofing, sniffing, and denial of service attacks.

Florian & Al [59] provided a general security analysis of the three LPWAN protocols, including LoRaWAN, Sigfox, and NB-IoT, and described packet forgery attacks against LoRaWAN. The study assumed the use of an unencrypted and unauthenticated connection between the gateway and network server.

**Threat modeling analysis of join procedure**

Threat modeling is the process of identifying potential security threat and vulnerabilities in a system that can compromise its confidentiality , integrity and availability.

In the context of LoRaWAN, the join procedure is a critical process that needs to be evaluated in terms of its security vulnerabilities and potential threats.

The join procedure involves a device sending a join request to the network server. The server then sends a join accept message to the device, containing the network session key (NwkSKey), application session key (AppSKey), and device address. The device uses these keys and address to securely communicate with the network.

Several security vulnerabilities exist in the join procedure that can be exploited by attackers.

Remember a threat vector exploits a vulnerability on an asset. In this section we identify key assets , possible threats to those assets and vulnerabilities exposing the assets in a LoRaWAN network.

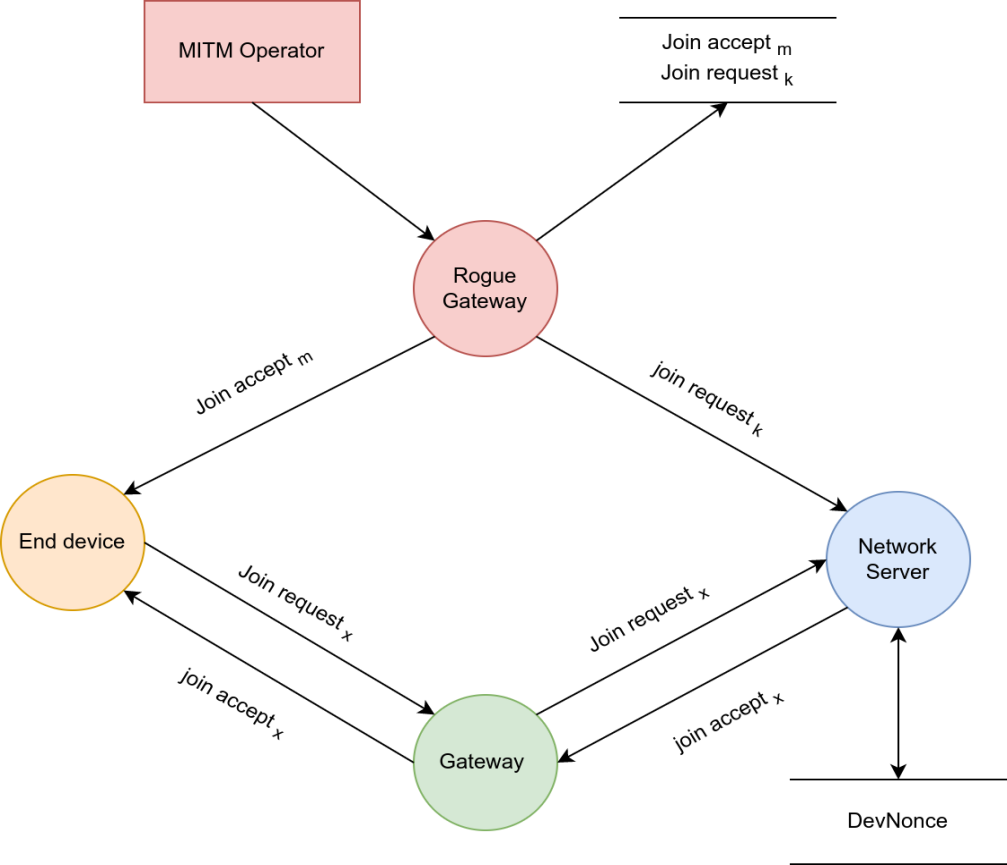


Figure 17. A Threat model for OTAA

**Threat identification**

In order to achieve our goals of producing normal and attack traffic traces for our simulated LoRaWAN network we methodically a process which combines both passive and active vectors.

**Passive attack vectors :**

**eavesdropping:** Since LoRaWAN uses wireless communication, it is vulnerable to eavesdropping attacks where an attacker can intercept the communication between the device and the network. To mitigate this threat, LoRaWAN uses encryption to secure the communication between the device and the network.

**Active attack vectors :**

**man-in-the-middle (MITM) attacks**: In an MITM attack, an attacker intercepts and alters the communication between a device and the network or between two devices in the network. LoRaWAN 1.0 uses AES encryption and MAC algorithms to prevent MITM attacks.

**replay attacks**: A replay attack involves an attacker intercepting a message and then retransmitting it to the network at a later time, causing the network to perform the same action twice. LoRaWAN 1.0 uses message integrity codes (MICs) and frame counters to prevent replay attacks.

**Denial of Service (DoS) attacks:** A DoS attack involves an attacker flooding the network with messages, causing it to become overloaded and unable to function properly. LoRaWAN 1.0 includes various mechanisms to prevent DoS attacks, such as limiting the number of messages a device can send in a given time period.

Some realistic consequences of a threat vector exploiting weaknesses in LoRaWAN 1.0 is the ability to replay and decrypt packets , perform denial of service (device disconnection) on end devices and network server.

**Asset identification**

In order to achieve our goals of producing normal and attack traffic traces for our simulated LoRaWAN network we methodically a process which combines both passive and active vectors.

**Target system assets :**

We identify what LoRaWAN network assets are essential for the success of our attack simulation.

Since the focus is on generating attack traces via OTAA procedure , we need to capture the join request and accept payloads from the simulation of our testbed under normal operational conditions:

***Attack Scenarios on Join Procedure***

1. **DOSing the End Device**

The main objective of this attack is to disconnect or desynchronize communication between a LoRa node and Network Server. This goal can be achieved by replaying join request and join accept payloads under the conditions defined below. Remember the join accept is encrypted and signed while the join request is just signed.

1. ***Replaying join accept to Node:***

*- Threat vector :* **Denial of Service**

- *Method* : Replay a previously capture join accept payload before the Network Server join accept payload.

