FLORIDA TECH
FLORIDA'S STEM UNIVERSITY™

Department of Computer Science

CSE 4820: Wireless and Mobile Security

19. LoRaWAN Security

Dr. Abdullah Aydeger
Location: Harris Inst #310
Email: aaydeger@fit.edu

Outline

LoRaWAN

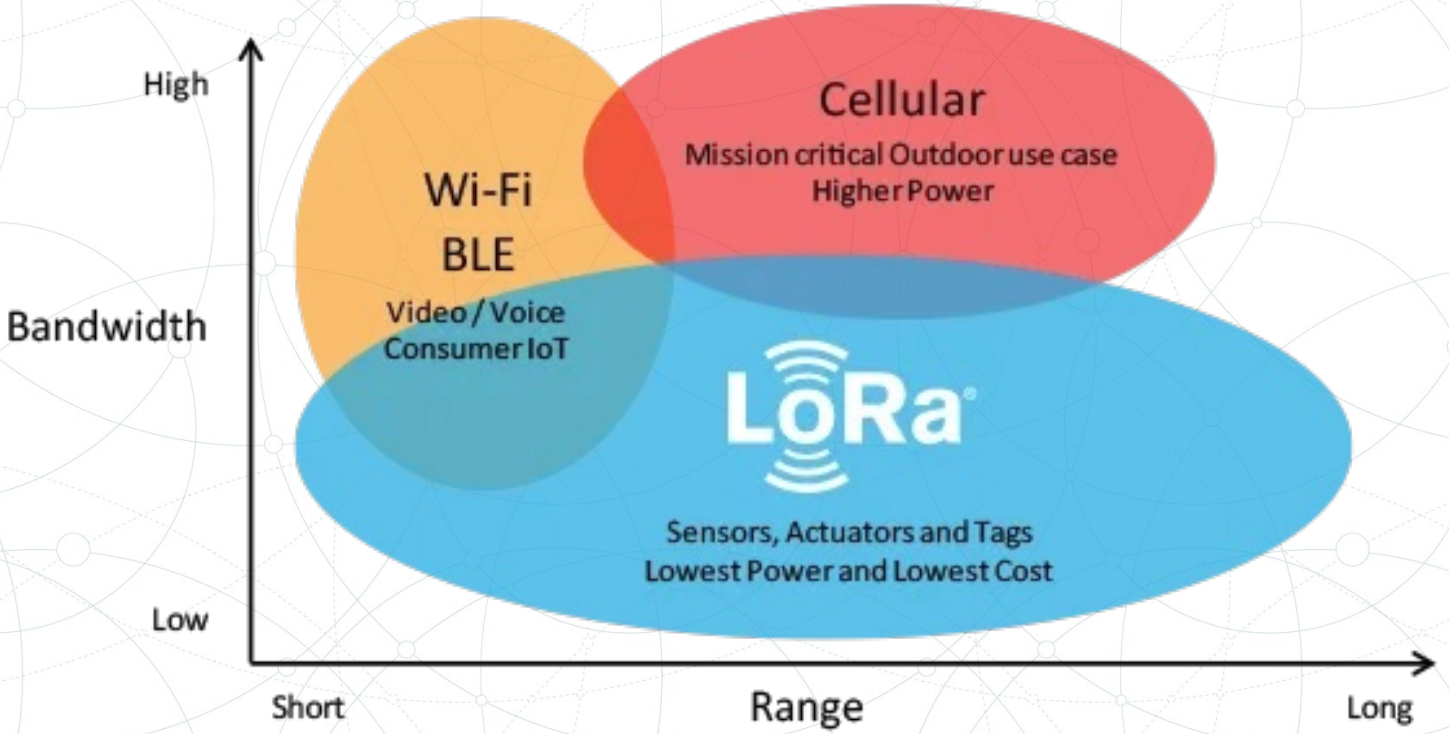
- Message Types
- Adaptive Data Rates
- Security

Recall: LoRaWAN

- Long Range (LoRa) Wide Area Network (WAN)
  - "A Low Power, Wide Area (LPWA) networking protocol designed to wirelessly connect battery operated 'things' to the internet in regional, national or global networks,
  - And support targets key Internet of Things (IoT) requirements such as bi-directional communication, end-to-end security, mobility and localization services"

