

LoRa Roaming Tutorial

- Sandoche BALAKRICHENAN



LoRa Architecture Components



End-Device



Gateway



Network Server



Application Server



Join Server



DNS Server

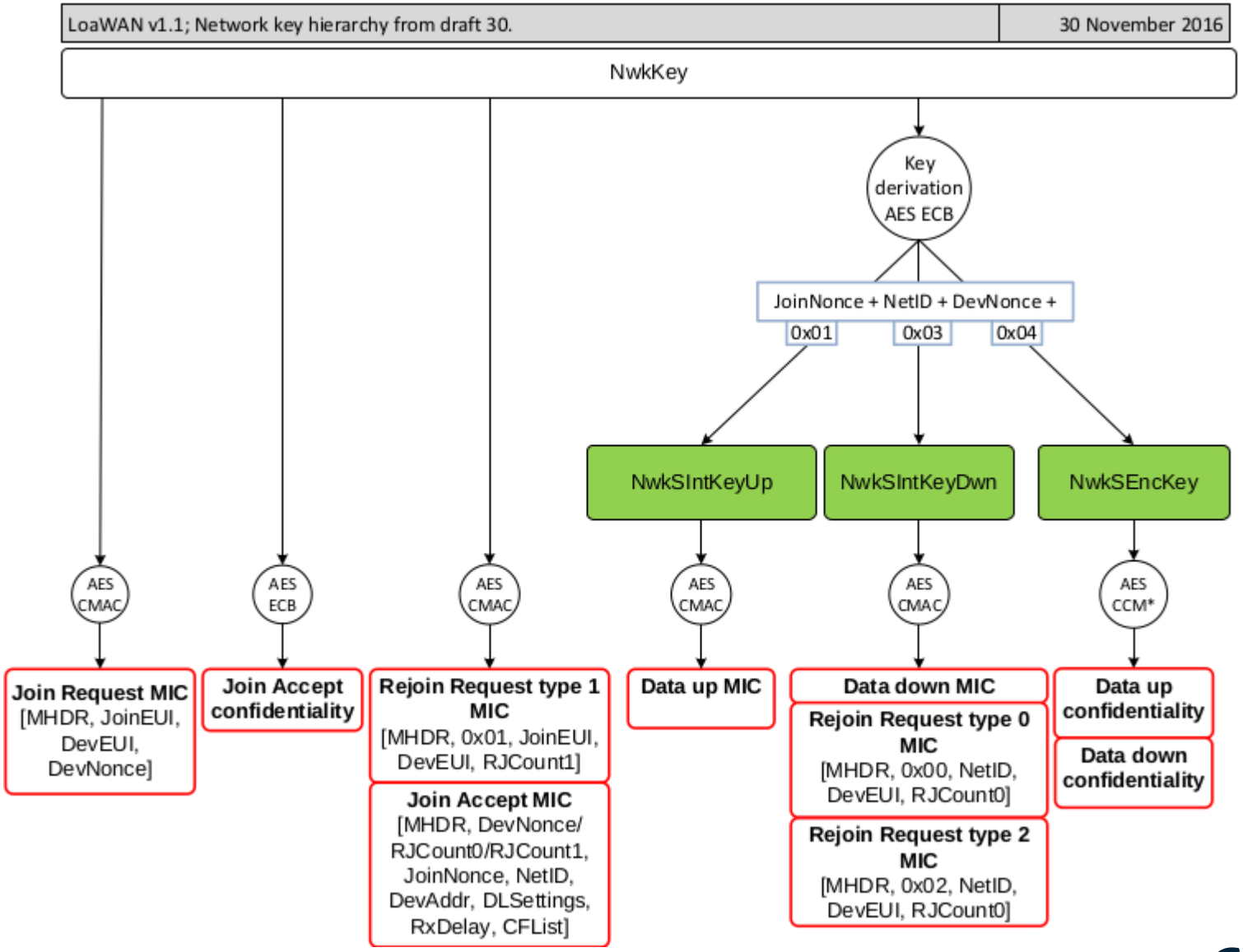
Explaining the different components

- End-device (ED) : *A sensor device attached to an object*
- Gateway (GW) : *Acts as bridge between the end-device and the 'NS'*
- Network Server (NS) : *Co-ordinates data between the GW and the 'AS'*
- Application Server (AS): *e.g. Cloud IoT application*
- Join Server (JS) : *Manages the OTA ED activation process*
- DNS Server (DS) : *Used for NetID & JS resolution*

Keys used in the LoRa data frame

	**AppKey**
	**NwKey**
	**AppSKey**
	**NwkSKey**
	**NwkSIntKeyUp**
	**NwkSIntKeyDwn**
	**NwkSEncKey**
	**Secret Key shared between JS and AS**

Network Key Hierarchy



Explaining the different parameters

- DevEUI : Identifies the MAC address in the sensor device. Provided by the device manufacturer. Unique (1 end-device = 1 DevEUI)
- AppEUI : Identifies the AS. Furnished by the Application provider. Unique to the application provider (N Same end-devices = 1 AppEUI)
- JoinEUI : Identifies the JS. (1 JoinEUI = 1 JS or multiple JoinEUIs could be pointing to a single JS)
- DevNonce : Random value generated by the end-device
- AppNonce : Unique ID provided by the NS and used by the end-device to derive the two session keys – 'NwKSKey' and 'AppSKey'
- DevAddr : Identifies the end-device within the network. Unique for the specific network
- NetID : Describes the network ID used by the end-node
- DevAddr : To identify the end-device within the network

Different types of activation procedures

- Activation-by-Personalization (ABP)
  - Before Activation : ED must have the following information
    - *DevAddr, AppSKey, NwkSKeyUp, NwkSKeyDown*
  - Before Activation : NS must have the following information
    - *DevAddr, NwkSKeyUp, NwkSKeyDown, AS info of the device*

- Over-the-Air Activation (OTAA)
  - Before Activation : End-device must have the following information
    - *DevEUI, NwkKey, AppKey, JoinEUI*
  - Before Activation : JS must have the following information
    - *DevEUI, AppKey, NwkKey*
  - No NS or AS has any info. about the device until it is activated

On buying the ED



The ED contains the DevEUI

*** DevEUI -** Identifies the MAC address in the sensor device. Provided by the device manufacturer. Unique (1 ED = 1 DevEUI)

Generic ED

On configuring the device with additional parameters (below), the device is called "Generic ED"



The ED contains the DevEUI, NwkKey, AppKey, JoinEUI

- **DevEUI -** Identifies the MAC address in the sensor device. Provided by the device manufacturer. Unique (1 ED = 1 DevEUI)
- **NwkKey -** AES-128 asymmetric key in the device before activation
- **AppKey -** AES-128 asymmetric key in the device before activation
- **JoinEUI -** Global application ID in IEEE EUI64 address space that uniquely identifies the JS

Information at the JS (before OTAA)



The JS contains the DevEUI, NwkKey, AppKey

- **JS - M**anages the OTAA End-Device activation process. There may be several JSs connected to a NS, and a JS may connect to several NSs.

Information at the AS (before OTAA)





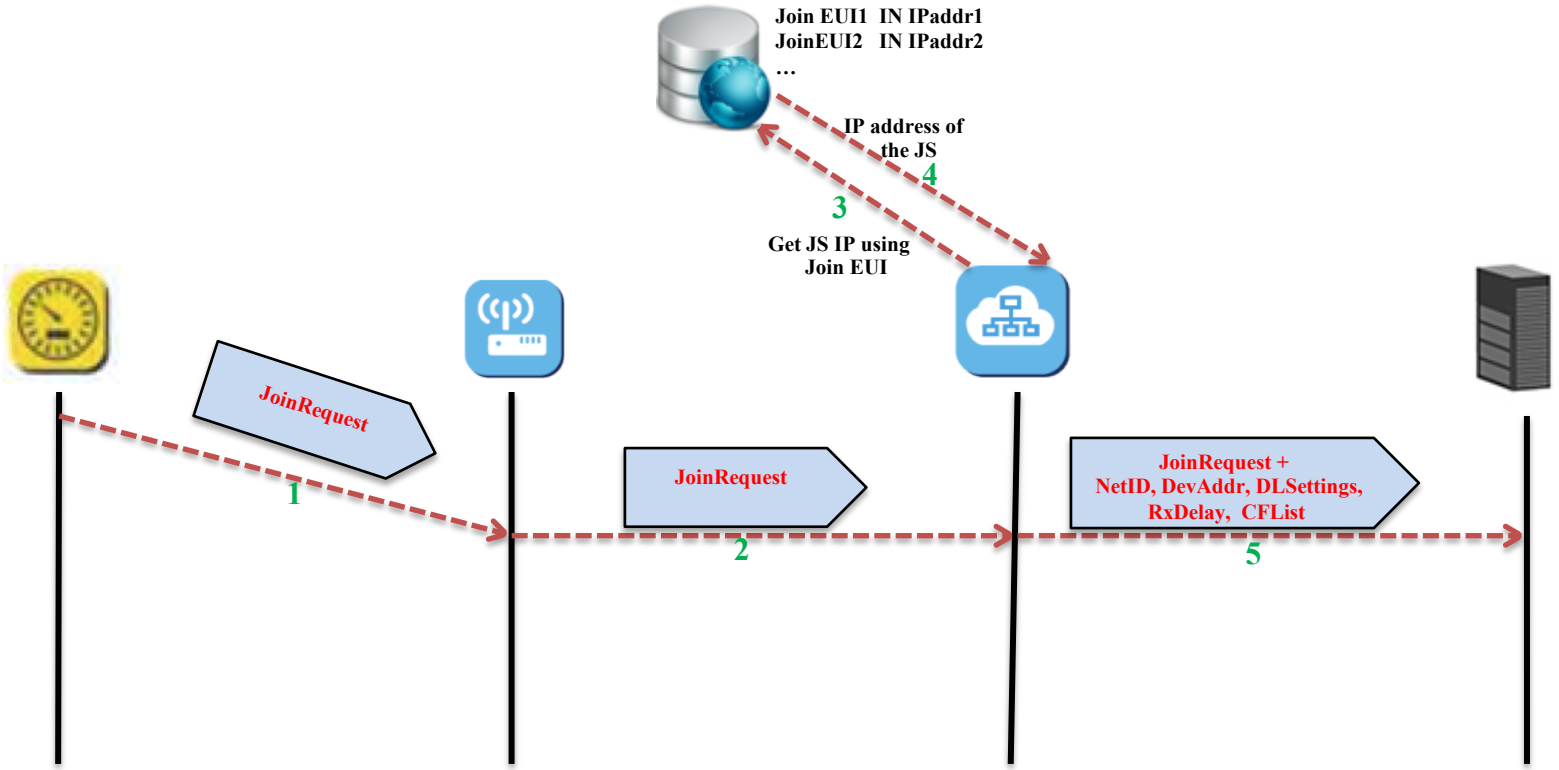
- **Secret Key** - The JS and AS will share a secret key to encrypt/decrypt data sent between them

Join Request for activation



DevNonce - is a random value. For each end-device, the network server keeps track of a certain number of **DevNonce** values used by the end-device in the past, and ignores join requests with any of these **DevNonce** values from that end-device.

