ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

inf | Informatik
Computer Science

LoRaWAN (In)Security

Master Thesis

Nico Schottelius, Kamila Součková
`nicosc@ethz.ch, skamila@ethz.ch`

November 8, 2017

Advisers:
Prof. Dr. Srdjan Čapkun
Patrick Leu

Department of Computer Science
ETH Zürich

Abstract

The objective of this distributed systems lab is to analyse the security of LoRaWAN. LoRaWAN is a long range, low bandwidth wireless area network standard commonly used for IoT devices. While LoRaWAN has been released in June 2015, the use of LoRaWAN is still in its early days and not much security-related research is available.

While the initial focus of the project was on general security-related aspects, the focus changed during the project to analyse the security of event-based IoT devices using LoRaWAN.

Contents

Chapter 1

Introduction

LoRaWAN uses the proprietary chirp spread spectrum radio modulation LoRa to achieve long range (>10 km) connectivity, while maintaining low energy consumption. The protocol definition and related materials are published by the LoRa Alliance [1]. As LoRaWAN is designed for internet of things (IoT) devices, it provides very low bandwidth (between 292 bit/s and 50 kilobit/s) at very low energy consumption (100 nA – 10 mA) [7]. Devices are usually built using a microcontroller only and do not contain a full-fledged operating system.

A summary of the LoRaWAN standard [1] follows.

LoRaWAN devices can operate in two different modes:

- ABP (activation by personalisation)
- OTAA (over the air activation)

The first mode, ABP, requires configuring two symmetric keys and a device address per device:

- AppSKey
- NwkSKey
- DevAddr

With ABP, devices are pre-configured and never change keys nor need any join procedure.

With OTAA, devices are configured with

- AppEUI
- AppKey

Devices using OTAA use a join procedure for joining the network: First, the devices sends a Join-Request message to the network, then the network replies with a Join-Accept message. AppSKey and NwkSKey are computed in this procedure.

The two messages in the standard are defined as follows [1]:

Size (bytes)	8	8	2
Join Request	AppEUI	DevEUI	DevNonce

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

When using OTAA, the AppSKey and NwkSKey are determined as follows [1]:

NwkSKey = aes128_encrypt(AppKey, 0x01 | AppNonce | NetID | DevNonce | pad_{16})
AppSKey = aes128_encrypt(AppKey, 0x02 | AppNonce | NetID | DevNonce | pad_{16})

Note that both the AppSKey and the NwkSKey are derived from the AppKey. We will elaborate on this aspect later.

1.1 Security mechanisms in LoRaWan

The standard defines the following security mechanisms:

1.1.1 Payload encryption

The payload of the message is encrypted using AES with 128-bit keys. The contents of the payload is thereby hidden from the attacker (except for its length). However, several important fields are not considered payload and therefore are sent in cleartext, namely:

- DevAddr: the device address – a device identifier. This identifies the device permanently in case of ABP or per-session in case of OTAA. A session can be arbitrarily long-lived, as re-initialization is not specified by the standard.
- FCnt: the frame counter. For both uplink and downlink messages, this is incremented each time a message from/to this device is sent (modulo 2^{16}).
- FCtrl and FOpts: contain settings for the frame and transmission. May identify the device type or manufacturer, as they depend on the capabilities of the device.
- FPort: a field used for multiplexing inside the application. This is application-level information, so it may be sensitive, but it is sent outside of the payload. The authors are surprised by this decision.

1.1.2 Message integrity protection

The important parts of the message (in particular all of the above fields) are protected by a Message Integrity Code derived from an AES-based MAC. The MAC is keyed by NwkSKey in case of messages to the network or AppSKey for application messages. The frame counter is part of the MAC calculation to prevent MIC re-use.

1.1.3 Replay attack protection

The standard defines that the frame counter may only increase in order to prevent replay attacks. Any packets with a frame counter lower than the last seen value should be discarded. However, in practice it is hard to implement a device that behaves accordingly: a LoRa device may lose power anytime, and storing the frame counter in non-volatile storage after every message is often seen as infeasible.[1] Therefore all providers in Switzerland give the option to ignore the frame counter: in this mode, a packet with a frame counter of 0 is allowed any time, with the extra effect of dropping any further packets with high counters.