Recall: LoRaWAN as Wireless Standard

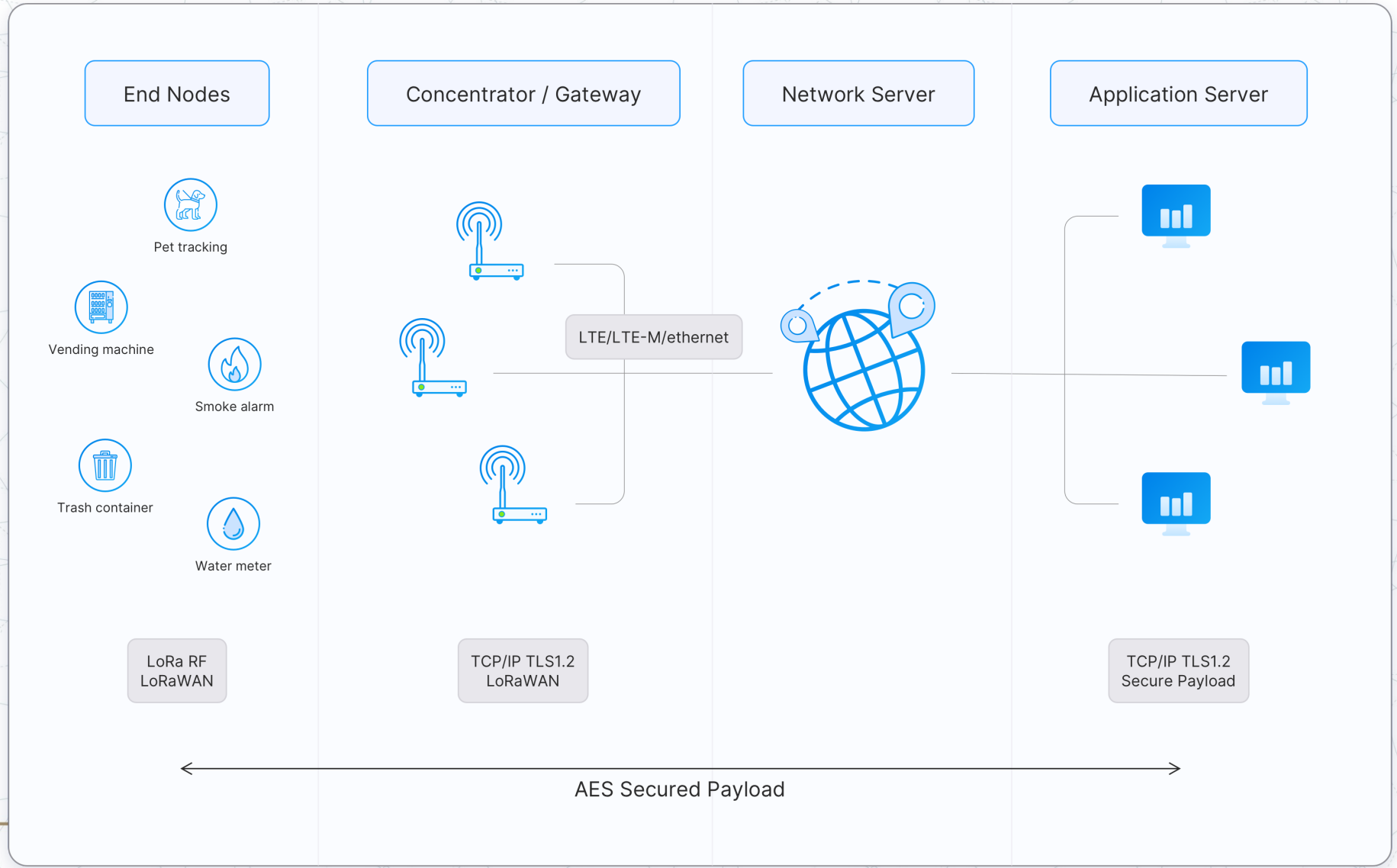
- Suitable for transmitting small size payloads (like sensor data) over long distances
  - Greater communication range with low bandwidths than other competing wireless data transmission technologies



Recall: LoRaWAN Network Devices

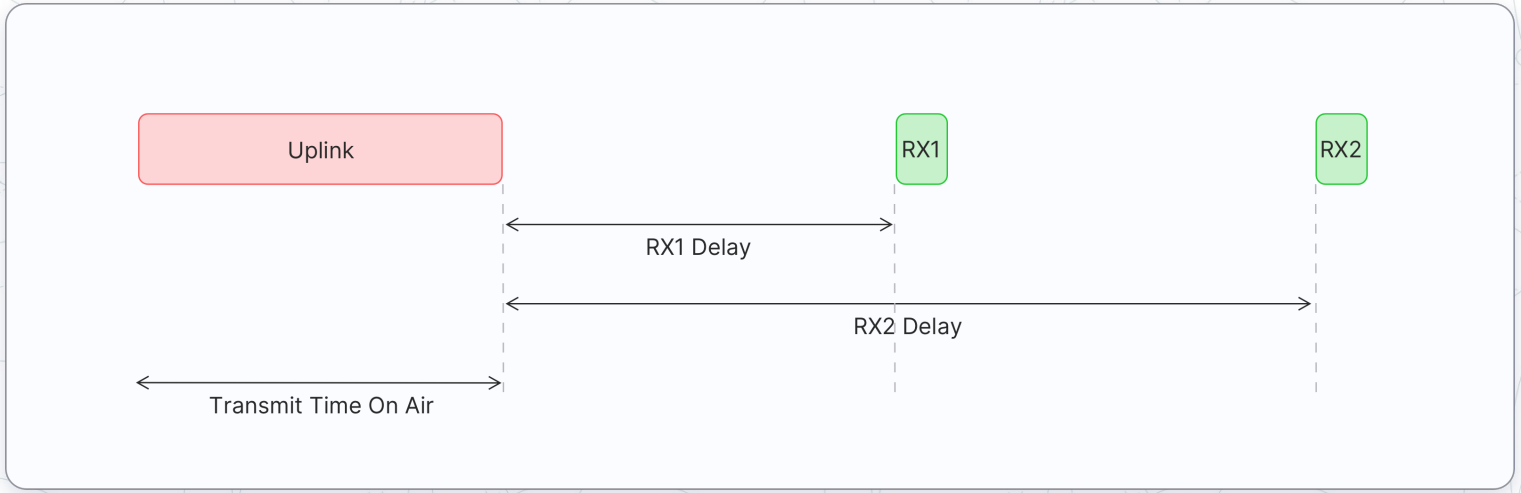
- LoRaWAN Device Types:
  - End Devices
  - Gateways
  - Network Server
  - Application servers
  - Join Server

