TREND MICRO | research

# The Current State of LoRaWAN Security

Technical Brief

Internet of things (IoT) devices have been adopted across many areas and use cases, from the production lines of manufacturing industries to the operation of smart cities. IoT is expected to have an even more significant impact on daily life in the near future; however, barriers such as increased power consumption prevent many organizations from adopting the technology. A subset of IoT technologies that use Low Power Wide Area Network (LPWAN)[1] was created to resolve the problem of power consumption. Among these LPWAN technologies, we can find cellular IoT systems known as Narrow-Band IoT (NB-IoT), SigFox, and LoRa.

The LoRa (long-range) protocol helps companies connect low-powered IoT devices to the internet using a wireless connection. Long Range Wide Area Network (LoRaWAN) is a radio-based technology that works in conjunction with LoRa. Since they are affordable and convenient to use, many industries have adopted LoRaWAN technology. Specifically, they have been used in soil management, animal monitoring, weather monitoring, water meter reading, environment sensors for dams, asset tracking and fleet management, failure prediction for buildings, and more. These technologies have been introduced as an alternative to short-range systems such as Bluetooth, ZigBee, and Z-Wave. However, questions must be raised when adopting these technologies in sensitive environments: are they secure, and do they present a risk to organizations using them?

This article will discuss LoRa modulation technology and the LoRaWAN protocol by first summarizing previous security research material on the topic. The succeeding sections will cover more solutions and introduce our tools that add to past research. We also provide insight into overcoming the limitations in defining attacks against these technologies and propose solutions that can reinforce LoRaWAN devices against radio and hardware attacks.

# Introduction

LoRa PHY is the physical layer commonly used by the LoRa Wide Area Network (LoRaWAN) Medium Access Control (MAC) stack, as shown in the OSI network model:

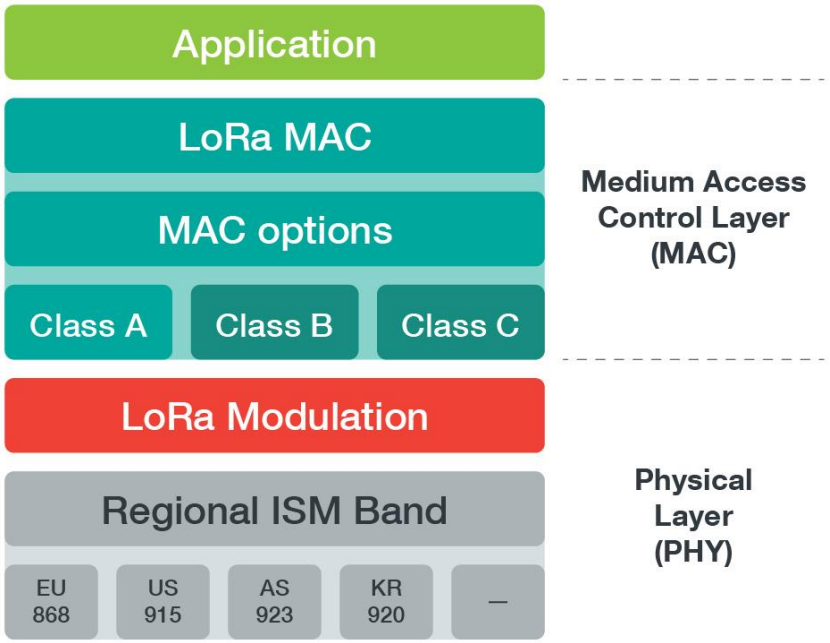


Figure 1. The LoRaWAN (MAC) protocol stack is implemented on top of LoRa modulation (PHY).[2]

There are three dominant LPWAn technologies: LoRaWAN, NB-IoT, and Sifox. Compared to NB-IoT, LoRaWAN is perfect for single-building applications as it can be set up and managed without any telecom operator dependence on a gateway. It also has a longer battery life but a lower data rate and more latency than NB-IoT. LoRaWAN has also started to be more widely deployed in Europe compared to Sigfox, although Sigfox's radio module costs less.

We also noticed other differences between the three technologies:

| | LoRaWAN | Sigfox | NB-IoT |
|---|---|---|---|
| Frequency bands | Unlicensed | Unlicensed | Licensed |
| Range (urban) | 5 km | 10 km | 1 km |
| Range (rural) | 20 km | 40 km | 10 km |
| Maximum data rate | 50 kbit/s | 0.1kbit/s | 200 kbit/s |
| Maximum messages per day | Unlimited | 140 Up, 4 Down | Unlimited |
| Modulation | CSS | BPSK | QPSK |
| Encryption | Yes | Yes | Yes |
| Adaptive Data Rate (ADR) | Yes | No | No |
| Private networks | Yes | No | No |
| Gateways locations determined by | Anyone | Operator | Operator |
| Localisation | RSSI & TDOA | RSSI | No |

Table 1. A Comparison of the different Low-Power Wide Area Network (LPWAN) technologies (fixed version from National Center for Biotechnology Information[3])

Both SigFox and LoRa use ISM bands[4] (i.e., 868 MHz in Europe, 915 MHz in North America, and 433 MHz in Asia). Like SigFox, LoRa also transmits signals using a proprietary spread spectrum technique modulation scheme[4] based on Chirp Spread Spectrum modulation (CSS) to encode information. This modulation technique was created by Semtech, which provided specifications on its website.[5] Invented initially for radar applications in the 1940s and then used by military and secure communications applications, CSS was adopted by the IEEE for long-range and mobility for the Low-Rate Wireless Personal Area Networks (LR-WPANs) standard, 802.15.4.

LoRa end-devices can also use different modulations as supported by the SX12xx series (like the SX1261):

- FSK (Frequency-Shift Keying)
- GFSK (Gaussian Frequency-Shift Keying)
- MSK (Minimum-Shift Keying)
- GMSK (Gaussian Minimum Shift Keying)
- LoRa

It should also be noted that LoRa modulation is used over Frequency Shift Keying (FSK) to send data over long distances because of the LoRa Spread Spectrum's high sensitivity. FSK provides a higher data rate.

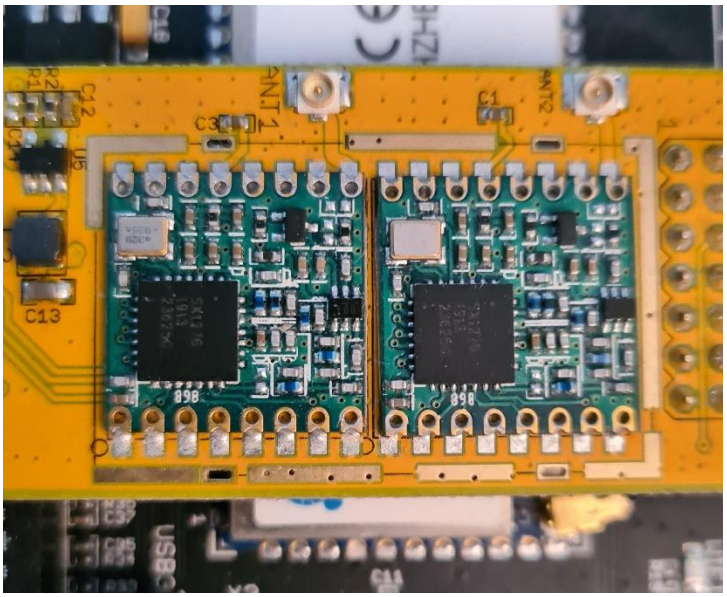


Figure 2. Two SX1276 transceivers used in a Dragino Gateway

# LoRa Modulation

Symbols are transmitted around a sin wave around a defined carrier frequency, noted as carrier frequency (fc), with a linear frequency that varies over time:

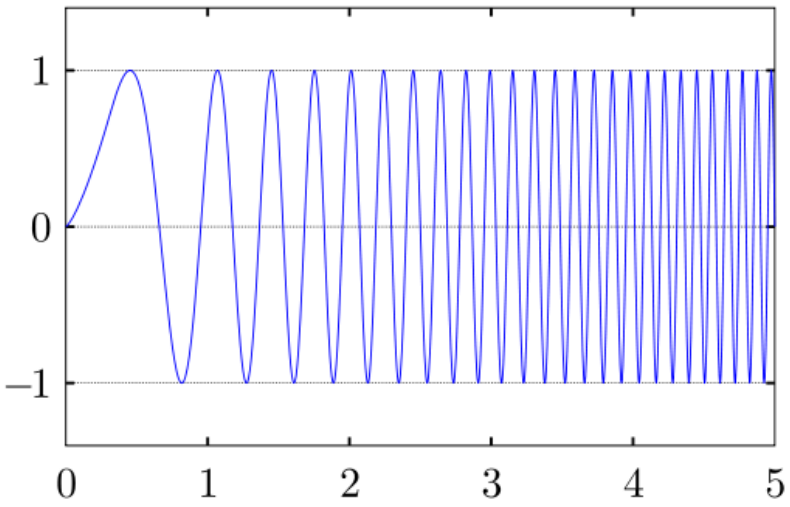


Figure 3. Linear frequency-modulated up-CHIRP[6]