[1]LoRa devices are usually power-constrained and writing data to non-volatile storage costs extra energy. Perhaps more importantly, it limits the lifetime of the device, because flash storage can only sustain a limited number of overwrites (on the order of thousands per one cell).

Chapter 2

General Security issues

2.1 Incomplete Standard

While the LoRaWAN standard defines several entities, it does not define the network provider entity clearly. Specifically, it does not define how key distribution happens, who defines the symmetric keys used or how they are exchanged. As LoRaWAN security is based on symmetric keys, the lack of a definition poses security risks in leakage of keys during configuration/distribution of devices.

2.2 Key Management

When using OTAA, the NwkSKey and AppSKey are calculated by the network. In theory the network provider could contact a service of the owner of the IoT device to retrieve the NwkSKey and thus not have access to the AppSKey nor the AppKey. In practice, this does not happen: in all network providers that we checked in Switzerland (Swisscom, TTN, Loriot), the network provider itself calculates both keys.

Even though the separated setup has been discussed in the TTN community, there is no implementation available that supports this separation.

The two different AES keys that are defined by the standard, can practically be seen as only one key, as they are derived from the same master key. Thus the objective to protect data from the network provider with two different keys is not achieved in reality. The network provider can decrypt and manipulate all messages, unless additional application-layer security is introduced.

2.3 Static keys

When using ABP, the session keys are fixed and will never be replaced. Thus brute force attacks can use every packet the device sends during its whole lifetime, and leaked keys compromise all future packets until manual re-configuration. While network providers recommend the use of OTAA, in reality many LoRaWAN devices are pre-configured with ABP.

2.4 Pre-configured devices

The test devices that were ordered from Ascoel (www.ascoel.it) contained pre-configured keys. While this might be convenient in some places, it also allows the manufacturer to decrypt the data sent by the devices. Furthermore, the devices came pre-configured in ABP mode and thus never change the session keys. For some devices it was impossible to change the keys or switch to OTAA mode.

2.5 Replay attacks

The LoRaWAN standard defines counters in the frame header:

Size (bytes)	4	1	2	0..15
FHDR	DevAddr	FCtrl	FCnt	FOpts

While in theory they should always be increasing during one session, in practice all network providers offer at least to ignore the counter. ABP devices always use the same session and thus when the battery is replaced, the counter will be reset (usually the counter is not stored in non-volatile memory). Thus network providers allow frame counter reset.

Thus an attacker has the ability to replay packets for ABP devices, if the network provider is configured to ignore the frame counter.

In OTAA the session is restarted every time the device follows the join procedure and thus it uses fresh session keys. It is though undefined, how often or if at all the device repeats the join procedure. The join request itself does not have any protection against replay attacks.

2.5.1 Denial of Service by Replaying Packets

The standard mandates that packets with a too large FCnt value (larger than the last seen + some constant) be dropped, presumably in order to protect from injecting packets (e.g. replaying packets captured in the previous cycle of the 16-bit counter) and/or to avoid silently missing too many packets. However, this opens the door for an effective denial of service attack: if a packet with a *lower* FCnt can be injected, further valid packets will be dropped. Because in practice network providers allow frame counter resets, a packet with FCnt 0 will always be accepted. Therefore replaying that one is sufficient to cause the network to drop all further valid packets from the device, because its FCnt values will be large (the device does not know that we triggered a counter reset).

2.6 Channel selection

The standard defines:

> The end-device changes channel in a pseudo-random fashion for every transmission. The resulting frequency diversity makes the system more robust to interferences.

In practice all tested devices just use round-robin: they select the next channel for the next packet. Thus an attacker can see if a packet was sent even if she did not receive it. This might be particularly useful for event based devices in which a distant attacker wants to distinguish between noise and real events. It also simplifies jamming, as predicting the channel for the next transmission is trivial.

Chapter 3

Security of event-based devices

3.1 Motivation

While it was show in the previous section that there are various obvious security issues within LoRaWAN. Importantly, the nature of a long range wireless network allows attacks from a far away adversary that are not possible with traditional wireless protocols like 802.11. If a device sends a message, it can be detected by a receiver a couple of kilometers away.