Recall: LoRaWAN Architecture



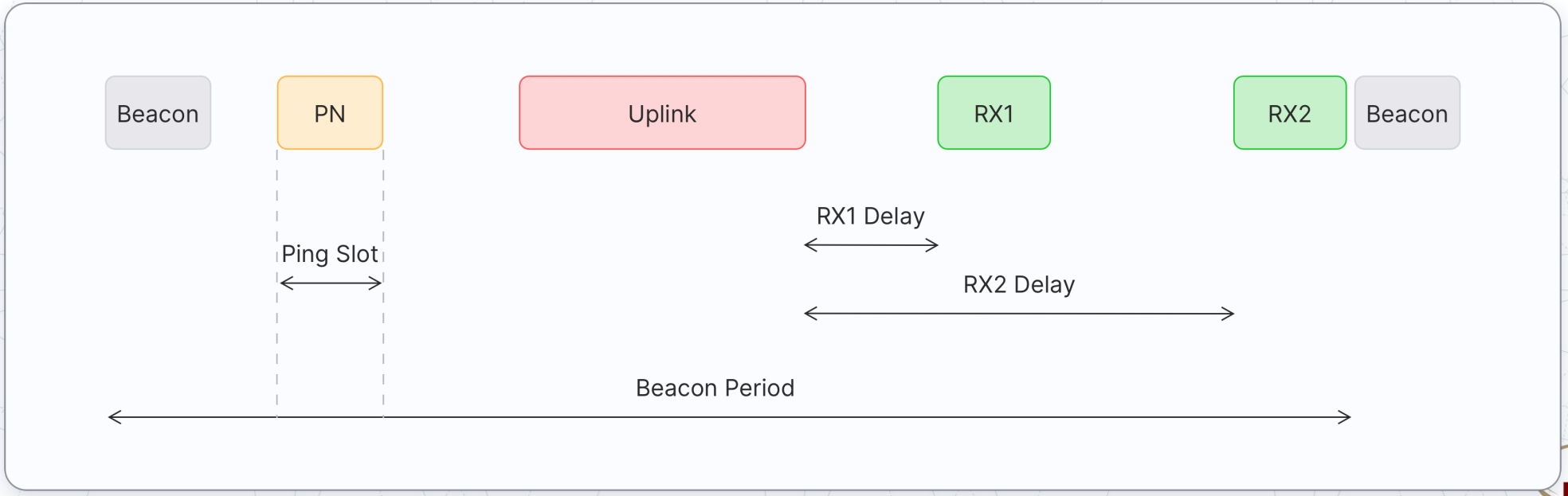
Recall: LoRaWAN Device Class A

- Communication is always initiated by the end-device
- A device can send an uplink message at any time
  - Once the uplink transmission is completed the device opens two short receive (downlink) windows



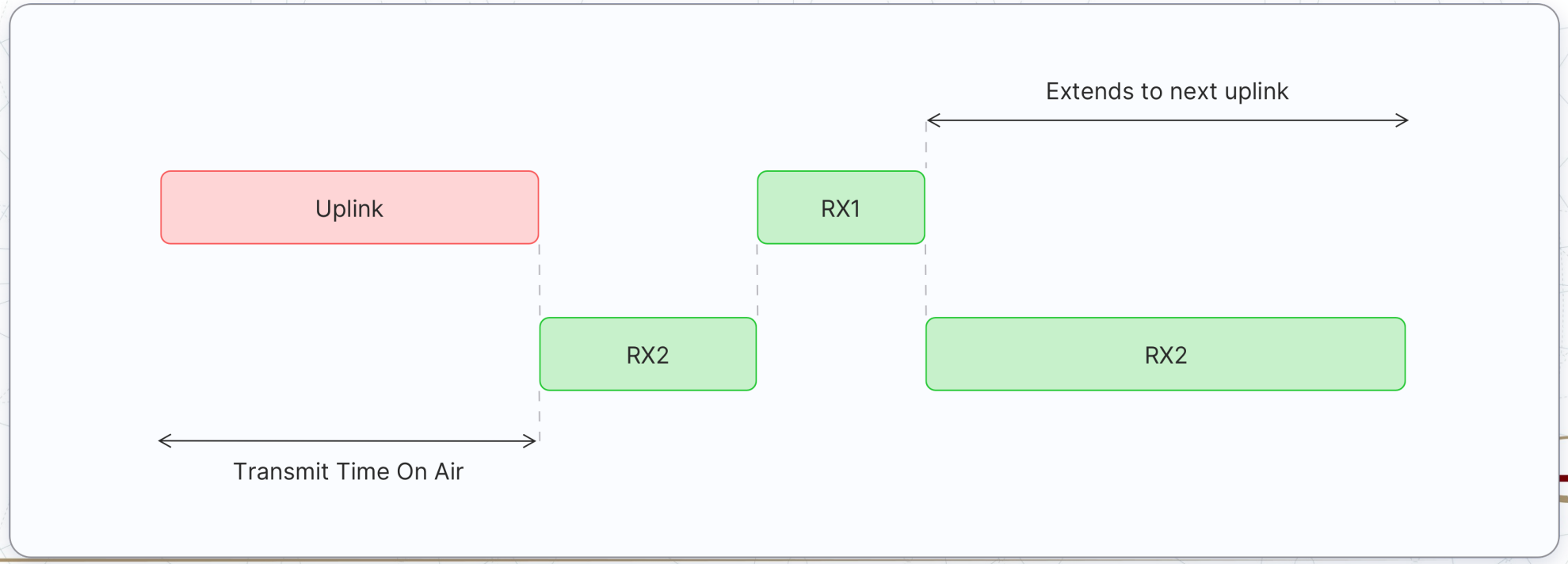
Recall: LoRaWAN Device Class B

- Open scheduled receive windows for receiving downlink messages from the network server



Recall: LoRaWAN Device Class C

- Extend Class A by keeping the receive windows open unless they are transmitting



LoRaWAN MAC Message Types

LoRaWAN 1.0.x	LoRaWAN 1.1	Description
Join-request	Join-request	An uplink message, used by the over-the-air activation (OTAA) procedure
Join-accept	Join-accept	A downlink message, used by the over-the-air activation (OTAA) procedure
Unconfirmed Data Up	Unconfirmed Data Up	An uplink data frame, confirmation is not required
Unconfirmed Data Down	Unconfirmed Data Down	A downlink data frame, confirmation is not required
Confirmed Data Up	Confirmed Data Up	An uplink data frame, confirmation is requested
Confirmed Data Down	Confirmed Data Down	A downlink data frame, confirmation is requested
RFU	Rejoin-request	1.0.x - Reserved for Future Usage1.1 - Uplink over-the-air activation (OTAA) Rejoin-request
Proprietary	Proprietary	Used to implement non-standard message formats

LoRaWAN: Join-Request

- The Join-request message is always initiated by an end device and sent to the Network Server
- In LoRaWAN versions earlier than 1.0.4 the Join-request message is forwarded by the Network Server to the Application Server
  - In LoRaWAN 1.1 and 1.0.4+, the Network Server forwards the Join-request message to the device's Join Server
- The Join-request message is not encrypted