- *Objective* : prevent end device from associating to network server

- *Complexity* : Race condition ( forward previous join accept before legitimate one )

- *Impact* : Availability ( Device Disconnection)

The success of this replay attack depends on timing of replayed join accept payload. At the moment the end device initiates a join request, the rogue gateway should forward its payload before the legitimate one does.Based on LoRaWAN 1.0 no protection mechanism is specified at the level of end device to prevent replay.

As a result the AppNonce received by the end device to derive the two session keys will be different from the expected one. The device won’t be able to establish a session with its Network Server since **AppNonce k ≠ AppNonce m**

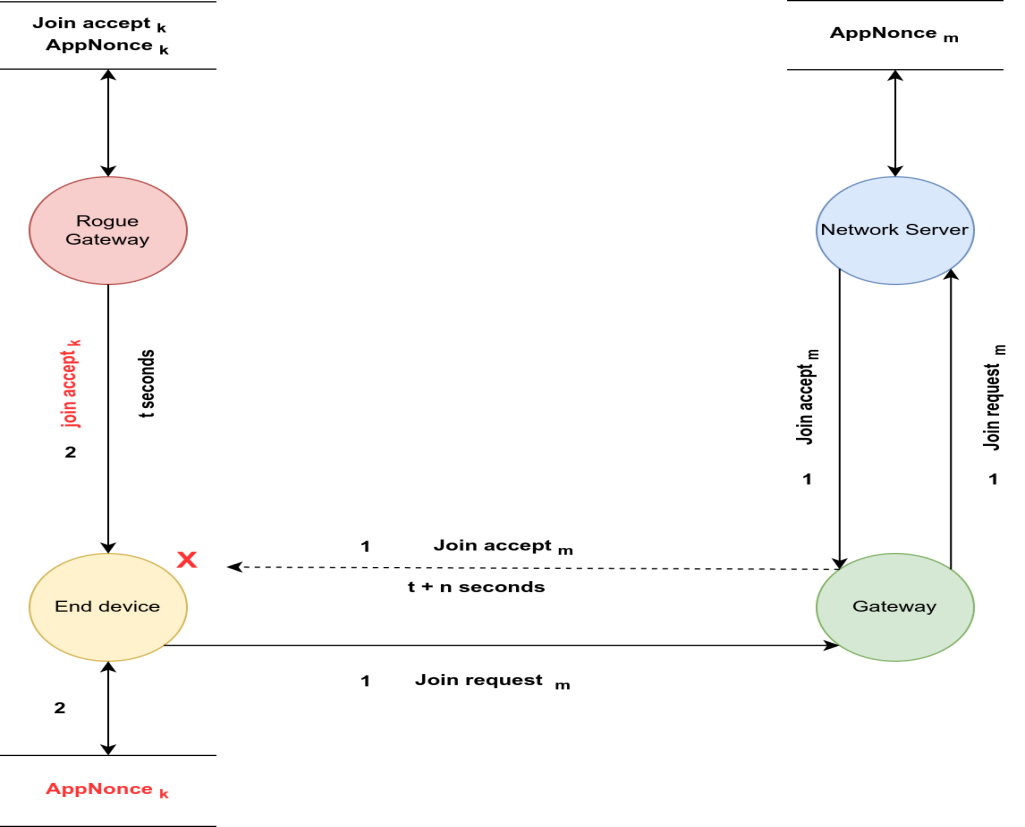


Figure 18. Join accept replay against end device

1. ***Replaying join request to Network Server***

*- Threat vector :* **Denial of Service**

- *Method* : Replay a previously captured join request message & forward to the network server

- *Objective* : Disconnect an end device on the network server from a current session

- *Complexity* : The DevNonce of the previously captured message should not be stored in Network Server memory

- *Impact* : Availability ( Device Disconnection )

For the attack to function the Device Nonce replayed in the join request must be different from DevNonce stored in the Network Server’s database. According to LoRaWAN specification the Network Server keeps record of previously accepted device nonces to prevent replay.

**DevNonce k  ≠  DevNonce n**

The aftermaths of a successfull join request is the Network Server computing new session keys such that the previous session keys are rendered invalid. Hence the legitimate end device will be disconnected from the server

**{ NwKSKey k, AppKSKey k}  ≠ { NwKSKey m, AppKSKey m}**

The threat model diagram below illustrates the join request replay attack against a Network Server.

Logic flow 1 denotes a current session with already negotiated session keys stored in End-device and Network Server memory.

Logic flow 2 demonstrates injection of a previous join request and the effect on communication between the Network Server and End Device.