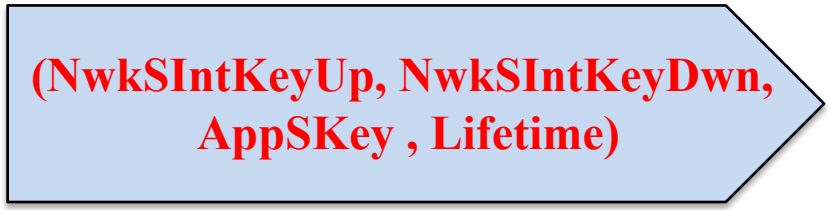
OTAA - JR to JS (Home network)



- **NetID -** Identifies the device home network
- **DevAddr -** To identify the end device within the network
- **DLSettings -** Contains the downlink configuration
- **RxDelay -** Delay between Transmission and Reception
- **CFList –** Optional channels to use

JoinAns from JS

JoinAns



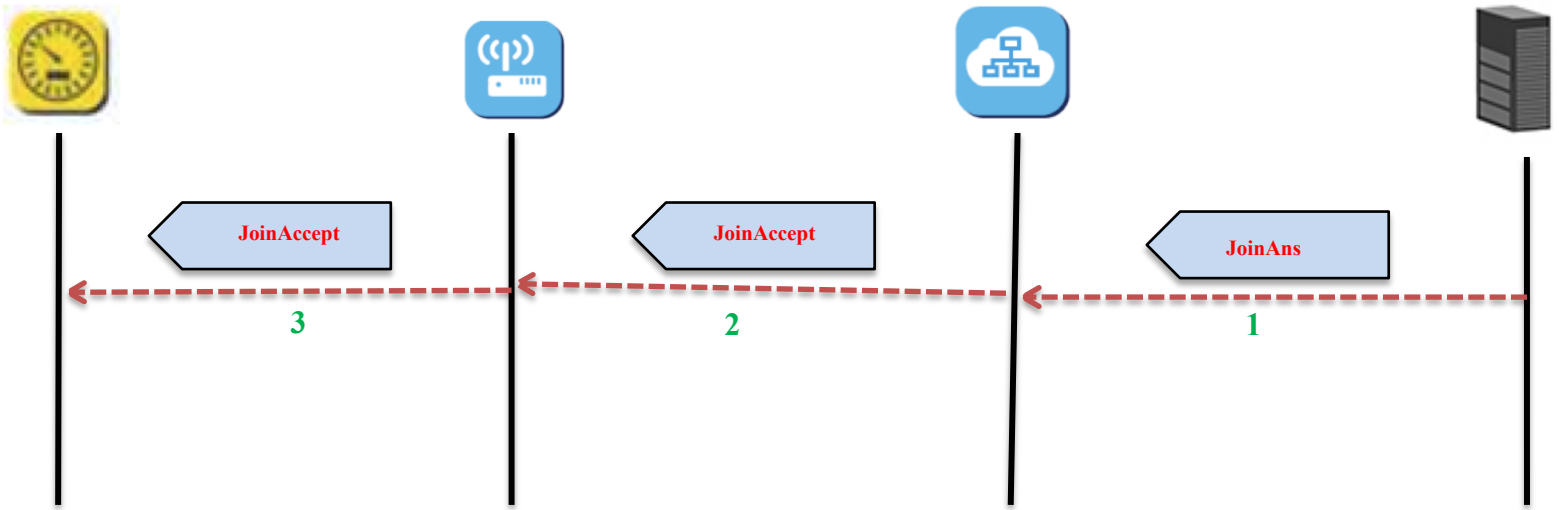
NwkSIntKeyUp - Network server session Key for the Uplink. Derived from NwkKey at the JS for the end-device.
NwkSIntKeyDwn - Network server session Key for the Downlink. Derived from NwkKey at the JS for the end-device.
AppSKey – Application server session key. Derived from the AppKey at the JS for the end-device.
Lifetime – Session Life time

Join Accept for activation

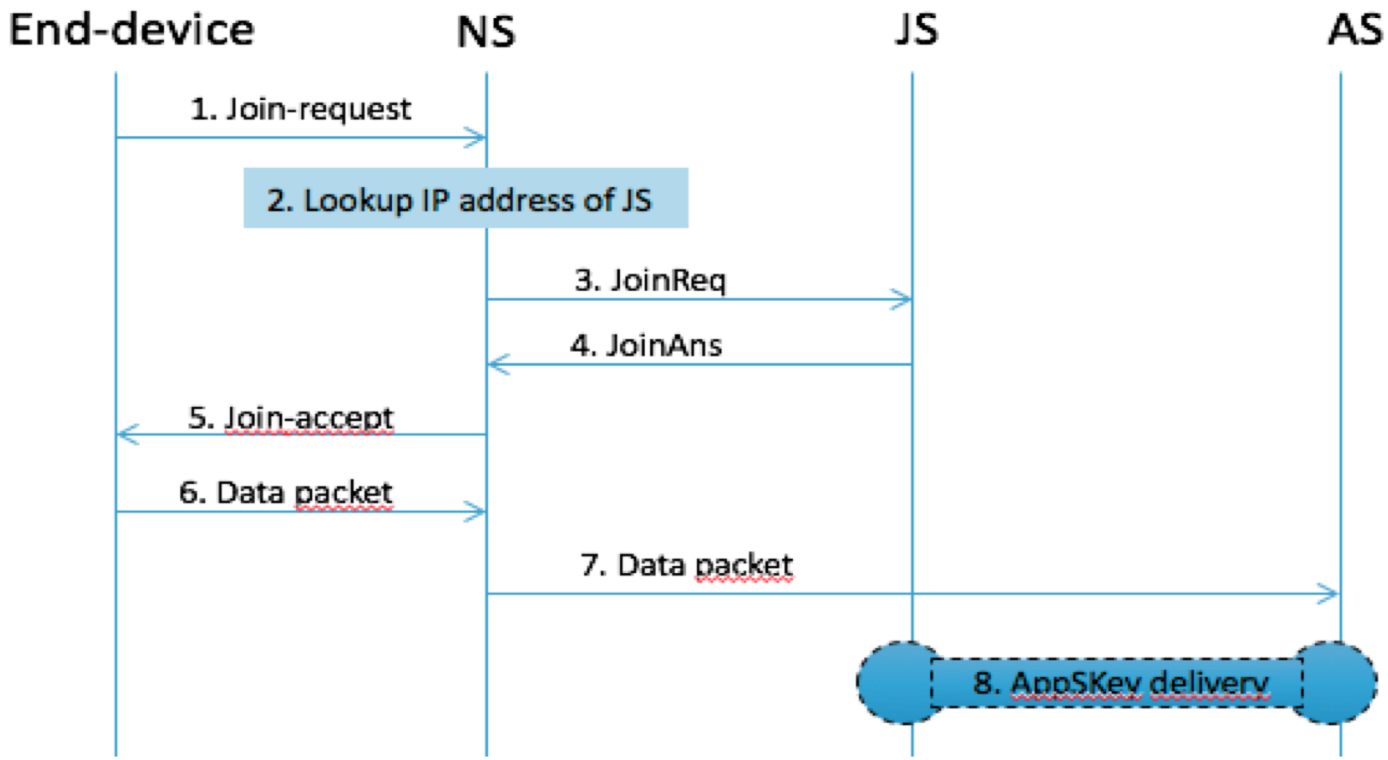


JoinNonce - random value or some form of unique ID provided by the network server and used by the end-device to derive the session keys

OTAA - JoinAccept from NS to the ED



A simple view of LoRa end-to-end connections



Parameters at the ED (After OTAA)

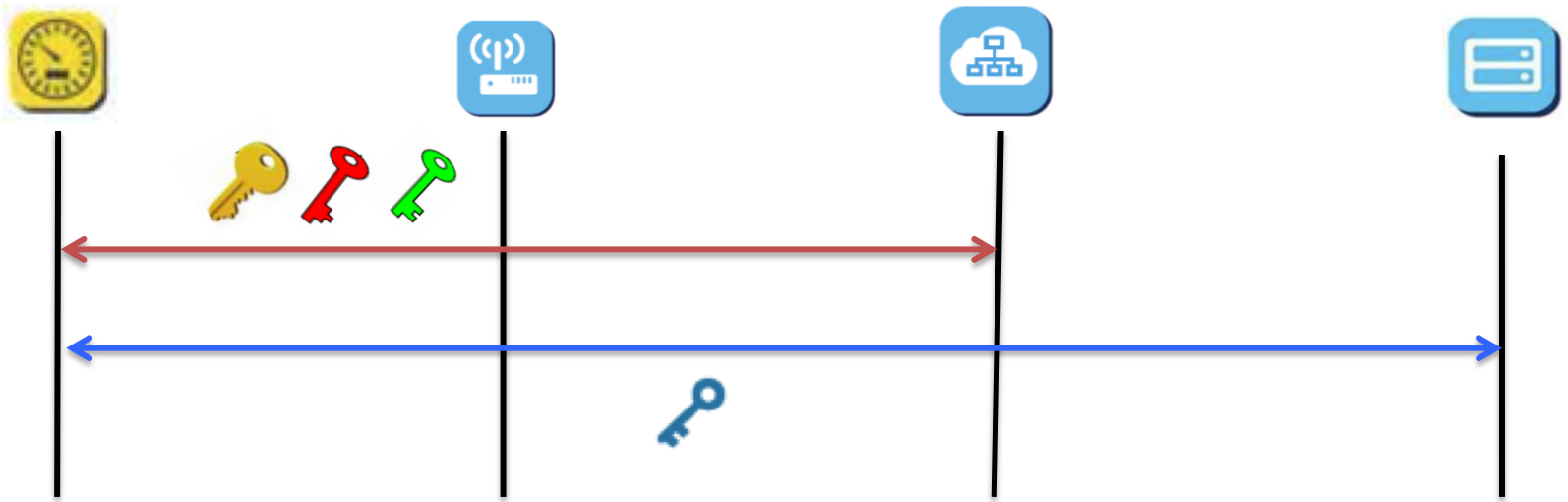


The ED contains the DevEUI, JoinEUI, NwkKey AppKey, JoinNonce, NetID, DevAddr , NwkSIntKeyUp, NwkSIntKeyDwn, NwkSEncKey