LoRaWAN: Join-Accept

- In LoRaWAN versions earlier than 1.0.4 the Join-accept message is generated by the Application Server
  - In LoRaWAN 1.1 and 1.0.4+ the Join-accept message is generated by the Join Server
  - In both cases the message passes through the Network Server
- Then the Network Server routes the Join-accept message to the correct end-device

LoRaWAN: Join-Accept

- The Join-accept message is encrypted as follows
  - In LoRaWAN 1.0, the Join-accept message is encrypted with the AppKey
  - In LoRaWAN 1.1, the Join-accept message is encrypted with different keys:

If triggered by	Encryption Key
Join-request	NwkKey
Rejoin-request type 0, 1, and 2	JSEncKey

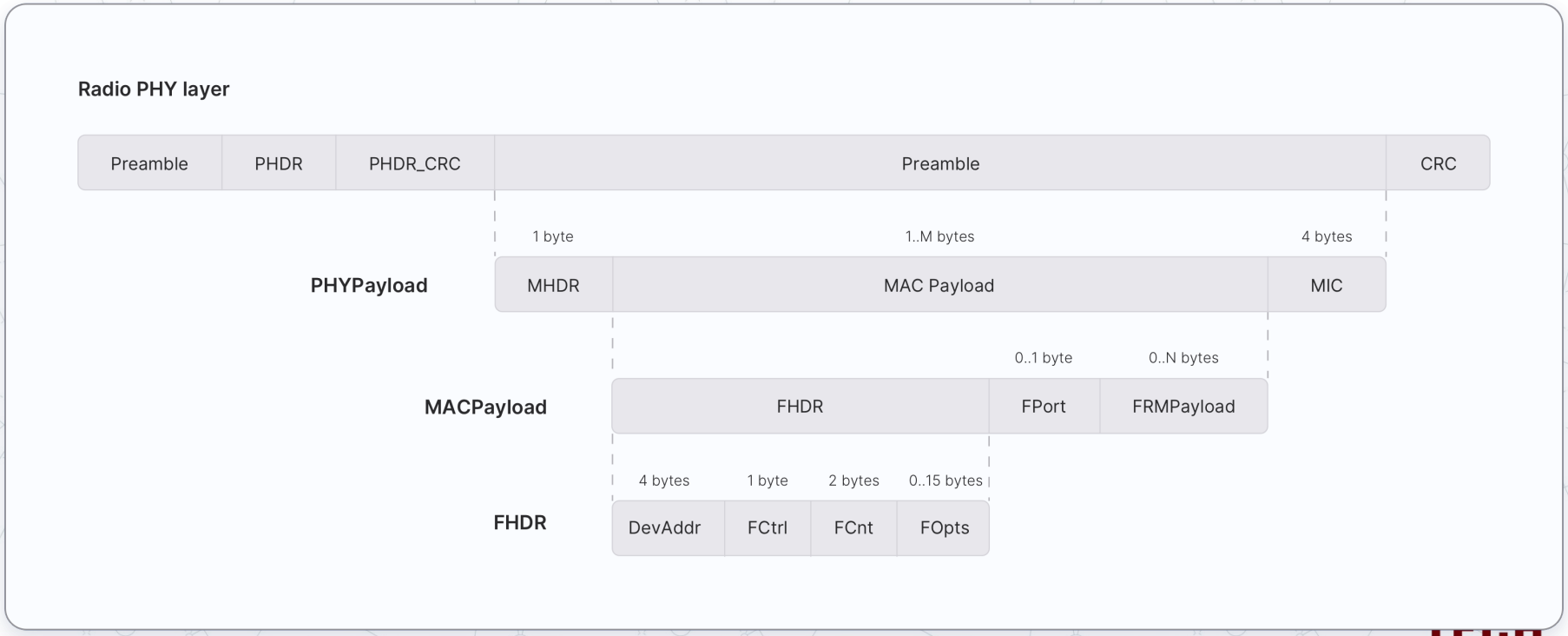
Message Integrity Code (MIC)

- The Message Integrity Code (MIC) ensures the integrity and authenticity of a message
- The message integrity code is calculated over all the fields in the message and then added to the message itself

Message Type	Fields
Join-request	MHDR | JoinEUI | DevEUI | DevNonce
Join-accept	MHDR | JoinNonce | NetID | DevAddr | DLSettings | RxDelay | CFList
Rejoin-request Type 0 and 2	MHDR | Rejoin Type | NetID | DevEUI | RJcount0
Rejoin-request Type 1	MHDR | Rejoin Type | JoinEUI | DevEUI | RJcount1
Data messages (up and down)	MHDR | FHDR | FPort | FRMPayload

LoRaWAN Data Messages

- Are used to transport both MAC commands and application data which can be combined together in a single message
- Data messages can be confirmed or unconfirmed



LoRaWAN End Device Activation

- Every end device must be registered with a network before sending and receiving messages
  - This procedure is known as activation
- There are two activation methods available:
  - Over-The-Air-Activation (OTAA)
  - Activation By Personalization (ABP)

OTAA

- The most secure and recommended activation method for end devices
- Devices perform a join procedure with the network, during which a dynamic device address is assigned and security keys are negotiated with the device
- The join procedure for LoRaWAN 1.0.x and 1.1 is slightly different