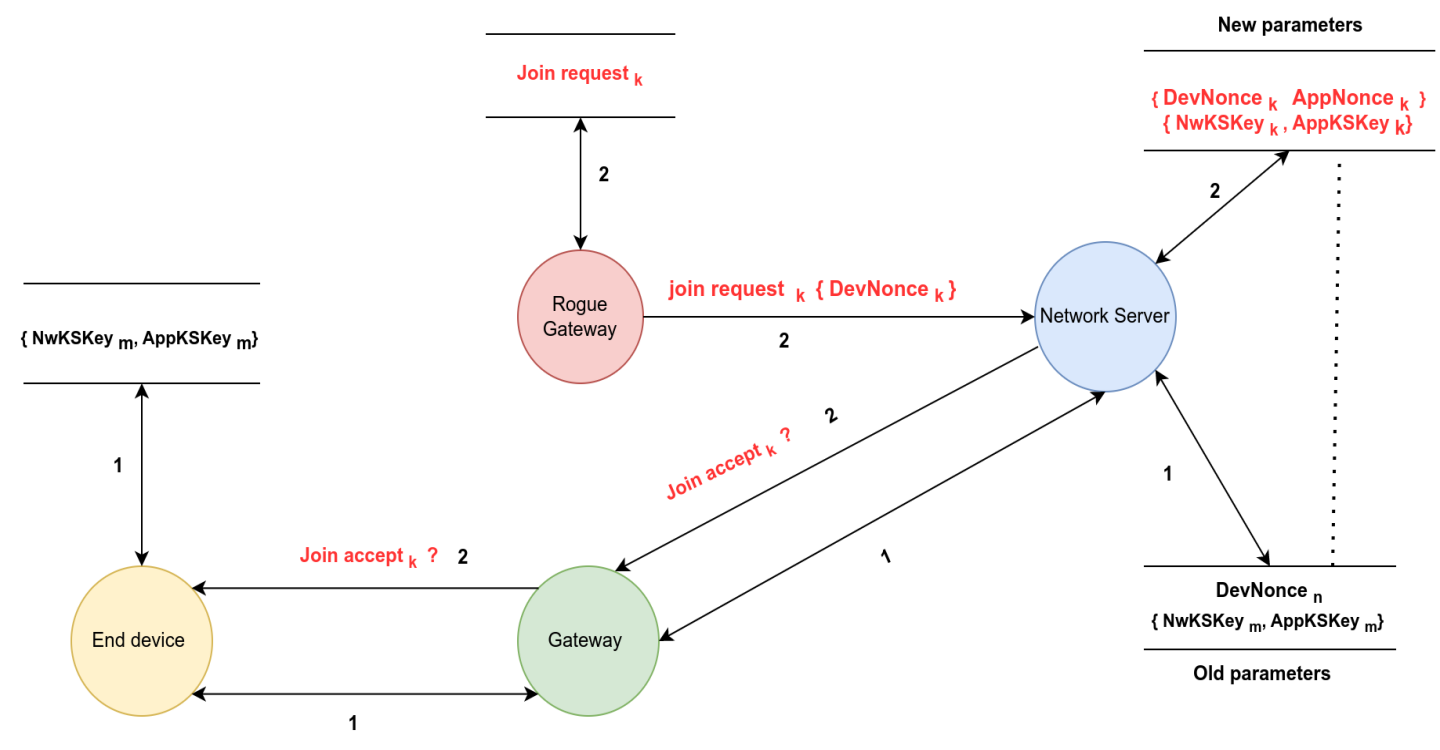


Figure 19. Join request replay against Network Server

**Chapter 4**

**Simulating attacks**

**4.1 Overview of existing simulation environments**

To generate normal & attack traffic traces we needed a simulation environment capable of producing realistic LoRa payload & abstract communication between node , gateway and network server. We reviewed a set of LoRaWAN simulators presented in [64] [65] amongst which FloRa, LoRaSIM & LoRaWAN NS-3. However the reviewed simulators cannot generate real LoRaWAN traffic because they are discrete-event types. A discrete event simulation with applicability to LoRaWAN can only model the operations and state of devices in the network as a sequence time based events.

On the other hand LWN Simulator [3] in conjunction with Chirpstack OpenSource Network Server [19] generate LoRaWAN traffic as expected from a real network.

**4.2 Simulation Architecture & Parameters**

In this section we present the architecture of the simulation environment together with constraints and parameters applied.

**4.2.1. Architecture**

**LWN Simulator**

The implemented simulation environment consist of a Web GUI interface that enables a user to configure virtual nodes and gateways, communicate with a real gateway and communicate with a network server via the packet forwarder module.

The function of the packet forwarder module is to pull and push packets for the gateway while interacting with the network server.

The communication between the node-gateway simulator and network server is established and maintained via the Semtech UDP protocol.

**Chirpstack**

The Chirpstack Open Source Network Server consist of a gateway bridge to receive and transmit LoRa packets , a network server to handle device association request and network session security and an application server that handles application level data and provides its own session key.

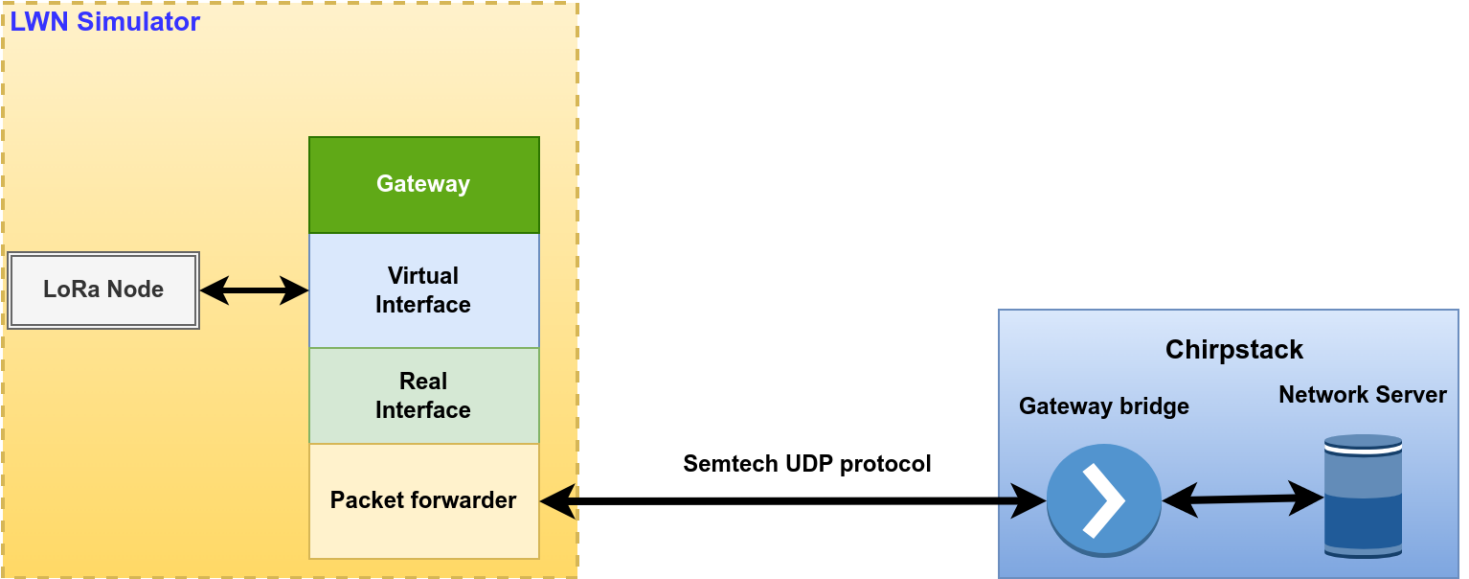


Figure 20. Simulation Architecture

**4.2.2. Environment , Constraints and Parameters**

To generate our normal traffic scenario , we set up 2 virtual machines for node-gateway simulator and the network server. Both machines are located in the private addressing space as configured in VirtualBox.

Both simulators are accessible and configurable via the Web GUI .

