LoRaWAN Insecurities

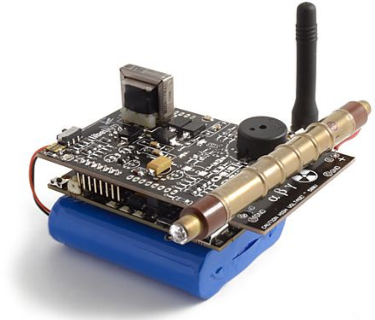
Nico Schottelius & Kamila Součková

LoRaWAN

- Long range (10km+)
- Low energy (years runtime on a single battery)
- Low bitrate
- Network
  - 868 Mhz
  - Datagram-based (separate packets, usually not ack’d)
  - Network provider separate from application

LoRaWAN

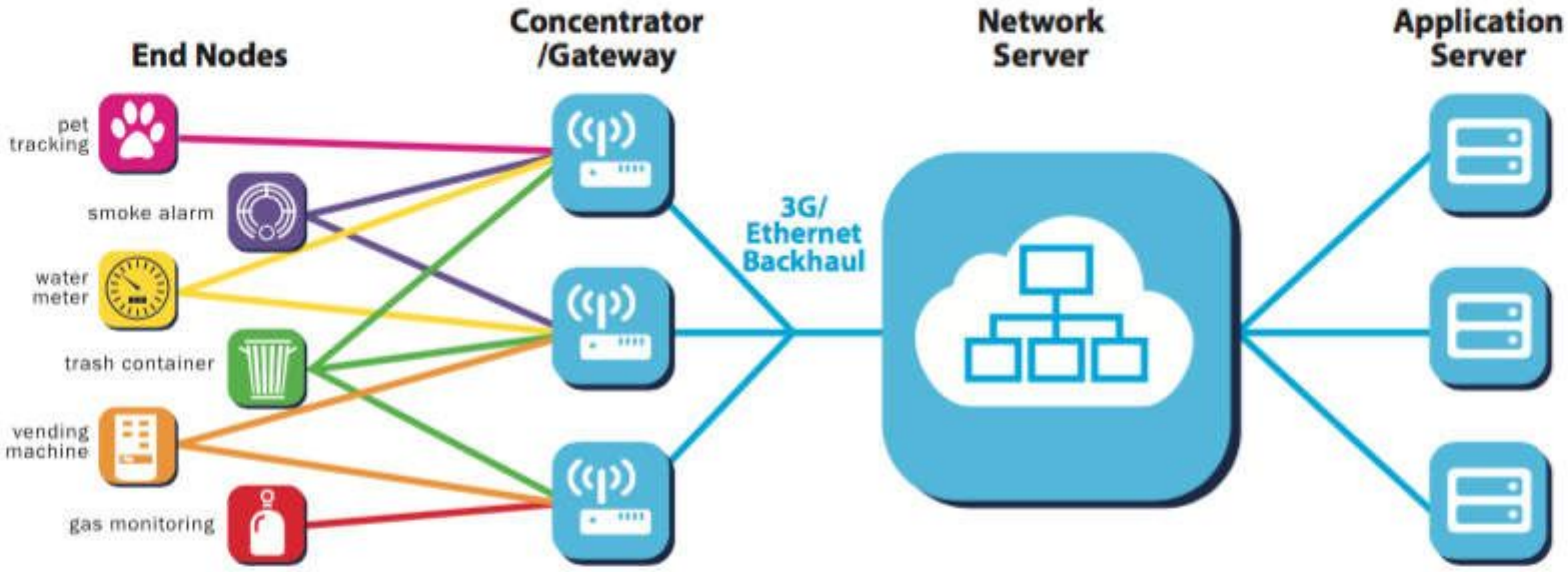
- Primary usage: IoT
  - Measurements (temperature, humidity, water / oil level)
  - Status Sensors (water, radioactivity, burglar alarm, parking)
  - Animal tracking
  - Buttons