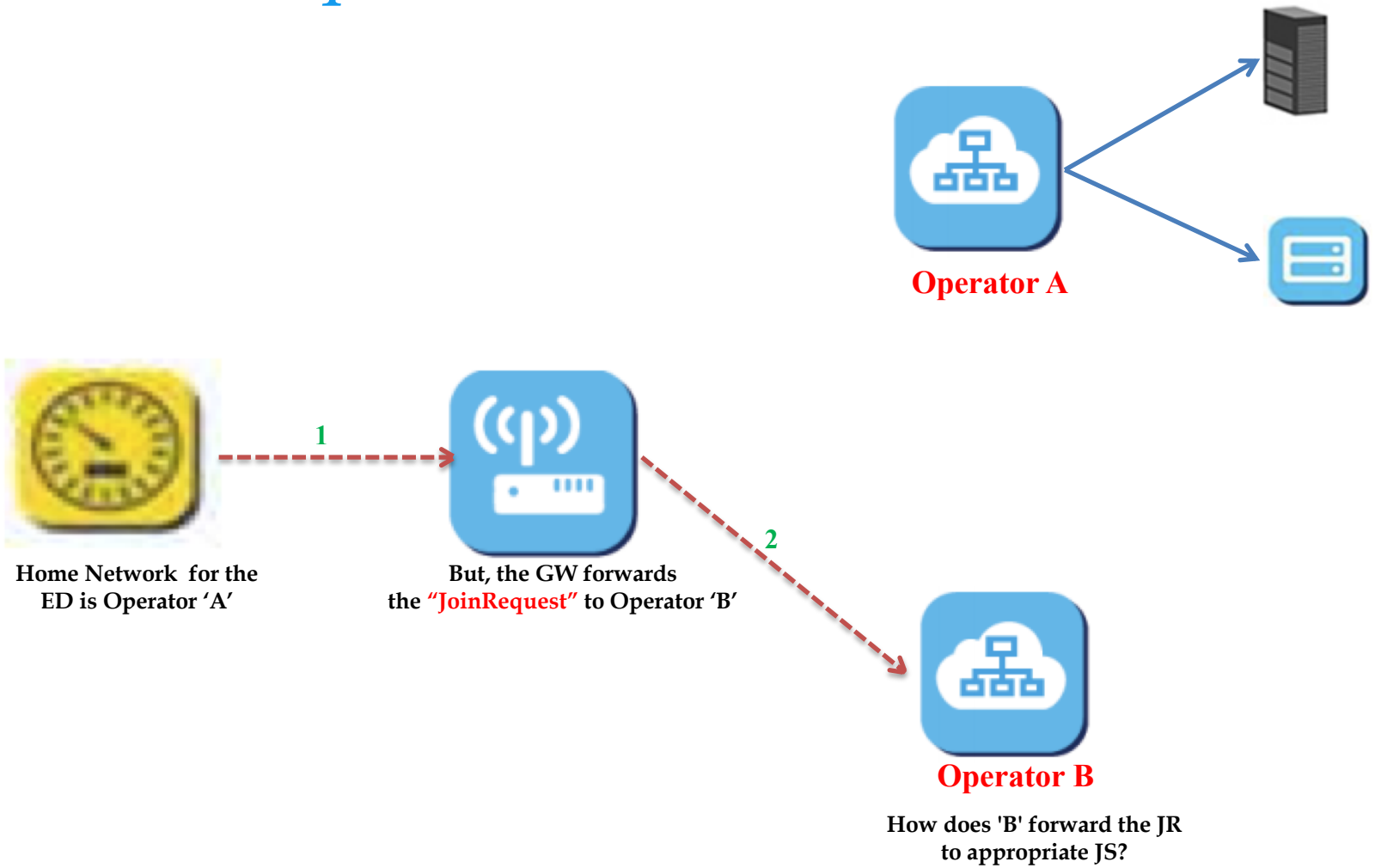
NwkSIntKeyUp - Network server session Key for the Uplink. Derived from NwkKey at the end-device.
NwkSIntKeyDwn - Network server session Key for the Downlink. Derived from NwkKey at the JS for the end-device.
NwkSEncKey – Used to encrypt & decrypt uplink & downlink MAC payloads transmitted on port 0

- AppSKey = aes128_encrypt(AppKey, 0x02 | JoinNonce | NetID | DevNonce | pad16)
- NwkSIntKeyUp = aes128_encrypt(NwkKey, 0x01 | JoinNonce | NetID | DevNonce | pad16
- NwkSIntKeyDwn = aes128_encrypt(NwkKey, 0x03 | JoinNonce | NetID | DevNonce | pad16)
- NwkSEncKey = aes128_encrypt(NwkKey, 0x04 | JoinNonce | NetID | DevNonce | pad16)

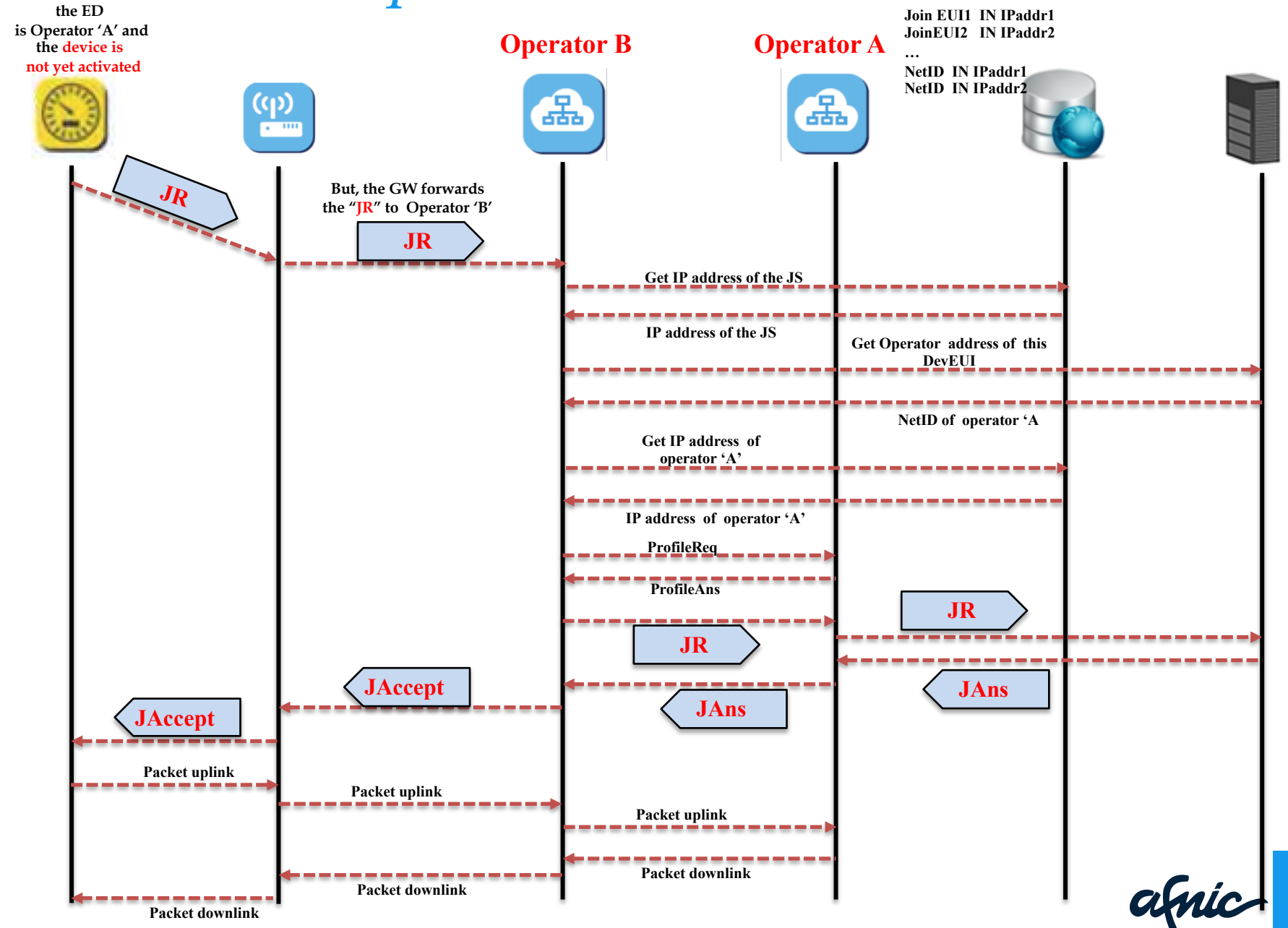
Secure data transmission from ED to NS and AS



