# Study on the security of LoRaWAN® networks

*Thierry Didi,*
*July 2025*
*Tidiwi*

Cellular technologies operating in licensed bands, such as LTE-M and NB-IoT, are now widely deployed. Their emergence could have marginalized competing solutions operating on unlicensed tapes, such as LoRaWAN® and Sigfox®. This is what happened for Sigfox. In France, in the same way, the operator Bouygues Telecom has terminated its LoRaWAN network. However, LoRaWAN has been able to "play its cards right" by establishing itself in many areas: smart cities, connected buildings, precision agriculture, etc.

Many metropolises have deployed private LoRaWAN networks to oversee a large fleet of urban sensors. There is also a strong presence of LoRaWAN sensors in the technical management of tertiary buildings as well as in agricultural infrastructures. The protocol also seeks to expand to other sectors, such as smart housing.

This document is structured as follows:

- LoRa® and LoRaWAN® technologies
  It begins with a presentation of the LoRaWAN protocol, which is essential to address security-related aspects. However, it does not detail low-level technical elements such as LoRa modulation, Spreading Factors or achievable bitrates, for which many specialized resources are available online.
- Threats to LoRaWAN networks
  The analysis then focuses on the threats these networks face, and the potential consequences for the end customer.
- Securing a LoRaWAN network
  Finally, it describes the best practices to be implemented to secure a LoRaWAN network depending on the topology deployed.

**Preliminary note** : This document relies exclusively on publicly available sources and specifications. No confidential information was used. The opinions and conclusions presented here are those of the author, and in no way reflect those of Semtech or the LoRa Alliance.

If you have any comments or suggestions for improvement, do not hesitate to let me know by email: thierry.didi[at]tidiwi[dot]com.

# 1 Introduction

Before going into the details of the protocols, it is useful to clarify a few elements of terminology:

- **ABP: "Activation By Regulation**" is one of the two methods of activating a LoRaWAN terminal.

- **AES:** is a robust symmetric encryption algorithm. In the LoRaWAN protocol, it is used with 128-bit keys. AES is used to encrypt 128-bit (16-byte) messages.

- **AES ECB:** AES-ECB is an encryption mode based on the AES algorithm, which allows messages to be encrypted in length that are a multiple of 16 bytes. Each 16-byte block is independently encrypted with the same AES key. This mode does not offer any protection against repetition of data: if the same message is encrypted twice with the same key, or if the message contains identical blocks, the encrypted blocks will also be identical. This makes AES-ECB unsuitable for most cryptographic uses.

- **AES CTR:** AES-CTR is an encryption mode based on the AES algorithm, which allows you to encrypt messages of arbitrary size. To ensure that encryption is secure, AES requires the use of a different initialization vector (IV)—or initial counter—each time it runs. Thus, if we encrypt the same message twice with different IVs, we will obtain different numerical results. On the other hand, if the same IV is reused with the same key, the results will be identical, which will compromise security.

- **AppKey:** An AES key known to the device and the network, which is used to derive the encryption keys that will be used to secure exchanges between a device and the network. The AppKey of a device is the most critical security parameter that must be protected.

- **ARM®** is a company that designs microprocessors, but does not manufacture them. Most of the processors and *SoCs* developed by major manufacturers such as STMicroelectronics, Texas Instruments, NXP Semiconductors... are based on ARM architectures.

- **Class A, B, C:** LoRaWAN defines three classes of equipment — A, B and C. Class A equipment is the most common. Their radio is turned off most of the time: they wake up autonomously to send a message, then open two short reception windows to eventually receive a response. Class C equipment, on the other hand, is in permanent listening (excluding transmission), which allows them to receive messages at any time. Finally, Class B equipment works like Class A equipment, but also opens regular reception windows synchronized with a beacon signal, in order to allow scheduled receptions.

- **Downlink:** describes the direction of communications, from the network to a radio terminal.

- **FUOTA: F**irmware **Update Over The Air.**

- **Gateway:** A device that receives messages from terminal equipment and transmits it to *the LNS*, and receives messages from the *LNS* to transmit it to the terminal equipment.

- **Join procedure**: procedure by which an OTAA-enabled LoRaWAN terminal obtains its address and encryption keys.

- **JS (Join Server**): A server that knows the *AppKey* keys of all the terminals on the network, and derives the session keys used to secure exchanges between the *LNS* and the terminals. The advantage of this server is that it can be operated by the end customer when the LNS is operated by an operator: the end customer is therefore the only one who knows the AppKey keys of his terminals. This server can also be integrated into the LNS if the LNS's knowledge of *the AppKey* keys does not pose a security problem.

- **LNS (L**oRaWAN **Network Server):** This server is responsible for registering endpoints on the LoRaWAN network and then managing the confidentiality and integrity of LoRaWAN messages. It communicates with endpoints using the LoRaWAN protocol through LoraWAN Gateways, and with application servers using the TCP/IP protocol via the Internet or a private network.

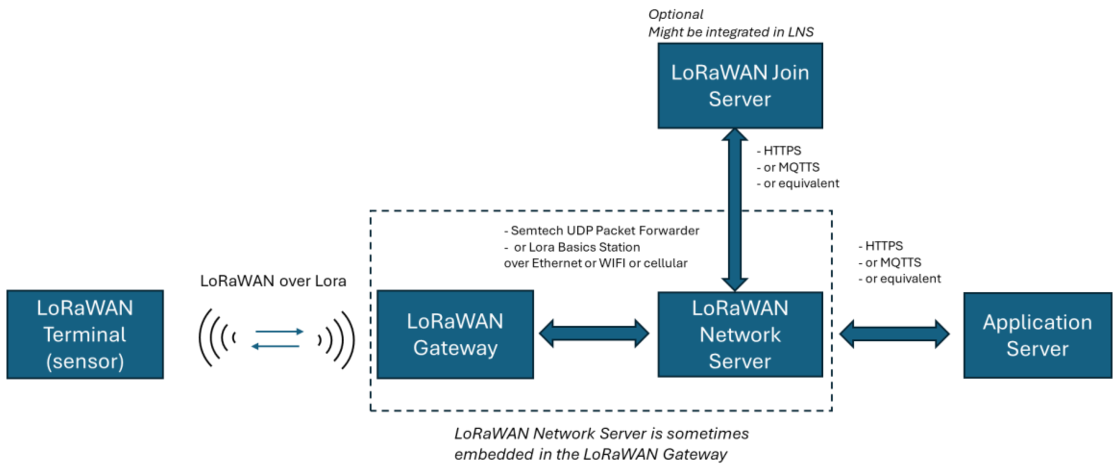


*Figure 1 - LoRaWAN Network Architecture*

-

**MIC:** Message Integrity Code. This is a 32-bit value calculated by the sender of a message, based on the content of the message and an encryption key. The sender adds this MIC code to the end of the transmitted message. The receiver, which has the same encryption key, can verify that the message has not been tampered with by a malicious actor.
Note: in traditional protocols, this code is rather called MAC code. But the term MAC has a special meaning in radio protocols, and so it has been replaced by MIC.

- **Radio module**: A radio module is composed of several electronic components assembled by a manufacturer on a printed circuit board, and most often encapsulated in a metal shield. The components integrated into a module can be processors, *transceivers*, *SoCs*, quartz, and some passive components (resistors, capacitors, chokes). They are pre-certified to comply with current radio standards, and their use therefore greatly simplifies the development of radio equipment. LoRaWAN modules are also certified against the "LoRaWAN certified" label[1].

- **OTAA: "Over The Air Activation**" is the preferred method to activate the registration of a LoRaWAN terminal with a *LoRaWAN Server*.

- **Payload:** A radio message contains a certain amount of information that allows the receiver to synchronize with the transmitter or to detect/correct transmission errors. But this radio message is intended to transmit an application message, which is called the payload of the radio message. A radio message is therefore composed of a payload, and a certain number of other bytes intended to carry this payload.