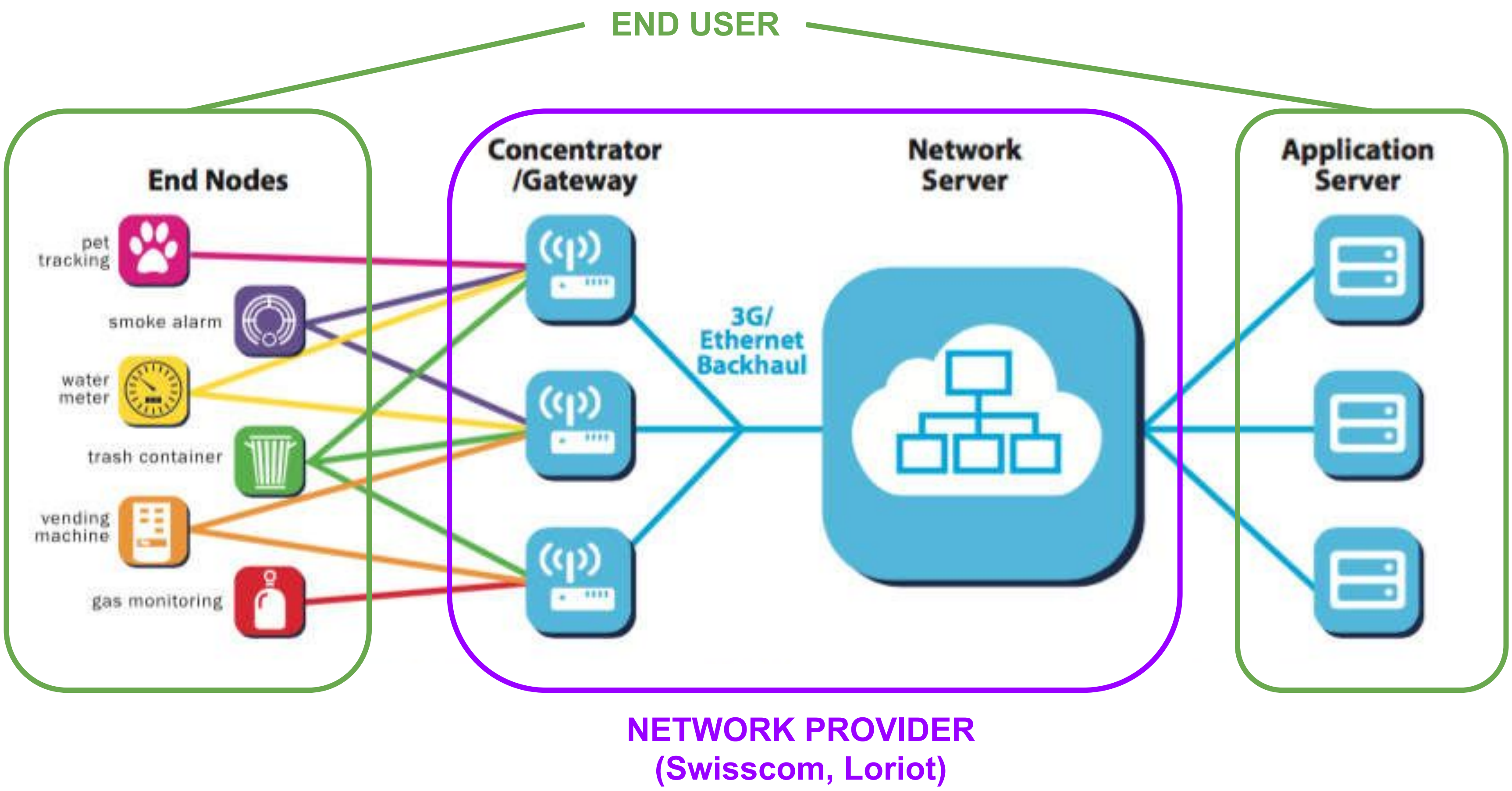


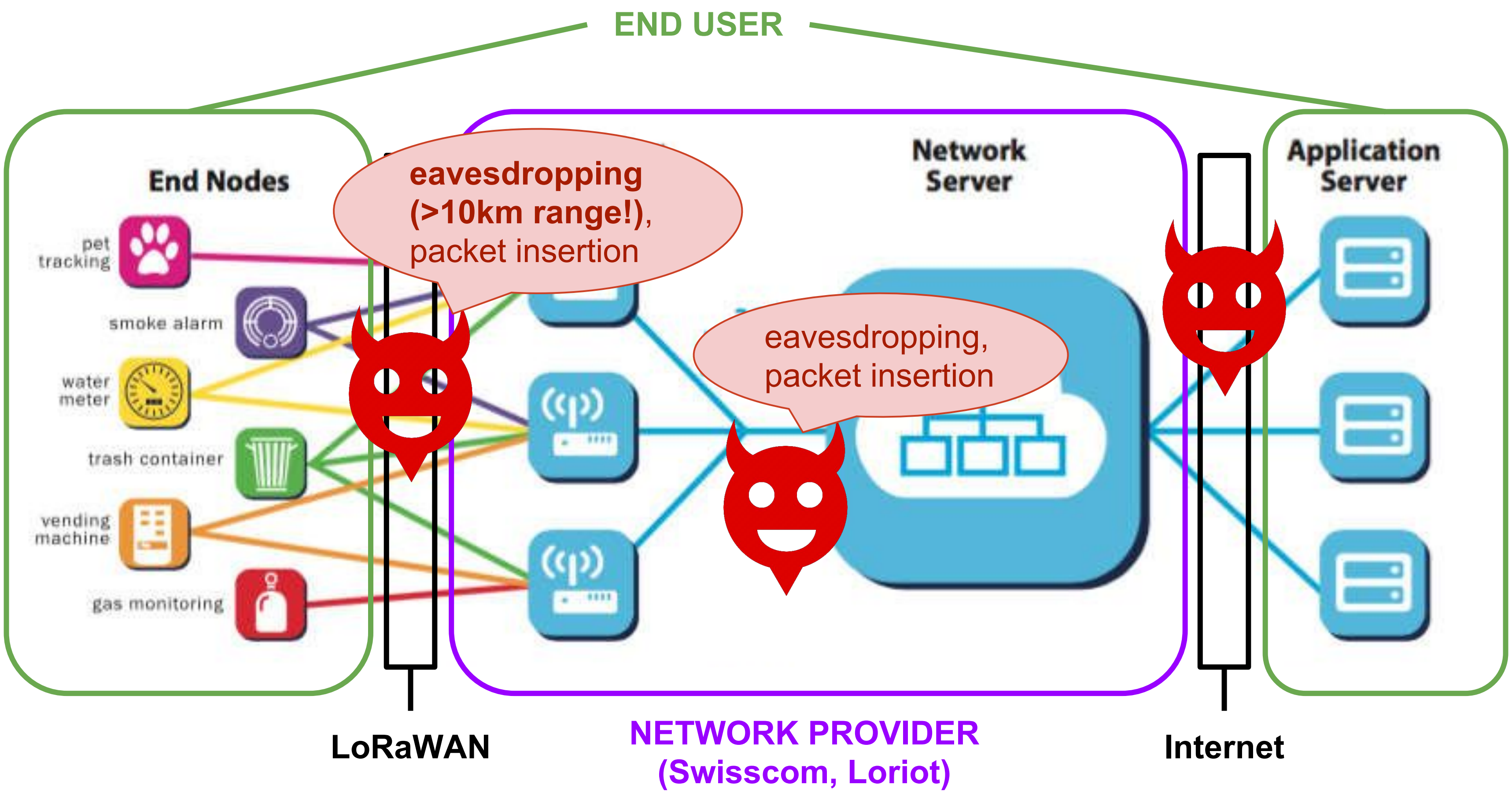
Motivation

- New protocol: Lessons learned or mistakes repeated?
- Wireless + Long Range + Energy-efficient = Very Interesting
- Huge Growth + Hype

LoRaWAN Architecture




END USER
End Nodes
pet tracking
smoke alarm
water meter
trash container
vending machine
gas monitoring
Concentrator /Gateway
3G/ Ethernet Backhaul
Network Server
Application Server
NETWORK PROVIDER
(Swisscom, Loriot)