This sin wave is called a CHIRP (Compressed High-Intensity Radar Pulse). A chirp is a signal that varies in frequency and can increase or decrease across the whole bandwidth. The following example shows an up-chirp containing a message, **m(t)**, with eight symbols encoded in 8 bits using a speed of 1 KBaud:

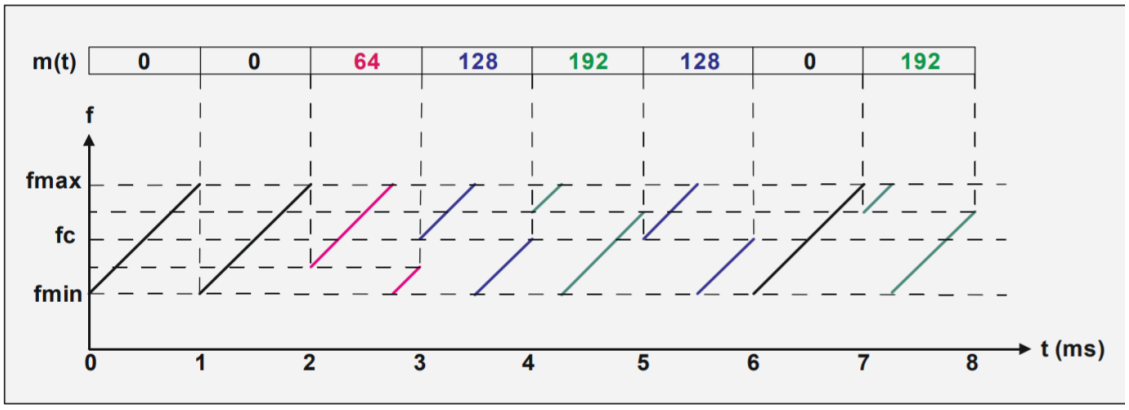


Figure 4. Byte associated to chirp symbols

In the case of down-chirps, the signal frequency decreases over time.

The CSS technique presents many advantages:

- Transmitting signal across vast distances
- Immunity to multipath and fading
- Resistance to doppler shift (moving devices, high clock tolerance)
- Good sensitivity
- Simple to implement
- Low power and adapted for low data rates

Two parameters define the data bit rate:

- Bandwidth
- Spreading factor

The following expression defines the modulation bit rate Rb as:

$$R_b = SF * \frac{1}{\left[\frac{2^{SF}}{BW}\right]} \; bits/sec$$

(where SF = spreading factor (7...12); BW = modulation bandwidth)

LoRa CSS modulation uses three bandwidths, using the 125 kHz bandwidth the most:

- 125 kHz
- 250 kHz
- 500 kHz

LoRa uses six different spreading factors, from SF7 to SF12, as shown in the following graphic:

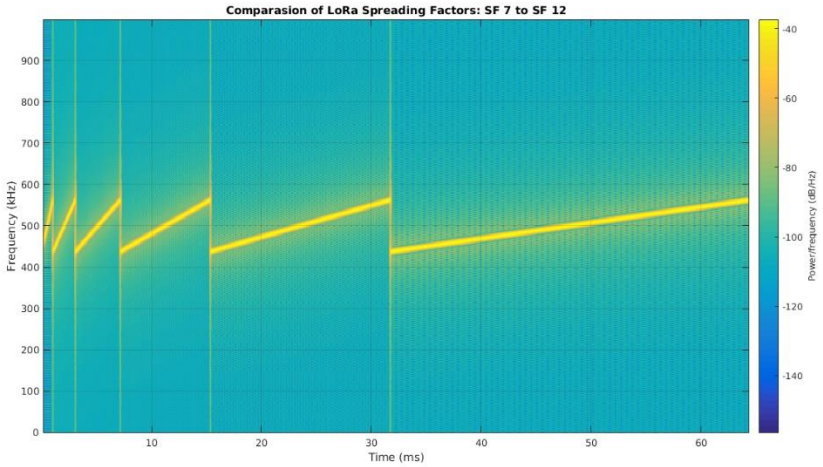


Figure 5. Comparison of LoRa Spreading Factors: SF7 to SF12[7]

This gives us an idea of the different configurations we can find in the EU868, EU433, CN780, and AS923 bands. The data rates[8] are as follows:

| Data Rate | Configuration | Bits/s | Max Payload |
|---|---|---|---|
| DR0 | SF12/125kHz | 250 | 59 |
| DR1 | SF11/125kHz | 440 | 59 |
| DR2 | SF10/125kHz | 980 | 59 |
| DR3 | SF9/125kHz | 1,760 | 123 |
| DR4 | SF8/125kHz | 3,125 | 230 |
| DR5 | SF7/125kHz | 5,470 | 230 |
| DR6 | SF7/250kHz | 11,000 | 230 |
| DR7 | FSK: 50kpbs | 50,000 | 230 |

Table 2. Data rates associated to Spreading Factor and bandwidth configuration

This allows us to conclude that a higher spreading factor equates to a higher over-the-air time, which means the information will take longer to send. On the other hand, a lower spreading factor equates to a higher data rate, which means the information will be sent quicker.

The LoRa physical layer includes eight up-chirps for preamble symbols (in red), two down-chirps synchronization message symbols (in green), a physical payload (in pink), plus an optional CRC depending on message direction:

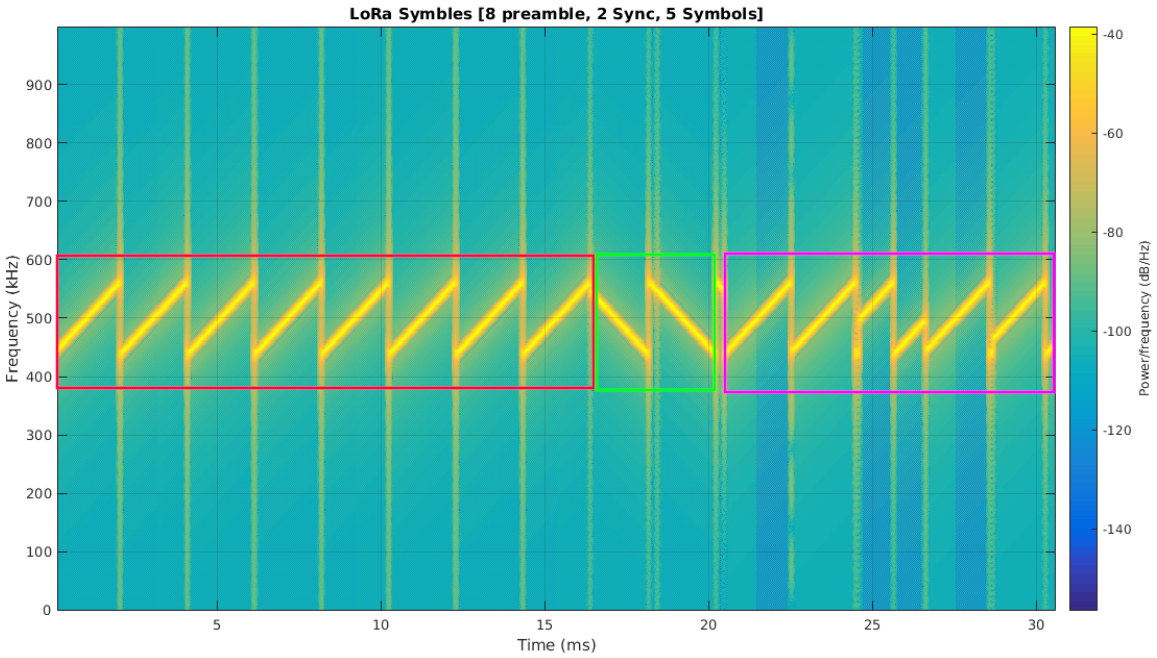


Figure 6. LoRa modulated up-link frame

The demodulation process of a LoRa modulated message can be done by multiplying the received signal **su(t)** with a computed inverse chirp as **ref(t)**, such as the signal **s(t)** before symbol extraction is **s(t) = su(t)*ref(t)**, in order to isolate significant components and obtain the symbols to extract after:

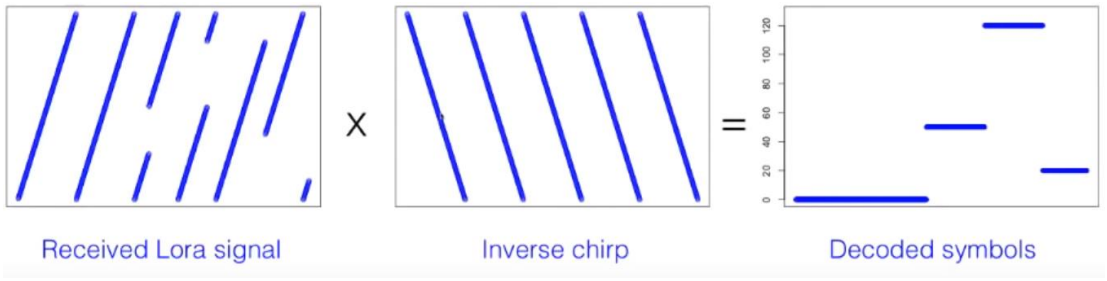


Figure 7. Decode LoRa modulated symbols[9]