- **Replay Attack:** A replay attack consists of capturing a message sent by a piece of equipment (a command to open a door for example), and "replaying" it later to make the receiver believe that a new command has been sent.

- **Application Server:** The Application Server is the "end" interlocutor of terminals connected to a LoRaWAN network (see Figure 1 - LoRaWAN Network Architecture). Messages from terminals on the radio interface (e.g. measurements taken by a sensor) are received by a *Gateway* LoRaWAN, which transmits them to a *LNS* optionally integrated into the LoRaWAN gateway. The *LNS* transmits them to the Application Server which interprets them. Messages from the Application Server to terminal equipment follow the opposite path.

- **SoC**: System On Chip. In this document, an SoC is an electronic component that integrates a processor, volatile memory, non-volatile memory, and a radio *transceiver*.

- **Terminals:** In this document, terminals are the sensors or actuators that use the LoRaWAN protocol to communicate.

- **Transceiver**. A transceiver is an electronic component that allows radio waves to **be transmitted** and **received.** It can be a discrete component that will be connected to a processor via a serial interface (SPI, UART, I2C), or it can be integrated inside an *SoC*.

- **Uplink:** Describes the direction of communications from a radio terminal to the network.

# 2 LoRa®/LoRaWAN® technology

The term LoRa is often used when talking about the LoRaWAN protocol. It is therefore useful to clarify what these two terms actually represent.

[1] https://lora-alliance.org/lorawan-certification/

## 2.1 Lora®

LoRa is a modulation patented by Semtech®, derived from CSS (Chirp Spread Spectrum) modulation[2]. Semtech is the only foundry to supply Lora transceivers. It is possible to obtain *SOCs* (System On Chip) that integrate an *ARM processor* and a LoRa *transceiver* from foundries such as ST Microelectronics[3], or *modules* that integrate a LoRa processor and *transceiver* from most Radio module suppliers. The advantage of Lora modulation is that it allows radio signals to be transmitted over distances of several kilometers and that it also offers good penetration in deep indoor environments.

***Note***
*Lora is therefore not an open standard, unlike other technologies such as LTE-M, NB-IOT, Thread or Zigbee. Semtech is the only supplier of this technology, although some foundries such as STMicroelectronics have entered into agreements with Semtech to integrate a Semtech transceiver into some of their SoCs. For example, there is no Lora SoC at Texas Instruments, Silicon Labs, NXP Semiconductors...*

*This point can be considered as a weakness of this technology (a single supplier), but also as a strength: the compatibility between two Lora transceivers is guaranteed since they come from the same foundry.*

LoRa does not define the security parameters, nor the technique of access to the radio interface, nor the addressing, but only the modulation to be used and a "very simple" Radio frame format, illustrated below:

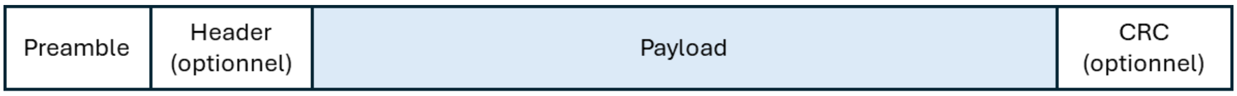


*Figure 2 – Lora frame structure*

A Lora radio frame contains a "Preamble" intended to wake up a receiver, a "Header" which contains information about the length of the *payload* and the coding used to transmit it, and essentially a *payload* which is the actual content of the message. A LoRa frame can also contain a "CRC" (Cyclic Redundancy Control) that allows the receiver to check if the content of the frame has not been altered during its transport over the radio interface.

The purpose of this document is not to describe in detail the LoRa modulation or the structure of LoRa frames. For more details, you can refer to the datasheet of the SX1276/77/78/79 transceivers[4] which explains this in detail.

It is possible to implement many protocols on top of Lora, in star networks or mesh network topologies, but the most widely used protocol at present is the LoRaWAN protocol, which is the subject of this document.

[2] https://en.wikipedia.org/wiki/Chirp_spread_spectrum
[3] ST Microelectronics: STM32WL5x - Wireless MCUs with LoRa support - STMicroelectronics
[4] https://semtech.my.salesforce.com/sfc/p/#E0000000JelG/a/2R0000001Rbr/6EfVZUorrpoKFfvaF_Fkpgp5kzjiNyiAbqcpqh9qSjE

## 2.2 LoRaWAN®

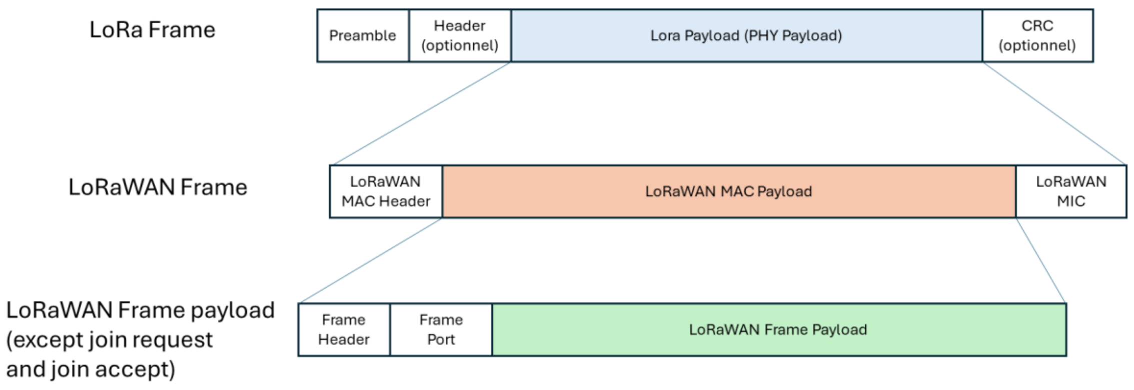
LoRaWAN is an open protocol whose specifications[5][6] are published by the Lora Alliance® and which defines:

- **The radio layer to use**
  - LoRaWAN works in principle on the Lora physical layer,
  - A recent evolution[7] has also specified LoRaWAN on another physical layer, LR-FHSS (Long Range Frequency Hopping Spread Spectrum). This other layer is suitable for very long-range communications (several hundred kilometers), especially for communications (uplink only) to satellites. LR-FHSS is also patented by Semtech. Currently, ST Microelectronics' STM32WL series SOCs include a Semtech *transceiver* that supports both Lora and LR-FHSS modulations.

- **The frequency range to be used**, which depends on the location of the devices.
  - LoRaWAN operates in the ISM 863-870MHz band in Europe[10].

- **The maximum payload size of LoRaWAN messages**
  This maximum size depends on the region.
  In Europe, this maximum size is defined in the document "RP002-1.0.3 LoRaWAN® Regional Parameters". It varies between 59 bytes and 230 bytes depending on the Spreading Factor used.

- **The format of the LoRaWAN frames** that will be transported in the payload of a LoRa frame (or possibly another modulation allowed in the specification). This format is shown below:



The contents of the fields represented above are described in the LoRaWAN protocol specification[7]. This specification describes:

  - The format of the device addresses (4 bytes),
  - LoRaWAN message encryption (AES-CTR),
  - LoRaWAN message integrity (AES-CMAC),

[5] LoraWan Technical Specification V1.0.4: https://resources.lora-alliance.org/technical-specifications/ts001-1-0-4-lorawan-l2-1-0-4-specification
[6] RP2-1.0.3 LoRaWAN® Regional Parameters: https://resources.lora-alliance.org/technical-specifications/rp2-1-0-3-lorawan-regional-parameters
[7] Lora Specification V1.0.4 – October 2020

- The procedures for sending/receiving data between a terminal and a server.

