Security Vulnerabilities in LoRaWAN

Presented by
Xinyuan Ma
(xma34@ucsc.edu)

Authors

Xueying Yang,

Evgenios Karampatzakis,

Christian Doerr,

Fernando Kuipers

Outline

- Introduction
- Security features of LoRaWAN
- Attacks and attack mitigations
    - Replay attack
    - Eavesdropping
    - Bit-flipping attack
    - ACK spoofing
    - LoRa class B attacks

Outline

- Introduction
- Security features of LoRaWAN
- Attacks and attack mitigations
    - Replay attack
    - Eavesdropping
    - Bit-flipping attack
    - ACK spoofing
    - LoRa class B attacks

IoT Devices and Security

- Insecure defaults or insecure, remotely exploitable code
- Lack of accepted reference architecture
- Deployed in exposed and complex environment

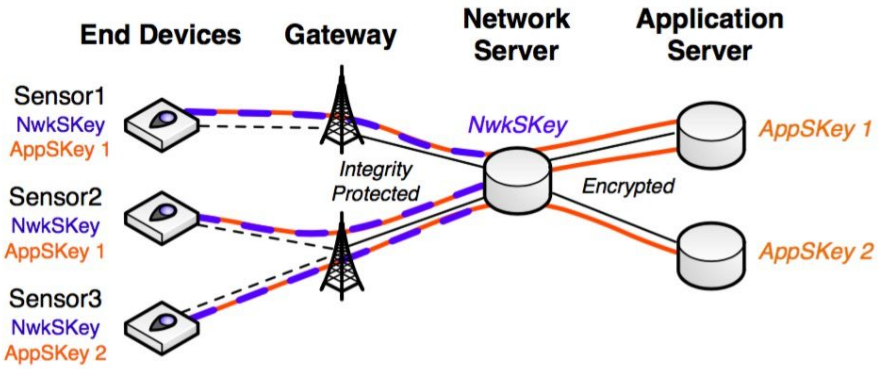
LoRaWAN

- LoRaWAN: Long Range Wide Area Network
- MAC Layer protocol
- Widely adopted in IoT

Outline

- Introduction
- Security features of LoRaWAN
- Attacks and attack mitigations
    - Replay attack
    - Eavesdropping
    - Bit-flipping attack
    - ACK spoofing
    - LoRa class B attacks

Channel Confidentiality

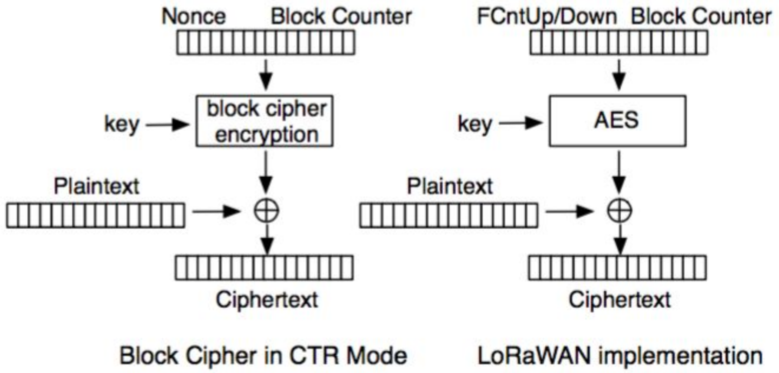


2 keys to ensure confidentiality:

Network Key (NwkSKey) → between the IoT device and the network infrastructure

Application Key (AppSKey) → between the IoT device and a third-party application provider in the backend

Channel Confidentiality - Message Encryption



LoRaWAN borrows idea from Block Cipher in CTR mode

AppSKey encryption procedure:

1. Use ***FCntUp*** and ***FCntDown*** as Nonce;
2. The block cipher (AES) generate a pseudo-random permutation, which is used as a keystream;
3. Apply exclusive OR (XOR) to encrypt the plaintext.

Enrollment Protocol

The 2 keys, NwkSKey and AppSKey, needs to be ‘enrolled’ into the protocol.

By:

1. OTAA (Over-the-Air Activation)
2. ABP (Activation by Personalization)

Enrollment Protocol - OTAA

End device: send a ***Join Request*** (contains a 3-byte DevNonce) to network server;

Network server checks if 1) not accepted, then do nothing; 2) accepted, then sends a ***Join Accept*** to the end device.