To extract these symbols, the recovered **s(t)** needs are passed to a fast Fourier transform (FFT) analysis process after being filtered. Symbols can then be extracted properly, as seen in the following flowgraph:

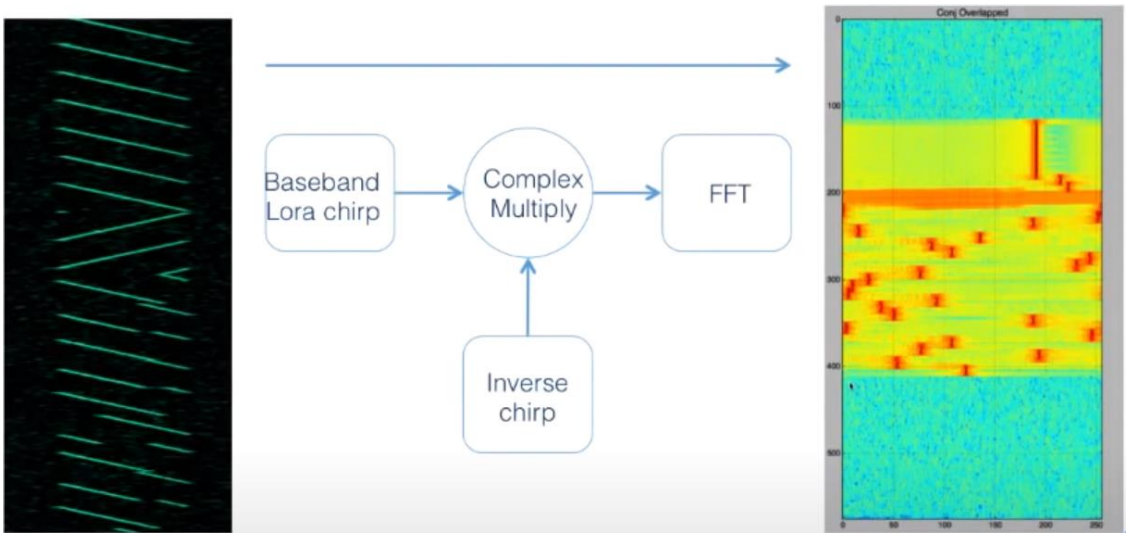


Figure 8. Decoded LoRa modulated IQ symbols[10]

The following diagram details a complete IQCSS transceiver:

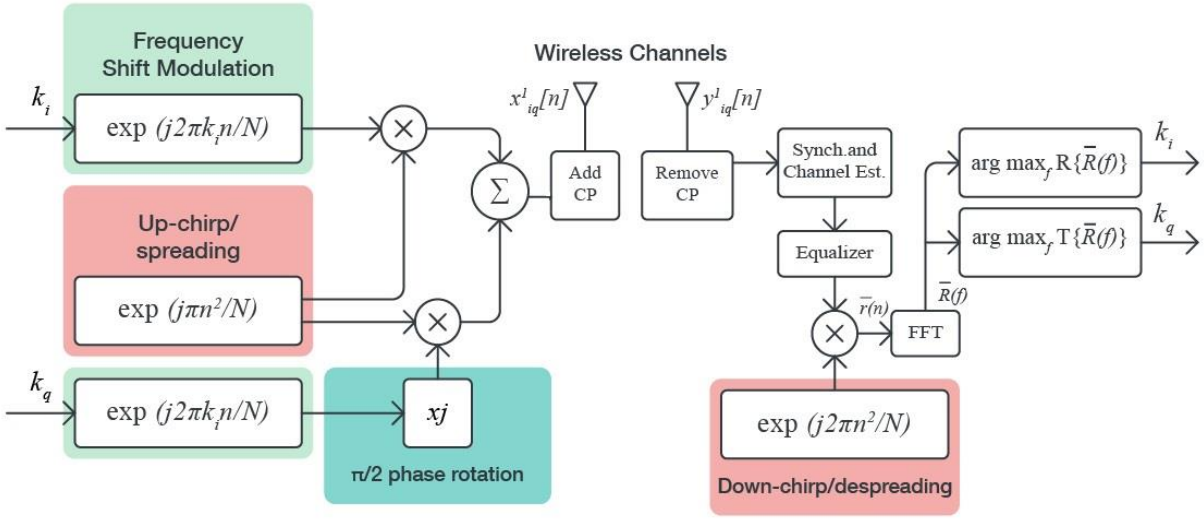


Figure 9. LoRa PHY transceiver block diagram[11]

To send and receive LoRa-modulated signals, Semtech provides cheap transceivers that can be used for uplink and downlink messages and are generally used by endpoint-devices as gateways.

But a few software-defined radio implementations, especially two GNU Radio modules, were also released to work on LoRa:

- *gr-lora* from rpp0 from Pieter Robyns, Eduard Marin, William Thenaers, and Clayton Smith
- *gr-lora* from Matt Knight of Bastille Threat Research Team

Before releasing the *gr-lora* module, Matt Knight made a very instructive presentation on LoRa PHY at the 33rd Chaos Communication Congress (33c3),[14] which should be viewed by anyone interested in LoRa PHY implementation. However, this implementation hasn't been updated in three years and needs to be readapted to the current version of GNU Radio. The next part of this blog series will discuss the use of the actively updated *gr-lora* module from the *rpp0* repository, which we can also directly use to decode uplink and downlink messages.

IoT devices can use LoRa PHY to transport messages, but since they do not provide native security mechanisms, developers need to implement their own security to protect messages from interception, injection, replay attacks, and other malicious activity.

Another layer on top of LoRA PHY, called LoRaWAN, has been created to simplify communications and address these security problems.

# LoRaWAN

LoRaWAN is the cloud-based MAC layer protocol. This layer is used to communicate between LPWAN gateways. The following diagram shows the classic architecture of a LoRaWAN network:

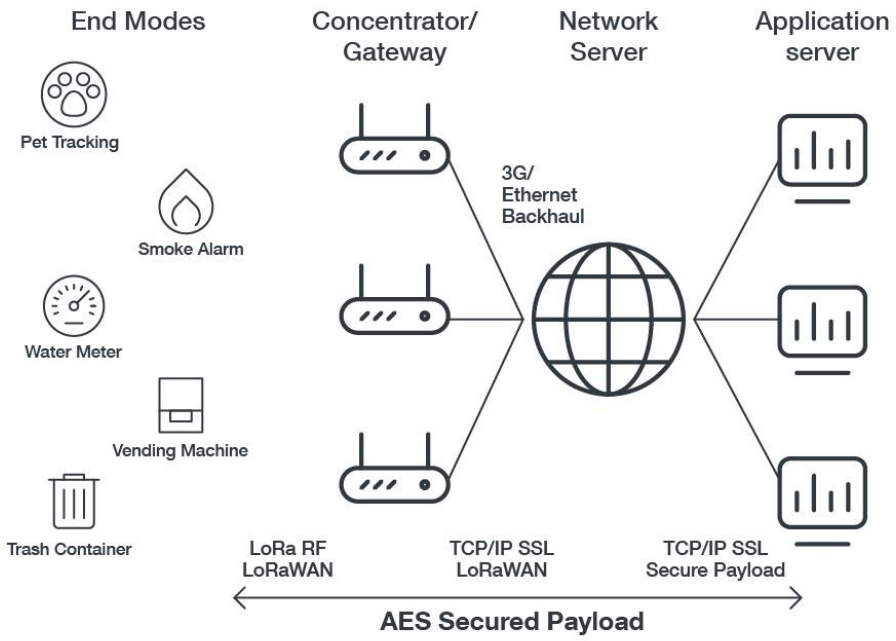


Figure 10. LoRaWAN architecture[12]

We reproduced a real-world environment with two development LoRaWAN kits, a LoRaWAN GPS tracking badge, and a LoRaWAN door sensor connected to a Dragino LG308 Gateway. In the following picture, we can observe the environment connected to one of The Things Network's servers:

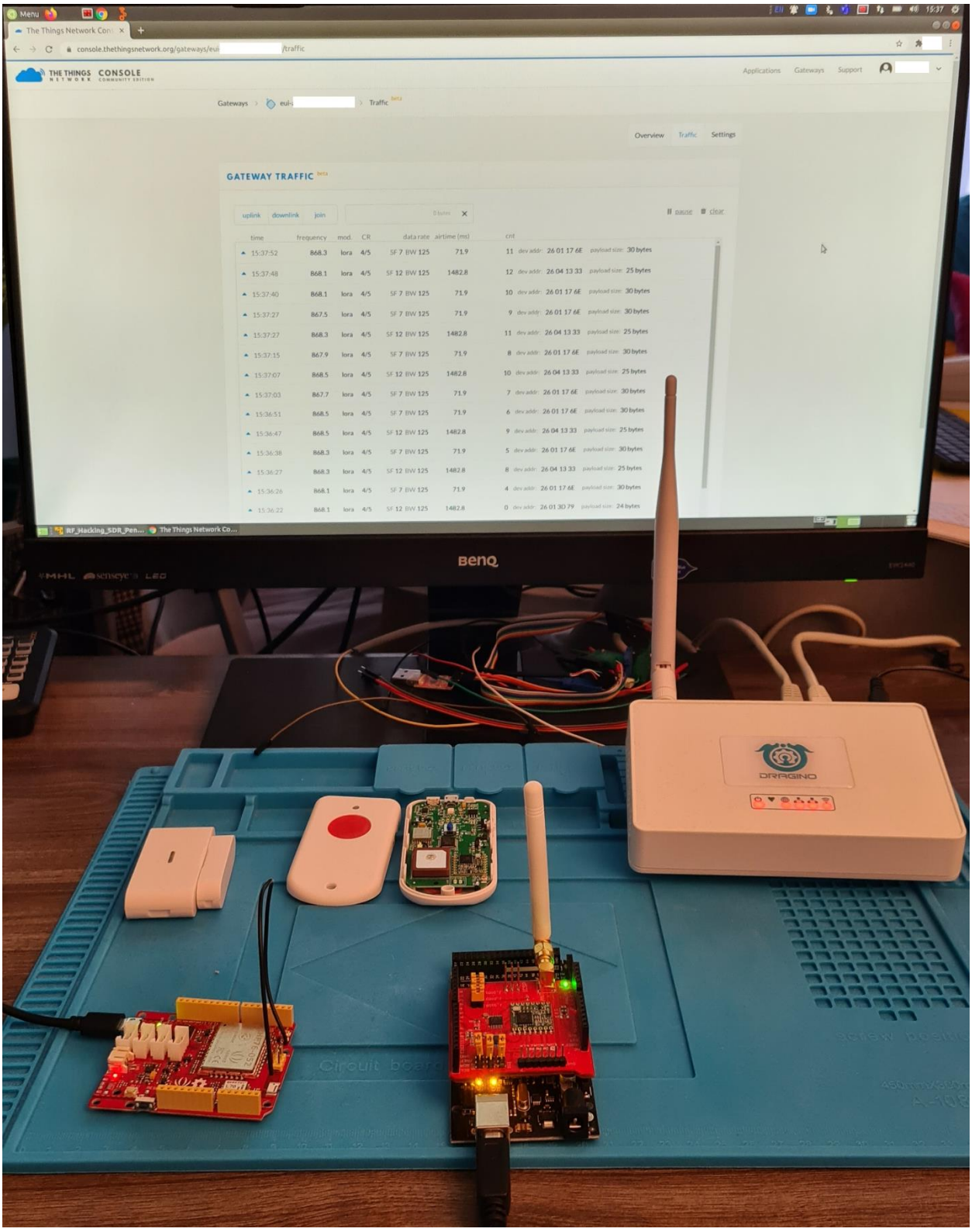


Figure 11. LoRaWAN real-world testbed with LoRaWAN sensors and a gateway

From LoRaWAN 1.1, a separate Join server can also be used to store and generate root keys as well as send them to the application server and the network server.

LoRaWAN is not mandatory when using a LoRa device. It is also possible to use raw MAC communication to directly send commands and messages, like in P2P (Peer-to-Peer), for example. But in that situation, 100% of the security depends on the user, as the raw MAC communication does not provide any encryption and integrity the same way LoRaWAN does.

Regional parameters of LoRaWAN can be found in documentation from the Lora Alliance.[13]

Different server solutions are available to connect gateways:

- The Things Network[14]
- Tencent cloud
- Custom/private solutions
- Network server solution for IoT devices using LoRaWAN and with security in mind, like LORIOT[15]

Different classes exist for LoRaWAN communications:

1. Class A: The uplink transmission of each end-device is followed by two short downlink receive windows.
2. Class B: End-devices of Class B allow for more receive slots — the end-device opens its receiving window at the scheduled time, and it receives a time-synchronized beacon from the gateway.
3. Class C: The window for receiving is always open; however, this results in increased battery consumption.

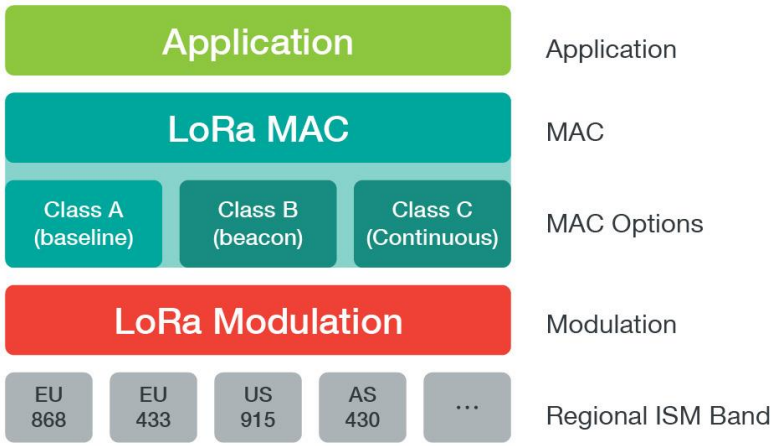


Figure 12. The LoRaWAN stack

Two different versions of LoRaWAN are currently in use: LoRaWan 1.1, and LoRAWAN v1.0.3, which can be found in many cheap devices sold in the market (such as Dragino gateways), including end-devices and shields. Solutions based on Mbed OS version 5.8 (the operating system that runs the LoRaWAN stack on most embedded devices) and more expensive solutions support LoRaWAN version 1.1 and provide even more security features for version 1.0.3.

We will review the security impact and different modes of operation in the following sections.

# LoRaWAN Security

There are two different modes that define/compute keys for MAC frame payload encryption:

- OTAA
- ABP

Regardless of the mode used with LoRaWAN communication, messages are protected by two session keys, **AppSKey** and **NwSKey**, which are used to encrypt messages in Counter with CBC-MAC (CCM) mode, a variation of Counter (CTR) mode[16]:

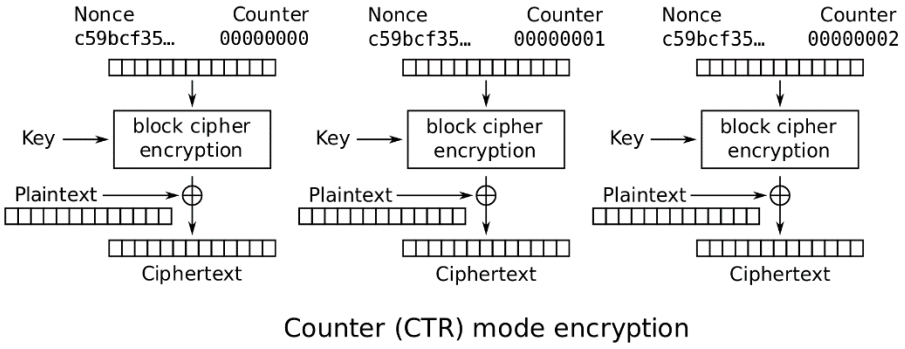


Figure 13. Use of CTR mode encryption within LoRaWAN

CTR mode is used to encrypt the payload using AppSKey and authenticates the message with a Cipher-based Message Authentication Code (CMAC) based on the NwkSkey.

In the different sections, we will review these different modes to understand their impact on security.

## OTAA mode: Over-The-Air Activation

If the end-device supports the *Join* function and can store dynamically generated keys, a join procedure can be performed to compute keys for encrypting and protecting packet integrity.

The process for computing new keys:

1. The end-device sends a Join Request
2. The network will generate keys
3. If the device can Join the network, a *Join Accept* message is sent by the network encrypted with *AppKey,* providing an App Nonce
4. With the given parameters sent via the Join Accept message, the end-device can compute the new encryption and integrity keys

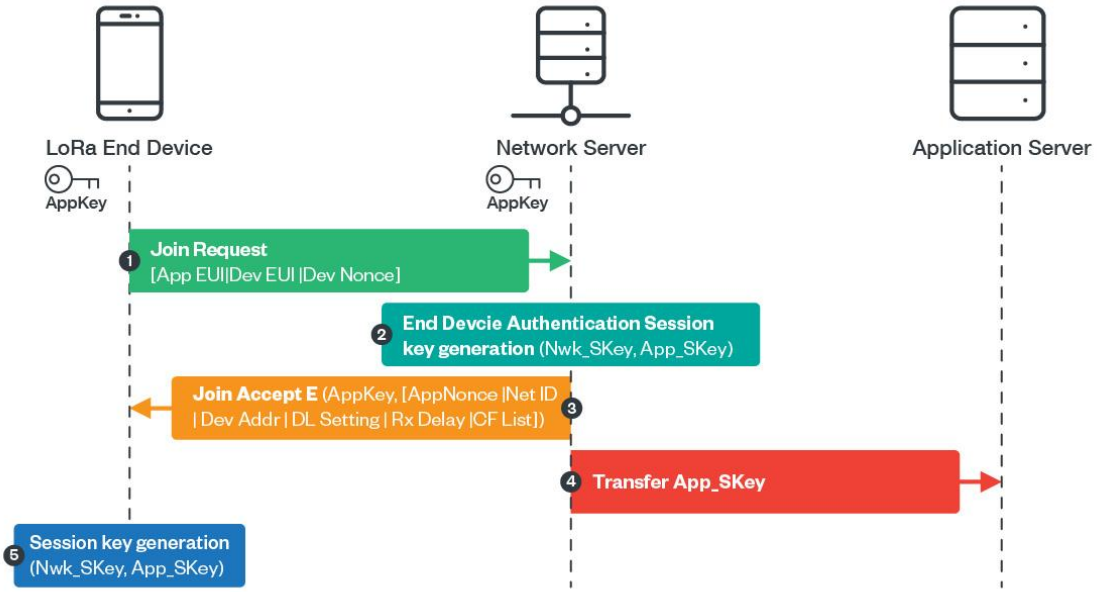


Figure 14. LoRaWAN 1.0 Join procedure in OTAA

Some differences exist between 1.0 and 1.1 in terms of security.