## 2.3 Key features of LoRaWAN

The following sections describe some important features of LoRaWAN

- Bidirectionality in LoRaWAN
- Radio channel management
- Activating a device
- Encryption management
- Integrity Management
- Repetition management
- Protection against replay attacks
- The protocol between the gateways and the LoRaWAN server
- Firmware Over The Air (*FUOTA)* updates

**Note**

*It is important to note that SemTech offers a full open-source implementation of the LoRaWAN specification and optional packages, including FUOTA. The code for this implementation, called Lora Basics Modem*[8]*, is available on github. This is important because the majority of LoraWAN terminals use this code, which greatly improves interoperability with gateways and LNS, knowing that interoperability is a major issue for the adoption of a technology.*

[8] https://github.com/Lora-net/SWL2001/tree/master

### 2.3.1 Bidirectionality in LoRaWAN

To save battery life, Class A LoRaWAN terminals most often keep their radios off, and therefore cannot receive messages. The following mechanism has been specified to allow these devices to review messages:

- After a transmission, a Class A device opens a reception window (RX1) on the same radio channel that it used to transmit. If it does not receive anything, it opens another window (RX2) a second later on a predefined channel. This mechanism is shown below:

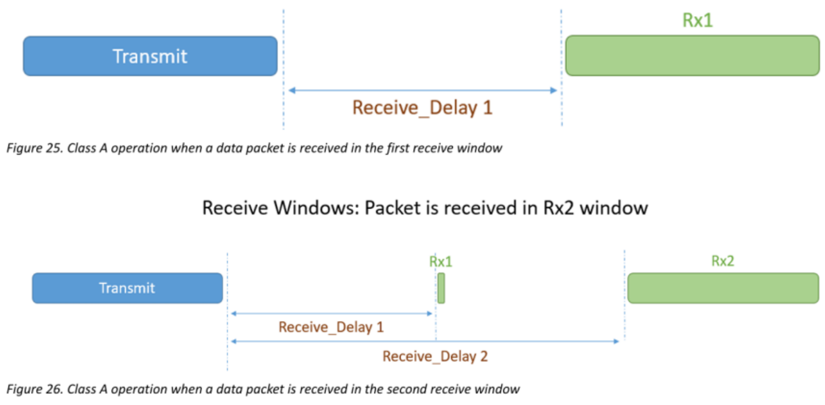

Transmit
Rx1
Receive_Delay 1
Figure 25. Class A operation when a data packet is received in the first receive window
Receive Windows: Packet is received in Rx2 window
Transmit
Rx1
Rx2
Receive_Delay 1
Receive_Delay 2
Figure 26. Class A operation when a data packet is received in the second receive window

*Figure 3 - Source: Semtech AN1200.86*

- The delays between the emission of a frame and the opening of the RX1 and RX2 windows are configured by the LNS[9].

### 2.3.2 Channels/frequencies defined by LoRaWAN

In Europe[10], a LoRaWAN terminal must implement[11] the following three channels that will be used in *Uplink* and *Downlink*: 861.1MHz, 861.3MHz, 861.5MHz. These channels are used in particular to send "join requests" messages to the *LoRaWAN Server*. In its response to the "Join Request", a *LoRaWAN server* can optionally add 5 additional channels. Finally, it can still 8 more channels in the 867-870MHz band in case of a very dense network via another protocol message[12].

The three channels that are required to be supported by all LoRaWAN terminals in Europe are as follows:

| Channel | Frequency | Bandwidth | DR (DataRate) / Bitrate | Duty cycle |
|---|---|---|---|---|
| 0 | 868.1 MHz | 125 kHz | DR0 to DR5 / 0.3-5kbps | 1% |
| 1 | 868.3 MHz | 125 kHz | DR0 to DR5 / 0.3-5kbps | 1% |
| 2 | 868.5 MHz | 125 kHz | DR0 to DR5 / 0.3-5kbps | 1% |

[9] The Receive_Delay1 and Receive_Delay2 timeouts are defined in the RXParamSetupReq message.
[10] For other regions, see RP002-1.0.3- LoRaWAN Regional Parameters
[11] https://resources.lora-alliance.org/technical-specifications/rp2-1-0-3-lorawan-regional-parameters
[12] The NewChannelReq message allows you to add or edit channels.

In Europe, the five channels configured by the server in the response to the join request are free, as long as they comply with the regulations, and could be configured by the server in the following way:

| Channel | Frequency | Bandwidth | DR (DataRate) / Bitrate | Duty Cycle |
|---|---|---|---|---|
| 4 | 867.1 MHz | 125 kHz | DR0 to DR5 / 0.3-5kbps | 1% |
| 5 | 867.3 MHz | 125 kHz | DR0 to DR5 / 0.3-5kbps | 1% |
| 6 | 867.5 MHz | 125 kHz | DR0 to DR5 / 0.3-5kbps | 1% |
| 7 | 867.7 MHz | 125 kHz | DR0 to DR5 / 0.3-5kbps | 1% |
| 8 | 867.9 MHz | 125 kHz | DR0 to DR5 / 0.3-5kbps | 1% |

- A special RX2 channel[13] is dedicated to *downlinking*, with better Duty Cycle and more power[14]. The *LNS* may decide, for its own reasons, to use this channel to send a request or acknowledgement to a terminal. This channel must therefore also be supported by terminals, receiving only.

[13]default = 869.525MHz, bandwidth=125kHz
[14] The RX2 channel supports a duty cycle of 10% instead of 1% for the other channels and allows transmitting at 27dBm instead of 14dBm for the other channels.

### 2.3.3 Activating a device

**Factory Setup**

First, a terminal must have been configured at the time of installation with the following minimum settings:

- **devEUI**: A unique 64-bit encoded terminal identifier. This setting can be public.
- **joinEUI**: A 64-bit encoded "join" Server identifier. This setting can be public.
- **AppKey**: An AES-128 secret key that will be used during the join procedure, which is only required in the case of an OTAA activation as described below. This key will also be known to the LNS (or Join Server). **This setting must be kept secret**.

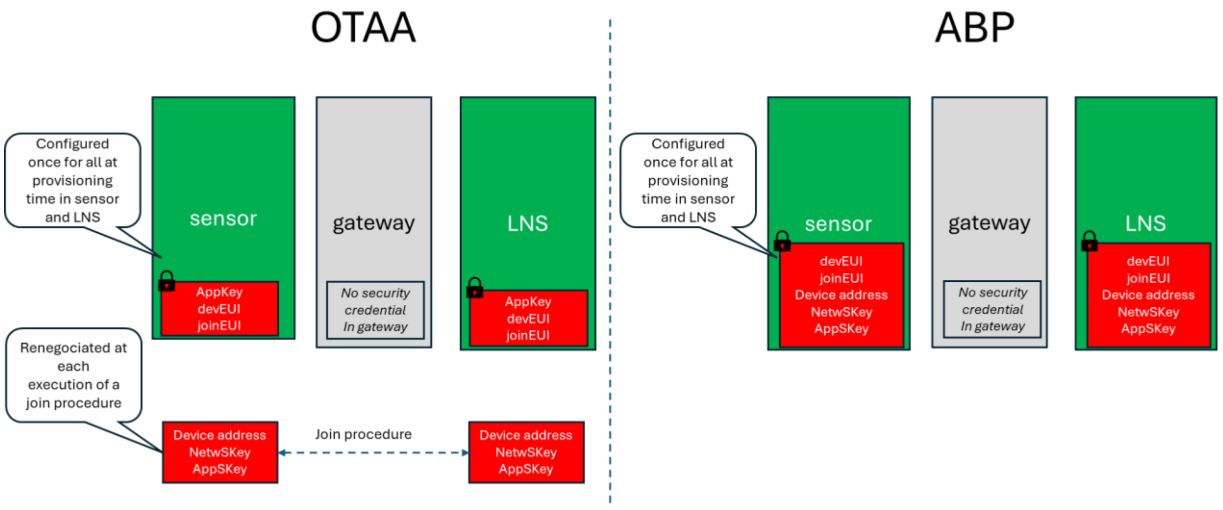
**Activation**

To be able to communicate with an LNS, this terminal must be activated. The purpose of this activation is to equip this equipment with:

- An **address** (32 Bits): allocated by the server in the case of *OTAA* or configured at the time of installation in the case *of ABP* as described below.
- A 128-bit AES key (**NwkSKey**) to encrypt the payload of messages containing network commands, and to compute the MIC code of LoRaWAN frames.
- A 128-bit AES key **(AppSKey**, not to be confused with **AppKey)** to encrypt the payload of messages containing data from the final application. The advantage of this second key is that the network provider will not be able to decrypt the application messages[15].