OTAA in LoRaWAN 1.1

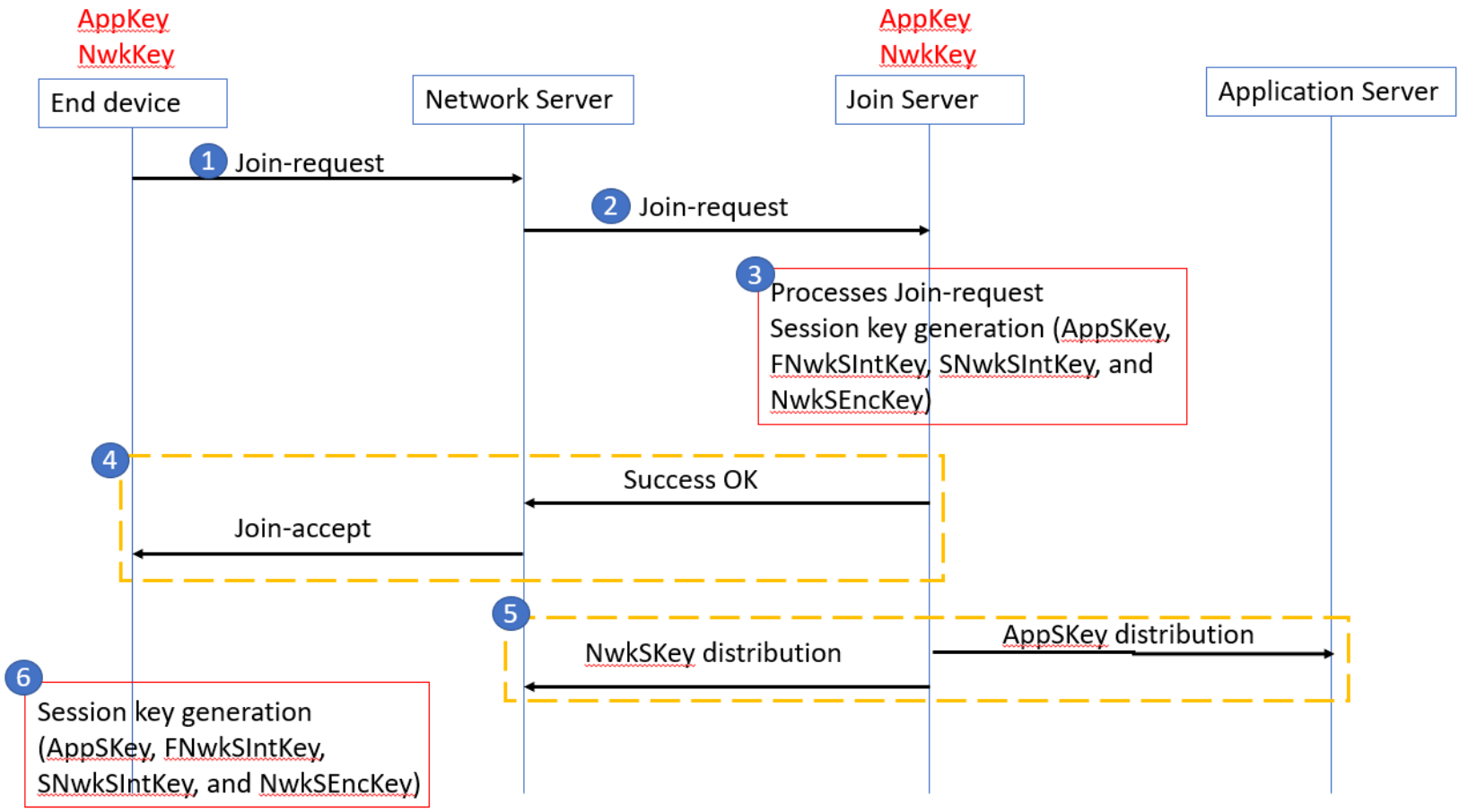
- The join procedure requires two MAC messages to be exchanged between the end device and the Join Server:
  - Join-request - from end device to the Join Server
  - Join-accept - from Join Server to the end device
- Before activation, the JoinEUI, DevEUI, AppKey, and NwkKey should be stored in the end device

OTAA in LoRaWAN 1.1

- The AppKey and NwkKey are AES-128 bit secret keys known as root keys
- The matching AppKey, NwkKey, and DevEUI should be provisioned onto the Join Server that will assist in the processing of the join procedure and session key derivation
- The JoinEUI and DevEUI are not secret and visible to everyone
- **Note**: The AppKey and NwkKey are never sent over the network

OTAA

- The join procedure is always initiated by the end device
- The end device sends the Join-request message to the network that is going to be joined



OTAA: Join Request #1

8 bytes	8 bytes	2 bytes
JoinEUI	DevEUI	DevNonce

- The Join-request message consists of the following fields
  - JoinEUI – a 8 bytes global application identifier that uniquely identifies the Join Server that can assist in the processing of the Join-request and derivation of the session keys
  - DevEUI – a 8 bytes global device identifier that uniquely identifies the end-device
  - DevNonce – a 2-byte counter, starting at 0 when the device is initially powered up and incremented with every Join-request
    - The DevNonce value is used to prevent replay attacks