Join Accept: contains a 3-byte AppNonce

Then both sides use this AppNonce to generate the NwkSKey and AppSKey

$$\text{NwkSKey} = \text{AES}_E\ (\text{AppKey},\ 0\text{x}01\ ||\ \text{AppNonce}\ ||\ \text{NetID}\ ||\ \text{DevNonce}\ ||\ \text{pad}\ )$$

$$\text{AppSKey} = \text{AES}_E\ (\text{AppKey},\ 0\text{x}02\ ||\ \text{AppNonce}\ ||\ \text{NetID}\ ||\ \text{DevNonce}\ ||\ \text{pad}\ )$$

Enrollment Protocol - ABP

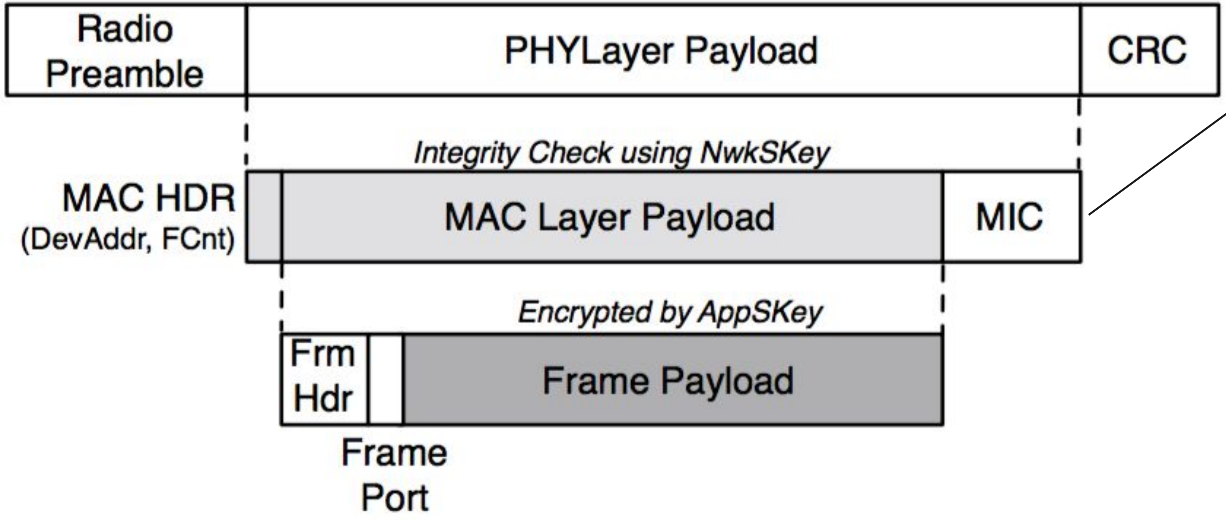
Different from OTAA: it skips the exchange of join messages;

DevAddr, NwkSKey, AppSKey are assigned to end devices, and stored in the server;

These unique parameters will be used across the sessions until updated in the device;

Encrypted messages are sent directly from an end device to server.

Integrity and Authenticity Validation



MIC: Message Integrity Code

It is calculated using ***NwkSKey*** and AES method discussed above.

Outline

- Introduction
- Security features of LoRaWAN
- Attacks and attack mitigations
    - Replay attack
    - Eavesdropping
    - Bit-flipping attack
    - ACK spoofing
    - LoRa class B attacks

Targets of Attacks

1. Adversary wants to ***eavesdrop*** and ***decrypt*** the content of a frame
2. It is possible that the content of a packet would be ***modified outside of integrity check***
3. Messages would be ***replayed***
4. A node would be tricked into believing that a message has been received successfully while it hasn't
5. Battery exhaustion attack

Outline

- Introduction
- Security features of LoRaWAN
- Attacks and attack mitigations
    - Replay attack
    - Eavesdropping
    - Bit-flipping attack
    - ACK spoofing
    - LoRa class B attacks

Replay Attack for ABP-activated Nodes

Recall: ABP used ***static*** keys which are programmed into the device

Consequence: ABP-activated end device ***reuses*** the frame counter value from 0 with the ***same*** keys

- When does this happen?
- When counter overflows and is reset to 0.