JoinRequest outside the home network



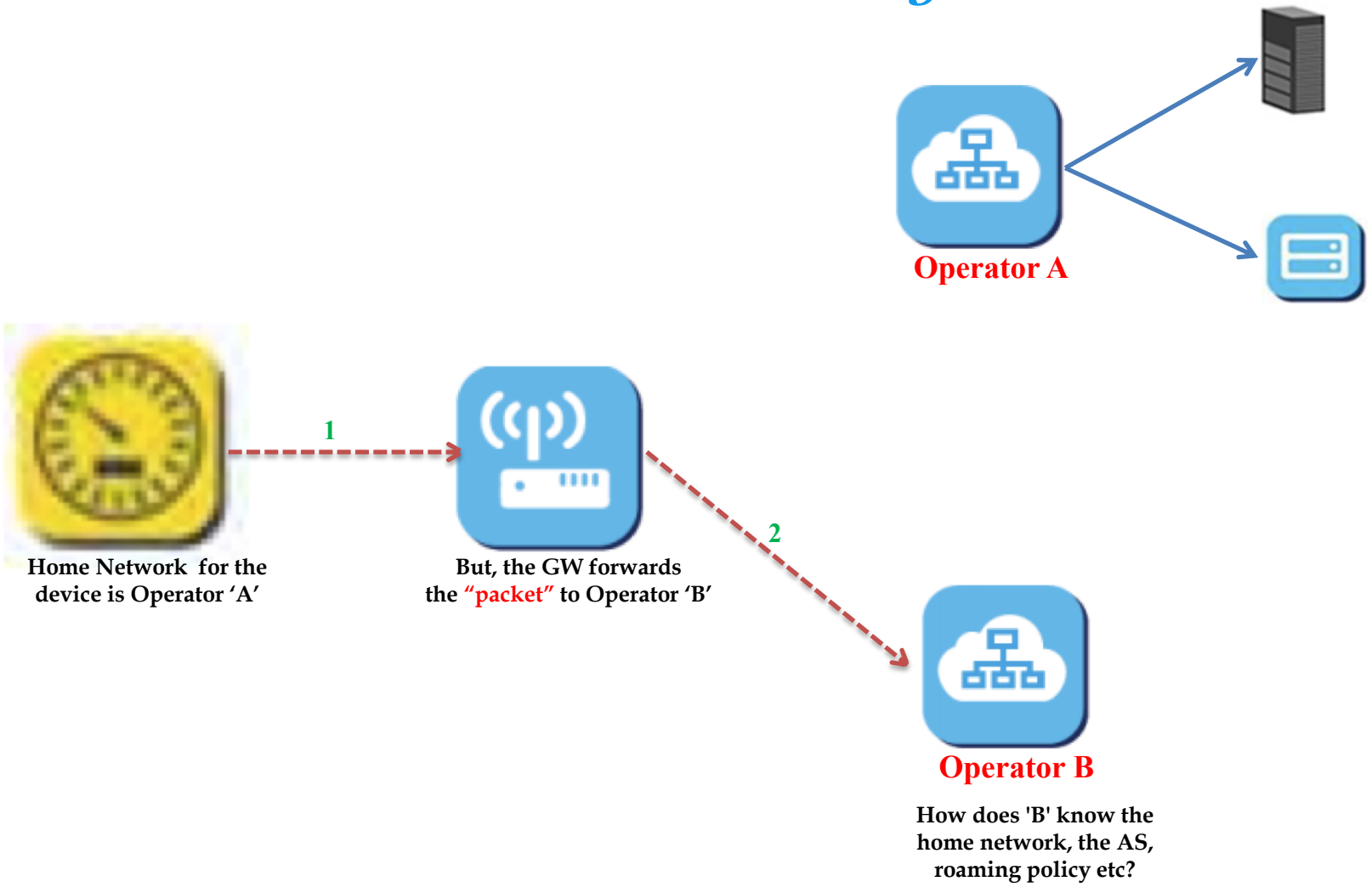
JoinRequest outside the home network



Roaming

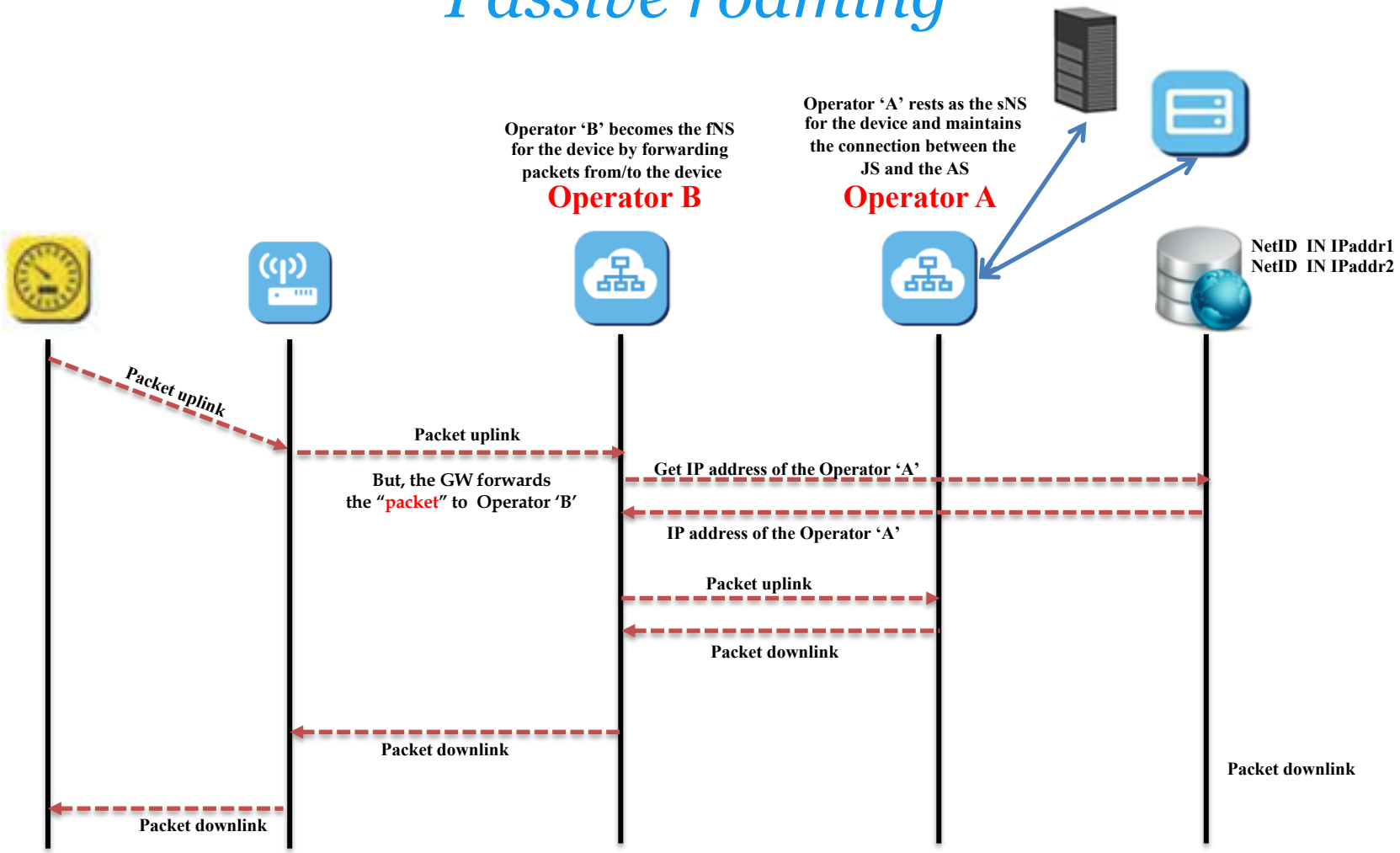
- ✓ Passive roaming - *Passive Roaming enables the Device to continue its active LoRa Session with a Network Server (NS), even when the Device is using the Gateway(s) (GWs) under the control of another NS*
  - ✓ Serving NS (sNS) – which actually has the MAC-layer control and process the communication session
  - ✓ Forwarding (fNS) – which forwards the data to/from the gateway
- ✓ Hand-over roaming - *enables the transfer of the MAC-layer control from one NS to another.*
  - ✓ Anchor NS (aNS) - The NS through which the Device was activated. aNS maintains the control-plane and data-plane with the JS and the AS even after the Device performs a Handover Roaming from one NS to another. aNS stays the same for a given LoRa Session until the Device performs the next Join Procedure.

Passive roaming



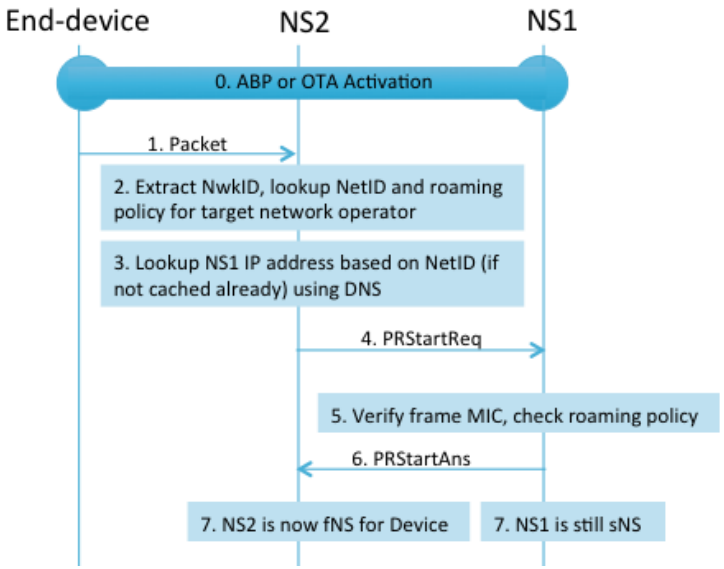
"packet" - Specifies in the process of data transmission after initial handshake using JoinRequest/JoinAccept is done.

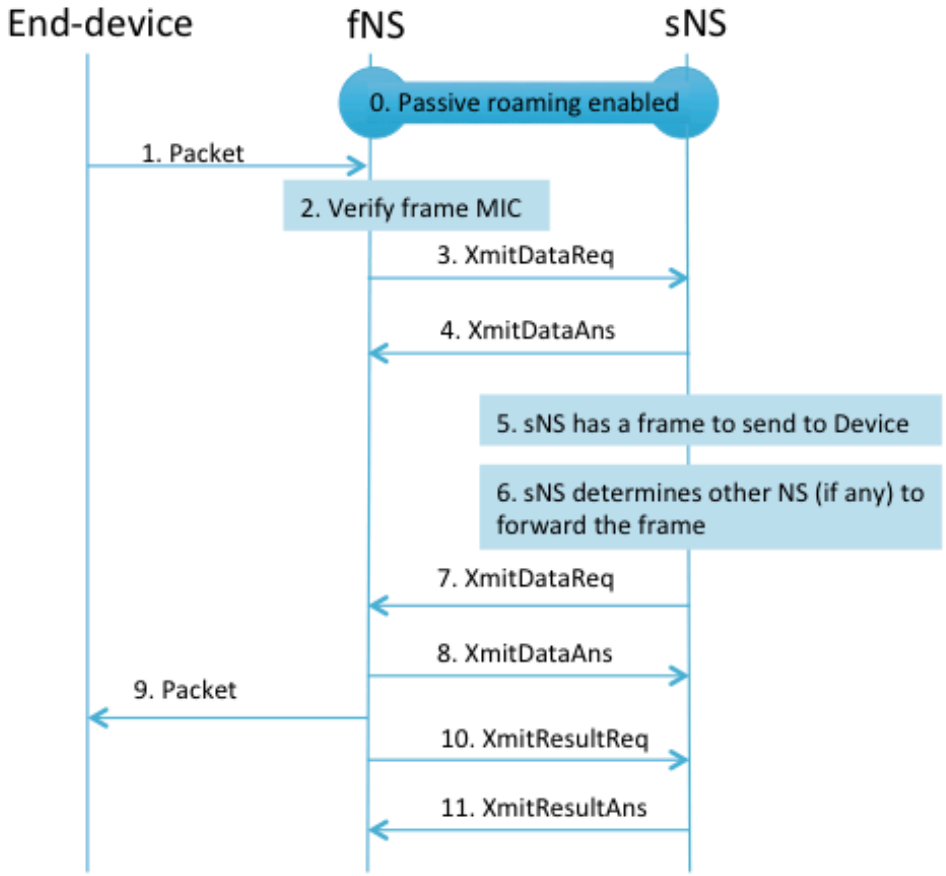
Passive roaming



Serving NS (sNS) - NS which has the functionality of maintaining the LoRa session and MAC layer control of the device
Forwarding NS (fNS) - NS which has the functionality of forwarding data to/from the air interface