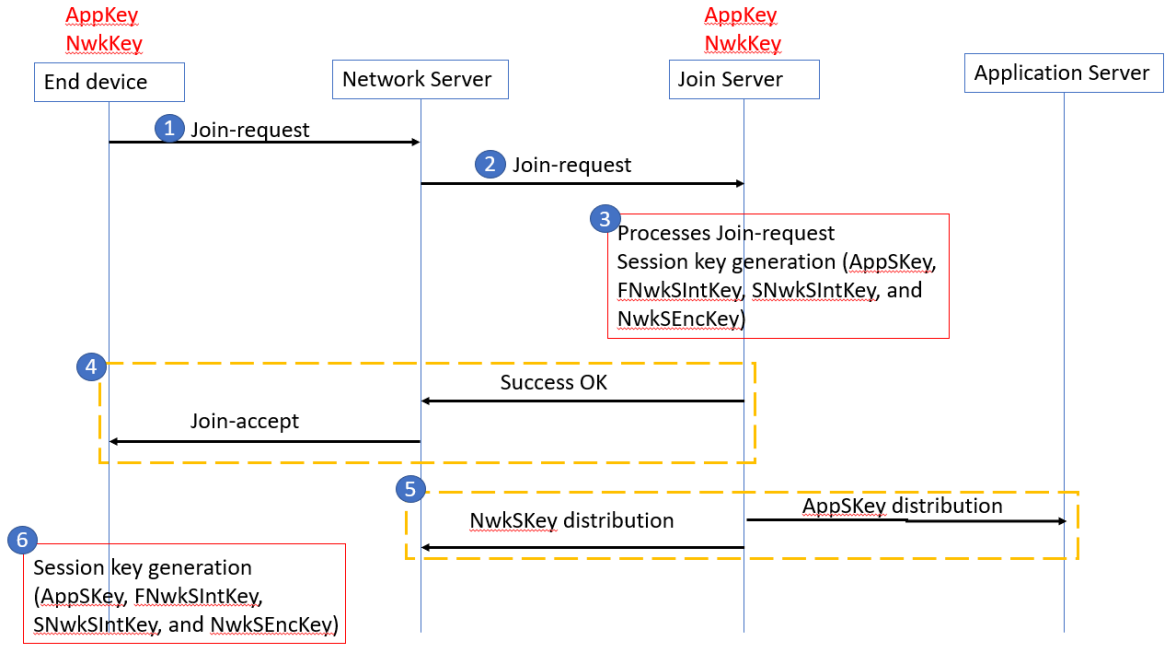
OTAA: Join Request #1

8 bytes	8 bytes	2 bytes
JoinEUI	DevEUI	DevNonce

- MIC is calculated over all the fields in the Join-request message using the NwkKey
  - The calculated MIC is then added to the Join-request message
- **Note**: The NwkKey is not sent with the Join-request message, and the Join-request message is not encrypted but sent as plain text
- The Join-request message can be transmitted using any data rate and using one of the region-specific join channels and travels through one or more gateways to the Network Server
- **Note**: No response is given to the end-device if the Join-request is not accepted

OTAA Steps

- Step 2:
  - The Network Server forwards the Join-request message to the corresponding Join Server
- Step 3:
  - The Join Server processes the Join-request message
  - The Join Server will generate all the session keys (AppSKey, FNwkSIntKey, SNwkSIntKey, and NwkSEncKey) if the end-device is permitted to join the network



OTAA Steps #4

- If the above step 2-3 gets success, the Network Server generates the Join-accept message
- The Join-accept message consists of the following fields:

1 byte	3 bytes	4 bytes	1 bytes	1 bytes	16 bytes
JoinNonce	NetID	DevAddr	DLSettings	RXDelay	CFList

  - JoinNonce – a device specific counter value provided by the Join Server and used by the end device to derive the session keys, FNwkSIntKey, SNwkSIntKey, NwkSEncKey, and AppSKey

OTAA Steps: #4

1 byte	3 bytes	4 bytes	1 bytes	1 bytes	16 bytes
JoinNonce	NetID	DevAddr	DLSettings	RXDelay	CFList

- NetID – a 24-bit unique network identifier
- DevAddr – a 32-bit device address assigned by the Network Server to identify the end device within the current network
- DLSettings – a 1-byte field consisting of downlink settings which the end device should use
- RxDelay – contains delay between TX and RX
- CFList – an optional list of channel frequencies for the network the end-device is joining
  - These frequencies are region-specific

OTAA Steps #4

- The Message Integrity Code (MIC) is calculated over all the fields in the Join-accept message using NwkKey (for LoRaWAN 1.0 devices) or JSIntKey (for LoRaWAN 1.1 devices)
  - The calculated MIC is then added to the Join-accept message
- The Join-accept message itself is then encrypted with the NwkKey
  - The Network Server uses an AES operation in ECB mode to encrypt the join-accept message

OTAA Steps #4

- The Join-accept message is encrypted with the NwkKey (if triggered by Join-request) or JSEncKey (if triggered by Rejoin-request)
- Then the Network Server sends the encrypted Join-accept message back to the end device as a normal downlink
- **Note**: No response is given to the end-device if the Join-request message is not accepted by the Network Server

OTAA Step #5-6

- Step 5:
  - The Join Server sends the AppSKey to the Application Server and the three network session keys (FNwkSIntKey, SNwkSIntKey, and NwkSEncKey) to the Network Server
- Step 6
  - The end-device decrypts the Join-accept message using AES operation
  - The end device uses AppKey, NwkKey, and JoinNonce to generate session keys

OTAA Step #6

- For LoRaWAN 1.1 devices,
  - AppSKey is derived from AppKey
  - FNwkSIntKey, SNwkSIntKey, and NwkSEncKey are derived from the NwkKey
- The end device is now activated on the Network

OTAA Step #6

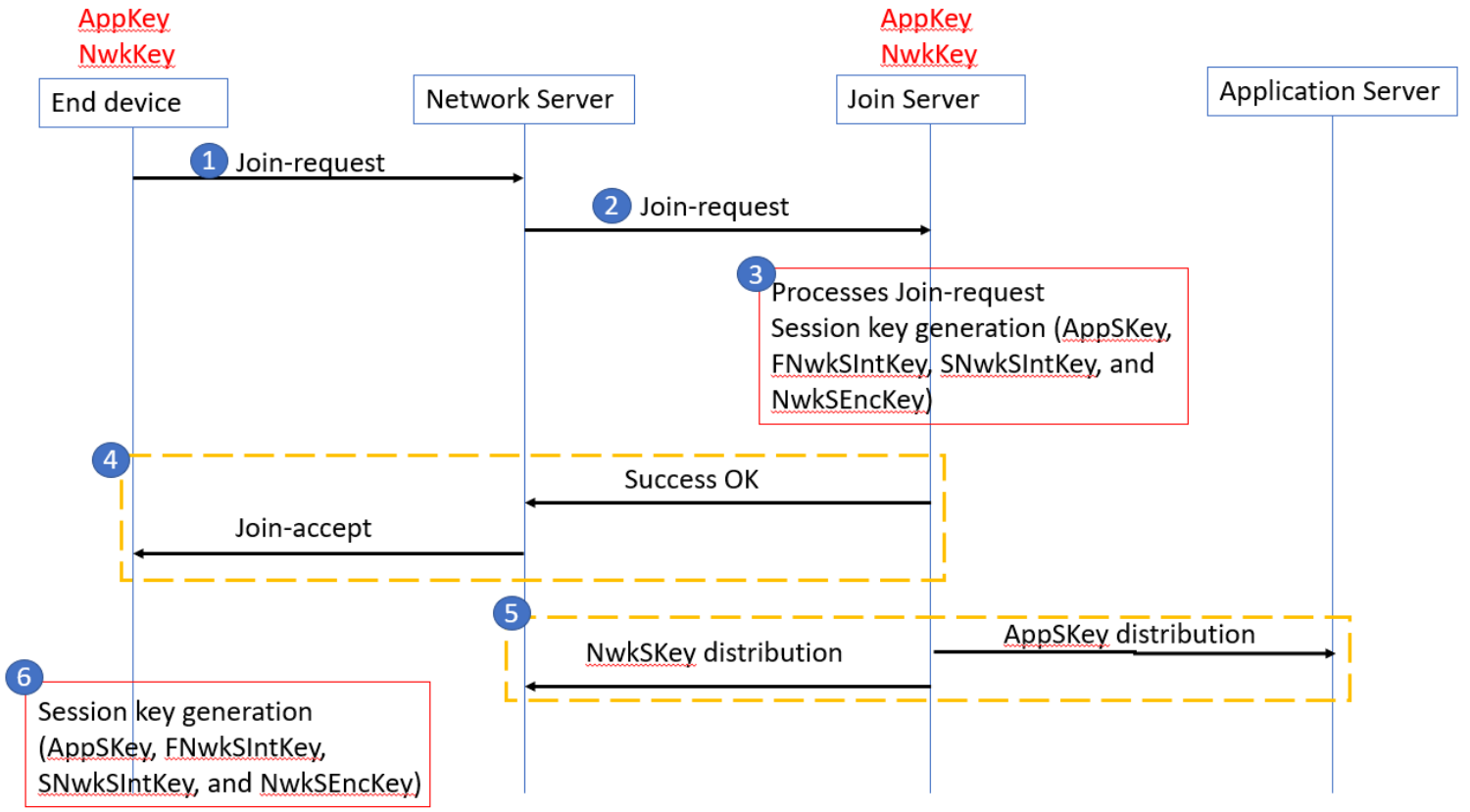
- After activation, the following additional information is stored in the end device:
- DevAddr:
  - 32-bit device address assigned by the Network Server to identify the end device within the current network
- FNwkSIntKey:
  - Network session key that is used by the end device to calculate the MIC (partially) of all uplink data messages for ensuring message integrity