3.2 Use cases and a list of devices

To illustrate the potential threats, a list of event-based devices found on the market [6] has been compiled. The attack vectors assume that the devices only send a message if an event has happened. In reality a device usually sends keep-alive packets in addition to event-related packets, but they are easily distinguishable, as they are sent at regular intervals.

Category	**Use case(s)**	**Attack vector / examples**
Water sensors	Detect overflowing bath tub, broken pipe	• A fake plumber could approach the house and get access, as the person knows that there is a water leakage. • Replay attacks could be used to cause a real plumber to visit the house and cause unnecessary costs

Burglar alarm	Detects if somebody is in the house, sends alarm if movement is detected.	• As above
Smoke detector	Detect smoke/fire in a house	• As above
Radioactivity detector	Detect radiation	• Cause panic using replay attacks
Buttons	Sends an event when pressed. Can be used as emergency notification ("HELP") or in case something is broken ("Repair request")	• Similar to water sensors
Remote controls	Control light, power, TV, windows with a LoRaWAN-based remote control	• Adversary can learn about the presence (or absence) of people in the house • Susceptible to replay attacks
Parking sensor	Detects if a parking lot is used or free.	• Adversary can learn about occupation of parking house and possibly can infer business success from the usage.
Bluetooth sensor	If a paired device (usually a smart phone) is out of range, an alarm is sent.	• Adversary can learn that the object or the phone is lost (and look for it, too)

Animal tracking	The position of an animal is sent when motion is detected.	• Adversary can learn about the behaviour of animals, infer information at a distance

While LoRaWAN uses AES encryption, it can be seen from the above list that sending (or not sending) a message might be enough information for an adversary to infer significant knowledge about the objects in question.

3.3 Example: Door sensor

The Ascoel CM868 door sensor[1] was shipped to us with pre-defined ABP keys and DevAddr. To be usable in TTN or any other Swiss network provider, the DevAddr and keys need to be changed. However, according to the manufacturer, the current firmware does not support changing the address nor keys. Thus the device was not usable in Switzerland.

Even if the device were usable, any user of this device would share the key with the manufacturer and could never change the key herself. Because it was not possible to connect the device to a network provider, no further tests have been conducted with this device.

3.4 Example: Burglar alarm

The Ascoel IR868LR family[2] is an infrared passive sensor for indoor applications. It is also shipped in ABP mode with pre-configured keys. It can, however, be reconfigured to OTAA and the parameters can be changed (DevAddr, NwkSKey, AppSKey for ABP; and AppEUI and AppKey for OTAA). After changing the device to OTAA, these packets are being sent to the network:

1. special packets after the device started
2. keep-alive packets every couple of minutes
3. an alarm packet when presence is detected

While all of these messages can be distinguished when the keys are known, the type of these messages can also be understood by an adversary, without

[1] http://www.ascoel.it/images/pdf_ascoel/CM868LRTH_eng.pdf
[2] http://www.ascoel.it/images/pdf_ascoel/IR868LR_eng.pdf

knowledge of the key: the device uses different ports for different types of messages. The port is sent in plaintext as specified by the LoRaWAN standard:

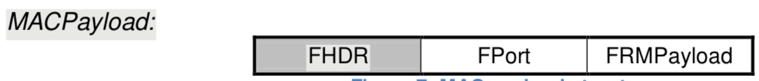
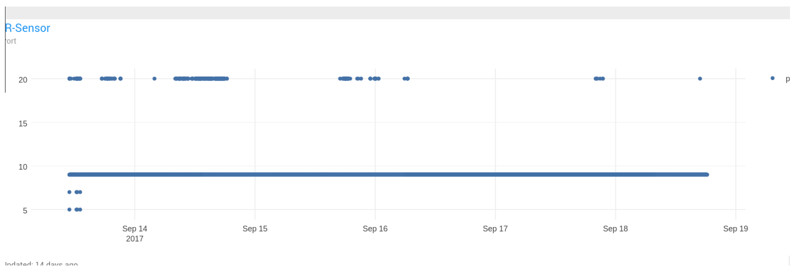


Figure 7: MAC payload structure

The following graph, depicting the ports of packets in time, was created from the received packets of the device:



It can be seen that the keep-alive packets are sent on port 9, boot-up messages on port 5 and 7, and the presence detection uses port 20. While these packets were received using the correct keys, the port information is known to any adversary in range of the device.