### *LoRaWAN 1.0*

As described in LoRaWAN 1.0.3 specifications,[17] the Join Request is sent in clear-text to the gateway with the following parameters:

- *DevEUI*: unique end-device identifier in IEEE EUI64 address space
- *AppEUI*: the application identifier in IEEE EUI64 address space
- A random *DevNonce* of 2 bytes

| **Size (bytes)** | 8 | 8 | 2 |
|---|---|---|---|
| **Join Request** | AppEUI | DevEUI | DevNonce |

From the specifications, *DevNonce* values are tracked to avoid replay attacks.

The message integrity code (MIC) for this message, which also enables the network to check if the *AppKey* is correct, is computed as follows:

```
cmac = aes128_cmac(AppKey, MHDR | AppEUI | DevEUI | DevNonce)
MIC = cmac[0..3]
```

If the device is allowed to join the network and the MIC is correct, the network sends an encrypted Join Accept message with the following fields:

- *AppNonce* in LoRaWAN 1.0: random value (3 bytes)

- *NetID*, called *Home_NetID* in 1.1: network ID (3 bytes)
- *DevAddr*: Device ID (3 bytes)
- *DL Settings*: downlink parameters
- *RxDelay*: delay between TX and RX (1 byte)
- *CFList*: optional list of channel frequencies (16 bytes)

| **Size (bytes)** | 3 | 3 | 4 | 1 | 1 | (16) Optional |
|---|---|---|---|---|---|---|
| **Join Accept** | AppNonce | NetID | DevAddr | DLSettings | RxDelay | CFList |

The *Join-accept* payload is encrypted as follows:

```
aes128_decrypt(AppKey, AppNonce | NetID | DevAddr | DLSettings | RxDelay | CFList | MIC)
```

The sessions keys are computed in the network and end-device sides as follows:

```
NwkSKey = aes128_encrypt(AppKey, 0x01 | AppNonce | NetID | DevNonce | pad16)
AppSKey = aes128_encrypt(AppKey, 0x02 | AppNonce | NetID | DevNonce | pad16)
```

All the communications will then be encrypted using *AppSKey,* and the integrity protected with *NwkSKey*.

Nevertheless, only one *AppKey* key is used to compute the MIC as well as AppSKey and NwkSKey, which leaves a bigger opening for an attacker to crack either the MIC of the Join procedure's message or the Join-accept message to retrieve this key. Retrieving the key could allow an attacker to eavesdrop on the messages between an end-device and a gateway.

### *LoRaWAN 1.1*

The 1.1 version introduces more key diversification to make key-cracking attacks more complex on LoRaWAN. Everything is documented in the LoRaWAN 1.1 specifications.[18]

In this version, LoRaWAN defines two root keys, *NwkKey* and *AppKey*, that are AES-128 keys specific to the end-device and assigned during the fabrication.

For the *Join-request* message, values transmitted by the end-device are generally the same, but they are renamed:

| **Size (bytes)** | 8 | 8 | 2 |
|---|---|---|---|
| **Join-request** | JoinEUI | DevEUI | DevNonce |

According to the specifications, *DevNonce* is a counter that starts at zero when the device is initially powered up and is incrementally increased by one with each *Join-request*. This *DevNonce* value will never be reused for a given *JoinEUI* value, but since some end-devices cannot store this counter in non-volatile memory, they may discard the *Join-request* of the device to the server. In that case, the *JoinEU* (AppEUI renamed in v1.1) must reset as well.

To protect the message's integrity, the MIC is computed using a dedicated *NwkKey*:

```
cmac = aes128_cmac(NwkKey, MHDR | JoinEUI | DevEUI | DevNonce)MIC = cmac[0..3]
```

If the device is allowed to join the network, a Join-accept message is sent to the device with the following encrypted parameters:

- *JoinNonce* in LoRaWAN 1.0: random value (3 bytes)
- *NetID*, called *Home_NetID* in 1.1: network ID (3 bytes)
- *DevAddr*: Device ID (3 bytes)

And this Join-Accept message is encrypted as follows using the *NwkKey*:

```
aes128_decrypt(NwkKey, JoinNonce | NetID | DevAddr | DLSettings | RxDelay | CFList | MIC)
```

- *DL Settings*: downlink parameters
- *RxDelay*: delay between TX and RX (1 byte)
- *CFList*: optional list of channel frequencies (16 bytes)

| **Size (bytes)** | 3 | 3 | 4 | 1 | 1 | (16) Optional |
|---|---|---|---|---|---|---|
| **Join Accept** | AppNonce | NetID | DevAddr | DLSettings | RxDelay | CFList |

The key used to encrypt the *Join-Accept* message depends on the *Join* or *ReJoin Request* message that triggers it.

| Triggering Join-request or Rejoin-request type | Join-accept Encryption Key |
|---|---|
| Join-request | **NwkKey** |
| Rejoin-request type 0 or 1 or 2 | **JSEncKey** |

The Join-Accept message is encrypted as follows:

```
aes128_decrypt(NwkKey or JSEncKey, JoinNonce | NetID | DevAddr | DLSettings | RxDelay |
CFList | MIC)
```

Like in the 1.0 version, an exception exists by using the *NwkKey* only instead of the *AppKey* to encrypt this message. Indeed, in the downlink configuration field *DLsettings*, 7-bit subfields are indicated if the Network Server implements the LoRaWAN 1.0 (unset) or 1.1 (set) protocol. Referring to the specifications, when this *OptNeg* bit is set:

- The protocol version is further (1.1 or later) negotiated between the end-device and the Network Server through the *RekeyInd/RekeyConf* MAC command exchange
- The device derives *FNwkSIntKey & SNwkSIntKey & NwkSEncKey* from the *NwkKey*
- The device derives *AppSKey* from the *AppKey*

| **Bits** | 7 | 6:4 | 3:0 |
|---|---|---|---|
| **DLsettings** | OptNeg | RX1DRoffset | RX2 Data rate |

The session keys are derived with the *NwkKey* as follows:

```
FNwkSIntKey = aes128_encrypt(NwkKey, 0x01 | JoinNonce | JoinEUI | DevNonce | pad16 )
SNwkSIntKey = aes128_encrypt(NwkKey, 0x03 | JoinNonce | JoinEUI | DevNonce | pad16)
NwkSEncKey = aes128_encrypt(NwkKey, 0x04 | JoinNonce | JoinEUI | DevNonce | pad16)
FNwkSIntKey = aes128_encrypt(NwkKey, 0x01 | JoinNonce | JoinEUI | DevNonce | pad16 )
SNwkSIntKey = aes128_encrypt(NwkKey, 0x03 | JoinNonce | JoinEUI | DevNonce | pad16)
```

And the *AppSKey* with the *AppKey*:

```
AppSKey = aes128_encrypt(AppKey, 0x02 | JoinNonce | JoinEUI | DevNonce | pad16)
```

The MIC is computed as follows:

```
cmac = aes128_cmac(JSIntKey,JoinReqType | JoinEUI | DevNonce | MHDR | JoinNonce | NetID
| DevAddr | DLSettings | RxDelay | CFList )

MIC = cmac[0..3]
```

Note that *JSIntKey* is used to get MIC Rejoin-Request type 1 messages and Join-Accept answers:

```
JSIntKey = aes128_encrypt(NwkKey, 0x06 | DevEUI | pad16)
```

But when this *OptNeg* bit is not set:

- The device reverts to LoRaWAN1.0, no options can be negotiated
- The device does not send the *RekeyInd* command
- The device derives *FNwkSIntKey* & *AppSKey* from the *NwkKey*
- The device sets *SNwkSIntKey* & *NwkSEncKey* equal to *FNwkSIntKey*

*AppSKey* and Session keys are then computed as follows with *NwkKey*:

```
AppSKey = aes128_encrypt(NwkKey, 0x02 | JoinNonce | NetID | DevNonce | pad16)
FNwkSIntKey = aes128_encrypt(NwkKey, 0x01 | JoinNonce | NetID | DevNonce | pad16)
SNwkSIntKey = NwkSEncKey = FNwkSIntKey.
```

And the MIC is computed as follows:

```
cmac = aes128_cmac(NwkKey, MHDR | JoinNonce | NetID | DevAddr | DLSettings | RxDelay |
CFList )
MIC = cmac[0..3]
```

# ABP: Activation by Personalization

The ABP method is simpler than OTAA as there is no Join Procedure. Nevertheless, it has downsides in terms of security, as session keys are hardcoded on versions 1.0 and 1.1.