OTAA Step #6

- SNwkSIntKey:
  - Network session key that is used by the end device to calculate the MIC (partially) of all uplink data messagse and calculate the MIC of all downlink data messages for ensuring message integrity
- NwkSEncKey:
  - Network session key that is used to encrypt and decrypt the payloads with MAC commands of the uplink and downlink data messages for ensuring message confidentiality

OTAA Step #6

- AppSKey:
  - Session key used by both the Application Server and the end device to encrypt and decrypt the application data in the data messages for ensuring message confidentiality

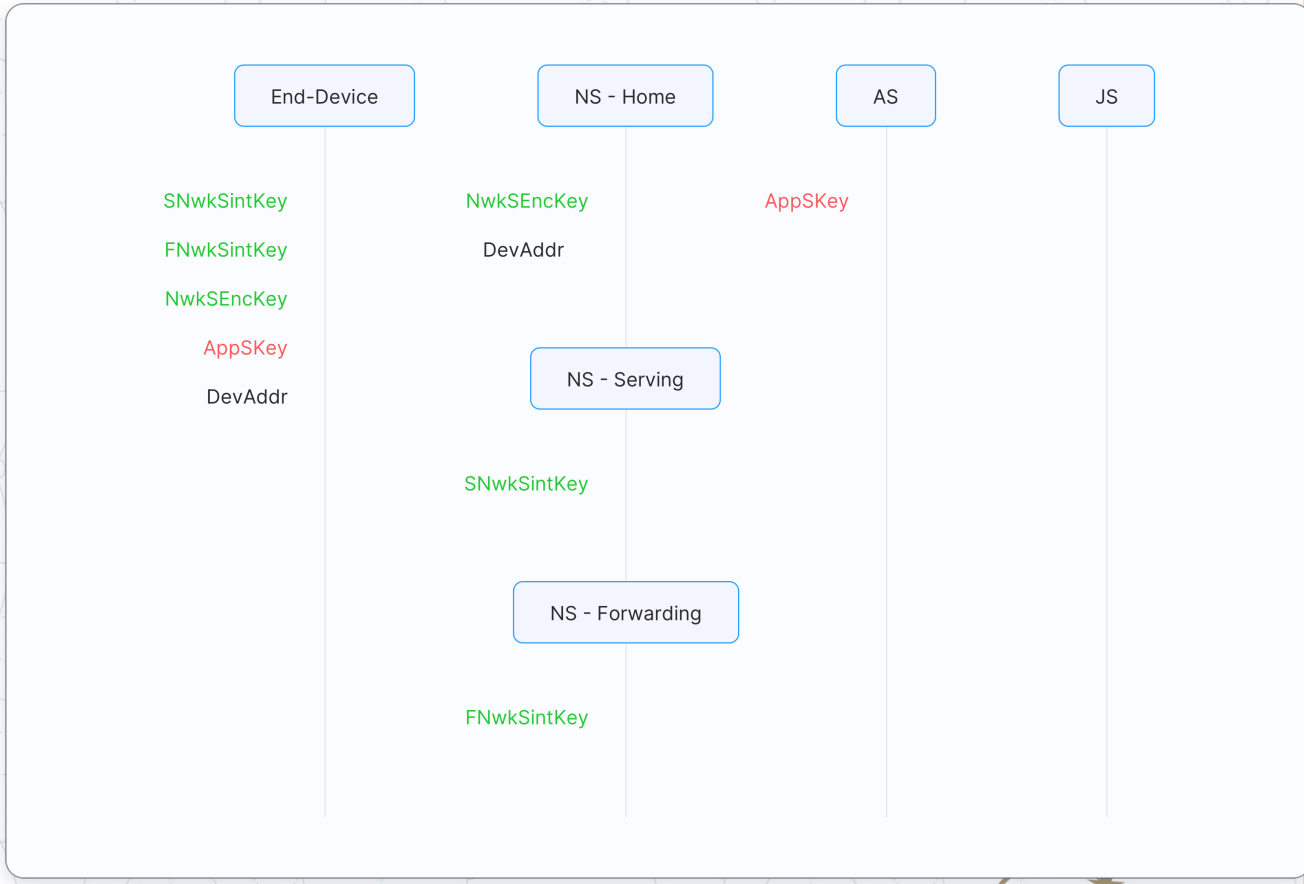


Activation By Personalization (ABP)