3.5 Counter-measures

We assume that the motivation of the manufacturer to use different ports was to make life easier for developers. However, as it is sent in plain text, the information is revealed to an adversary. This problem could easily be fixed by including the message type within the encrypted payload and always using the same port. This would have to be done on the application level, as encryption of the port itself is not possible in the current standard.

Even without the port information, the regularity of non-event packets (or simply the number of packets including the regular ones) can often be used to infer information. If event-based devices are not time-critical (such as e.g. parking sensors), they can opt for sending packets in fixed intervals in order to hide the events.

Further, the frame counter (or the channel, see 2.6) can be used by an attacker to learn the number of events that happened even before she started capturing packets.

To sufficiently hide events without delaying the packets, one needs to introduce dummy traffic to mask the real events. Depending on the target difficulty of guessing the real event, the amount of dummy traffic needs to be varied. However, this approach will consume more energy and thus might results in lower lifetime of battery-powered devices, if not compensated for otherwise.

Chapter 4

Changes in LoRaWAN 1.1

The new LoRaWAN 1.1 standard brings several changes to the table. Perhaps the most significant one is the addition of the backend definition. We will present the summary of the changes and their security impact below.

Note: As of the time of writing, only a draft version of the standard is available. Nevertheless, we present the changes here, as the functional aspects are more-or-less finished.

4.1 Frame counters

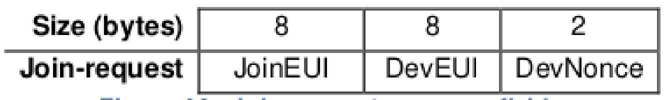
4.1.1 Avoiding replay attacks and nonce re-use

The LoRaWAN 1.1 standard defines three instead of two frame counters. While 1.0 defined one frame counter for uplink messages, 1.1 defines one counter for messages with fport = 0 and one counter for all other uplink messages. All frame counters are now 32-bit values rather than the original 16-bit. More importantly, LoRaWAN 1.1 forbids frame counter resets. Specifically:

> ABP devices have their Frame Counters initialized to 0 at fabrication. In ABP devices the frame counters MUST NEVER be reset during the device's life time. If the end-device is susceptible of [sic] losing power during its life time (battery replacement for example), the frame counters SHALL persist during such event.
>
> The end-device shall not reuse the same FCntUp value, except for retransmission, with the same application and network session keys.

In case of OTAA/Network Join, the counter is called DevNonce and is reset on every join:

> DevNonce is a counter starting at 0 when the device is initially powered up and incremented with every Join-request. A DevNonce



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

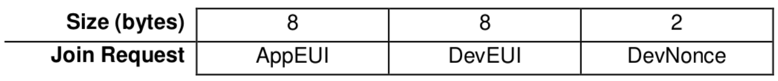
Figure 44 : Join-request message fields

> value SHALL NEVER be reused for a given JoinEUI value. If the end-device can be power-cycled then DevNonce SHALL be persistent (stored in a [sic] non-volatile memory). Resetting DevNonce without changing JoinEUI will cause the Network Server to discard the Join-requests of the device. For each end-device, the Network Server keeps track of the last DevNonce value used by the end-device, and ignores Join-requests if DevNonce is not incremented.

Thus ABP devices need to have permanent storage for their frame counter, while OTAA devices can keep their counter in volatile memory, because they can reset their counter with new keys by a new join request.

However OTAA devices receive a JoinNonce from the network that will always increase and thus needs to be stored in non-volatile memory:

> Note: This mechanism prevents replay attacks by sending previously recorded join-request messages with the intention of disconnecting the respective end-device from the network. Any time the Network Server processes a Join-Request and generates a Join-accept frame, it shall maintain both the old security context (keys and counters, if any) and the new one until it receives the first successful uplink frame containing the RekeyInd command using the new context, after which the old context can be safely removed.
>
> The JoinNonce is a device specific counter value (that never repeats itself) provided by the Join Server and used by the end-device to derive the session keys FNwkSIntKey, SNwkSIntKey, NwkSEncKey and AppSKey. JoinNonce is incremented with every Join-accept message.