Detailed diagram of passive roaming





ReJoin Request

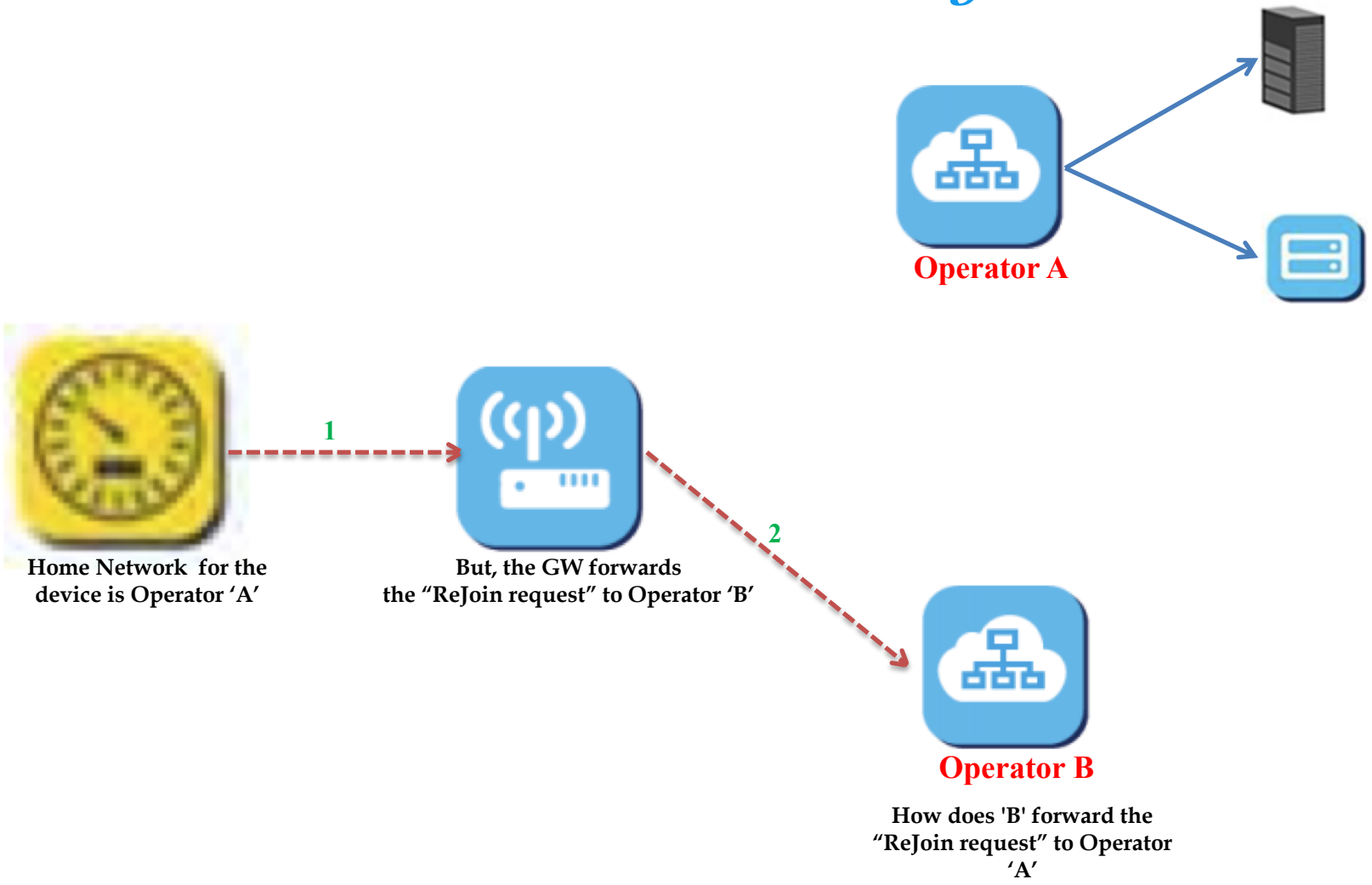
ReJoin Request



ReJoin - This Rejoin-request message periodically gives the network the opportunity to initialize a new session context for the end-device. For this purpose the network replies with a Join-Accept message. This may be used to hand-over a device between two networks or to rekey and/or change DevAddr of a device on a given network.

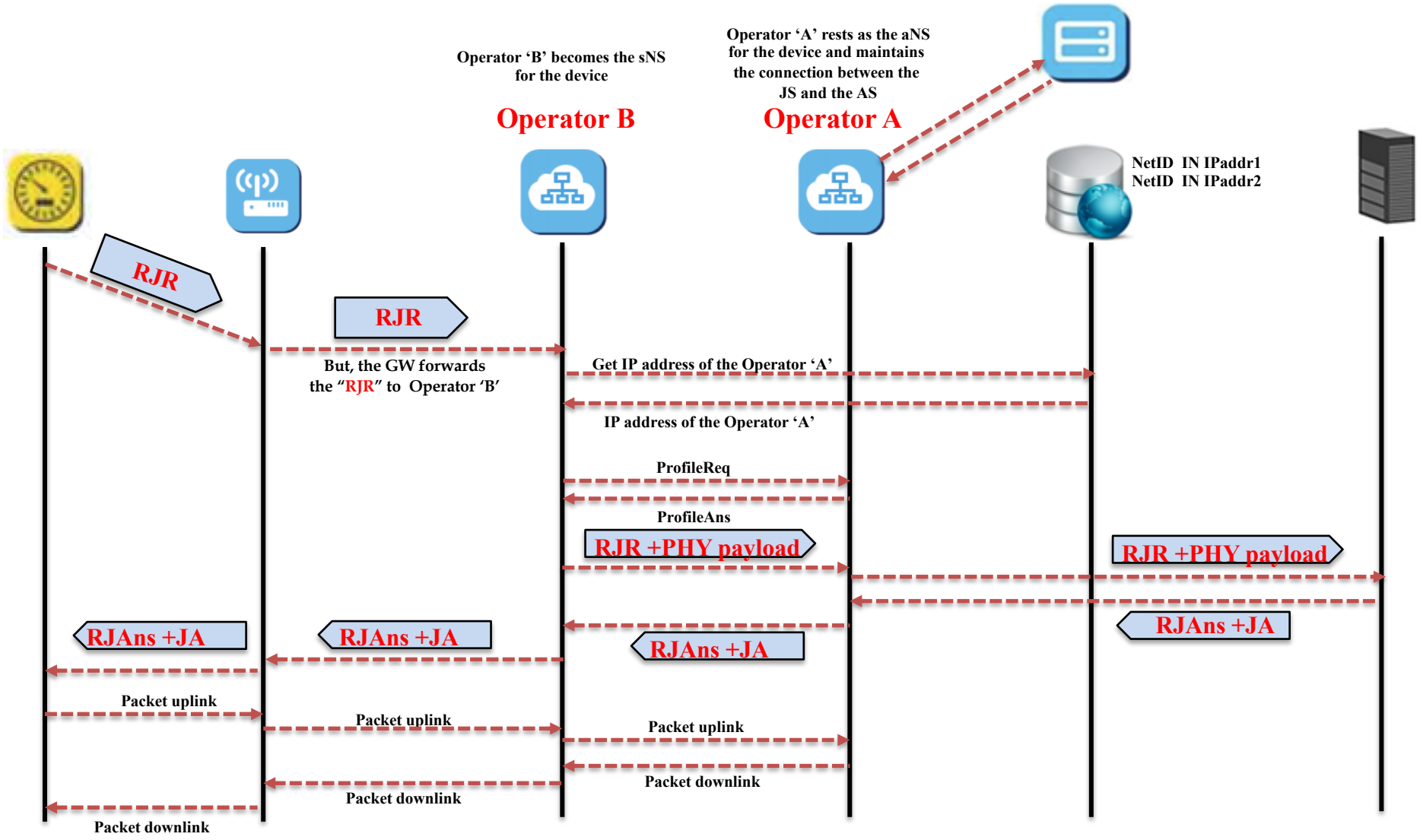
Three types of Rejoin requests:
Type0 - Contains NetID+DevEUI. Used to reset a device context including all radio parameters (DevAddr, session keys, Frame counters, radio parameters, ..). This message can only be routed to the device's home network server by the receiving network server, not to the device's JoinServer
Type1 - Contains JoinEUI+DevEUI. Exactly equivalent to the initial Join-Request message but may be transmitted on top of normal applicative traffic without disconnecting the device. Can only be routed to the device's JoinServer by the receiving network server. Used to restore a lost device context (Example, network server has lost the session keys and cannot associate the device to a JoinServer)
Type2 - Contains NetID+DevEUI. Used to rekey a device or change its DevAddr (DevAddr, session keys, Frame counters). Radio parameters are kept unchanged. This message can only be routed to the device's home network server , not to the device's JoinServer