This activation can be done in two ways:

- Activation By Personalization (*ABP*): Activation mode NOT recommended.
- Activation Over The Air (*OTAA*): Recommended activation mode.


OTAA
ABP
Configured once for all at provisioning time in sensor and LNS
sensor
AppKey devEUI joinEUI
gateway
No security credential In gateway
LNS
AppKey devEUI joinEUI
Renegociated at each execution of a join procedure
Device address NetwSKey AppSKey
Join procedure
Device address NetwSKey AppSKey
Configured once for all at provisioning time in sensor and LNS
sensor
devEUI joinEUI Device address NetwSKey AppSKey
gateway
No security credential In gateway
LNS
devEUI joinEUI Device address NetwSKey AppSKey

[15] If the Join server is integrated in the LNS, this point is a bit "theoretical" since the LNS will know the AppSKey keys. So, if the LNS is compromised, so are the application messages.

## Activation OTAA

OTAA (Over-The-Air Activation) is the recommended mode to ensure a high level of security. Unlike the ABP method (described below), OTAA allows you to dynamically generate two session keys that will be used to ensure the confidentiality and integrity of communications: the NwkSKey (Network Session Key) and the AppSKey (Application Session Key). These keys are not pre-registered in the equipment at the factory and are renewed with each new activation procedure, which greatly enhances the security of the device.

In detail, this procedure works as follows:

### Sending a Join-Request message

To join the network, the terminal sends a Join-Request message, whose payload contains:

- DevEUI (Device-Unique Identifier)
- AppEUI (application or network identifier)
- The DevNonce, a unique number used to avoid replay attacks.

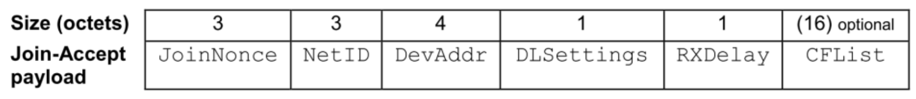
> DevNonce should never be reused, even after a restart or battery change so as not to compromise the safety of the system. It must therefore be stored in non-volatile memory, usually as an incremental counter for each activation attempt.

The Join-Request message is sent in the clear on one of the three mandatory channels (0 to 2) with any allowed LoRa rate. Message Integrity Code (MIC) is calculated using the AppKey, which is preconfigured at the factory.

If no response is received, the terminal retransmits the message by randomly jumping to another authorized channel, until it receives a valid response.

### Receiving a Join-Accept message

In response, the server sends a Join-Accept, whose payload is:



| **Size (octets)** | 3 | 3 | 4 | 1 | 1 | (16) optional |
|---|---|---|---|---|---|---|
| **Join-Accept payload** | JoinNonce | NetID | DevAddr | DLSettings | RXDelay | CFList |

Table 54: Join-Accept payload format

This payload essentially contains:

- JoinNonce: A nonce generated by the LNS
- DevAddr: the terminal address allocated by the LNS
- NetID, DLSettings, RXDelay: network settings.
- CFList: If present, allows you to specify up to 5 additional channels (channels 3 to 7) that the terminal can use. Subsequently, the LNS can also dynamically adjust the channels and rates via LinkADRReq messages (ADR = Adaptive Data Rate).

From the information contained in the Join-Accept, including the JoinNonce, the terminal and the LNS derive the two session keys in a synchronized manner:

- NwkSKey: Used for the integrity of messages exchanged with the network.
- AppSKey: Used for encryption/decryption of application messages.

The AppSKey key is transmitted by the LNS to the application server, so that the latter can interpret the messages coming from the terminal.

**Activation by Personalization (APB)**

This mode is not recommended because the security is minimalist. The terminal address as well as the **AppSKey** and **nwkSKey** keys are configured in the terminal before it can connect to the network. There is therefore no *join* procedure. These keys must be unique for each device, so that compromising one device doesn't compromise the entire network. These keys can never be changed, which goes against the new RED directive in Europe which requires that security assets can be updated if the device handles personal data.

### 2.3.4 Privacy management

LoRaWAN uses symmetric encryption algorithms to ensure the confidentiality and integrity of the messages exchanged. The same encryption keys must therefore be known by the two communicating entities, namely a terminal and an LNS.

If a LoRaWAN frame contains a payload, it is encrypted with the AES-CTR algorithm, and a 128-bit AES key.

- If the payload contains only LoRaWAN commands, it is encrypted with the **NwkSKey key**.
- Otherwise, it contains application data and is encrypted with the **AppSKey** key. This ensures that the network manager will not be able to decrypt the application messages.

### 2.3.5 Integrity Management

All LoRaWAN frames contain a *Message* Integrity Code (MIC). This *MIC* code is computed by the AES-CMAC algorithm, using the **NwkSKey** key, except for the messages used during the join procedure in the case of *OTAA* activation where the *MIC* code is computed with the **AppKey** since the **NwkSKey key** has not yet been negotiated between the terminal and the network.

### 2.3.6 Rehearsal management

To improve network quality, *LNS* can tell endpoints to repeat each message sent a certain number of times[16]. In this case, the terminals will repeat the messages several times (unless they receive a message from the *LoRaWAN Server*), changing channels with each retransmission. This mechanism will make it possible to compensate for radio interference generated by equipment that would be located near the Lora gateway and that would generate a radio signal in one of the Lora channels used by the terminal, or possible collisions with another message from another sensor.

[16] NbTrans parameter of the LinkADRReq message

### 2.3.7 Protection against replay attacks

To protect against replay attacks, the LoRaWAN specification defines for each terminal two 32-bit frame counters for each terminal to *The Uplink*[17] and the *downlink*[17] :

- Each time it sends a new message (so not in the case of repeating the same message), a terminal increments its *uplink counter*, and integrates this counter[18] into its message. The LNS memorizes this counter, and will ignore frames that have the same counter, and that are not repeats.
    - Therefore, it will be impossible for a malicious user to replay a message that he has captured because this message would be considered a repetition of a message already received, and it would therefore be ignored by the network.

- Each time it sends a new message to a terminal, the server increments its *downlink* counter. A terminal must therefore ignore a frame that would contain a *downlink counter* that has already been received.

- If the device supports "activation over the air" (OTAA), the *uplink* and *downlink* counters are reset to zero at the end of each activation sequence. Otherwise, the terminal supports "activation by customization" (ABP) and the counters should never be reset, even in the event of a restart, even in the case of battery replacement.

### 2.3.8 The protocol between gateways and LoRaWAN servers

There are several protocols for interfacing a gateway with an LNS.

- The "Semtech UDP packet forwarder" protocol
- Lora Basics$^{TM}$ Station

**Semtech UDP packet forwarder**

Historically, Semtech had specified the "Semtech UDP packet forwarder" protocol[19] between LoRaWAN gateways and LNS. But this protocol is based on UDP, and does not provide any privacy or authentication mechanisms. It is therefore possible to spy on communications between an LNS and a Gateway, or to send messages to an LNS by pretending to be an official gateway, and vice versa.

This protocol is obviously not recommended for deployment, unless the gateways and the LNS are part of the same private network, which would itself be protected against external attacks.