END USER
End Nodes
pet tracking
smoke alarm
water meter
trash container
vending machine
gas monitoring
eavesdropping (>10km range!), packet insertion
Network Server
eavesdropping, packet insertion
Application Server
LoRaWAN
NETWORK PROVIDER (Swisscom, Loriot)
Internet

LoRaWAN Security Features

- Devices can operate in one of two modes:

  ABP (activation by personalization)
  - Static device address
  - Static long term keys

  OTAA (over the air activation)
  - Device actively ***joins*** the network
  - Derived short term keys & address

- Symmetric encryption (AES-128)
  - Messages to network encrypted with NwkSKey, to application with AppSKey
- Replay protection: frame counters
- Integrity protection: AES-based MAC

LoRaWAN Security Features

The devil is in the details...

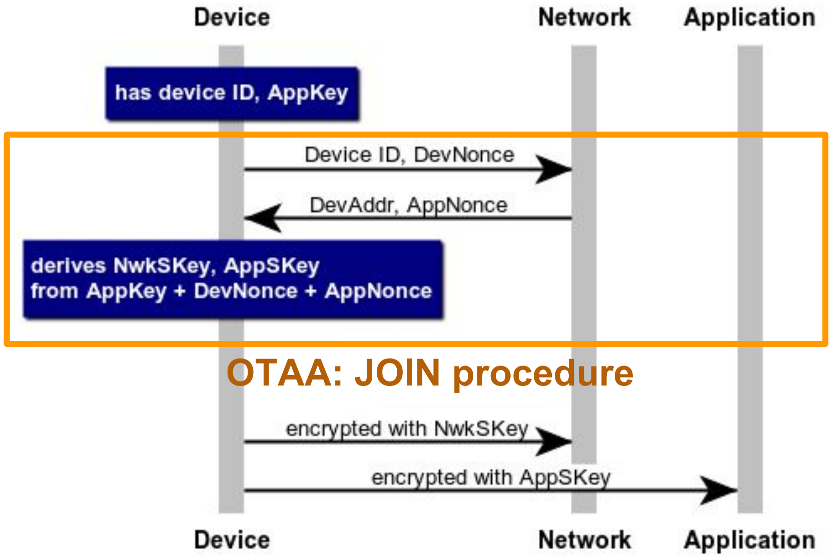
Incomplete Standard: Keys

- Nothing about key distribution
  - Symmetric encryption ⇒ key distribution not trivial
  - Some devices come with hard-coded keys and IDs that cannot be changed
- ABP shouldn't be used in production
  - Keys never change automatically with ABP
  - But some devices only support ABP

Ascoel CM868 door sensor:
ABP-only, with hard-coded DevAddr and keys
(fully standard-compliant)

Incomplete Standard: Network Provider

- Undefined: who generates **master** keys + IDs?
    - In the real world, it is the network
- For OTAA: **session** keys
    - NwkSKey and AppSKey both derived from AppKey
    - Computed by the network provider in practice
    - In theory, application could provide keys as a Service to the network, but nobody does that
    - ⇒ AppSKey known to the network