The LWN Simulator generates a series of discrete events seen on the CLI after launching the script.

The Chirpstack Network Server generates logs that are accessibles via MQTT broker.

The figure below depicts the simulation environment.

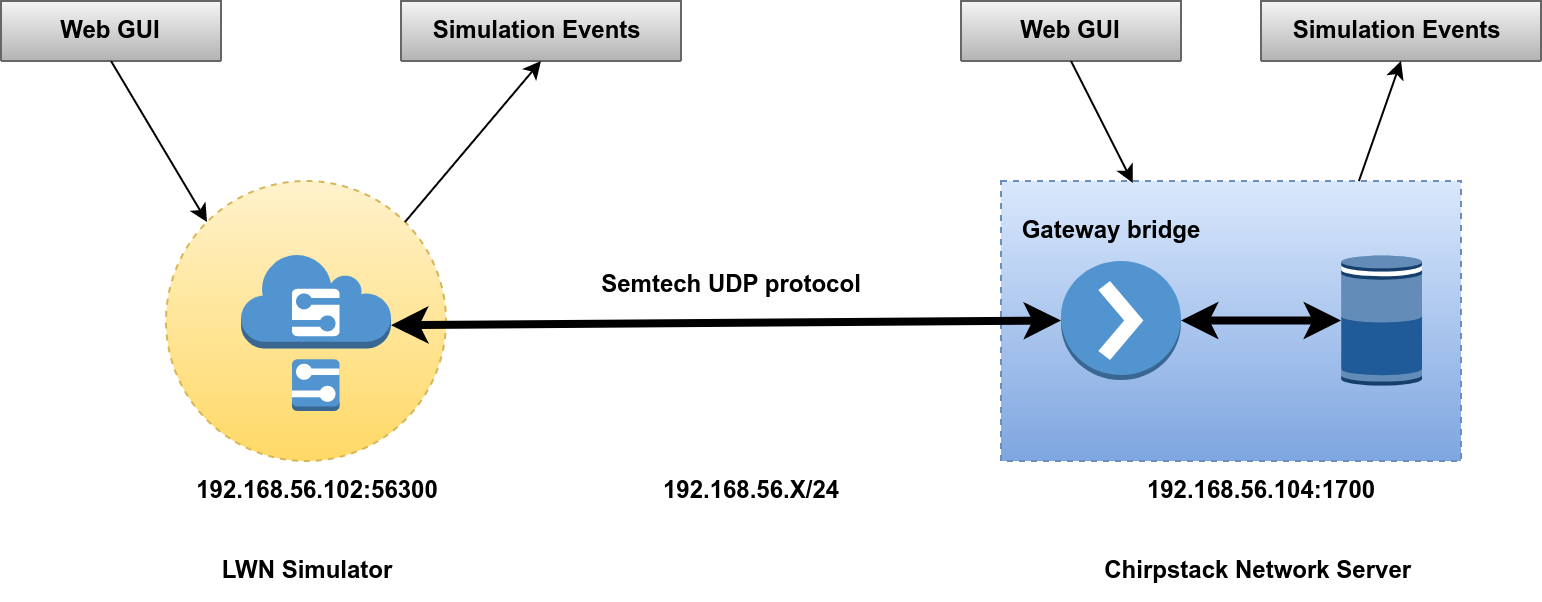
****

Figure 21. Simulation Environment

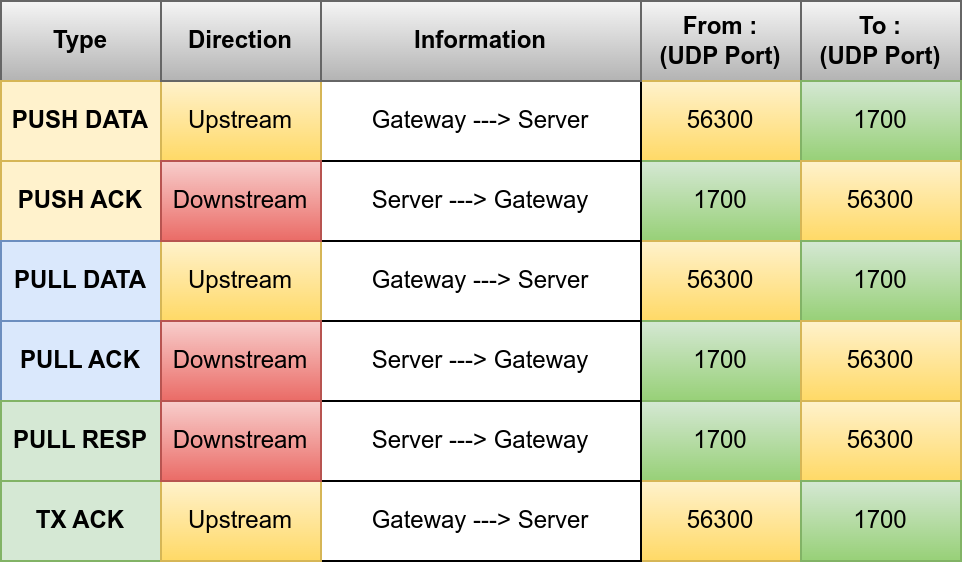
****

Figure 22. Protocol transactions

Virtualization platform : Virtualbox 7.0

LWN Simulation OS : Ubuntu Server 22 LTS

Chirpstack Simulation OS : Ubuntu Server 22 LTS

**LWN Simulator**

**End Device**

LoRaWAN version : 1.0

Semtech UDP protocol version : 2

N° End device(s) : 1

Class of End device : A

Association Method : OTAA

Regional parameters : EU864

Confirmed Downlink Message : False

**Gateway:**

IP : 192.168.56.102

Port : 56300

Gateway bridge : 192.168.56.104:1700

**Chirpstack**

**Network Server:**

IP : 192.168.56.104

Port : 1700

**4.2.3. Normal Traffic Collection :**

**4.3 Attack Simulation**

To perform the attack scenarios analysed in chapter 3 , we need to follow a set of steps illustrated in figure 20.

