LoRaWAN1.1: Over The Air Activation

Department of Computer, Control, and Management Engineering Antonio Ruberti

Sapienza Università di Roma

Stefano Milani

End-device activation

To ensure a **secure communication** between the end-device and the Join/Application server, each device have to be personalized and activated.

There exists two types of activations:

- **Activation By Personalization (ABP)**: less secure, pre-determinate keys hardcoded into the device.

- **Over The Air Activation (OTAA)**: more secure, handshake between the two parties to agree on keys.

The focus of this presentation is on the OTAA (LoRaWAN1.1 specification)

OTAA: Needed data

Before activation

- **JoinEUI:** Global application ID that uniquely identifies the Join Server. It must be stored in the end-device before starting the OTAA procedure. 8 Bytes

- **DevEUI:** Global end-device ID that uniquely identifies the end-device. It must be stored in the end-device before starting the OTAA procedure. 8 Bytes

- **NwkKey/AppKey:** AES-128 root keys assigned to the end-device during fabrication. Secure provisioning, storage and using of these two keys on the device and on the backend are intrinsic to the overall security of the solution.

- **JSIntKwy:** used to calculate the MIC on the Rejoin-Request type 1 and Join-Accept message.

$$\boldsymbol{JSIntKey = aes128\_encrypt(NwkKey, 0x06|DevEUI|pad_{16})}$$

- **JSEncKey:** used to encrypt the Join-Accept triggered by a Rejoin-Request.

$$\boldsymbol{JSEncKey = aes128\_encrypt(NwkKey, 0x05|DevEUI|pad_{16})}$$

After activation

- **DevAddr:** 32-bits that identifies the end-device within the current network. It is allocated by the Network server of the end-device.

Address Prefix	Network address
[31 … 32 – N]	[31 – N … 0]

- **Forwarding Network Session Integrity Key (FNwkSIntKey):** used by the end-device to calculate the MIC of all the uplink messages.

- **Serving Session Encryption Key (SNwkSIntKey):** used by the end-device to check the MIC of all the downlink messages.

- **Network Session Encryption Key (NwkSEncKey):** used to encrypt uplink and downlink MAC commands transmitted as payload on port 0 or in the *Fopt* field.

- **Application Session Key (AppSKey):** specific for end-device, used by both partied to encrypt the payload field of the application–specific messages

OTAA: Procedure

Join-Request message

The Join procedure is started by the end-device by sending a **Join-Request message**, composed as follows:

JoinEUI	DevEUI	DevNonce
8 Bytes	8 Bytes	2 Bytes

The **DevNonce** is a counter starting from 0, incremented at every **Join-Request**. This message is **NOT** encrypted.
The **Message Integrity Code (MIC)** is computed as follows:

$$cmac = aes128\_cmac(NwkKey, MHDR|JoinEUI|\mathrm{DevEui}|\mathrm{DevNonce})$$
$$MIC = cmac[0..3]$$

The **MAC Header (MHDR)** is composed as follows:

MType	RFU	Major

The **Mtype** field specifies the type of the message:

Bits	Type	Bits	Type
000	Join-Request	100	Confirmed Data Up
001	Join-Accept	101	Confirmed Data Down
010	Unconfirmed Data Up	110	Rejoin-Request
011	Unconfirmed Data Down	111	Propetary

By default bits of **RFU** are set to 0 by the transmitter are shall be ignored by the receiver.
The **Major** field specifies the format of the message in the Join procedure.

Join-Accept message

The **Join-Accept** message contains:

JoinNonce	Home_NetID	DevAddr	DLSettings	RxDelay	CFList
3 Bytes	3 Bytes	4 Bytes	1 Byte	1 Byte	16 Bytes (optional)

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, AppSKey. The JoinNonce is incremented at each Join-Request.

The **DLSettings** contains the downlink configuration:

OptNeg	Rx1Doffsett	Rx3DataRate
1 bit	3 bits	4 bits

The **OptNeg** bit indicates whether the Join Server implements LoRaWAN1.0 (unset) or LoRaWAN1.1 (set).