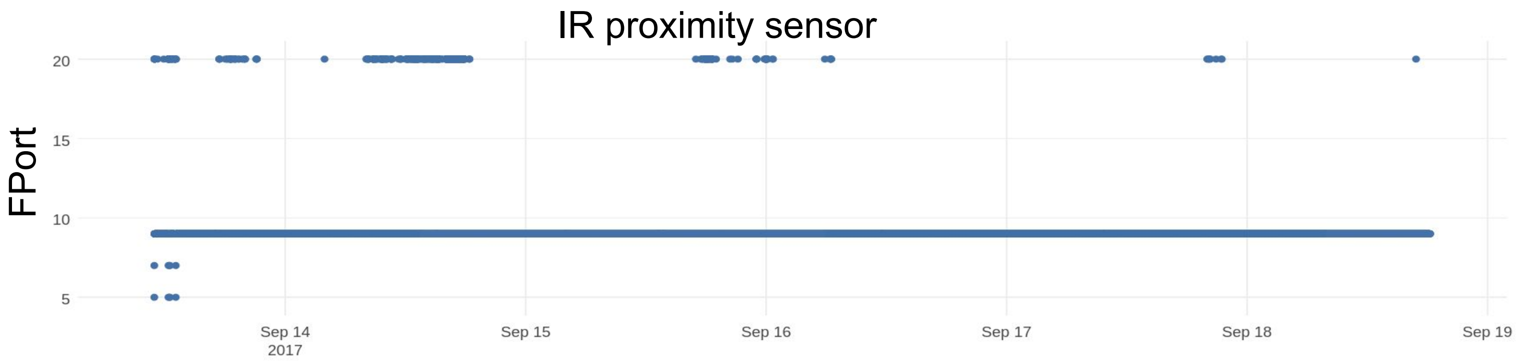


Replay Attacks

- In theory: frame counter should always increase
- In practice: storing frame counter in non-volatile memory is impractical ⇒ all network providers allow frame counter resets
  - Replaying frame #0 works
  - ⇒ DoS:
    - *Frame counter gap*: current - last seen
    - Standard mandates dropping packets if gap too big
    - Replay frame 0, device won’t know a reset occurred ⇒ legit packets will be ignored
- OTAA: Join requests have no replay protection ⇒ DoS

Plain Text Fields

- Packet body encrypted (⇒ only length leaked), header in plain text
- Notable header fields:
  - **DevAddr: permanent ID for ABP; long-lived and easy to associate for OTAA**
    - ⇒ no anonymity
  - **FPort: used for message multiplexing <u>inside the application</u>**



Event-based devices

- Many devices send packet when something happens
- **Existence of packet reveals information**
- Example “use cases”:

Fake plumber attack

- water sensors
- burglar alarms / motion detectors
- smoke detectors

Information leakage

- radioactivity detectors
- door sensors
- buttons
- remote controls
- parking sensors
- Bluetooth range sensor
- animal tracking

Event-based Devices: Counter-measures

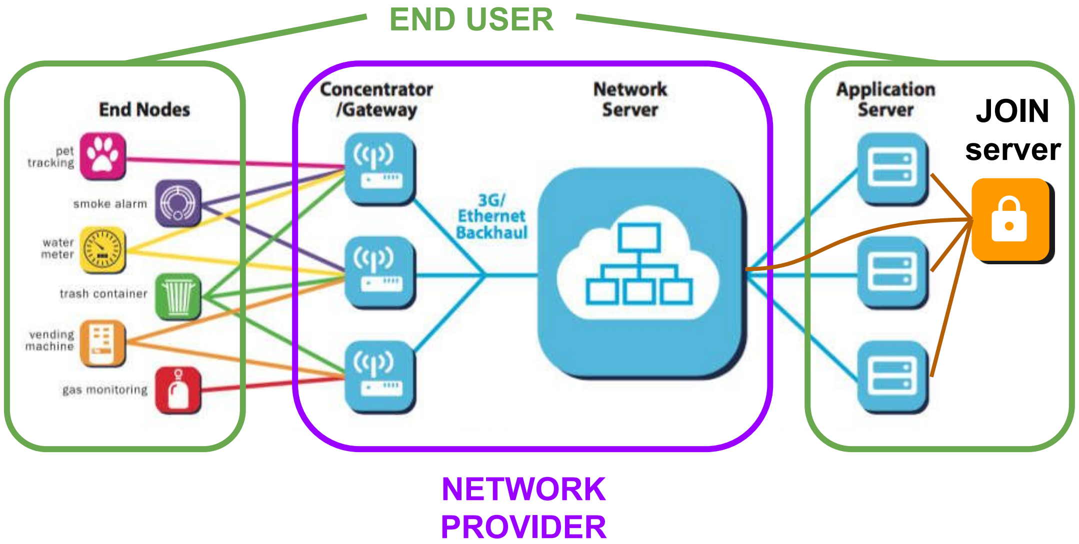
- Do not use the FPort field
  - Also: send fixed-size packets
- Inject random fake packets to hide event timing and total number of events
  - Tradeoff: better hiding of information ⇒ more extra packets ⇒ higher energy consumption
- If not time-critical: bundle events and send at regular intervals

LoRaWAN 1.1

Newer == Better ?