[17] Uplink Counter = FCntUp and Downlink Counter = FCntDown
[18] In fact, only the 16 bits of low weight of this meter
[19] semtech-udp/PROTOCOL.TXT at master · Helium/Semtech-UDP · GitHub

**Lora Basics™ Station**

Subsequently, Semtech developed an open source implementation of a "packet forwarder" that offers secure communications: Lora Basics Station[20]. This protocol allows you to define several levels of security between a gateway and an LNS. Depending on the configuration of the system, the communication can be in clear and without mutual authentication, or impose authentication of the LNS by the gateway, or impose mutual authentication between the gateway and the LNS. In the latter cases, HTTPS[21] or WSS[22] protocols are used.

In addition, some LoRaWAN gateways integrate an LNS. They therefore make it possible to implement a LoRaWAN network without any interaction with the outside world. In this case, the application messages are transmitted to the *Application Server* using standard protocols such as MQTTS or HTTPS that are completely outside the scope of the specifications or recommendations published by Semtech or by the LoRaWAN alliance.

### 2.3.9 Over The Air Terminal Firmware Updates (*FUOTA)*

The Lora Alliance has published a set of technical specifications[23] to describe the firmware update of LoRaWAN terminals. The implementation of FUOTA is optional.

The operation is based on the following steps:

- At the request of the application server, a dedicated server, called MDS (Multicast Distribution Server), is a multicast group of all the terminals whose firmware needs to be updated.

- The server notifies each targeted device individually of its inclusion in the group. It also communicates the associated multicast address to the user, as well as the specific encryption keys that will be used to secure the transmission of the data blocks containing the new firmware.

- The MDS then instructs these terminals to set up periodic receive windows in order to receive the firmware blocks. In concrete terms, this amounts to temporarily activating a class B type behavior, for the duration of the download session.

- Firmware blocks can be reissued multiple times to compensate for possible transmission losses or errors.

- Once the download is complete, the MDS checks device by device to see if all the new firmware has been received. If successful, it can then ask the terminal to validate and activate the new firmware, either immediately or at a later date.

*This feature is particularly strategic to ensure that equipment complies with the RED Directive, in particular by facilitating large-scale updates.*

[20] https://doc.sm.tc/station/index.html
[21] https://fr.wikipedia.org/wiki/Hypertext_Transfer_Protocol_Secure
[22] https://fr.wikipedia.org/wiki/WS-Security
[23] https://resources.lora-alliance.org/technical-specifications

# 3 Threats to LoRaWAN networks

Before looking in detail at the measures to be put in place to secure a LoRaWAN network, it is important to identify the threats and risks associated with this type of deployment. It will then be up to the end user to judge whether these threats are relevant to the use they want to make of the network, and to put in place the measures to protect themselves against these threats. Most of these threats are common to all wireless communication protocols. In summary, the risks are as follows:

- Make the network completely inoperable, and therefore reduce the return on investment to zero.
- The theft of personal data, if the network is used to manage private data (door openings, windows, etc.). in a smart home deployment for example).
- Theft of monetary data, if the network is used to manage data with a monetary value (water meter, parking meters, etc.). In a Smart City deployment, for example).

The main threats are described in the following sections:

- Cloning Devices
- Identity theft
- Replay Attacks
- Data theft
- Alteration of transmitted data
- Denial Of Service

## 3.1 Cloning Devices

This threat concerns the device manufacturer rather than the end user. It will still be up to the end user to ensure that they have acquired original terminals and not copies.

The associated risks are as follows for the user:

- A non-original endpoint can include malware that will transmit fake data, or disrupt the LoRaWAN network. It can therefore render the deployed system completely unusable.
- Loss of the manufacturer's warranty, and potential problem with the quality of the terminals.

## 3.2 Identity theft

A malicious actor emits erroneous data to a gateway by pretending to be a device that is part of the network, or emits erroneous data to the LNS by pretending to be an official gateway, or vice-versa.

The associated risks are as follows for the user:

- If a single terminal is compromised, that terminal will become completely unusable in the system since the LNS will receive inconsistent data from it, consisting of a mix of real data and fake data. If the real terminal itself is attacked (destroyed, or simply prevented from transmitting data) the deployed system will be unusable.

### 3.3 Replay Attacks

A malicious actor listens for a communication between a sensor and a gateway, or between a gateway and an LNS, and then replays that message.

- For example, an actor could listen to a network command asking to open a valve, or a lock... then replay it much later to force a new execution of this action.
- If the role of the sensor is to generate alarms, a malicious actor could also trigger nuisance alarms by replaying a real alarm message and disrupt the operation of a tertiary building, for example.

### 3.4 Data theft (Passive eavesdropping)

A malicious actor spies on exchanges between a sensor and a gateway, or between a gateway and an LNS, and extracts information from them.

- If the terminals are used in an access control system, for example, passive eavesdropping will allow a malicious actor to identify entry and exit times (in a smart home application, for example), and possibly the identities of people entering and exiting depending on the system deployed (in a smart building application, for example).

### 3.5 Alteration of transmitted data

A malicious actor modifies the data emitted by a device or gateway before it arrives at the LNS, or vice-versa.

- For example, he could change the value of the index of a water meter to reduce his bill.
- In a city or agricultural irrigation management system, the transmission of false soil moisture data could compromise crops and render the system unusable.
- In a traffic management or parking space management system, the system would once again become completely unusable.

### 3.6 Denial Of Service (DOS) Attacks

A malicious actor prevents data from sensors or gateways from reaching the LNS, or vice versa.

- This type of attack renders the network completely inoperable.

# 4 Securing a LoRaWAN network

**Preliminary note on the security of the AES algorithm**

The AES algorithm allows you to encrypt a message from a symmetric key. If you encrypt the same message twice with the same key, you get the same result. This aspect is a weakness of this type of algorithm, which can be exploited by a malicious actor to break the security of the protocol.

For this reason, AES-based constructs such as AES-CTR add a "nonce" to the beginning of the message to be encrypted: this "nonce" is a number that should never be reused with the same encryption key. Therefore, if we encrypt the same message twice with the same key, we will obtain two different results, provided that we have not reused a "nonce" that has already been used.

Most LoRaWAN messages (outside of the join procedure) are encrypted with the NetwSKey key and the AES-CTR algorithm. In this case, the nonce is a simple counter, incremented with each new cryptographic operation. It is therefore important that this nonce is not reset when a device is restarted, because a malicious actor who has physical access to the device could cause reboots, and the device would have to encrypt a message with the same nounce several times, which would be a major security flaw.

The Join Request message is not encrypted, but it includes a MIC code calculated with the AppKey key. To protect against replay attacks, the payload of the Join Request message itself contains a nonce that should never be repeated, even if the device is restarted or its battery is replaced.

## 4.1 The LoRaWAN protocol version

The LoRaWAN protocol has evolved over time, in particular to improve its security. The versions currently deployed are as follows:

- 1.0.2
- 1.0.3
- 1.0.4

There is also a version 1.1 but it is not widely deployed to my knowledge. However, the major security changes introduced in version 1.1 have been carried over to version 1.0.4.

Versions 1.0.2 and 1.0.3 have a number of security vulnerabilities[24] that are fixed in version 1.1 (and therefore in version 1.0.4). Some flaws are quite "theoretical" because they assume that the attacker knows exactly when messages are transmitted, but other flaws are easier to exploit. Notably, the most important flaws fixed in versions 1.1 and 1.04 are the following[25]:

- The frame counters (nonce) of ABP-enabled devices are reset to zero when a device is rebooted.
    - **Compromised security of ABP-enabled devices**
      A malicious actor with access to a device could cause reboots for the device to generate multiple frames with the same nonce. This would completely compromise the security offered by AES-CTR.