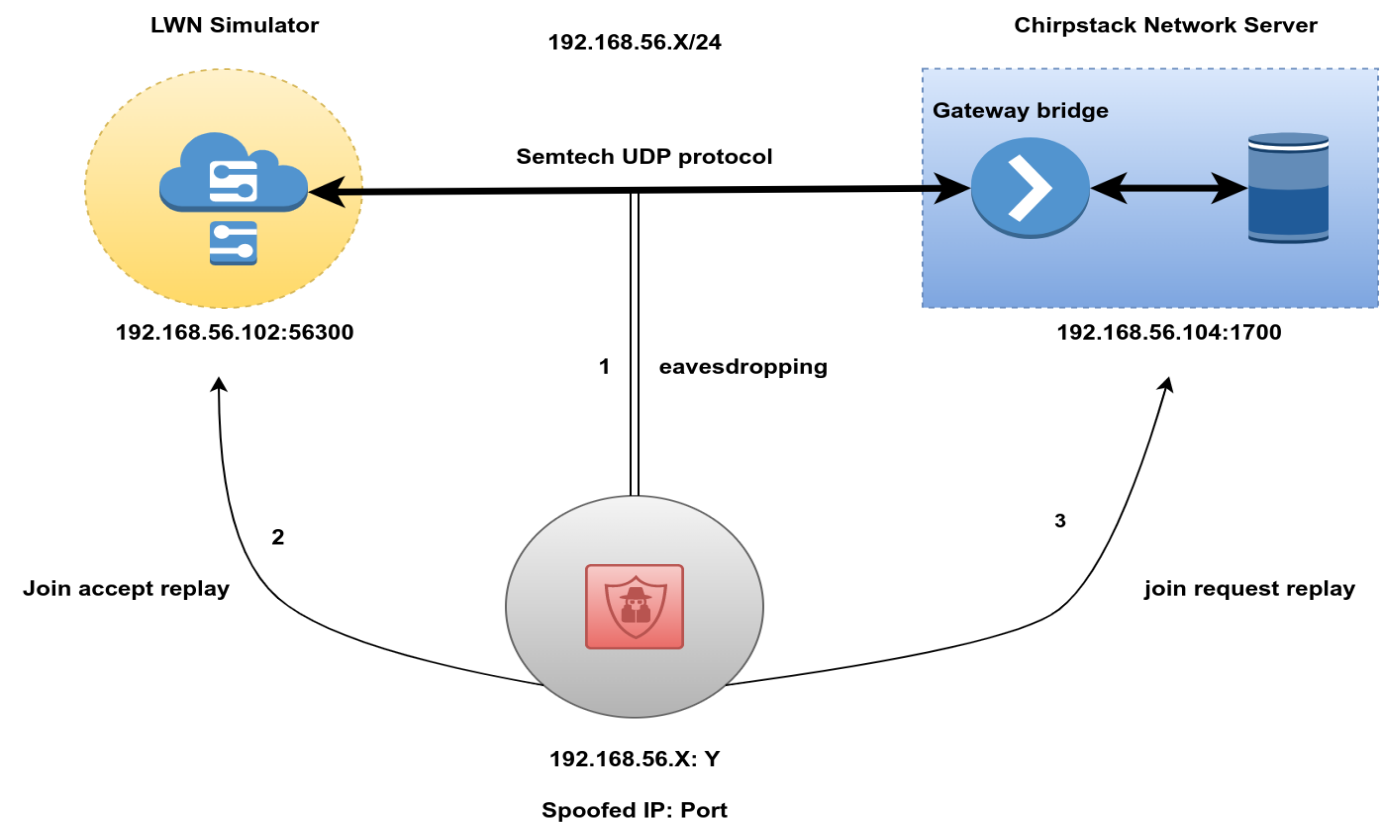


Figure 23. Attack Simulation

1. **Eavesdropping :**

- The simulation environment is configured and launched

- The rogue system has a MITM position in the subnetwork with ability to capture traffic.

- A listener is set to capture traffic in the subnet with the following command :

tcpdump -XUq -i (interface) port 1700 | -w loratraces.txt

- The payloads of interest in the capture file are the join request packet and join accept packets. They are located at position xxxxx in the screen capture below

Results :

1. **Replay attack towards end device**

- Identify the join accept trace in the capture file and save the payload to xxx format

- Restart the simulation to trigger a join request & simultaneously execute the attack script to replay join accept payload towards LWN Simulator

Results :

1. **Replay attack towards network server**

- Identify the join request trace in the capture file and save the payload to xxx format

- Flush Chirpstack DevNonce queue to empty list

- Restart Simulation

- Run script to forward join accept payload to network server

Results :

**Chapter 5**

**Experimentation results and analysis**

A LoRaWAN (Low Range Wide Area Network) attack trace dataset should contain various types of information to help identify and analyze potential vulnerabilities or malicious activities in a LoRaWAN network. LoRaWAN is designed for long-range, low-power communication, often used in IoT applications. Here are some essential elements that a LoRaWAN attack trace dataset should contain:

1. Timestamps: Precise time information for each event in the dataset, which helps in analyzing the sequence of events and identifying patterns.

2. Device identifiers: Unique identifiers for devices in the network, such as DevEUI (Device Extended Unique Identifier), AppEUI (Application Extended Unique Identifier), and DevAddr (Device Address).

3. Packet data: Detailed packet data, including payload, headers, and metadata, for both uplink (device-to-server) and downlink (server-to-device) communication.

4. Attack type: Classification of the type of attack being performed, such as replay attacks, jamming, denial of service (DoS), man-in-the-middle (MITM), or cryptographic attacks.

5. Attack parameters: Specific parameters related to the attack, such as the targeted devices, frequency of malicious packets, and attack duration.

6. Network and application layer information: Data from both the network layer (LoRaWAN) and the application layer (e.g., MQTT or HTTP) to analyze the impact of the attack on various network components.

7. Signal strength and quality: Information about the received signal strength indicator (RSSI) and signal-to-noise ratio (SNR) to understand the impact of the attack on communication quality.

8. Geolocation data: If available, the geographical location of the devices and gateways can help identify spatial patterns in the attacks.

9. Countermeasures: Any mitigation techniques employed during the attack, such as changes in cryptographic keys, packet rate limiting, or device authentication mechanisms.

10. Impact analysis: A record of the attack's impact on network performance, such as packet loss, latency, and battery life of the devices.