Replay Attack for ABP-activated Nodes (Cont'd)

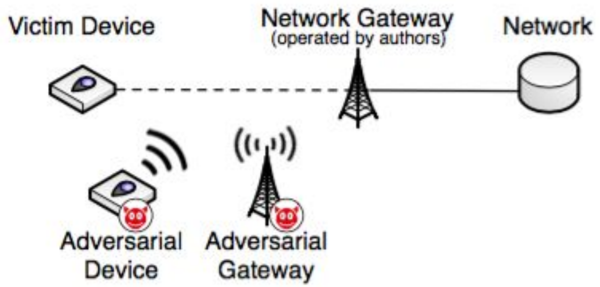
Steps for adversary to perform an attack:

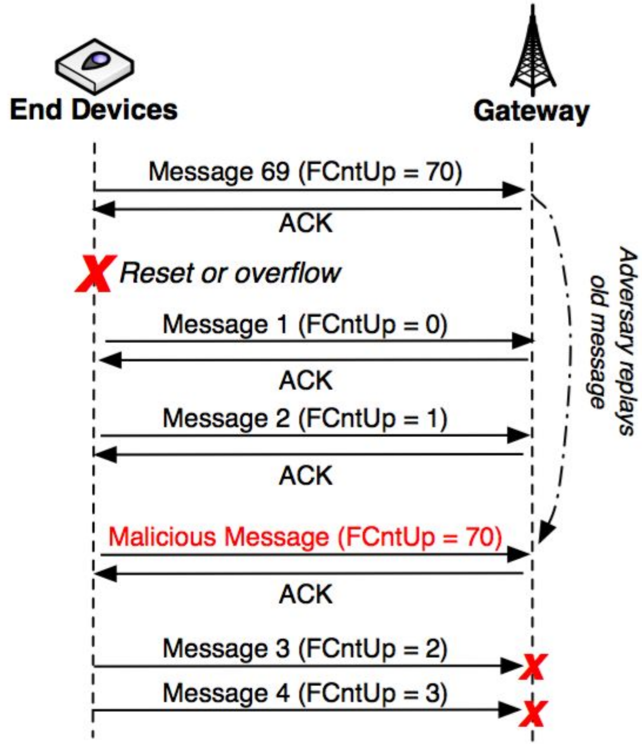
1.Monitor, store, keep in memory the uplink messages

2.Wait and prepare for the counter to be reset

3.Replay the malicious message when the time comes

*4.Replay this message again and again to block end device permanently (DoS attack)






```
Thu Apr 13 16:04:50 2017
DevAddr 89140126 , Counter number is 3 , Physical Payload is 4089140126000300530cb6cea1637e08d3c8240257
Thu Apr 13 16:05:49 2017
DevAddr 89140126 , Counter number is 5 , Physical Payload is 40891401260005008646398lfa78962f244c5624f0
DevAddr 24170126 , Counter number is 49817 , Physical Payload is 40241701260099c20371b1fe383188ac82
DevAddr 89140126 , Counter number is 6 , Physical Payload is 408914012600060003d226c33a4882c44af7c5bac9b
Thu Apr 13 16:06:48 2017
DevAddr 24170126 , Counter number is 49819 , Physical Payload is 4024170126009bc203dd7d7ba55fd710d2
DevAddr 89140126 , Counter number is 7 , Physical Payload is 4089140126000700297259 7f1f3eab3c254bccb946
DevAddr 24170126 , Counter number is 49820 , Physical Payload is 4024170126009cc20337ed4acfba5046fd
Thu Apr 13 16:07:47 2017
Thu Apr 13 16:08:46 2017
DevAddr 89140126 , Counter number is 10 , Physical Payload is 4089140126000a0031d5ef2a97d488b8232c8c9f39
DevAddr 89140126 , Counter number is 0 , Physical Payload is 40891401260000000473663cb1f6a23ec3bf98c4798
Here is a reset!
[3, 5, 6, 7, 10, 0]
>>RN2483 1.0.1 Dec 15 2015 09:38:09

 Attacking......
 Thu Apr 13 16:09:48 2017
 DevAddr 89140126 , Counter number is 10 , Physical Payload is 4089140126000a0031d5ef2a97d488b8232c8c9f39
 Thu Apr 13 16:10:47 2017
 DevAddr 89140126 , Counter number is 2 , Physical Payload is 408914012600020 0455e51f71a43d61cba6736abcc
 DevAddr 89140126 , Counter number is 3 , Physical Payload is 4089140126000300 2b0cb4c2a1637e0bd3d68a025f
 DevAddr 89140126 , Counter number is 4 , Physical Payload is 408914012600040 05477e5b703fea2f3644548a6bf
 Thu Apr 13 16:11:46 2017
 DevAddr 24170126 , Counter number is 49838 , Physical Payload is 402417012600aec203808788497e5c79a6
 DevAddr 89140126 , Counter number is 5 , Physical Payload is 408914012600050 04b46358dfa78962e24a11899da
 Thu Apr 13 16:12:45 2017
 DevAddr 89140126 , Counter number is 6 , Physical Payload is 408914012600060 045206133a4882c42af70467d84
 Thu Apr 13 16:13:44 2017
 DevAddr 89140126 , Counter number is 8 , Physical Payload is 408914012600080 022c12e31a31c5b626b4c5b62eb
 DevAddr 89140126 , Counter number is 9 , Physical Payload is 408914012600090 03e507f00b4b0878653e65329af
 Thu Apr 13 16:14:43 2017
 DevAddr 89140126 , Counter number is 10 , Physical Payload is 4089140126000a0015d4e52297d488bd23a28bfe84
 Thu Apr 13 16:15:42 2017
 DevAddr 89140126 , Counter number is 11 , Physical Payload is 4089140126000b00143e307772c1eaeb47678fb066
 DevAddr 89140126 , Counter number is 12 , Physical Payload is 4089140126000c003d4905e528298ffad1830f2529
```