    - **Denial Of Service on ABP-enabled devices**
      A malicious actor listening to communications from an ABP-enabled device could force a device to restart and then replay recorded messages. This would have the effect that the messages sent by the real terminal after it has been started would be ignored by the LNS because the counters of these messages would be lower than the

[24] The uplink frame counters are reset to zero at each start in case of ABP activation, which leads to a security vulnerability. The nonces used in join request messages are random values, so there is a risk that the same nonce will be reused after a certain number of messages. In version 1.0.4, a counter incremented with each new join procedure takes the place of a nonce. This counter should not be reset to zero in the event of a reset.
[25] LoRaWAN: Part 2, Protocol Attacks - Kereval

counters of the replayed frames: this is equivalent to executing a Denial Of Service attack on this terminal.

- **Denial Of Service on OTAA-enabled devices**
  The 1.0.3 specification states that the nonces used in Join Request and Join Accept messages should be chosen randomly by devices and LNS. To protect against replay attacks, servers and terminals must memorize "a certain number" of nonces already received. Servers have enough memory to store 65535 nonces for a terminal, but terminals surely don't have enough memory to do the same. An attacker can therefore respond to a join request message sent by a terminal with a join accept that he has memorized, and which dates back some time. From that point on, the terminal would be connected to a "phantom LNS", and would be unable to communicate with the real LNS.

These flaws assume that the attacker can listen to radio exchanges, which is within the reach of anyone in possession of a Lora receiver. They are therefore easily exploitable. They sometimes assume that the attacker has physical access to the endpoints, and that they are able to trigger a reboot of the endpoint. This point is also very plausible in a Smart City or Smart Building type deployment since the sensors are not physically protected. In addition, simply removing the batteries from the sensor will cause a restart.

Therefore, it is recommended to use terminals that implement a protocol version greater than or equal to 1.0.4.

## 4.2 Terminal certification

The LoRa Alliance has defined a certification program[26] to ensure that a device complies with the LoRaWAN specification. The use of terminals with this certification ensures that they correctly manage the security aspects defined in the LoRaWAN specification. Especially:

- The negotiation of AppSKey and NtwSKey keys in the OTAA procedure.
- LoRaWAN encryption and frame integrity management.
- The handling of the nonces used in the join procedure, and the persistence of these nonces in the event of a terminal reboot.
- The management of uplink and downlink counters used as IVs for the encryption of LoRaWAN frames.

## 4.3 Securing AppKey Keys

The security asset that is absolutely critical in a LoRaWAN network is the *AppKey* stored in the terminals and in the *Join Server*, which is itself often integrated into the *LNS*. This is because this key is used to execute the *join procedure*, which will itself be used to define the session keys. Also, this key can't be changed once the device has been deployed.

> **An attacker who has this key can execute all the attacks mentioned above. It can therefore render a terminal, or even the network completely unusable without it being possible to correct the problem without physically accessing the terminal.**

[26] https://lora-alliance.org/lorawan-certification/

Since this key is known to the terminal and the Join Server, protecting it is not a simple problem:

- A malicious actor could easily steal a deployed device, take it to their lab to analyze it, and try to extract the AppKey from it.
- A user (possibly a trainee, etc.) who has access to the Join Server (or the LNS if the Join Server is integrated into the LNS) can access all the AppKey keys of all the deployed terminals.

Recommended measures to secure these keys include:

- **Use a unique AppKey for each device**
  This key must be unique for each device to prevent the compromise of one device from compromising the entire network. This key should not be easily deduced from the serial number of the product, for example, but rather obtained through a "derivation function" that implements known information and a secret key[27].

- **Control access to AppKeys in servers**
  AppKeys are also stored in the Join Server (or LNS). They are therefore potentially accessible to an operator who has access to the LNS or the Join Server. It is therefore also necessary to ensure that access to these keys is only available to a very limited number of users, and not to all the operators of these servers.

  In the event that the end customer outsources the infrastructure to a telecom operator or integrator, it is recommended to use a Join Server that is separate from the LNS. This *Join Server* should be managed by the end customer and not by the operator. In this case, the LNS operators will not have access to the keys of the terminals, and it will "suffice" to secure access to the Join Server. This option is only possible from the LoRaWAN 1.0.4 specification.

- **Verify the security of keys in endpoints**
  This verification is not necessarily feasible by the end user, who most often does not have access inside the terminals. However, it is possible to audit endpoint vendors to find out their strategy for protecting security assets. In particular, the following strategies are recommended:

  - DEBUG interfaces (including the JTAG interface) must be disabled on terminals. Indeed, a malicious actor who could connect to these interfaces could access the security keys stored in the device, including the *AppKey* that allows the device to register on the network. It could also replace the terminal's firmware with firmware intended to disrupt the network, by sending false data, or by sending too much data, or by executing numerous join procedures...

  - Security keys, especially the AppKey, must not be readable in the terminal. Ideally, this key should be stored in a Secure Element, but there are few low-cost sensors that are equipped with this type of functionality. These keys can also be stored in the terminal's internal flash since the JTAG interface is disabled, or stored in an external memory. In the latter case, they must be encrypted with a key that cannot be read from the outside or

[27] https://cyber.gouv.fr/sites/default/files/2021/03/anssi-guide-selection_crypto-1.0.pdf

inferred from publicly known information (such as the device's serial number, for example).

- If the sensor data is transmitted over the internet - which is almost always the case, except in the case of a completely private deployment where the sensors, gateways and LNS are co-located - and if it is possible to identify a user, a natural person, from this data - which is less often the case except in smart home applications - the terminal must implement the *FUOTA* functionality (Firmware Update Over The Air), or at least it must provide a way to update its firmware. This constraint is introduced by the RED Directive.

## 4.4 Fraud detection

- **Detect anomalous behavior at the LNS level**
  Since no protection mechanism is infallible, it is recommended to be able to detect abnormal behavior of one or more terminals so that they can be isolated or replaced in the event of suspected fraud. This mechanism should be implemented in the LNS, which has a global vision of the network.

- **Endpoint Intrusion Detection**
  Ideally, the sensor should implement an intrusion detection feature, and be able to generate an alert to the application server in the event of a suspected intrusion.

## 4.5 Terminal activation

If they do not have physical access to the terminals or the LNS, a malicious user can still spy on the radio interface, since the frames emitted by the sensors or gateways can be listened to by anyone, even at a great distance, as long as that person has a Lora receiver.

Cryptographic algorithms such as AES-CTR are vulnerable if the "nonces" used to encrypt messages are used more than once. The LoRaWAN specification recommends that these nonces should never be reused, even if the device is rebooted.

**Prefer OTAA activation to ABP**

The LoRaWAN specification also plans to reduce the period during which the system is "attackable" by using ephemeral keys, redefined with each new join procedure, rather than keys stored once and for all in the terminals, without being able to be modified. This is one of the advantages of the OTAA activation procedure: the AppKey is only used once during the join procedure, to encrypt the join accept message. Once the join procedure is complete, the next messages will be encrypted and authenticated with the **AppSKey** and **NetwSKey** keys that will be negotiated between the terminal and the LNS during the join procedure. A new join procedure can be triggered if fraud is suspected, although it is not recommended to run this procedure too frequently[28].

**Remark**

[28] No more than once a month in the worst case according to https://lora-alliance.org/wp-content/uploads/2021/05/TR007_Developing_LoRaWAN_Devices-v1.0.0.pdf?utm_source=chatgpt.com

> The OTAA procedure also allows you to change your LoraWAN network provider without having to physically intervene on the sensors to program the new provider's netSKey appSKey keys.