- ABP directly ties an end-device to a pre-selected network, bypassing the over-the-air-activation procedure
  - Requires hardcoding the device address as well as the security keys in the device
- ABP is the less secure activation method, and also has the downside that devices can not switch network providers without manually changing keys in the device
  - A Join Server is not involved in the ABP process
- An end device activated using the ABP method can only work with a single network and keeps the same security session for its entire lifetime

ABP in LoRaWAN 1.1

- The DevAddr and the four-session keys FNwkSIntKey, SNwkSIntKey, NwkSEncKey, and AppSKey are directly stored into the end device instead of the DevEUI, JoinEUI, AppKey, and NwkKey
- The same DevAddr, FNwkSIntKey, SNwkSIntKey, and NwkSEncKey should be stored in the Network Server and the and AppSKey should be stored in the Application Server



LoRaWAN Data Rates

- In addition to frequency hopping, all communication packets between end-devices and gateways also include a variable 'Data rate' (DR) setting
- The selection of the DR allows a dynamic trade-off between communication range and message duration
- Also, due to the spread spectrum technology, communications with different DRs do not interfere with each other and create a set of virtual 'code' channels increasing the capacity of the gateway

LoRaWAN Data Rates

- To maximize both battery life of the end-devices and overall network capacity, the LoRaWAN network server manages the DR setting and RF output power for each end-device individually by means of an Adaptive Data Rate (ADR) scheme
- LoRaWAN baud rates range from 0.3 kbps to 50 kbps

LoRaWAN Adaptive Data Rate

- Adaptive Data Rate (ADR) is a mechanism for optimizing data rates, airtime and energy consumption in the network
- The ADR mechanism controls the following transmission parameters of an end device:
  - Spreading factor
  - Bandwidth
  - Transmission power

LoRaWAN ADR

- ADR can optimize device power consumption while ensuring that messages are still received at gateways
- When ADR is in use, the network server will indicate to the end device that it should reduce transmission power or increase data rate
- End devices which are close to gateways should use a lower spreading factor and higher data rate, while devices further away should use a high spreading factor because they need a higher link budget

LoRaWAN ADR

- ADR should be enabled whenever an end device has sufficiently stable RF conditions
  - This means that it can generally be enabled for static devices
- If the static end device can determine that RF conditions are unstable (for example, when a car is parked on top of a parking sensor), ADR should (temporarily) be disabled
- Mobile end devices should be able to detect when they are stationary for a longer times, and enable ADR during those times
- End devices decide if ADR should be used or not, not the application or the network

LoRaWAN Security

- LoRaWAN 1.0 specifies a number of security keys: NwkSKey, AppSKey and AppKey
- All keys have a length of 128 bits
- The algorithm used for this is AES-128, similar to the algorithm used in the 802.15.4 standard

LoRaWAN Session Keys

- When a device joins the network (this is called a join or activation), an application session key AppSKey and a network session key NwkSKey are generated
  - The NwkSKey is shared with the network, while the AppSKey is kept private
  - These session keys will be used for the duration of the session

LoRaWAN Session Keys

- The Network Session Key (NwkSKey) is used for interaction between the Node and the Network Server
- This key is used to validate the integrity of each message by its Message Integrity Code (MIC check)
- This MIC is similar to a checksum, except that it prevents intentional tampering with a message

LoRaWAN Session Keys

- For this MIC, LoRaWAN uses AES-CMAC
  - This validation is also used to map a non-unique device address (DevAddr) to a unique DevEUI and AppEUI

The MIC of the Join-accept message is computed using the NwkKey:

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

LoRaWAN Session Keys

- The Application Session Key (AppSKey) is used for encryption and decryption of the payload
- The payload is fully encrypted between the Node and the Handler / Application Server component of The Things Network, which you can run on your own server
  - This means that nobody except you is able to read the contents of messages you send or receive

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

LoRaWAN Session Keys

- These two session keys (NwkSKey and AppSKey) are unique per device, per session
- If you dynamically activate your device (OTAA), these keys are re-generated on every activation
- If you statically activate your device (ABP), these keys stay the same until you change them

```
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)
```

LoRaWAN Application Keys

- The application key (AppKey) is only known by the device and by the application
- Dynamically activated devices (OTAA) use the Application Key (AppKey) to derive the two session keys during the activation procedure
- In The Things Network you can have a default AppKey which will be used to activate all devices or customize the AppKey per device

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

LoRaWAN Security

- By providing these two keys, it becomes possible to implement 'multi-tenant' shared networks without the network operator having visibility of the users payload data
- The keys can be Activated By Personalisation (ABP) on the production line or during commissioning, or can be Over-The-Air Activated (OTAA) in the field
  - OTAA allows devices to be re-keyed if necessary

LoRaWAN: Frame Counters

- Anyone will be able to capture and store messages
  - It's not possible to read these messages without the AppSKey, because they're encrypted
  - Nor is it possible to tamper with them without the NwkSKey, because this will make the MIC check fail
  - It is however possible to re-transmit the messages
- These so-called replay attacks can be detected and blocked using frame counters

LoRaWAN: Frame Counters

- When a device is activated, these frame counters (FCntUp and FCntDown) are both set to 0
- Every time the device transmits an uplink message, the FCntUp is incremented and every time the network sends a downlink message, the FCntDown is incremented
- If either the device or the network receives a message with a frame counter that is lower than the last one, the message is ignored

LoRaWAN: Frame Counters

- These frame counters reset to 0 every time the device restarts (when you flash the firmware or when you unplug it)
- As a result, The Things Network will block all messages from the device until the FCntUp becomes higher than the previous FCntUp
- Therefore, you should re-register your device in the backend every time you reset it

FLORIDA TECH

FLORIDA'S STEM UNIVERSITY™

Thank you. Questions?

Dr. Abdullah Aydeger