LoRaWAN 1.1 Security: Incomplete Standard

- Key distribution: “unique keys” (impossible to enforce)
- Network provider:
  - **OTAA: Who generates session keys?**
  - **How does it interact with application?**
    - Providers SHALL use subdomains under JOINEUIS.LORA-ALLIANCE.ORG and NETIDS.LORA-ALLIANCE.ORG ⇒ single point of failure for **ALL** LoRaWAN infrastructure: just DDoS the DNS



LoRaWAN 1.1 Security: Replay attacks **fixed :-)**

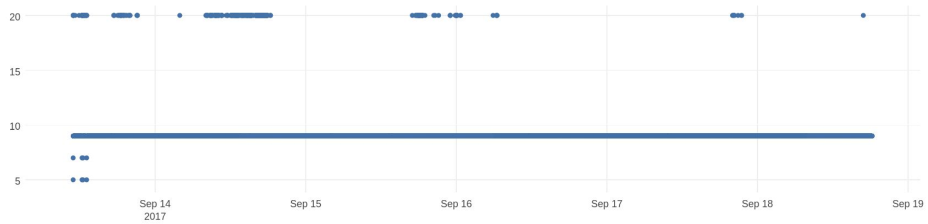
- Counter resets forbidden – counters *really* always increase (mod 2^{16})
- OTAA: JOIN requests have a counter, frame counter starts from 0 each session
  - JOIN counter stored in non-volatile memory – not that often ⇒ okay
  - Frame counters don't have to be stored
- Frame counter gap definition removed – no counter-related DoS

LoRaWAN 1.1 Security: Plain text fields

- **FPort still not encrypted :-(**

Conclusion

- LoRaWAN
  - Key distribution and derivation
  - Replay attacks and DoS
  - FPort leaks application information



- LoRaWAN 1.1
  - Key distribution: “unique” but no way to enforce
  - Key derivation: defined :-) but complex :-(⇒ actual use is questionable
  - Interaction between network and application: overdefined, with new SPOFs
  - Replay attacks: fixed
  - **FPort still leaks application information**
- Event based devices
  - Packet presence = information
  - Tradeoff between energy/security

Appendix

Types of use cases

periodic communication

- sensors: report value of X every Y seconds
- usual security mechanisms (e.g encryption) sufficient

1

event-based

- send packet whenever something happens
- **existence of packet reveals information**
- the event must be masked somehow

2

LoRaWAN 1.1: Frame counters

- Resets are forbidden now!
- Solves replay problems
- Unclear: How / where do ABP devices permanently store counter?
- Insecurity: fixed
- Implementation for ABP: unclear

LoRaWAN 1.1: Counter gap

- Definition removed
- Fixed: DoS prevented

LoRaWAN 1.1: FPort

- Was: Plain text field
- Is: Plain text field
- Problem not fixed :(

LoRaWAN 1.1: LoRa Alliance DNS

- Providers SHALL use JOINEUIS.LORA-ALLIANCE.ORG and NETIDS.LORA-ALLIANCE.ORG
- NEW External single point of failure in backend architecture

LoRaWAN 1.1: Gateway communication

- Previously: undefined
- Previous reality: JSON via UDP (no authentication, no confidentiality)
  - Swisscom: IPSec
- New: “via secured IP connections”