## 4.6 Choose the network topology

The risks associated with deploying a LoRaWAN-based solution depend primarily on the architecture deployed. One of the following topologies can be considered:

- LoRaWAN network operated
- Private LoRaWAN Network with Hosted LNS
- Private LoRaWAN Network with Private LNS

### 4.6.1 LoRaWAN network operated

The end user can delegate the management of his network to a mobile operator or an integrator. This type of deployment can be found in smart cities, for example. In this case, a large part of the security of the system will rest on the shoulders of the subcontractor (operator or integrator), whose cybersecurity capabilities will have to be audited:

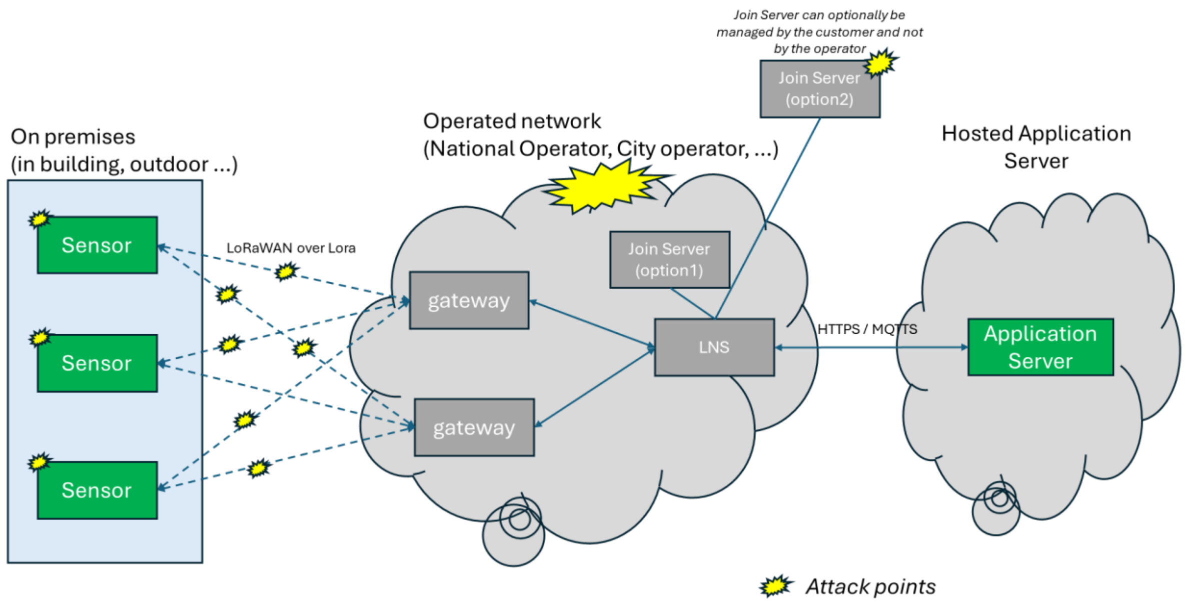

Join Server can optionally be managed by the customer and not by the operator
Join Server (option2)
On premises (in building, outdoor ...)
Operated network (National Operator, City operator, ...)
Hosted Application Server
Sensor
Sensor
Sensor
LoRaWAN over Lora
gateway
gateway
Join Server (option1)
LNS
HTTPS / MQTTS
Application Server
Attack points

*Figure 4 - LoRaWAN Operated network*

In this case, apart from the sensors themselves and the radio interface, safety management is essentially the responsibility of the operator or subcontractor who manages the network.

- **Jamming protection**

  In this type of deployment, radio jamming (at least in uplink) seems very difficult because it would be necessary to emit a radio signal strong enough to disturb the operator's gateways. In addition, the signals emitted by the sensors are usually received by several gateways, which makes the problem even more complicated for a malicious actor. Finally, insofar as an operator's gateways manage a large number of terminals, a malfunction of the gateways would be very quickly detected by the operator. This problem is therefore not critical in this type of deployment, at least for uplink communications.

In downlink, the LNS can request that messages be acknowledged by the terminal. This mechanism will therefore make it possible to detect a malfunction of the terminal, related to jamming or another cause.

- **La protection des clés AppKey**

  - **Manage access in the LNS**
    If the LNS integrates the Join Server, it knows the appKeys of all endpoints, which represents a major risk to network security. Indeed, depending on the way in which the access rights to the LNS will be configured by the subcontractor, unauthorized persons (possibly interns, etc.) could have access to these keys.

  - **Use an End Customer-Managed Join Server**
    Since it seems impossible for a customer to audit the operator's LNS configuration, the solution is to use a separate Join Server, managed by the customer itself and not by the operator. In this case, this Join Server will be the only one that knows the AppKey keys of the terminals, and the LNS of the operator will query this Join Server each time a terminal requests a join.
    The majority of LNS operators offer this option. The same goes for the open-source LNS Chirpstack. Obviously, it will be necessary to correctly configure the access rights to the Join Server to ensure that only a very small number of administrators will have access to the AppKey keys.

  - **Pair the Join Server with a Secure Element**
    To avoid the risk of misconfiguring access rights to the Join Server's AppKey keys, the Join Server can be coupled to a Secure Element[29], which will manage the AppKey keys of the terminals. In this case, even the operators of the Join Server will not have access to the AppKeys keys.

It should be noted that, even in the case of a separate Join Server, the compromise of the LNS managed by the subcontractor will lead to catastrophic consequences on the security of the network. The LNS does not have access to AppKeys but will know the session keys used to secure the exchanges. The security of the LNS is the responsibility of the network operator.

*Note: We are not interested in the interface between the LNS and the application server because it is completely independent of LoRaWAN. Securing this interface requires the use of secure protocols commonly used on the Internet (SSL/TLS, HTTPS, MQTTS, etc.).*

### 4.6.2 Private LoRaWAN Network with Hosted LNS

In smaller deployments (in terms of surface area to be covered), typically in smart building applications, the building manager will be able to deploy and operate its own gateways. In general, it will use a hosted LNS (e.g. "*TTN*", **The T**hings **Network**[30] or ThinkPark[31], ...). Again, even when using secure protocols between the gateways and the LNS, part of the security of the solution will rest with the hosted LNS solution provider.

[29] A tamper-proof physical component that stores keys and performs encryption operations
[30] https://www.thethingsnetwork.org/
[31] https://www.thingpark.com/thingpark-enterprise/

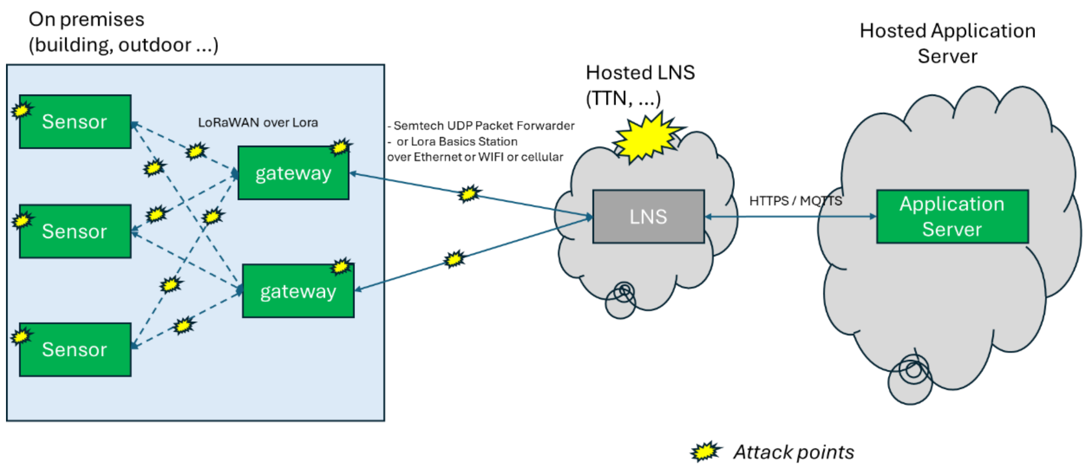


*Figure 5 - Private Network with hosted LNS*

- **Jamming protection**
  As a rule, it is not possible to protect against interference. The only possible measures are to detect the jamming, to trigger an investigative action.