Size (bytes)	8	8	2
Join Request	AppEUI	DevEUI	DevNonce

> The device keeps track of the JoinNonce value used in the last successfully processed Join-accept (corresponding to the last successful key derivation). The device SHALL accept the Join-accept only if the MIC field is correct and the JoinNonce is strictly greater than the recorded one. In that case the new JoinNonce value re-

places the previously stored one. If the device is susceptible of being power cycled the JoinNonce SHALL be persistent (stored in a non-volatile memory).

4.1.2 Packets no longer dropped after a frame counter gap

The 1.0 standard defines that when a counter with a too big gap was detected, further frames are to be dropped (see 2.5.1). 1.1 removes this definition. The removal of this behavior prevents denial of service attacks by replaying packets with a smaller frame counter.

4.2 FPort still not encrypted

The FPort is still sent as plain text in LoraWAN 1.1. Thus the event indications that were observed with the burglar alarm in our experiments (section 3.4) are still observable. To improve security, the LoRaWAN standard should move the FPort part into the encrypted message part.

4.3 NEW: Lora alliance DNS as a single point of failure

The new standard defines to use the DNS names `JOINEUIS.LORA-ALLIANCE.ORG` and `NETIDS.LORA-ALLIANCE.ORG` that SHALL be used for finding the Join Server and NetID, respectively.

In this regard the domain name `LORA-ALLIANCE.ORG`, and its associated name servers, become **new** single points of failure for the whole system.

Compare with chapter 19.1 “DNS configuration” of the backend definition.

4.4 Communication between gateway and network

In LoRaWAN 1.1 gateways are connected to the Network Server via secured IP connections, while previously it was undefined. In practice packets of LoRaWAN 1.0 gateways are usually forwarded using plain UDP without any encryption or signature. Defining this is therefore an improvement.

4.5 NetID defined

Before version 1.1 the handling of the NetID was undefined and could be chosen by a network provider. Since 1.1 they are allocated by LoRa Alliance. Even though this will regulate the use more clearly, it also adds an additional attack vector: now an attacker can lookup the NetID from the DNS database and specifically attack the chosen network.

4.6 Communication between AS/NS/JS

The LoRaWAN 1.1 standard defines a reference backend network model as follows:

3 Network Reference Model

Figure 1 and Figure 2 show the Network Reference Model (NRM) for the LoRaWAN architecture.