The primary goal of collecting and analyzing this data is to understand potential vulnerabilities in a LoRaWAN network, develop effective countermeasures, and improve the overall security of the system. Keep in mind that the availability and granularity of the data in the dataset may vary depending on the specific LoRaWAN deployment and monitoring capabilities.

**Chapter 6**

**Conclusion**

In conclusion, this master thesis has made a substantial contribution to the understanding and analysis of LoRaWAN security by generating a meticulously designed dataset of normal and attack traces, specifically addressing join replay attacks in LoRaWAN 1.0 within a simulated environment. The use of a simulated environment has enabled precise control over various network parameters, allowing for the creation of a diverse and representative dataset. This dataset not only advances the knowledge of security risks inherent in the LoRaWAN protocol but also serves as a valuable asset for researchers and practitioners to develop and assess the effectiveness of intrusion detection systems and other security measures.

The simulated environment facilitated the systematic evaluation of the dataset using a range of machine learning techniques, such as supervised and unsupervised learning algorithms. The results demonstrated the potential of the dataset to identify and counter join replay attacks in LoRaWAN networks effectively. This dataset is anticipated to inspire further research on the development of comprehensive security mechanisms for LoRaWAN networks, consequently leading to greater resilience and trustworthiness of this emerging IoT technology. By making the dataset publicly available, the research community can build on this work to validate and compare new approaches, promoting collaboration and accelerating progress in the field of LoRaWAN security.

Future work should focus on extending the dataset to incorporate traces from other types of attacks and exploring newer versions of the LoRaWAN protocol. Additionally, efforts should be directed towards transitioning from a simulated environment to real-world deployments, ensuring that the dataset reflects the complexities and challenges faced in practical settings. Moreover, the integration of artificial intelligence and machine learning techniques to proactively detect and mitigate attacks in real-time should be investigated, further enhancing the security posture of LoRaWAN networks. In summary, this thesis has laid a strong foundation for future research efforts, paving the way for the development of secure, reliable, and enduring LoRaWAN-enabled IoT ecosystems.

.

**References**

1. LoRa Alliance. 2015. LoRaWAN Specification v1.0. (2015). <https://www.lora-alliance.org/.>
2. LoRa Alliance. 2016. LoRaWAN Specification v1.0.2. (2016). <https://www.lora-alliance.org/.>
3. LWN Simulator. <https://github.com/UniCT-ARSLab/LWN-Simulator.>
4. Gildas Avoine and Loïc Ferreiras. 2016.Rescuing LoRaWAN 1.0. (2016). <https://eprint.iacr.org/2017/651.>
5. Kuntke F, Romanenko V, Linsner S, Steinbrink E, Reuter C. LoRaWAN security issues and mitigation options by the example of agricultural IoT scenarios. Trans Emerging Tel Tech. 2022;e4452. doi: 10.1002/ett.4452
6. Poliana de Moraes and Arlindo Flavio da Conceição. 2021. A Systematic Review of Security in the LoRaWAN Network Protocol. ACM Comput. Surv. 30, 3, Article 102 (March 2021), 20 pages.
7. Xueying Yang. 2017. LoRaWAN: Vulnerability Analysis and Practical Exploitation

(MSc thesis, Delft University of Technology). (2017).

1. Emekcan Aras, Gowri Sankar Ramachandran, Piers Lawrence, and Danny Hughes.2017. Exploring the security vulnerabilities of LoRa. In IEEE International Conference on Cybernetics (CYBCONF). <https://doi.org/10.1109/CYBConf.2017.> 7985777
2. X. Yang, E. Karampatzakis, C. Doerr, and F. Kuipers, “Security vulnerabilities in lorawan,” in 2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI), 2018, pp. 129–140.
3. Van Es, E., Vranken, H., & Hommersom, A. (2018). Denial-of-Service Attacks on LoRaWAN. In Proceedings ofthe 13th International Conference on Availability, Reliability and Security [17] acm.<https://doi.org/10.1145/3230833.3232804.>
4. James Forshaw. 2017. Attacking Network Protocols: A Hacker's Guide to Capture, Analysis, and Exploitation (1st. ed.). No Starch Press, USA.
5. Fotios Chantzis, Ioannis Stais, Paulino Calderon, Evangelos Deirmentzoglou, Beau Woods. 2021. Practical IoT Hacking : The Definitive Guide to Attacking the Internet of Things. No Starch Press, Inc., 2020.
6. S. Tomasin, S. Zulian and L. Vangelista, “Security Analysis of LoRaWAN Join Procedure for Internet of Things Networks,” 2017 IEEE Wireless Communications and Networking Conference Workshops (WCNCW), San Francisco, CA, 2017, pp. 1-6. doi: 10.1109/WCNCW.2017.7919091
7. Notes on LoRaWAN security. Feb 1, 2017. URL: https://medium.com/@brocaar/notes-on-lorawan-security-7e741a8ee4fa (visited on 2023).
8. SeungJae Na, DongYeop Hwang, WoonSeob Shin and Ki-Hyung Kim, “Scenario and countermeasure for replay attack using join request messages in LoRaWAN,” 2017 International Conference on Information Networking (ICOIN), Da Nang, 2017, pp. 718-720.