Handover roaming



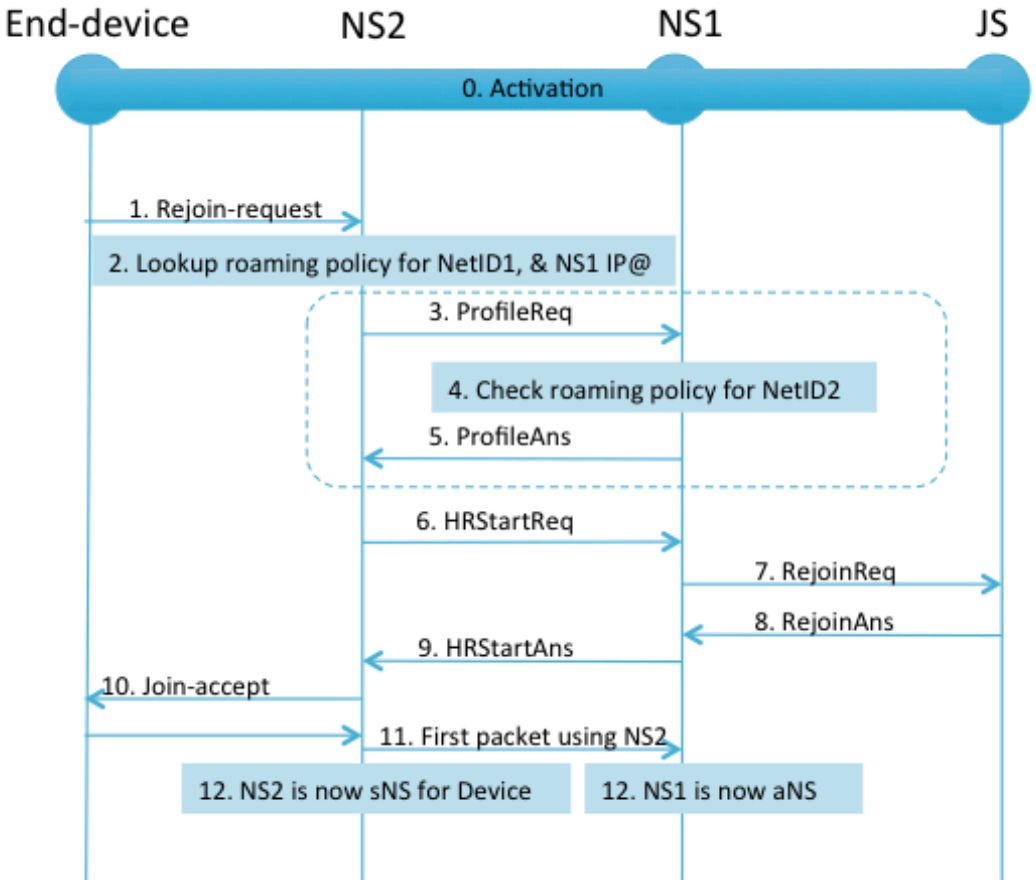
Handover Roaming - enables the transfer of the MAC-layer control from one NS to another.

Hand-over roaming



Anchor NS (aNS) - The NS through which the device was activated. Maintains the control and data plane with the JS and AS even after the device performs a hand-over roaming from one NS to another

Handover roaming using DNS



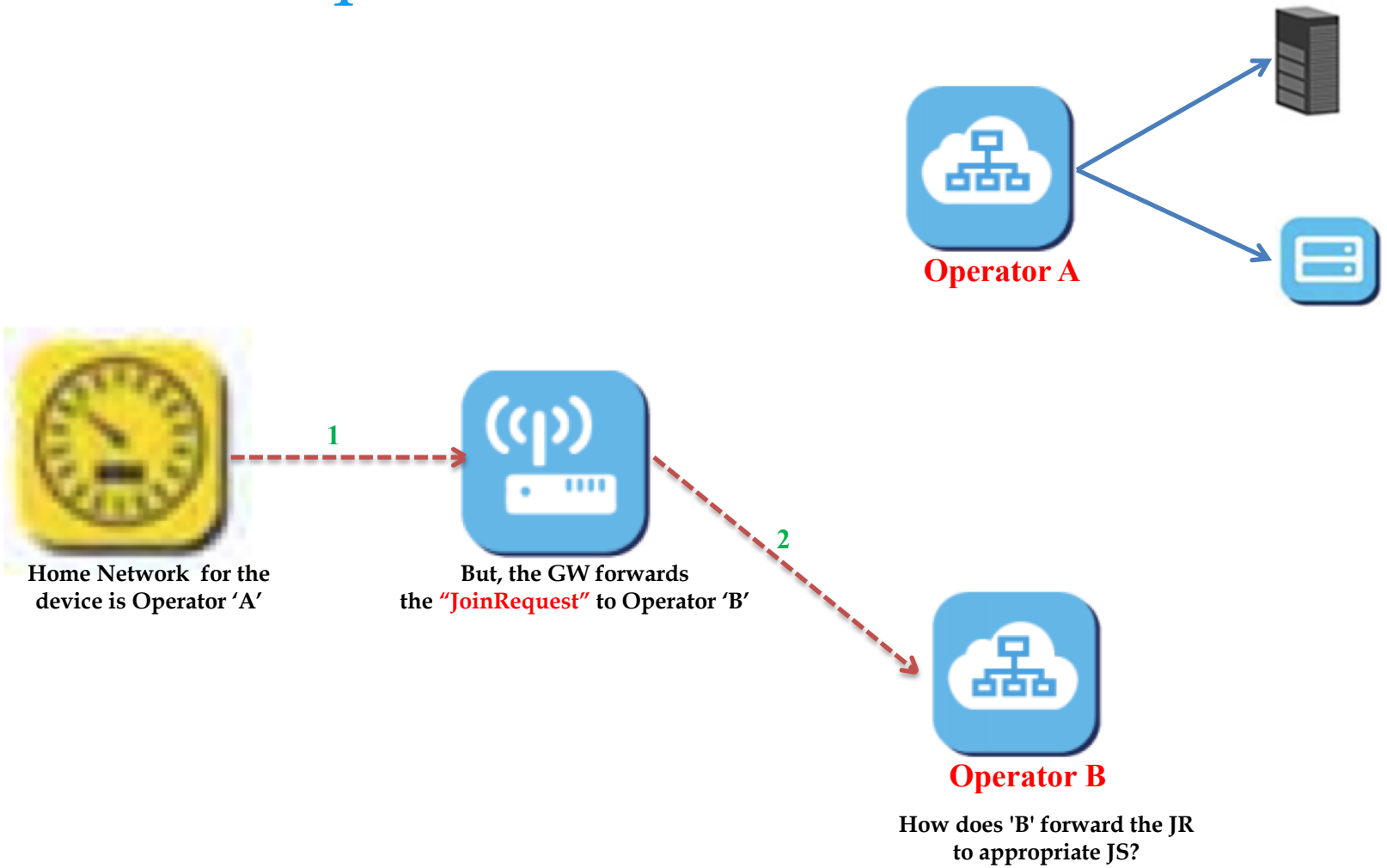
Merci !



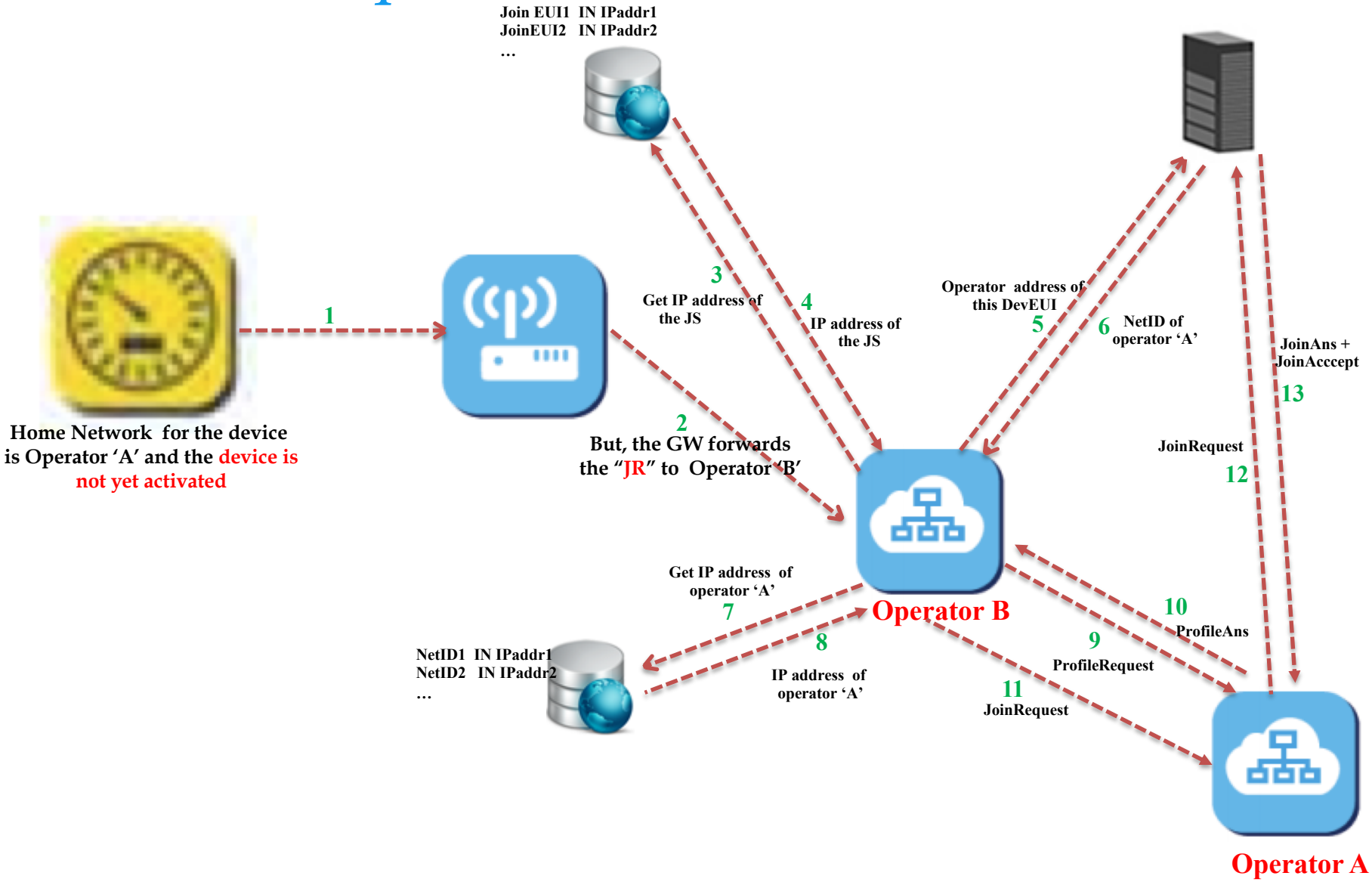
www.afnic.fr
contact@afnic.fr
Twitter : @AFNIC
Facebook : afnic.fr