LoRaWAN 1.1: Backend communication

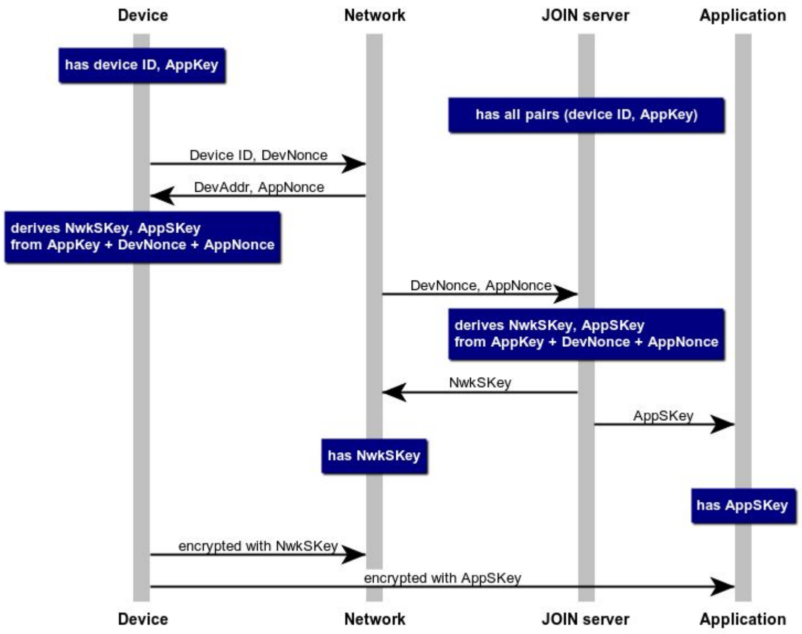
- Previously: undefined
- New: Symmetric keys usage
- Public key crypto not mentioned
- Conclusion: better, but not optimal

LoRaWAN 1.1: Key usage

- Previous: undefined
- New: unique key per device
- Insecurity fixed (?)
  - How to enforce in reality?

LoRaWAN 1.1: Key derivation

- Objective: have 2 symmetric keys
  - User data
  - Network commands
- Previous: Network provider has both keys
- New: User could run “join server”
- Insecurity fixed (?)
  - High degree of complexity
  - Very unlikely to happen in reality



LoRaWAN 1.1: Overdefinition

- Previous: no backend definition
- New: partially detailed definition
  - HTTP transport, POST-based, JSON
- Questionable improvement

Conclusion: LoRaWAN Security

- **incomplete standard**:
  - key distribution
  - network provider not defined:
    - how does it get keys?
    - how does it interact with application?
  - ABP-only devices
- **replay attacks:**
  - counter resets + frame counter gap ⇒ **DoS**
  - JOIN replay ⇒ **DoS**
- **plain text fields**:
  - permanent-ish device ID; device type might be leaked
  - **FPort leaks application information**

Conclusion: LoRaWAN 1.1 Security

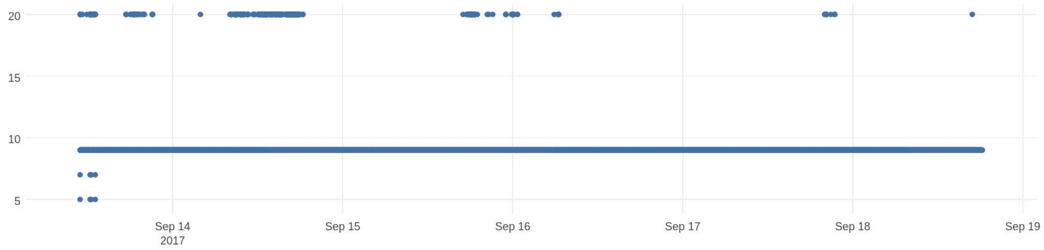
- **incomplete standard**:
    - key distribution: “unique keys” (impossible to enforce)
    - network provider:
        - how does it get keys? – defined :-) but complex :-(⇒ actual use is questionable
        - how does it interact with application? – overdefined, with SPOFs :-(
    - ABP devices allowed, but harder to implement
- ~~**replay attacks:**~~

  fixed :-)

- **plain text fields**
    - permanent-ish device ID; device type might be leaked
    - **FPort still leaks application information**

Conclusion: Event-based Devices

- Do not leak more than necessary
  - Do not use FPort…



  - Hide packet length
- Inject fake packets to hide real events
  - Tradeoff: attacker information gain vs. # of extra packets
- If not time-critical: bundle and send at regular (or random) intervals