doi: 10.1109/ICOIN.2017.7899580

1. Kim, J.; Song, J. A simple and efficient replay attack prevention scheme for LoRaWAN. In Proceedings of the 7th International Conference on Communication and Network Security, Tokyo, Japan, 24–26 November 2017; ACM: New York, NY, USA, 2017; pp. 32–36.
2. Sung, W.; Ahn, H.; Kim, J.; Choi, S. Protecting end-device from replay attack on LoRaWAN. In Proceedings of the 20th International Conference on Advanced Communication Technology (ICACT), Chuncheon-si, Gangwon-do, Korea, 11–14 February 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 167–171.
3. Thomas, J.; Cherian, S.; Chandran, S.; Pavithran, V. Man in the Middle Attack Mitigation in LoRaWAN. In Proceedings of the International Conference on Inventive ComputationTechnologies (ICICT), Coimbatore, India, 26–28 February 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 353–358.
4. Orne Brocaar. ChirpStack Network Server. Available online: <https://github.com/brocaar/chirpstack-network-server.> (accessed on 20 April 2023).
5. Hessel, F.; Almon, L.; Álvarez, F. ChirpOTLE: A Framework for Practical LoRaWAN Security Evaluation. In Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec’20), Linz, Austria, 8–10 July 2020; Association for Computing Machinery: New York, NY, USA, 2020; pp. 306–316.
6. CLAVERIE, Tristan; LOPES ESTEVES, Jose. A LoRaWAN Security Assessment Test Bench. Proceedings of the GNU Radio Conference, [S.l.], v. 2, n. 1, jan. 2021. Available at: <[https://pubs.gnuradio.org/index.php/grcon/article/view/88](https://pubs.gnuradio.org/index.php/grcon/article/view/88" \t "/home/seam/Documents\\x/_new)>. Date accessed: 19 apr. 2023.
7. LoRa Alliance. 2017. LoRaWAN v1.1 Specification. (2017). <https://www.lora-alliance.org/.>
8. Simone Zulian. 2016. Security threat analysis and countermeasures for LoRaWAN join procedure (MSc thesis, University of Padova). (2016). <http://tesi.cab.unipd.it/53210/1/zulian_simone_tesi.pdf.>
9. Robert Miller. 2015.LoRa Security - Building a Secure LoRa Solution.(2015).

<https://labs.mwrinfosecurity.com/assets/BlogFiles/mwri-LoRa-security-guide-1.2-2016-03-22.pdf.>

1. SmartCitiesWorld: Semtech LoRa chosen for new IoT network in New Zealand.

[https://smartcitiesworld](https://smartcitiesworld.net/news/news/semtech-lora-chosen-for-new)*[.](https://smartcitiesworld.net/news/news/semtech-lora-chosen-for-new)*[net/news/news/semtech-lora-chosen-for-new](https://smartcitiesworld.net/news/news/semtech-lora-chosen-for-new)iot-network-in-new-zealand-949 (September 2016)

1. National Institute of Standards and Technology (NIST). 2001. Announcing the advanced encryption standard (AES). (2001), 8–12 pages. <https://doi.org/10.1016/S1353-4858(10)70006-4.>
2. Song, J. H. and Poovendran, R. and Lee, J. and Iwata, T.: The AES-CMAC Algorithm. RFC 4493 (Jun 2006)
3. Ole Martin Dahl and Stephen D Wolthusen. 2006. Modeling and Execution of Complex Attack Scenarios using Interval Timed Colored Petri Nets. In IEEE International Workshop on Information Assurance (IWIA’06). pp.157–168. <https://doi.org/10.1109/IWIA.2006.17.>
4. Butun, I.; Pereira, N.; Gidlund, M. Security risk analysis of LoRaWAN and future directions. Future Internet 2019, 11, 3.
5. Sanchez-Iborra, R.; Sánchez-Gómez, J.; Pérez, S.; Fernández, P.J.; Santa, J.; Hernández-Ramos, J.L.; Skarmeta, A.F. Enhancing lorawan security through a lightweight and authenticated key management approach. Sensors 2018, 18, 1833.
6. Tsai, K.; Leu, F.; Hung, L.; Ko, C. Secure Session Key Generation Method for LoRaWAN Servers. IEEE Access 2020, 8, 54631–54640.
7. S. M. Danish, H. K. Qureshi, and S. Jangsher. Jamming Attack Analysis of Wireless Power Transfer on LoRaWAN Join Procedure. In 2018 IEEE Globecom Workshops (GC Wk

shps), pages 1–6, December 2018.

1. Lin, J.; Shen, Z.; Miao, C.; Liu, S. Using blockchain to build trusted LoRaWAN sharing server. Int. J. Crowd Sci. 2017, 1, 270–280.
2. Mohamed Eldefrawy, Ismail Butun, Nuno Pereira, and Mikael Gidlund. Formal security analysis of lorawan. Computer Networks, 148:328 – 339, 2019.
3. JungWoon Lee, DongYeop Hwang, JiHong Park, and Ki-Hyung Kim. Risk analysis and

countermeasure for bit-flflipping attack in LoRaWAN. In 2017 International Conference on

Information Networking (ICOIN), pages 549– 551, Da Nang, January 2017.

1. S. M. Danish, A. Nasir, H. K. Qureshi, A. B. Ashfaq, S. Mumtaz, and J. Rodriguez. Net

work Intrusion Detection System for Jamming Attack in LoRaWAN Join Procedure. In 2018

IEEE International Conference on Communications (ICC), pages 1–6, May 2018.

1. Sornin, N. and Luis, M. and Eirich, T. and Kramp, T. and Hersent, O.: LoRaWAN Specification (Jul 2016), LoRa Alliance, version 1.0.2
2. Tahsin C. M. Donmez, and Ethiopia Nigussie, “Security of LoRaWAN v1.1 in Backward Compatibility Scenarios,” accepted in The 15th International Conference on Mobile Systems and Pervasive Computing, 2018.
3. Renaud Lifchitz. Security review of LoRaWAN networks. In Hardwear.io, The Hague, Netherlands, 2016.
4. Robert Miller. LoRa the explorer: Attacking and Defending LoRa Systems. In Syscan 360

Singapore, Singapore, 2016.

1. The Things Network: [https://www](https://www.thethingsnetwork.org.)*[.](https://www.thethingsnetwork.org.)*[thethingsnetwork](https://www.thethingsnetwork.org.)*[.](https://www.thethingsnetwork.org.)*[org.](https://www.thethingsnetwork.org.)
2. N.E. L. M. N. A. E. F. A. Silva FSD, Silva E, “Taxonomy of ddos attack mitigation approaches featured by sdn technologies in iot scenarios,” Sensors (Basel), 2020.
3. H. Noura, T. Hatoum, O. Salman, J.-P. Yaacoub, and A. Chehab, “Lorawan security survey: Issues, threats and possible mitigation techniques,” Internet of Things, vol. 12, p. 100303, 2020. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2542660520301359.>
4. Fukushima, D. Marion, Y. Nakano, A. Facon, S. Kiyomoto, and S. Guilley, “Evaluation of side-channel key-recovery attacks on lorawan end-device,” in Information Systems Security and Privacy, P. Mori, S. Furnell, and O. Camp, Eds. Cham: Springer International Publishing,

2020, pp. 74–92.