Indeed, session keys stay the same until the user manually changes them or when a firmware update/upgrade is applied. Because of this, ABP is more vulnerable to a cryptanalysis attack compared to OTAA.

## *MAC Frame Payload Encryption (FRMPayload)*

With given session keys, we can protect MAC frame payloads on Data Up and Down messages.

**LoRaWAN 1.0**

First, the key "K" is chosen depending on the *FPort* of the data message:

| FPort | K |
|---|---|
| 0 | NwkSKey |
| 1..255 | AppSKey |

For the encryption, the message is split into a sequence of blocks: "$A_i$" for *i = 1..k with k =ceil(len(pld) / 16)* (with pld = *FRMPayload*).

| **Size (bytes)** | 1 | 4 | 1 | 4 | 4 | 1 | 1 |
|---|---|---|---|---|---|---|---|
| ***Ai*** | 0x01 | 4 x 0x00 | Dir | DevAddr | FCntUp or FCntDown | 0x00 | *i* |

Note that the Direction (Dir) is defined by "1" for a downlink and "0" for an uplink.

So, the block Ai is encrypted as follows:

```
Si = aes128_encrypt(K, Ai) for i = 1..k
S = S1 | S2 | .. | Sk
```

Encryption and decryption of the blocks are then performed as follows:

```
(pld | pad16) xor S
```

The MIC is computed as follows:

```
msg = MHDR | FHDR | FPort | FRMPayload
cmac = aes128_cmac(NwkSKey, B0 | msg)
MIC = cmac[0..3]
```

Where B0 is defined with the following parameters:

| **Size (bytes)** | 1 | 4 | 1 | 4 | 4 | 1 | 1 |
|---|---|---|---|---|---|---|---|
| $B_0$ | 0x49 | 4 x 0x00 | Dir | DevAddr | FCntUp or FCntDown | 0x00 | len(*msg*) |

## *LoRaWAN 1.1*

The key “K” used depends on the FPort as for LoraWAN 1.0 version:

| | | |
|---|---|---|
| 0 | Uplink/downlink | NwkSEncKey |
| 1..255 | Uplink/downlink | AppSKey |

For each data message, the algorithm defines a sequence of Blocks $A_i$ for *i = 1..k with k =ceil(len(pld) / 16)*:

| **Size (bytes)** | 1 | 4 | 1 | 4 | 4 | 1 | 1 |
|---|---|---|---|---|---|---|---|
| $A_i$ | 0x01 | 4 x 0x00 | Dir | DevAddr | FCntUp or NFCntDwn or AFCntDnw | 0x00 | *i* |

Blocks are computed and encrypted/decrypted the same way as for LoRaWAN 1.0:

```
Si = aes128_encrypt(K, Ai) for i = 1..k
S = S1 | S2 | .. | Sk
Encrypted/decrypt = (pld | pad16) xor S
```

For computing the MIC, there are differences between Uplink frames and Downlink Frame.

**Downlink Frames**

For Downlink frames, the MIC is computed with the *SNwkSIntKey* as follows:

```
cmac = aes128_cmac(SNwkSIntKey, B0 | msg)
MIC = cmac[0..3]
```

And the $B_0$ blocks is defined differently by introducing a *ConfFCnt* field:

| Size (bytes) | 1 | 2 | 2 | 1 | 4 | 4 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|
| $B_0$ | 0x49 | ConfFCnt | 2 x 0x00 | Dir = 0x01 | DevAddr | AFCntDwn or NFCntDwn | 0x00 | len(*msg*) |

In all cases, this *ConfFCnt = 0x0000*, but if the device is connected to a LoRaWAN 1.1 network and the ACK bit of DL is set (acknowledging an uplink "confirmed" frame), then the *ConfFCnt* is a frame counter with a value of "confirmed" uplink modulo 2^16.

#### Uplink frames

If the device is connected to a LoRaWAN 1.0 Network Server, the MIC uses $B_0$ block defined as follows:

| Size (bytes) | 1 | 4 | 1 | 4 | 4 | 1 | 1 |
|---|---|---|---|---|---|---|---|
| $B_0$ | 0x49 | 0x0000 | Dir = 0x00 | DevAddr | FCntUp | 0x00 | len(*msg*) |

And so the MIC is computed as follows:

```
cmacF = aes128_cmac(FNwkSIntKey, B0 | msg)
MIC = cmacF[0..3]
```

If the device is connected to a 1.1 Network Server, the MIC then uses $B_0$, but also a $B_1$ block defined as follows:

| Size (bytes) | 1 | 2 | 1 | 1 | 1 | 4 | 4 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|
| $B_1$ | 0x49 | ConfFCnt | TxDr | TxCh | Dir = 0x00 | DevAddr | FCntUp | 0x00 | len(*msg*) |

And the MIC is computed as follows:

```
cmacS = aes128_cmac(SNwkSIntKey, B1 | msg)
MIC = cmacS[0..1] | cmacF[0..1]
```

# Attacks against LoRaWAN

Most of the attacks documented below have been performed on LoRaWAN class A devices. However, a class B attack is also possible; these are usually done to drain the end-device's battery.

### DoS in ABP mode

Given that the counter in *FRMPayload* is only 16 bits long In LoRa 1.0, a denial-of-service (DoS) attack is actually possible. Xueying Yang, Evgenios Karampatzakis, Christian Doerr, and Fernando Kuiper (a team of security researchers based in the Netherlands)[19] demonstrated this. They posited that a malicious actor could replay a captured packet, wait until the counter overflows, and replay this packet to realize a DoS attack:

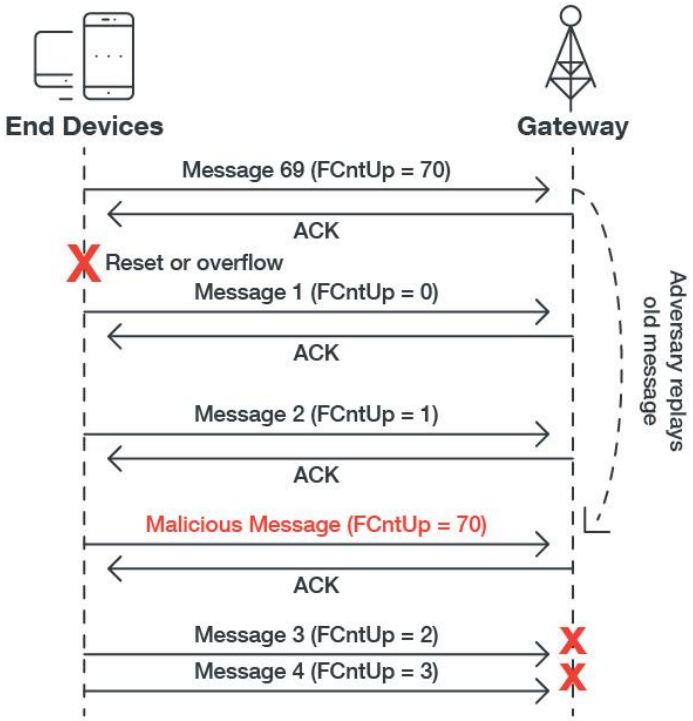


Figure 15. An example of a replay attack for ABP[20]

Session keys stay the same until manually changed or with a firmware update/upgrade. Because of this, ABP could be more vulnerable to a cryptanalysis attack than OTAA.

## Eavesdropping

In their paper,[21] Xueying Yang, Evgenios Karampatzakis, Christian Doerr, and Fernando Kuiper also highlighted that the key could be retrieved if the counter is reset, as it is not used securely in LoRaWAN 1.0. Indeed, when the counter is reset, the keystream will be reused and could allow an eavesdropping attack. Another factor that leaves it vulnerable is the use of CTR to encrypt information.

The same researchers also highlighted a bit-flipping issue in v1.0 that is backward compatible with v1.1.

## Bit-Flipping

Regardless of the version, the integrity of LoRaWAN messages is protected thanks to the MIC and encryption. Nevertheless, the messages decrypted by the network server and sent to the application are no longer protected:

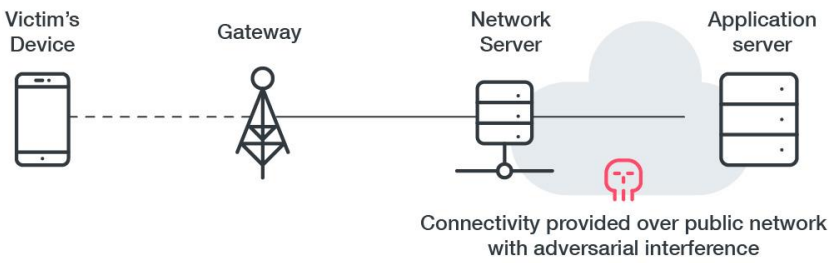


Figure 16. The setup of a bit-flipping attack

To protect messages from the network server to the application server, it would require designing the network to use an SSL tunnel, preferably with a client SSL certificate in order to protect the communication or by using another MIC field inside the MAC Layer Payload computed by the *AppSKey*.