afnic

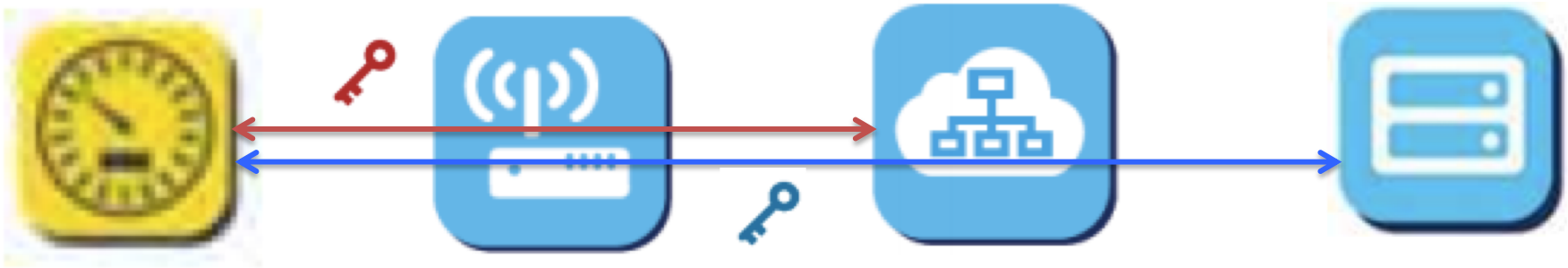
JoinRequest outside the home network



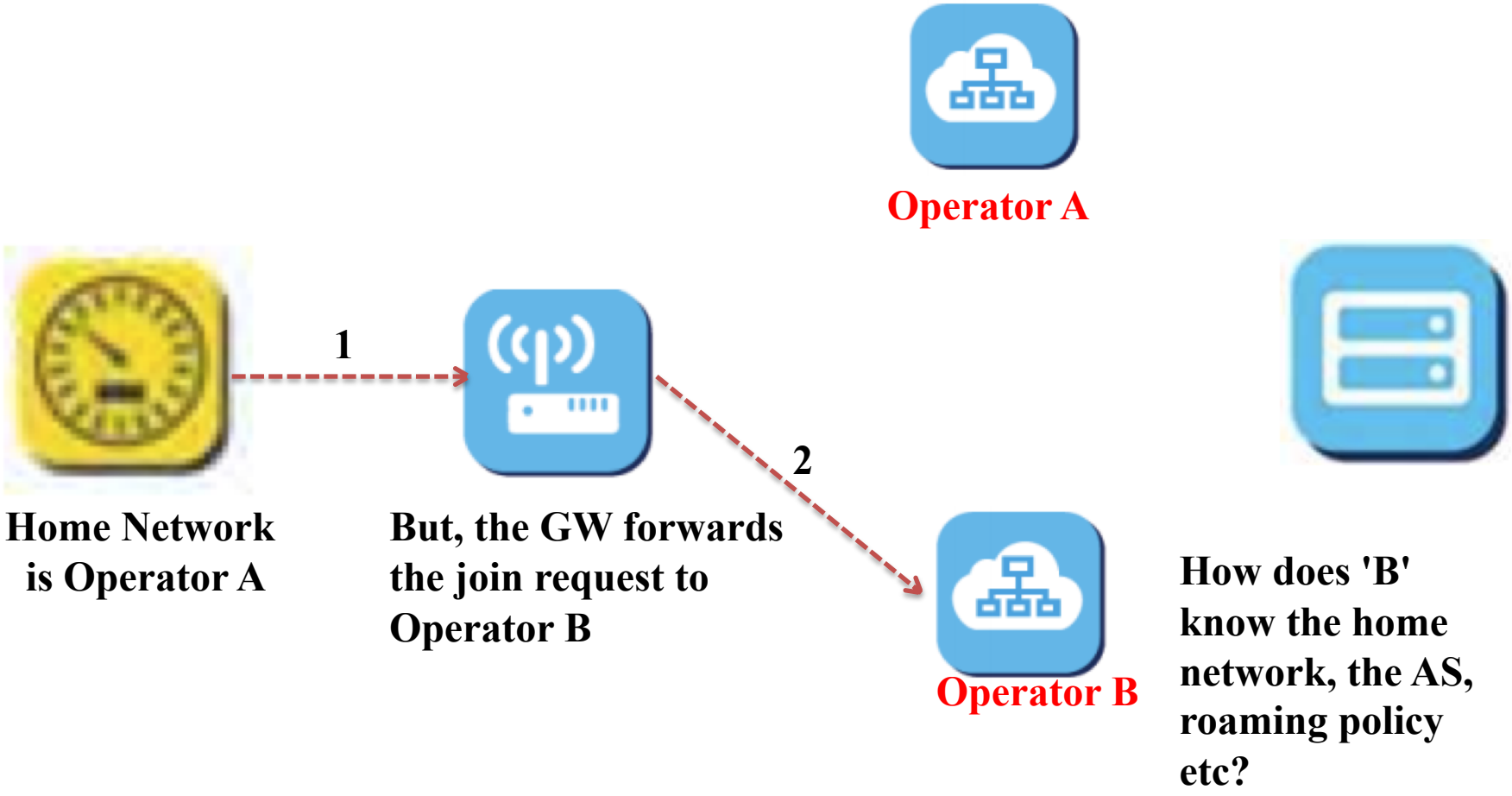
JoinRequest outside the home network



Secure data transmission



Use Case - 1 (Activation away from home network)



Possible solution for Use Case 1

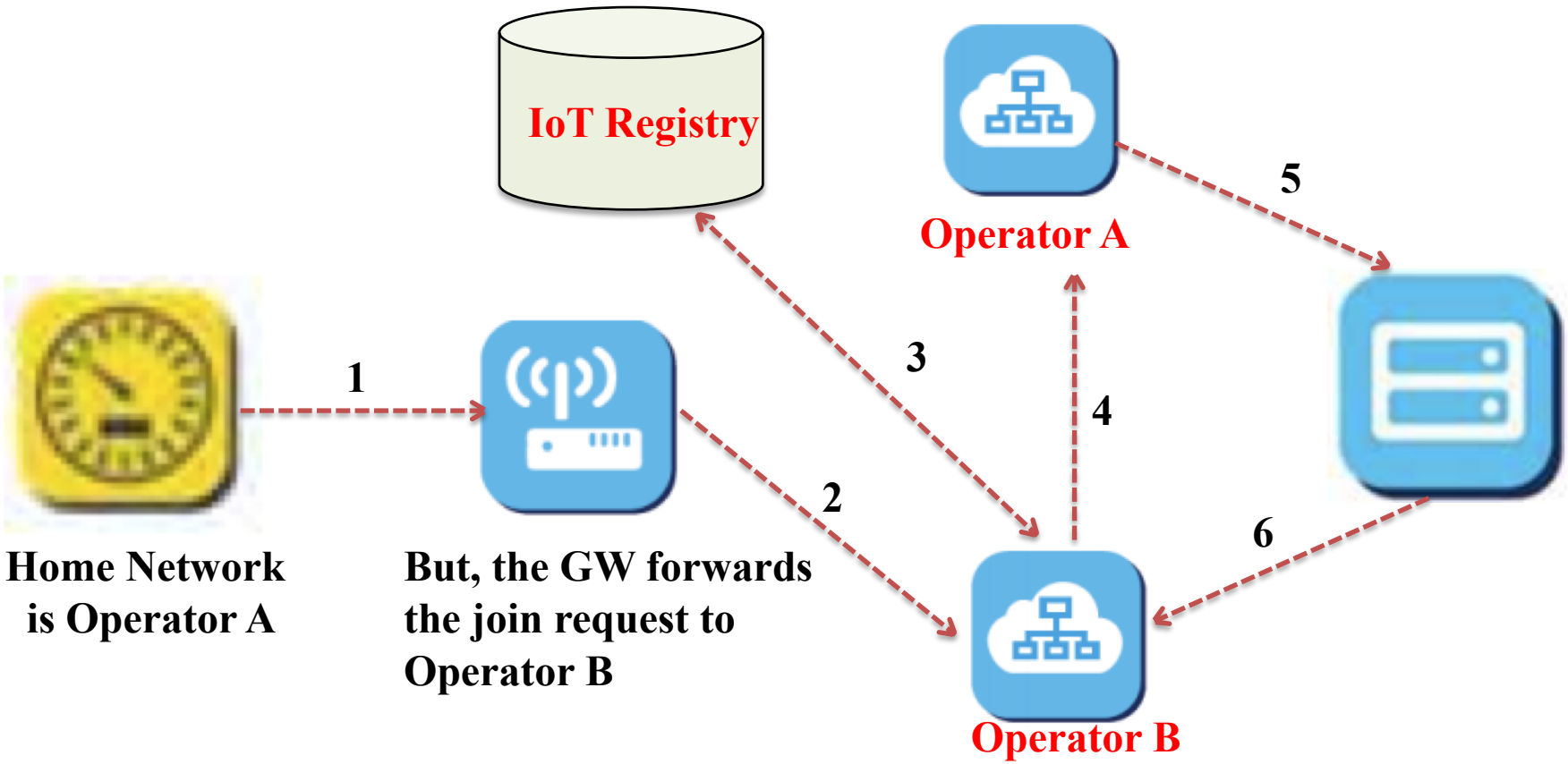
Before activation, the device has the following information

The device contains the DevEUI, AppEUI, AppKey

Have a DNS mapping table as below in a well known fallback address

AppEUI	Operator IP Address

Possible solution for use case - 1



Use Case for DNS – After activation

After activation, the device has the following information

The device contains the DevEUI, AppEUI, AppKey, DevAddr , NetID, NwKSKey, AppSKey

Have a DNS mapping table at each Operator

NetID	Operator IP address

NetID – Network Identifier

Explaining the different parameters

- ✓ DevEUI – Identifies the MAC address in the sensor device. Provided by the device manufacturer. Unique (1 end-device = 1 DevEUI)
- ✓ AppEUI – Identifies the AS. Furnished by the Application provider. Unique to the application provider (N Same end-devices = 1 AppEUI)
- ✓ JoinEUI - Identifies the JS. (1 JoinEUI = 1 JS)?
- ✓ DevNonce - Random value generated by the end-device
- ✓ AppNonce - Unique ID provided by the NS and used by the end-device to derive the two session keys – 'NwKSKey' and 'AppSKey'
- ✓ AppKey – Provided by the AS. Unique (1 end-device = 1AppKey)
- ✓ AppSKey – Generated by the AS, using the following data (AppKey, AppNonce, NetID)
- ✓ NwKSkey - Generated by the AS, using the following data (AppKey, AppNonce, NetID)
- ✓ DevAddr – Identifies the end-device within the network. Unique for the specific network
- ✓ NetID – Describes the network ID used by the end-node
- ✓ DevAddr – To identify the end-device within the network