1. L’Héréec, F., Joulain, N.: Sécurité LoRaWAN. In: Computer & Electronics Security Applications Rendez-vous – C&ESAR (2016)
2. Laksh Bhatia, Michael Breza, Ramona Marfievici, and Julie A. McCann. 2020. Dataset: LoED: The LoRaWAN at the Edge Dataset. In Proceedings of the Third Workshop on Data Acquisition To Analysis (DATA ’20), November 16– 19, 2020, Virtual Event, Japan. ACM, New York, NY, USA, 2 pages. <https://doi.org/10.1145/3419016.343149.>
3. Rachel Cardell-Oliver et al. 2019. Dataset: LoRa Underground Farm Sensor Net

work. In Proc. of DATA.

1. LoRa Alliance: LoRaWAN Certified Products. https://www.lora-alliance.org/ certified-products, last consulted December 8, 2017.
2. LoRa Alliance Technical committee: LoRaWAN Regional Parameters (Jul 2016), LoRa Alliance, version 1.0
3. LORIOT.io: <https://www.loriot.io>
4. Coman FL, Malarski KM, Petersen MN, Ruepp S. Security issues in internet of things: vulnerability analysis of LoRaWAN, Sigfox and NB-IoT. Proceedings of the 2019 Global IoT Summit (GIoTS); 2019:1-6.
5. Vladislav Skorpil, Vaclav Oujezsky, and Ludek Palenik. Internet of things security overview and practical demonstration. In 2018 10th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), pages 1–7. IEEE, 2018.
6. Gu C, Jiang L, Tan R, Li M, Huang J. Attack-aware data timestamping in low-power synchronization-free LoRaWAN. Proceedings of the 2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS); 2020:100-110.
7. LoRaWAN Auditing Framework . <https://github.com/IOActive/laf> (accessed on 23/04/2023)
8. Xu J, Tang Y, Wang Y, Wang X. A practical side-channel attack of a LoRaWAN module using deep learning. Proceedings of the 2019 IEEE 13th International Conference on Anti-counterfeiting, Security, and Identification (ASID); 2019:17-21.
9. Benkahla N, Belgacem B, Frikha M. Security analysis in enhanced LoRaWAN duty cycle. Proceedings of the 2018 7th International Conference on Communications and Networking (ComNet); 2018:1-7; IEEE.
10. Mundt T, Gladisch A, Rietschel S, Bauer J, Goltz J, Wiedenmann S. General security considerations of LoRaWAN version 1.1 infrastructures. Proceedings of the 16th ACM International Symposium on Mobility Management and Wireless Access (MobiWac). Association for Computing Machinery; 2018:118-123; New York, NY.
11. Shanay Behrad, Stéphane Tuffin, Emmanuel Bertin, and Noel Crespi. Network access control for the IoT: A comparison between cellular, wi-fi and LoRaWAN. In 2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN), pages 195–200. IEEE, 2019.
12. Florian Laurentiu Coman, Krzysztof Mateusz Malarski, Martin Nordal Petersen, and Sarah Ruepp. Security issues in internet of things: Vulnerability analysis of LoRaWAN, Sigfox and NB-IoT. In 2019 Global IoT Summit, pages 1–6. IEEE, 2019.
13. Syed Muhammad Danish, Marios Lestas, Waqar Asif, Hassaan Khaliq Qureshi, and Muttukrishnan Rajarajan. A lightweight blockchain based two factor authentication mechanism for LoRaWAN join procedure. In 2019 IEEE International Conference on Communications Workshops (ICC Workshops), pages 1–6. IEEE, 2019.
14. Erik Gresak and Miroslav Voznak. Protecting gateway from ABP replay attack on LoRaWAN. In International Conference on Advanced Engineering Theory and Applications, pages 400–408. Springer, 2019.
15. LoRa packet. <https://github.com/anthonykirby/lora-packet.>
16. Semtech UDP protocol. <https://github.com/Lora-net/packet_forwarder/blob/master/PROTOCOL.TXT>
17. Almuhaya, M.A.M., Jabbar, W.A., Sulaiman, N., Sulaiman, A.H.A. (2022). An Overview on LoRaWAN Technology Simulation Tools. In: Saeed, F., Mohammed, F., Ghaleb, F. (eds) Advances on Intelligent Informatics and Computing. IRICT 2021. Lecture Notes on Data Engineering and Communications Technologies, vol 127. Springer,Cham.

<https://doi.org/10.1007/978-3-030-98741-1_29>

1. Idris, S.; Karunathilake, T.Förster, A. Survey and Comparative Study of LoRa-Enabled Simulators for Internet of Things and Wireless Sensor Networks. Sensors 2022, 225546. <https://doi.org/10.3390/s22155546>
2. Oracle. VirtualBox. Accessed: April 2023. URL : <https://www.virtualbox.org/>
3. Wireshark v4. (2022). The Wireshark Foundation. Accessed: Jan. 13, 2023. [Online]. Available: <https://www.wireshark.org>
4. Ns-3 Network Simulator v3.37. (2008). nsnam. Accessed: Jan. 13, 2023.[Online]. Available: <https://www.nsnam.org>
5. The Tcpdump Group. Tcpdump. https : / / www . tcpdump . org/. [Accessed May 2023]. 2022
6. Chao Zhang, Chenguang Lu, Xuejun Cai, et al. “A Virtualized LoRa Testbed and Experimental Results for Resource Pooling”. In: 2021 IEEE 7th World Forum on Internet of Things (WF-IoT). IEEE. 2021, pp. 871–876
7. <https://www.univ-smb.fr/lorawan/wp-content/uploads/2022/01/Book-LoRa-LoRaWAN-and-Internet-of-Things.pdf>
8. ELoRa. (2023). Orange. Accessed: Mar. 2, 2023. [Online]. Available: <https://github.com/non-det-alle/elora-docker>
9. Scapy. Accessed : April 23. <https://scapy.readthedocs.io/en/latest/introduction.html>
10. Alessandro Aimi, Stephane Rovedakis, Fabrice Guillemin, Stefano Secci. ELoRa: End-to-end Emulation of Massive IoT LoRaWAN Infrastructures. 2023 IEEE/IFIP Network Operations and Management Symposium (NOMS), May 2023, Miami, FL, United States. hal-04025834