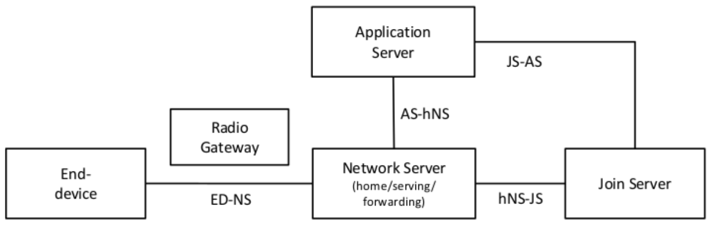


Figure 1 LoRaWAN Network Reference Model (NRM), End-Device at home

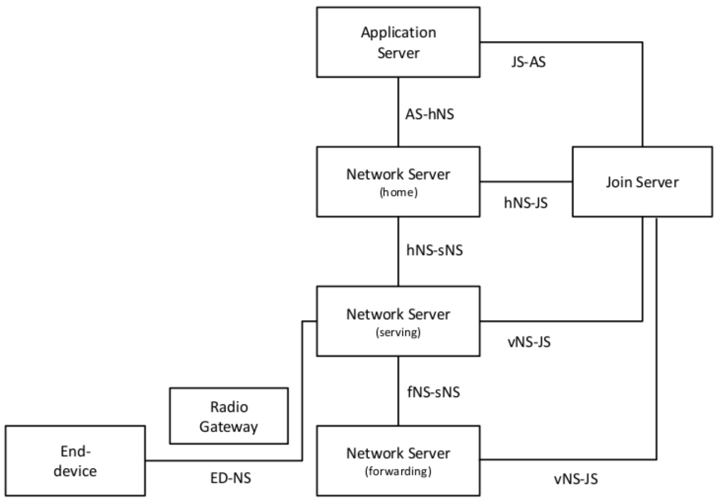


Figure 2 LoRaWAN Network Reference Model (NRM), roaming End-Device

The LoRaWAN Standard defines only using shared keys between the application server, network server and join server. However, public key crypto would also be feasible, and would simplify key management and configuration.

4.7 Unique Key usage

Version 1.1. defines that a key should be unique per device:

> Since all end-devices are equipped with unique application and network root keys specific for each end-device, extracting the App-

Key/NwkKey from an end-device only compromises this one end-device. [6.1.1.3 footnote 1].

Prior to 1.1 this is undefined and shared keys are used in different devices, especially for ABP devices.

4.8 Key Derivation

One of the challenging aspects of LoRaWAN is a safe key derivation method to separate payload and MAC command encryption. LoRaWAN 1.0 does not define the backend. Thus deciding which part of the infrastructure generates the keys is left open to the network provider. In practice this led to the situation that network providers (TTN, Loriot, Swisscom) all have knowledge of both keys and user data is visible to the network provider.

Even though in LoRaWAN backend services are defined and the Join Server handles the key management, it is highly unlikely that a user actually operates the join server, as it takes a core role in the network provider's backend.

It is defined that the application server will request the keys from the join server on arrival of encrypted messages with payload inside. That means that the join server and the application server would need to be operated by the user to be able to hide the payload encryption key.

4.9 Dangers of overdefinition

While LoRaWAN 1.0 did not define the backend part at all, in LoRaWAN 1.1 various aspects of communication (like HTTP transport, POST-based, JSON) are half-defined. It seems that the omission of the backend definition in 1.0 caused the authors to include an incomplete communication scheme that might not be fully thought through.

Chapter 5

Conclusion

While the designers of the LoRaWAN standard abstained from reinventing cryptographic methods and chose to use AES for encryption, the lack of specification of the network design and key management introduces several security flaws. The choice of symmetric keys (for simplicity?) shifts the burden of key management and distribution to users, manufacturers, and network providers. This could be simplified by the use of public key cryptography, or by specifying or recommending well-thought-out procedures in the standard. The impact of public-key cryptography on energy consumption has not been studied.

While these issues might be easily addressed in a newer version of the standard, the visibility of events over long distances needs more sophisticated countermeasures, as events leak information by their sheer presence.

The draft of the upcoming LoRaWAN 1.1 standard addresses some, but not nearly all, of the outlined issues, while also introducing some new problems. Therefore LoRaWAN security is still far from solved.

Bibliography

[1] *LoRaWAN^{TM} Specification*, N. Sornin (Semtech), M. Luis (Semtech), T. Eirich (IBM), T. Kramp (IBM), O.Hersent (Actility). Available at https://www.lora-alliance.org/ after registration.

[2] *Understanding the Limits of LoRaWAN*, F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Meli-Segu, T. Watteyne. https://arxiv.org/pdf/1607.08011.pdf

[3] *LoraWAN Overview*, TTN. https://www.thethingsnetwork.org/wiki/LoRaWAN/Home

[4] *LoRaWAN Security*, TTN, talk by Johan Stokking. https://www.youtube.com/watch?v=Nu_yZelDMZI

[5] *Reversing LoRa*, MATT KNIGHT. http://static1.squarespace.com/static/54cecce7e4b054df1848b5f9/t/57489e6e07eaa0105215dc6c/1464376943218/Reversing-Lora-Knight.pdf

[6] *Use case overview*, Semtech. http://www.semtech.com/wireless-rf/internet-of-things/lora-applications/briefs

[7] *Long Range Wide Area Network*, Wikipedia DE. https://de.wikipedia.org/wiki/Long_Range_Wide_Area_Network, retrieved Oct 2017.