  - **Radio jamming**
    In this type of topology, it is quite easy for an attacker to locate a gateway inside the building and jam the signal near that gateway. Even if there is no protection against jamming, it is safer to use several gateways located in different locations to complicate the attacker's task: messages from the sensors will be received by several gateways in this case.
    **However, Lora modulation is particularly robust against this type of interference[32].** An attacker will therefore have to install jammers near each gateway, or use a powerful jammer that will be more easily detectable.

  - **Protocol jamming (MAC-level jamming)**
    Since Lora modulation is resistant to radio jamming, attackers will prefer "protocol jamming" which consists of generating a large number of LoraWAN packets from a LoraWAN terminal, so that these packets collide with real packets at the gateways.
    Gateways are not capable of detecting this type of jamming. It will therefore be necessary to use "intelligent" algorithms at the LNS level to detect abnormal behavior at a particular site, and trigger investigations in the event of suspected jamming. Abnormal behaviors are, for example, receiving a large number of packets with an incorrect MIC, the absence of data from sensors that are supposed to periodically emit...

- **Protection des clés AppKey**
  In this scenario, the LNS and Join Server considerations mentioned in the previous section also apply.

- **Securing the interface between gateways and LNS**

[32] https://www4.comp.polyu.edu.hk/~csyqzheng/papers/LoRaJamming-INFOCOM21.pdf

This scenario presents an additional weakness in the interface between the gateways and the LNS. Some commercial gateways use the UDP Packet Forwarder protocol to communicate with the LNS. This UDP-based protocol does not provide any security, transmission guarantee, or mutual authentication mechanism between gateways and servers. Therefore, this protocol should not be used in this type of deployment, but rather use an SSL/TLS-based protocol such as Lora Basics$^{TM}$ Station[33].

### 4.6.3 Private LoRaWAN Network with Private LNS

Finally, in even smaller deployments, or those requiring a higher level of security, the end user will be able to host an LNS on their premises. The data will therefore not pass over the internet, and the end user will not depend on a third party to ensure the security of his data. On the other hand, in this case, it will need to have the necessary skills to fully manage its gateways, and deploy and manage an LNS on a private server. Hosted LNS providers often provide the option to host their LNS on a server that is located on the customer's premises. The end user will also be able to deploy an open source LNS on a local server, but this will require specific skills to ensure the security of his LNS. For example, ChirpStark[34] is an Open Source LNS that can run on a small linux server (or even a Raspberry Pi platform).

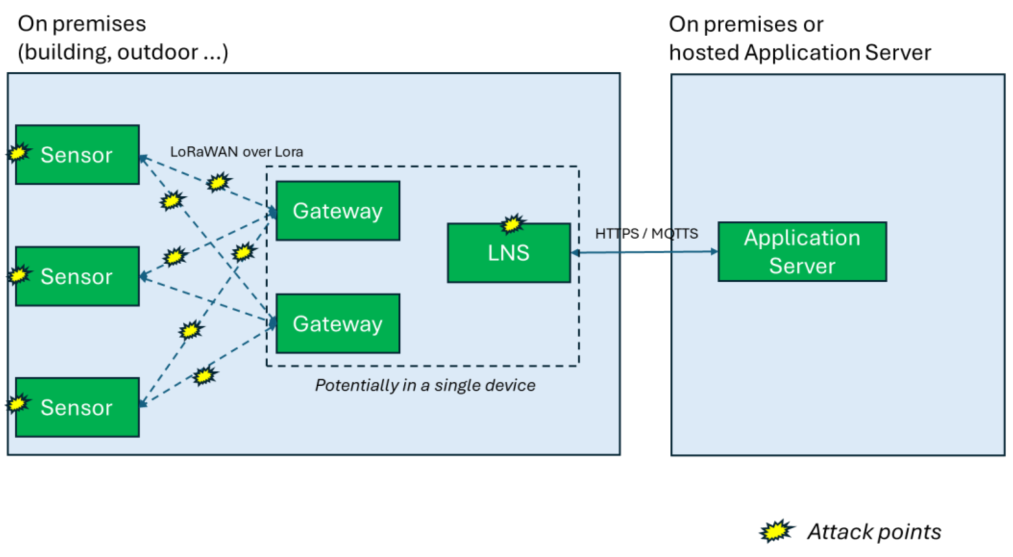


*Figure 6 - Private Nework with private LNS*

An even simpler solution is to use a Gateway that itself integrates an LNS, like some Multitech gateways[35], or gateways based on Raspberry Pi[36].

In this type of deployment:

- Jamming protection

[33] https://doc.sm.tc/station/
[34] https://www.chirpstack.io/
[35] https://multitech.com/all-products/cellular/cellular-gateways/conduit-300/
[36] https://www.chirpstack.io/docs/chirpstack-gateway-os/index.html

The interference considerations apply in the same manner as in the previous case. It will therefore be necessary to implement an "intelligent" algorithm at the LNS level to detect abnormal situations.

- Protection des clés AppKey
  There is no need to separate the Join Server from the LNS. However, it will be necessary to properly manage access rights to the LNS, and possibly associate a Secure Element with the LNS to prevent people with access to the LNS from being able to access the AppKeys of the terminals.

- It doesn't matter which protocol is used between the gateway and the LNS, and even the Semtec UDP Packet forwarder protocol is acceptable.

## 4.7 Implement Firmware Over The Air Update

As discussed above, it is impossible to design a system with the certainty that an attacker will not one day find a security vulnerability. For this reason, it is very important to plan for the Over The Air software update.

If the devices contain "personal data", this update is also made mandatory by the cybersecurity changes of the RED directive in Europe.

## 4.8 Protection against the corruption of application data

**Application data privacy**

The application data is encrypted with the AES-CTR algorithm and the *AppSKey* key, known only to the device and the application server. As a reminder, the *AppSKey key* was negotiated between the terminal and the Join Server at the time of OTAA activation, or configured in the terminal during the installation procedure in the case of ABP activation.

The confidentiality of application data is therefore ensured.

**Application Data Integrity**

The MIC code defined in the LoRaWAN protocol specification is used to verify the integrity of the frames exchanged between a terminal and an LNS. But it does not allow the integrity of application messages to be verified. It would therefore be possible for a corrupted LNS to modify the content of an application message without the application server noticing, even if the application message is encrypted with the **AppSKey** key. Indeed, **the encryption of a frame ensures its confidentiality, but not its integrity**.

If application data integrity is required, the application will need to implement its own message integrity verification mechanism. This will be the case in particular if this data is used to bill a customer (in the case of a water or gas meter for example).

# 5 About TIDIWI

TIDIWI is a consulting company specializing in the design and development of IOT projects. In particular, TIDIWI has a strong experience in wireless communication protocols and in ultra-low power consumption systems.

TIDIWI is involved in consulting, embedded software development, electronic board development, development of development teams, and project management.

For more information, visit Tidiwi's website: www.tidiwi.com

Or contact thierry.didi[at]tidiwi[dot]com

# 6 Reference Documents

LoraWan Technical Specification V1.0.3: https://lora-alliance.org/wp-content/uploads/2020/11/lorawan1.0.3.pdf

LoraWan Technical Specification V1.0.4: https://resources.lora-alliance.org/technical-specifications/ts001-1-0-4-lorawan-l2-1-0-4-specification

RP2-1.0.3 LoRaWAN® Regional Parameters: https://resources.lora-alliance.org/technical-specifications/rp2-1-0-3-lorawan-regional-parameters

LoRaWAN Security FAQ: https://resources.lora-alliance.org/faq/lorawan-security-faq

Semtech SX1276/77/78/7 Datasheet:
https://semtech.my.salesforce.com/sfc/p/#E0000000JelG/a/2R0000001Rbr/6EfVZUorrpoKFfvaF_Fkpgp5kzjiNyiAbqcpqh9qSjE

Jamming of LoRa PHY and Countermeasure :
https://www4.comp.polyu.edu.hk/~csyqzheng/papers/LoRaJamming-INFOCOM21.pdf