time	counter	port	dev id	
16:16:00	13	6	22	34 34 37 20 30 32 34 00
16:15:25	12	61	22	34 39 36 20 30 32 34 00
16:14:51	11	20	22	35 34 33 20 30 32 31 00
16:08:49	10	49	22	34 38 30 20 30 32 31 00
16:08:34	0	71	22	31 39 32 20 30 32 32 00
16:07:59	10	49	22	34 38 30 20 30 32 31 00
16:06:16	7	41	22	35 32 37 20 30 32 33 00
16:05:42	6	61	22	36 38 37 20 30 32 34 00
16:05:07	5	134	22	34 39 34 20 30 32 33 00
16:03:59	3	83	22	34 34 38 20 30 32 32 00

Replay Attack Mitigation

Minimize the use of ABP, and use OTAA if possible

If insist on using ABP:

1. Adopt new keys periodically

2. Protect end devices physically, i.e. use non-volatile memory to preserve the counter value and avoid sudden change

3. Rekey every time the counter reaches its maximum value (If using OTAA: go through OTAA activation procedure again)

Outline

- Introduction
- Security features of LoRaWAN
- Attacks and attack mitigations
  - Replay attack
  - Eavesdropping
  - Bit-flipping attack
  - ACK spoofing
  - LoRa class B attacks

Eavesdropping

Reason for this attack: the block cipher (AES) recreates exactly the same key material every time the counter values repeat.

Specifically:

(plaintext P) XOR (key stream K) = ciphertext C

When given 2 plaintexts P1 and P2 encrypted under the same keystream K, then we have

$$\begin{aligned} C_1 \oplus C_2 &= (P_1 \oplus K) \oplus (P_2 \oplus K) \\ &= P_1 \oplus P_2 \oplus \underbrace{(K \oplus K)}_{\text{cancels out}} \\ &= P_1 \oplus P_2. \end{aligned}$$

Eavesdropping Cont’d

Step 1: Guess a part of the content in P1

Step 2: Derive the part of P2 at the corresponding position

If all the plaintexts are readable, the guess is possibly correct

Validation and ‘enhance’ learning: the more resets, the more likely it is to recover the message

Eavesdropping Mitigation

It’s the counter value’s mistake again.

1. Replace the counter value by a nonce, which is generated from a cryptographically-secure pseudo random number generator