## Ack Spoofing

An acknowledgment mechanism was introduced in LoRaWAN to maximize battery life by reducing the time the radio needs to be powered up. But as highlighted by Xueying Yang, Evgenios Karampatzakis, Christian Doerr, and Fernando Kuiper once again in their paper,[22] the ACK message does not state which message it is confirming.

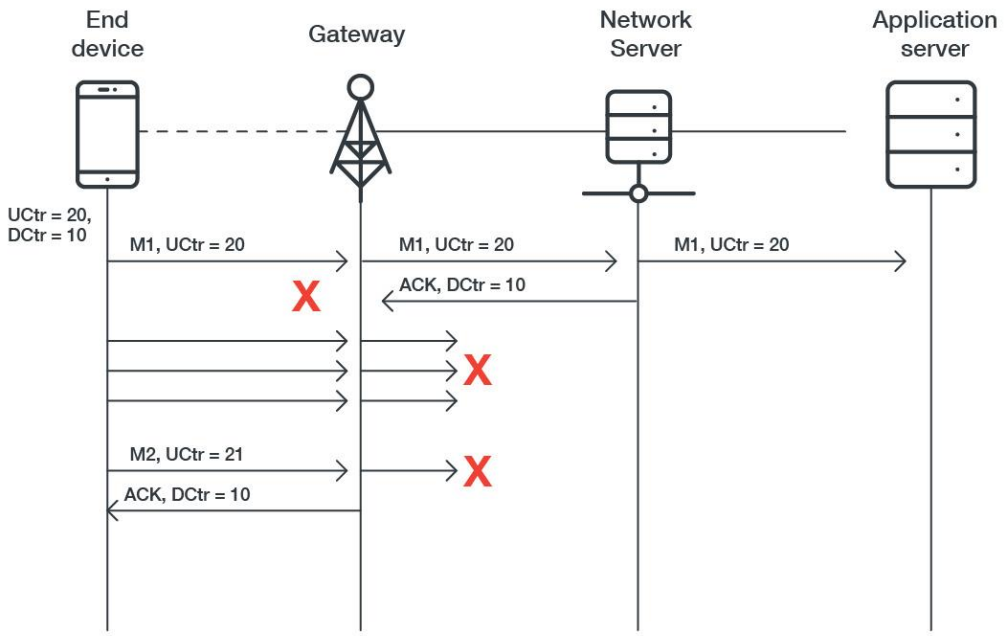


Figure 17. ACK messages may be repurposed to acknowledge frames other than the ones the application provider originally received

To illustrate this issue, researchers proposed to selectively jam the downlink when an end-device sends a confirmed message to the gateway that will confirm the reception. The confirmation message will never get to the end-device, and the confirmed message will be retransmitted seven times. Then, the message will be considered lost or refused, and the status “mac_err” will be raised.

But during the jamming session, the attacker can capture the downlink message for the confirmation and can play the confirmation for the first message when the end-device sends a second confirmation message.

## LoRa Class B Attacks

Most setups use the class A mode, which specifies that downlink traffic must/can follow an uplink one. Class B tends to reduce the amount of spent energy by telling end-devices to wake up periodically to wait for any incoming messages during a receiving window. The durations of the receiving windows are specified by beacon broadcast messages that are comprised of a PHY layer header followed by a beacon payload:

| **Size (bytes)** | 2/3 | 4 | 2 | 7 | 0/1 | 2 |
|---|---|---|---|---|---|---|
| **BCNPayload** | RFU | Time | CRC | GwSpecific | RFU | CRC |

Table 3. Beacon frame content for the EU 863- 870MHz ISM band[23, 24]

An attacker can easily compute the different publicly known fields with malicious parameters[25] to:

- Find the location of a LoRa gateway thanks to a *GwSpecific* field with an *InfoDesc* subfield containing GPS coordinates
- Drain the battery by sending crafted beacons framed with an extreme wakeup time value

According to their paper[26] (as well as another LoRaWAN security evaluation paper by Frank Hessel, Lars Almon, Flor Álvarez[27]) and LoRaWAN v1.1 specifications,[28] the integrity of beacon frames is still an issue. It would be better to use a MIC instead of a CRC to check the time's field integrity.

### Root Key Management

On top of all the security mechanisms we previously discussed is the management of keys. Indeed, the backend is exposed to the internet, which leaves it open to attacks (LFI, SQL injection, deserialization vulnerability, etc.). A malicious actor would be able to get the secret key, read the data, craft downlink packets, and more. All risks related to key management procedures are enumerated in a complete paper by F-Secure Labs (ex-MWR Labs).

Here are some security points to check in a LoRaWAN setup:

- Use randomly generated keys
- Avoid the exposition of key management servers and services (exposed key management service accessible on the internet)
- Preferably use HSM (Hardware Security Module) to keep the keys
- Preferably use OTAA mode and LoRa version 1.1.

# Further Reading

An interesting paper on the MobiSPC 2018[29] highlights security improvements brought by LoRaWAN v1.1. To understand the differences between LoRaWAN v1.0 and v1.1, we also suggest watching Renaud Lifchitz's presentation at The Things Conference 2019.[30] During the presentation, Renaud stated that there will be some vulnerabilities present even if replay attacks are used to transmit an earlier captured packet; he also outlined how LoRaWAN v1.1 has better authentication, confidentiality, and integrity. Indeed, message padding enforcement is not present to prevent default ECB mode encryption risks such as information leaks of the plaintext (length, prefixes, common substrings, etc.). Moreover, backend networks do not have a defined secure standard to protect messages between the network server and other servers because it is sent in plaintext in UDP by default, rather than using MQTT within a TLS session.

# The State of Security

When looking at LoRaWAN, we can see that very few security tools exist for the technologies released starting this year. Some researchers from IO Activ released a framework called LAF[31] (LoRaWAN Auditing Framework). They also provided a tool that can parse, send, craft, analyze, audit a setup, and crack some LoRaWAN packets using weak/default keys. In their paper, the researchers presented LoRaWAN 1.0.3 attacks mainly using their tool.

Nevertheless, this framework still has some limitations we can try to overcome:

- It only works with a Gateway
- It can only listen to uplink packets
- It can only listen to eight out of 64 channels
- Generation and fuzzing depends on LoRaWAN (Go) using an inflexible format such as JSON

Over the same period, another type of tool called “LoRa Craft”[32] was released to intercept packets using Software Defined-Radio and craft packets using dedicated LoRaWAN v1.0 and v1.1 Scapy layers developed for this tool. But this tool is mainly a do-it-yourself (DIY) tool and needs much more assistance than those already released, like the crypto-helpers for Join-Accept payloads and MIC to help crack weak keys.

Later in May, another framework called ChirpOTLE[33] was released by the SEEMOO Lab. This demonstrated two attacks affecting the availability of LoRaWAN networks, like time drifting in LoRa class B and a novel ADR spoofing attack to manipulate frame metadata. But still, the researchers’ setup was limited since they only chose a few default channels to demonstrate their attack on each node.

# Conclusion: Let’s Go Further!

In this first technical brief, we introduced LoRa and LoRaWAN technologies as well as security improvements from version 1.0 to 1.1. We also outlined research and practical tools to help against attacks on LoRaWAN networks and noted the limitations encountered by security researchers.

LoRaWAN devices may be low-powered, but they are used in critical areas and should be secured as comprehensively as possible. For example, in 2016, a large-scale LoRaWAN network was deployed in Taiwan,[34] and cities in the country have adopted many LoRaWAN devices over the years. They are used to monitor water levels in rivers, dam management, waste disposal, and more. Attacks against sensors used in smart cities could potentially affect the safety of a sizeable population. In terms of the private sector, organizations could also be fed falsified information on monitoring devices, which could impact their bottom line. These are only some of the issues that need to be resolved.

The next part of the series will explain how we managed to overcome security research limitations by going in another direction. We look into the use of Software-Defined Radio (SDR) and optimization when attacking both uplink and downlink on multiple channels, as well as multiple spreading factors using a very cheap SDR device. We will also introduce tools that would help analyze LoRa PHY as well as LoRaWAN radio communications. As discussed earlier, the final part of the series will be on hardware attacks and security mechanisms concerning LoRa end-devices.

# References


[1] TechTarget. (n.d.). *IoT Agenda.* “LPWAN (low-power wide area network).” Accessed on Jan. 11, 2020 at https://internetofthingsagenda.techtarget.com/definition/LPWAN-low-power-wide-area-network.

[2] Dong-Hoon Kim,Eun-Kyu Lee, and Jibum Kim. (March 30, 2019). *MDPI*. “Experiencing LoRa Network Establishment on a Smart Energy Campus Testbed”. Accessed on Jan. 11, 2020 at https://www.mdpi.com/2071-1050/11/7/1917.

[3] Philip J. Basford et al. (Jan. 20, 2020). *The National Center for Biotechnology Information*. “LoRaWAN for Smart City IoT Deployments: A Long-Term Evaluation.” Accessed on Jan. 11, 2020 at https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7038353/.