Associating the end-device to the AS (before activation)

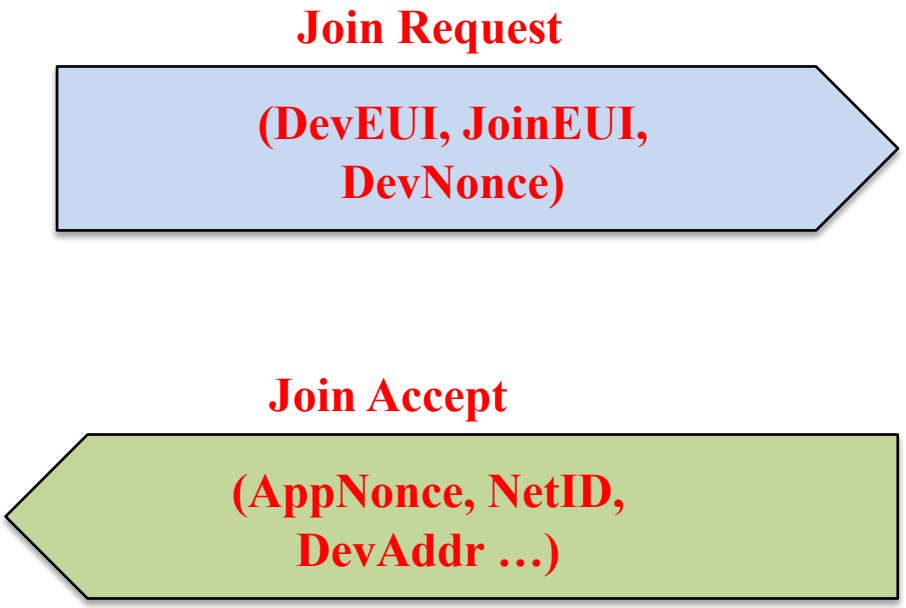


The device contains the DevEUI, JoinEUI, AppKey, NwkKey

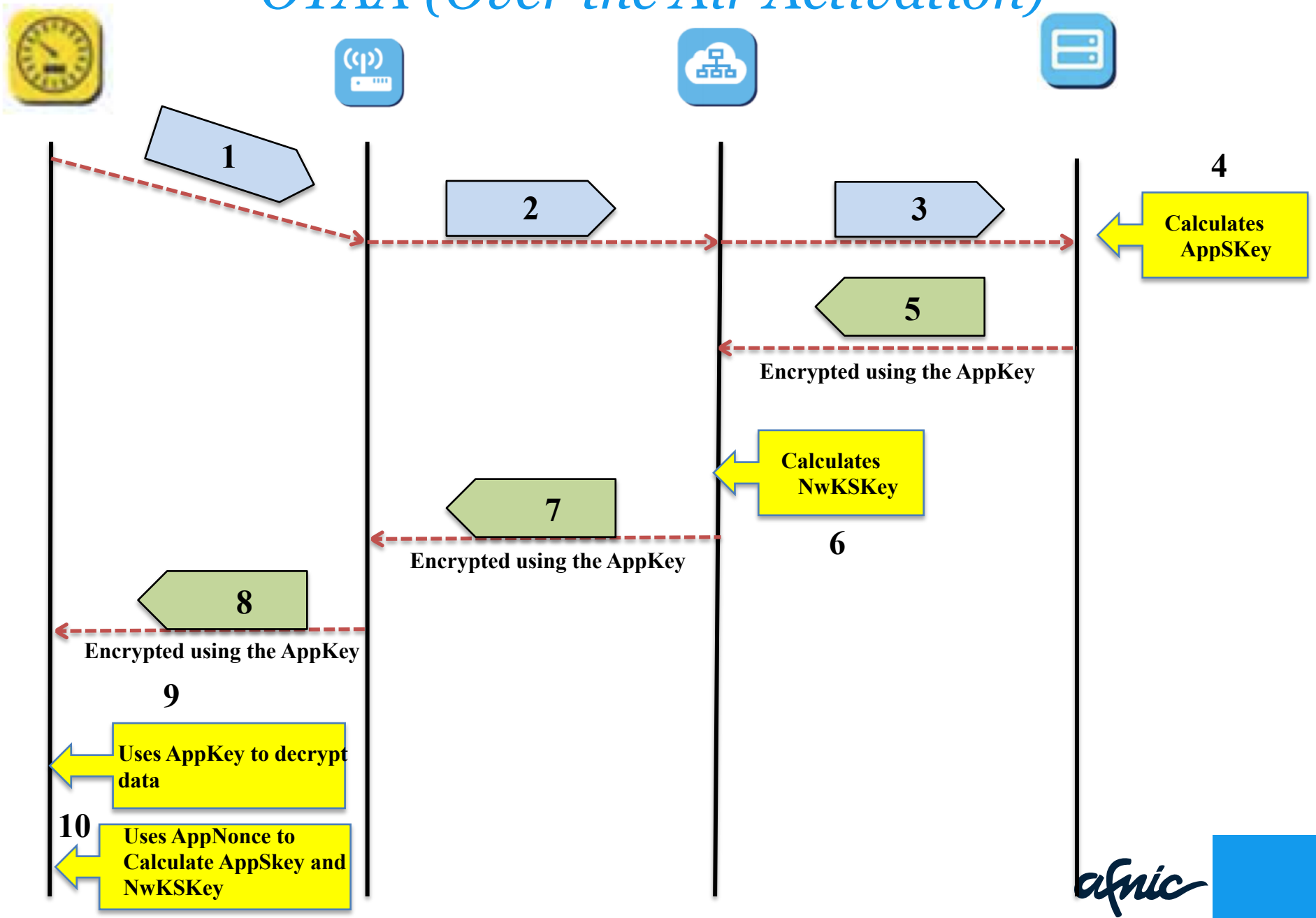


The AS contains the AppKey

Initial Handshake requests for activation



OTAA (Over the Air Activation)



Parameters & Keys in the LoRa data frame

DevEUI

AppEUI

JoinEUI

DevNonce

AppNonce

DevAddr

NetID


AppKey


AppSKey


NwkSKey


NwkSKeyUp

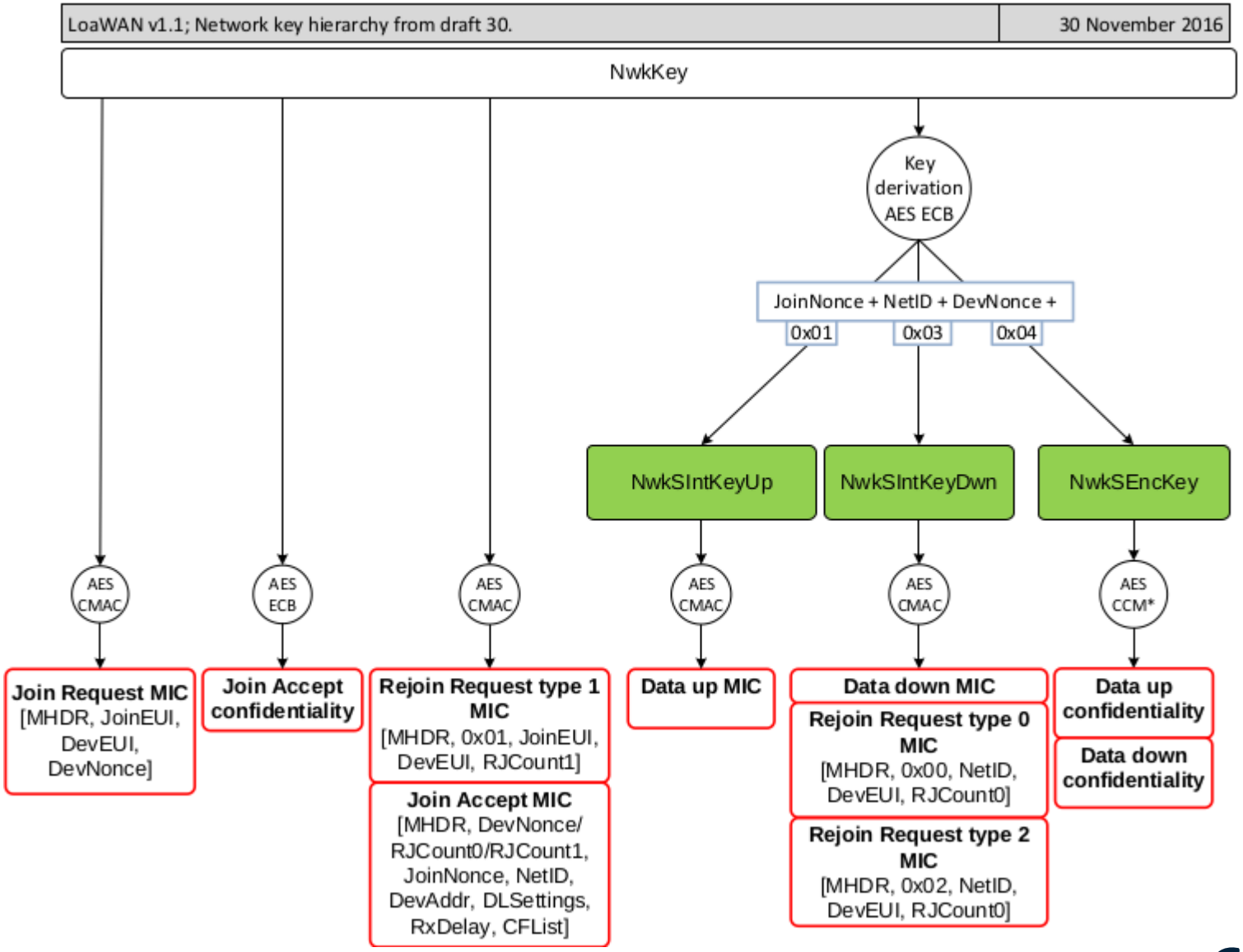

NwkSKeyDwn


NwkSEncKey

Different Cryptographic Keys used

- ✓ Keys assigned during fabrication
  - ✓ AppKey – AES-128 symmetric key
  - ✓ NwkKey – AES-128 symmetric key
- ✓ AppSKey –
- ✓ NwkSIntKeyUp –
- ✓ NwkSIntKeyDwn –
- ✓ NwkSEncKey –

Key Hierarchy



Passive roaming using DNS