2. Rekey on reset (easier to achieve by OTAA as well)

Outline

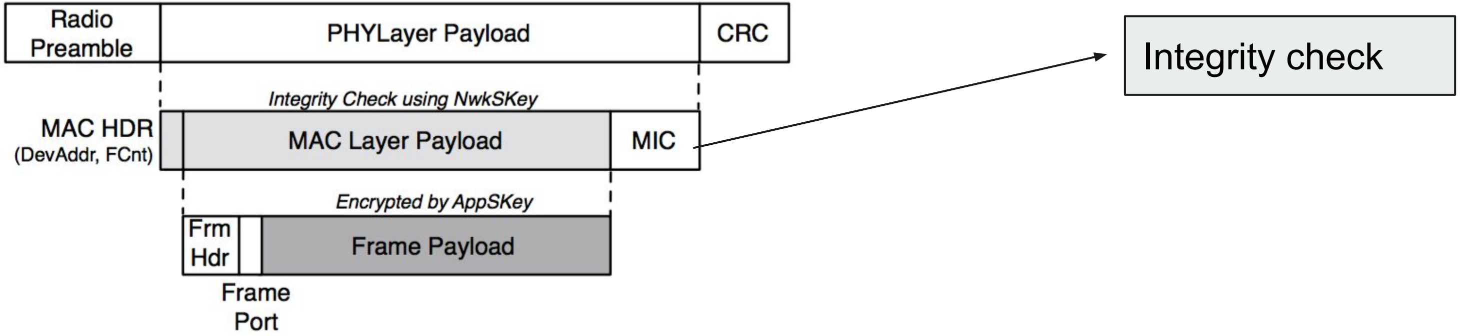
- Introduction
- Security features of LoRaWAN
- Attacks and attack mitigations
  - Replay attack
  - Eavesdropping
  - Bit-flipping attack
  - ACK spoofing
  - LoRa class B attacks

Bit-flipping Attack

There should have been ***integrity checking*** to prevent this from happening. But why?

Reason: Encryption and integrity check do NOT happen at the same scope.

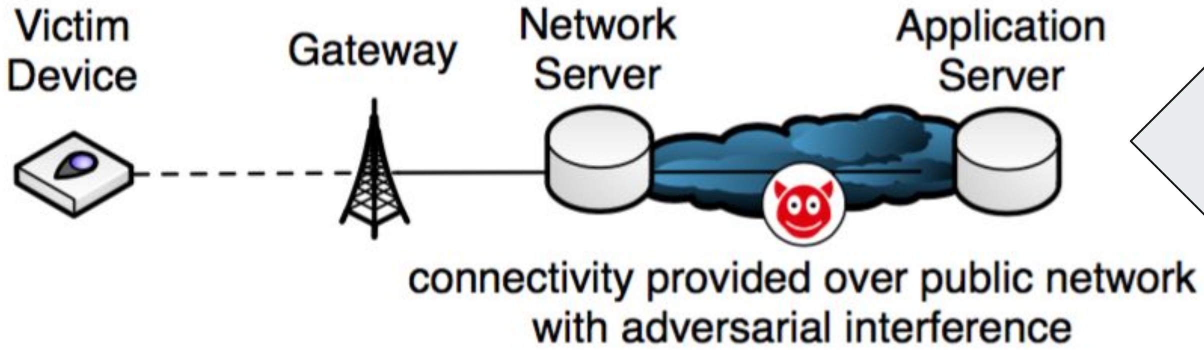
Recall this figure:



Bit-flipping Attack Cont'd

In between the infrastructure operator's ***network server*** and the IoT solution provider's ***application server***, the content cannot be checked for integrity and authenticity

i.e. The integrity check is terminated too early and there is no integrity check throughout transmission

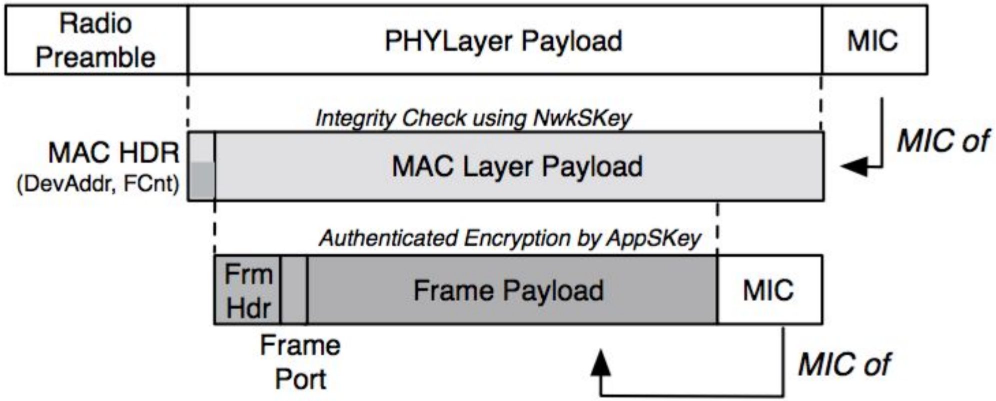


Man-in-the-middle: routing-based attacks, physical and link-layer based attacks, etc.