[4] Peter McNeil. (March 22, 2018). *Pasternack*. “What are the ISM Bands, and What Are They Used For?” Accessed on Jan. 11, 2020 at https://blog.pasternack.com/uncategorized/what-are-the-ism-bands-and-what-are-they-used-for/.

[5] Semtech. (May 2, 2015). *Semtech*. “AN1200.22 LoRa™ Modulation Basics.” Accessed on Jan. 11, 2020 at http://wiki.lahoud.fr/lib/exe/fetch.php?media=an1200.22.pdf.

[6] IoTOne. (n.d.). *IoTOne*. “Chirp Spread Spectrum.” Accessed on Jan. 11, 2020 at https://www.iotone.com/term/chirp-spread-spectrum/t110.

[7]Sakshama Ghoslya. (n.d.). *Sghoslya*. “LoRa: Symbol Generation.” Accessed on Jan. 11, 2020 at https://www.sghoslya.com/p/lora-is-chirp-spread-spectrum.html.

[8]The Things Network. (n.d.). *The Things Network.* “Modulation and Data Rate.” Accessed on Jan. 11, 2020 at https://www.thethingsnetwork.org/docs/lorawan/modulation-data-rate.html.

[9]Thomas Telkamp. (Nov. 10, 2016). *The Things Network*. “LoRa Crash Course.” Accessed on Jan. 11, 2020 at https://www.youtube.com/watch?v=T3dGLqZrjIQ.

[10]Thomas Telkamp. (Nov. 10, 2016). *The Things Network*. “LoRa Crash Course.” Accessed on Jan. 11, 2020 at https://www.youtube.com/watch?v=T3dGLqZrjIQ.

[11] Ivo Bizon Franco de Almeida et al. (Sept. 22, 2020). *Arvix.* “In-phase and Quadrature Chirp Spread Spectrum for IoT Communications In-phase and Quadrature Chirp Spread Spectrum for IoT Communications.” Accessed on Jan. 11, 2020 at https://arxiv.org/pdf/2009.10421.pdf.

[12]The Things Network. (n.d.). *The Things NetworkI.* “LoRaWAN Architecture.” Accessed on Jan. 11, 2020 at https://www.thethingsnetwork.org/docs/lorawan/architecture.html.

[13] LoRa Alliance. (Feb. 20, 2020). *LoRa Alliance.* “RP002-1.0.1 LoRaWAN® Regional 40 Parameters.” Accessed on Jan. 11, 2020 at https://lora-alliance.org/sites/default/files/2020-06/rp_2-1.0.1.pdf.

[14] The Things Network. (n.d.). *The Things Network.* “The Things Network.” Accessed on Jan. 11, 2020 at https://www.thethingsnetwork.org/.

[15]Loriot. (n.d.). *Loriot.* “Loriot.” Accessed on Jan. 11, 2020 at https://www.loriot.io/.

[16]Matt Knight. (Dec. 29, 2016). *RC3 33c3*. “Decoding the LoRa PHY.” Accessed on Jan. 11, 2020 at https://www.youtube.com/watch?v=NoquBA7IMNc.

[17]LoRa Alliance. (2018). *LoRa Alliance.* “LoRaWAN 1.0.3 Specification.” Accessed on Jan. 11, 2020 at https://lora-alliance.org/sites/default/files/2018-07/lorawan1.0.3.pdf.

[18]LoRa Alliance. (Oct. 11, 2017). *LoRa Alliance*. “LoRaWAN™ 1.1 Specification.” Accessed on Jan. 11, 2020 at https://lora-alliance.org/sites/default/files/2018-04/lorawantm_specification_-v1.1.pdf.

[19]Xueying Yang et al. (April 17, 2020). *IEEE Xplore.* "Security Vulnerabilities in LoRaWAN." Accessed on Jan. 11, 2020 at https://ieeexplore.ieee.org/document/8366983/footnotes#footnotes.

[20]Xueying Yang et al. (April 17, 2020). *IEEE Xplore.* "Security Vulnerabilities in LoRaWAN." Accessed on Jan. 11, 2020 at https://ieeexplore.ieee.org/document/8366983/footnotes#footnotes.

[21]Xueying Yang et al. (April 17, 2020). *IEEE Xplore.* "Security Vulnerabilities in LoRaWAN." Accessed on Jan. 11, 2020 at https://ieeexplore.ieee.org/document/8366983/footnotes#footnotes.

[22]Xueying Yang et al. (April 17, 2020). *IEEE Xplore.* "Security Vulnerabilities in LoRaWAN." Accessed on Jan. 11, 2020 at https://ieeexplore.ieee.org/document/8366983/footnotes#footnotes.

[23] LoRa Alliance. (2018). *LoRa Alliance.* "LoRaWAN 1.0.3 Specification." Accessed on Jan. 11, 2020 at https://lora-alliance.org/sites/default/files/2018-07/lorawan1.0.3.pdf.

[24] LoRa Alliance. (2018). *LoRa Alliance.* "LoRaWAN 1.1 Specification." Accessed on Jan. 11, 2020 at https://lora-alliance.org/sites/default/files/2018-04/lorawantm_specification_-v1.1.pdf.

[25]Xueying Yang et al. (April 17, 2020). *IEEE Xplore.* "Security Vulnerabilities in LoRaWAN." Accessed on Jan. 11, 2020 at https://ieeexplore.ieee.org/document/8366983/footnotes#footnotes.

[26]Xueying Yang et al. (April 17, 2020). *IEEE Xplore.* "Security Vulnerabilities in LoRaWAN." Accessed on Jan. 11, 2020 at https://ieeexplore.ieee.org/document/8366983/footnotes#footnotes.

[27]Frank Hessel, Lars Almon, and Flor Álvarez. (May 23, 2020). *Arvix*. "ChirpOTLE: A Framework for Practical LoRaWAN Security Evaluation." Accessed on Jan. 11, 2020 at https://arxiv.org/pdf/2005.11555.pdf.

[28]LoRa Alliance. (2018). *LoRa Alliance.* "LoRaWAN 1.1 Specification." Accessed on Jan. 11, 2020 at https://lora-alliance.org/sites/default/files/2018-04/lorawantm_specification_-v1.1.pdf.

[29] Tahsin C.M. Dönmez and Ethiopia Nigussie. (Jan. 2018). *ScienceDirect*. "Security of LoRaWAN v1.1 in Backward Compatibility Scenarios." Accessed on Jan. 11, 2020 at https://www.sciencedirect.com/science/article/pii/S1877050918311062.

[30]Renaud Lifchitz. (Feb. 11, 2019). *The Things Network*. "From LoRaWAN 1.0 to 1.1: Security Enhancements." Accessed on Jan. 11, 2020 at https://www.youtube.com/watch?v=FsO5zxYHfKw.

[31]Cesar Cerrudo, Esteban Martinez Fayo, and Matias Sequeira. (Jan. 2020). *IO Active*. "LoRaWAN Networks Susceptible to Hacking: Common Cyber Security Problems, How to Detect and Prevent Them." Accessed on Jan. 11, 2020 at https://act-on.ioactive.com/acton/attachment/34793/f-87b45f5f-f181-44fc-82a8-8e53c501dc4e/1/-/-/-/-/LoRaWAN%20Networks%20Susceptible%20to%20Hacking.pdf.

[32]Cesar Cerrudo, Esteban Martinez Fayo, and Matias Sequeira. (Jan. 2020). *IO Active*. "LoRaWAN Networks Susceptible to Hacking: Common Cyber Security Problems, How to Detect and Prevent Them." Accessed on Jan. 11, 2020 at https://act-on.ioactive.com/acton/attachment/34793/f-87b45f5f-f181-44fc-82a8-8e53c501dc4e/1/-/-/-/-/LoRaWAN%20Networks%20Susceptible%20to%20Hacking.pdf.

[33]Frank Hessel, Lars Almon, and Flor Álvarez. (May 23, 2020). *Arvix*. "ChirpOTLE: A Framework for Practical LoRaWAN Security Evaluation." Accessed on Jan. 11, 2020 at https://arxiv.org/pdf/2005.11555.pdf.

[34] Semtech. (Aug. 16, 2016). *Semtech*. "Semtech LoRa® RF Technology Adopted by Asia Pacific Telecom for Rollout of Commercial Low Power Wide Area Internet of Things Network in Taiwan." Accessed on Jan. 15, 2020 at https://www.semtech.com/company/press/semtech-lora-rf-technology-adopted-by-asia-pacific-telecom-for-rollout-of-commercial-low-power-wide-area-internet-of-things-network-in-taiwan.

**TREND MICRO™ RESEARCH**

Trend Micro, a global leader in cybersecurity, helps to make the world safe for exchanging digital information.

Trend Micro Research is powered by experts who are passionate about discovering new threats, sharing key insights, and supporting efforts to stop cybercriminals. Our global team helps identify millions of threats daily, leads the industry in vulnerability disclosures, and publishes innovative research on new threats techniques. We continually work to anticipate new threats and deliver thought-provoking research.

**www.trendmicro.com**