The **MIC** is computed as follows:

$$\boldsymbol{cmac = aes128\_cmac(JSIntKey, JoinReqType|JoinEUI|DevNonce|MHDR|JoinNonce|NetID|DevAddr|DLSettings|RxDelay|CFList)}$$

$$\boldsymbol{MIC = cmac[0 \dots 3]}$$

To encrypt the message is used: the **NwkKey** in case of Join-Request, the **JSEncKey** in case of Rejoin-Request.

$$\boldsymbol{aes128\_decrypt(NwkKey\ /\ JSEncKey, JoinnNonce|NetID|DevAddr|DLSettings|RxDelay|CFList|MIC)}$$

The Join Server use the decrypt function to encrypt the message, in such a way the end-device need to implement only the encrypt function both to encrypt and decrypt the messages. The **Join-Accept** message can be 16 or 32 bytes long, in case of 32 bytes it is needed the ECB cipher mode of operation.

The **Join Request Type** or **Rejoin Request Type**:

Value	Type
0xFF	Join-Request
0x00	Rejoin-Request of type 0
0x01	Rejoin-Request of type 1
0x02	Rejoin-Request of type 2

Key Derivation

$$FNwkSIntKey = aes128\_encrypt(NwkKey, 0x01|JoinNonce|JoinEUI|DevNonce|pad_{16})$$

$$SNwkSIntKey = aes128\_encrypt(NwkKey, 0x03|JoinNonce|JoinEUI|DevNonce|pad_{16})$$

$$NwkSEncKey = aes128\_encrypt(NwkKey, 0x04|JoinNonce|JoinEUI|DevNonce|pad_{16})$$

$$AppSKey = aes128\_encrypt(NwkKey, 0x02|JoinNonce|JoinEUI|DevNonce|pad_{16})$$

OTAA: Rejoin

Rejoin-Request message

Once activated a device may periodically transmit a **Rejoin-Request** message, that gives the backend the possibility to initialize a new session for an end-device. The Network server can also use it to transmit a normal confirmed or unconfirmed downlink message.

There exists three types of **Rejoin-Request**:

- **Type0:** Contains NetID+DevEUI. Used to reset device context including all radio parameters.

- **Type1:** Contains JoinEUI+DevEUI. Equivalent to Type0, but it may be transmitted on top pf normal applicative traffic without disconnecting the device.

- **Type2:** Contains NetID+DevEUI. Used to rekey a device or change its DevAddr. Radio parameters remains unchanged.

Rejoin-Request of type 0 and 2

Type (0 or 2)	NetID	DevEUI	RJcount0
1 Bytes	3 Bytes	8 Bytes	2 Bytes

RJcount0 is a counter incremented with every type 0 or type 2 **Rejoin-Request** frame transmitted. Initialized to 0 at each **Join-Accept** message is successfully processed by the end-device.

If the RJcount0 **reaches** $2^{16} - 16$ the device shall stop transmitting **Rejoin-Request (type 0/2)** and go back to Join state.

$$\boldsymbol{cmac = aes128\_cmac(SNwkSIntKey, MHDR|RejoinType|NetID|DevEUI|RJcount0)}$$
$$\boldsymbol{MIC = cmac[0 \ldots 3]}$$

The **Rejoin-Request Type 0 or 2** message is NOT encrypted.

Rejoin-Request of type 1

Type (1)	JoinEUI	DevEUI	RJcount1
1 Bytes	8 Bytes	8 Bytes	2 Bytes

RJcount1 is a counter incremented with every type 1 **Rejoin-Request** frame transmitted. Initialized to 0 at each **Join-Accept** message is successfully processed by the end-device.

This counter shall never wrap around. The transmission periodicity of **Rejoin-Request Type 1** message shall be such that this wrap around cannot happen for a lifetime of the device for a given **JoinEUI** value.

$$\boldsymbol{cmac = aes128\_cmac(JSIntKey, MHDR|RejoinType|JoinEUI|DevEUI|RJcount1)}$$
$$\boldsymbol{MIC = cmac[0 \dots 3]}$$

The **Rejoin-Request Type 1** message is NOT encrypted.