Bit-flipping Attack Mitigation

1.Run the integrity check value at the ***application server*** instead of the network server

2.Repurpose protocol fields: replace the CRC by a MIC



Outline

- Introduction
- Security features of LoRaWAN
- Attacks and attack mitigations
    - Replay attack
    - Eavesdropping
    - Bit-flipping attack
    - ACK spoofing
    - LoRa class B attacks

ACK Spoofing

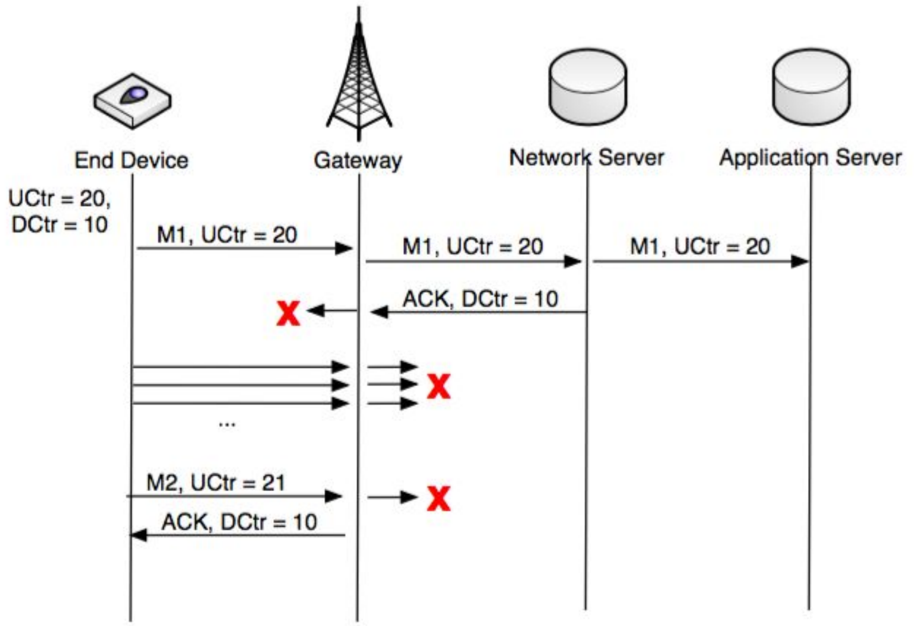
To maximize battery life and reduce the time the radio needs to be powered up, ACK messages from network server do NOT specify which message it is confirming:

PHYSICAL PAYLOAD FORMAT OF AN ACK MESSAGE.

MHDR	DevAddr	FCtrl	FCnt	MIC
60	88889999	20	0B00	BAE1557A

sequential number of all downstream messages

ACK Spoofing Cont'd



Assume that the gateway is malicious, selectively suppressing certain frames from transmission.

First it blocks the ACK from network server, making end device believe that M1 fails to transfer.

Then when end device send M2, this malicious gateway responds using ACK it blocked before to fool end device, making the end device believe M2 is successfully delivered.

ACK Spoofing Mitigation

ACK spoofing happens because the ACK doesn't specify which message it actually confirms.

Recall in *Bit-flipping Attack Mitigation* part that we use MIC to guarantee integrity throughout the whole transmission.

Apply MIC to both the connection to the network server and the application server:

- then it is possible to add a cryptographic checksum with the returned ACK, including the entire packet as sent by the field device
- IoT device can confirm that ACK belongs to this message and the value of the message remains unchanged during the transmission

Outline

- Introduction
- Security features of LoRaWAN
- Attacks and attack mitigations
    - Replay attack
    - Eavesdropping
    - Bit-flipping attack
    - ACK spoofing
    - LoRa class B attacks

LoRa Class B Attacks

LoRa Class B network: field devices periodically wake up to wait for any incoming messages, the durations of which are specified by the ***beacons broadcast*** by the gateway

Class B network balances power consumption and the possibility to periodically relay downlink instructions.

Problem: ***beacons*** are not encrypted, nor protected against malicious modification.

- Spoofing the location of a LoRa gateway (location information can be modified)
- Exhausting the battery

LoRa Class B Attacks Mitigation

Background: beacon frames are not protected, and are especially lacking in integrity check value.

Solution: also change the PHY CRC to a MIC → authenticating the beacon frames