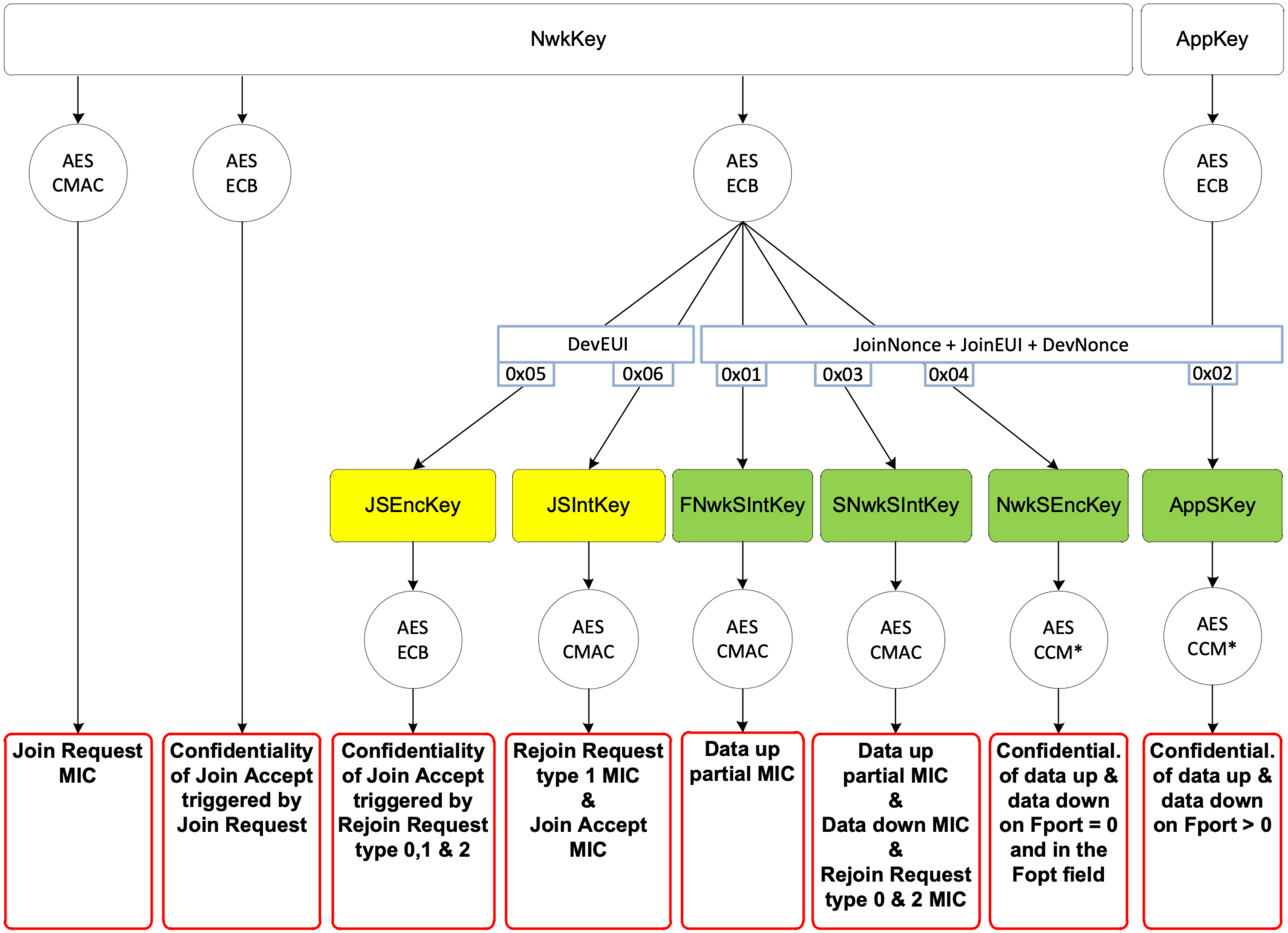
Rejoin-Request message processing

For all types of **Rejoin-Request** the Network Server may respond with:

- **Join-Accept** message if it wants to modify the device's network identity. In that case **RJcount0** or **RJcount1** replaces the DevNonce field in the key derivation process.
- A normal downlink frame optionally containing MAC commands. This downlink shall be sent on the same channel, with the same data rate and the same delay that the **Join-Accept** message it replace.

In most cases following a **Rejoin-Request Type 0 or 1** the Network Server will not respond.

LoRAWAN1.1 key derivation scheme



Reference

- Alliance, LoRa. "LoRaWAN 1.1 Specification." technical specification (2017).