26 Otc 2022

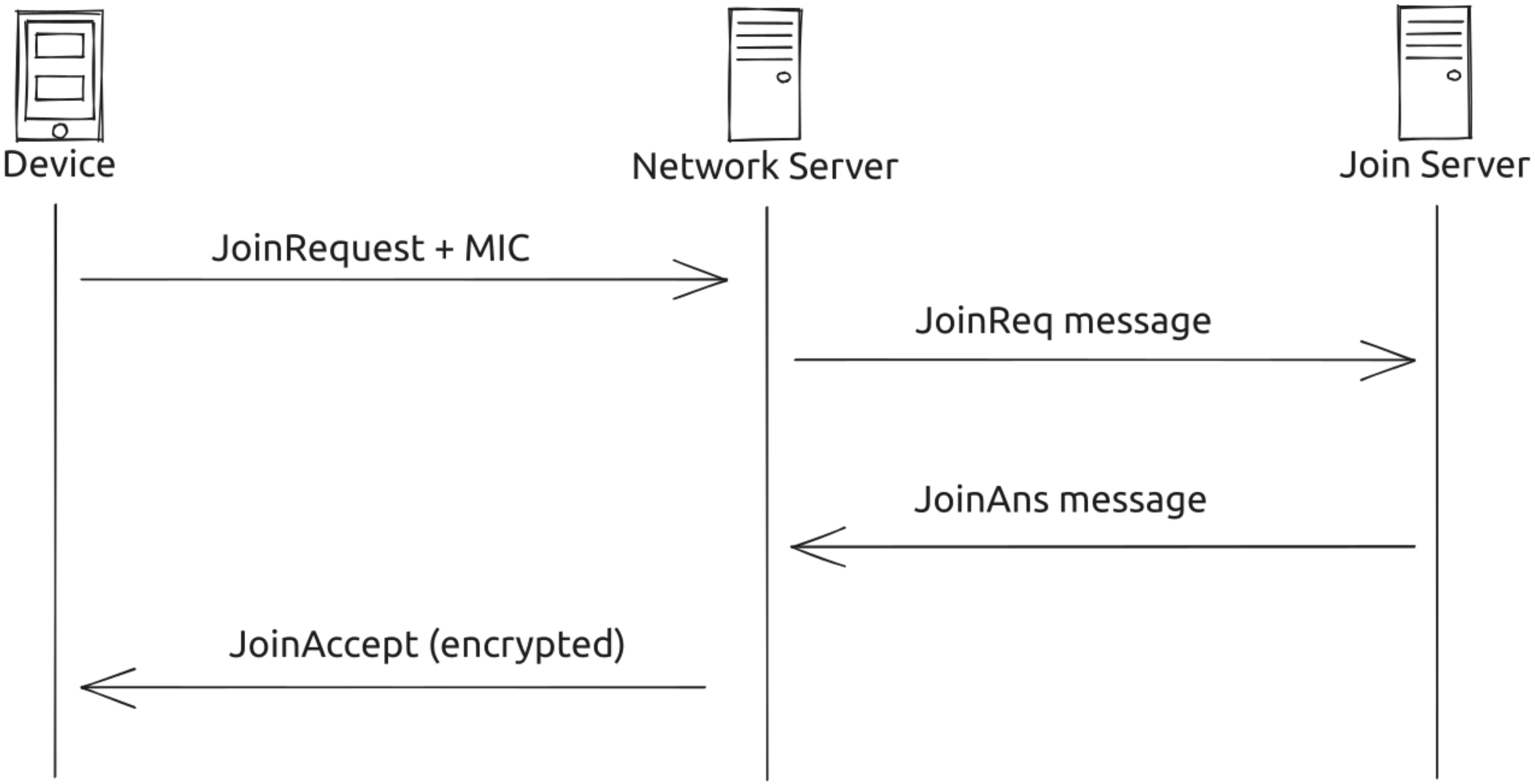
Using DANCE to authenticate devices in an IoT use-case

Gaël Berthaud-Müller – Afinc

Internet made in France

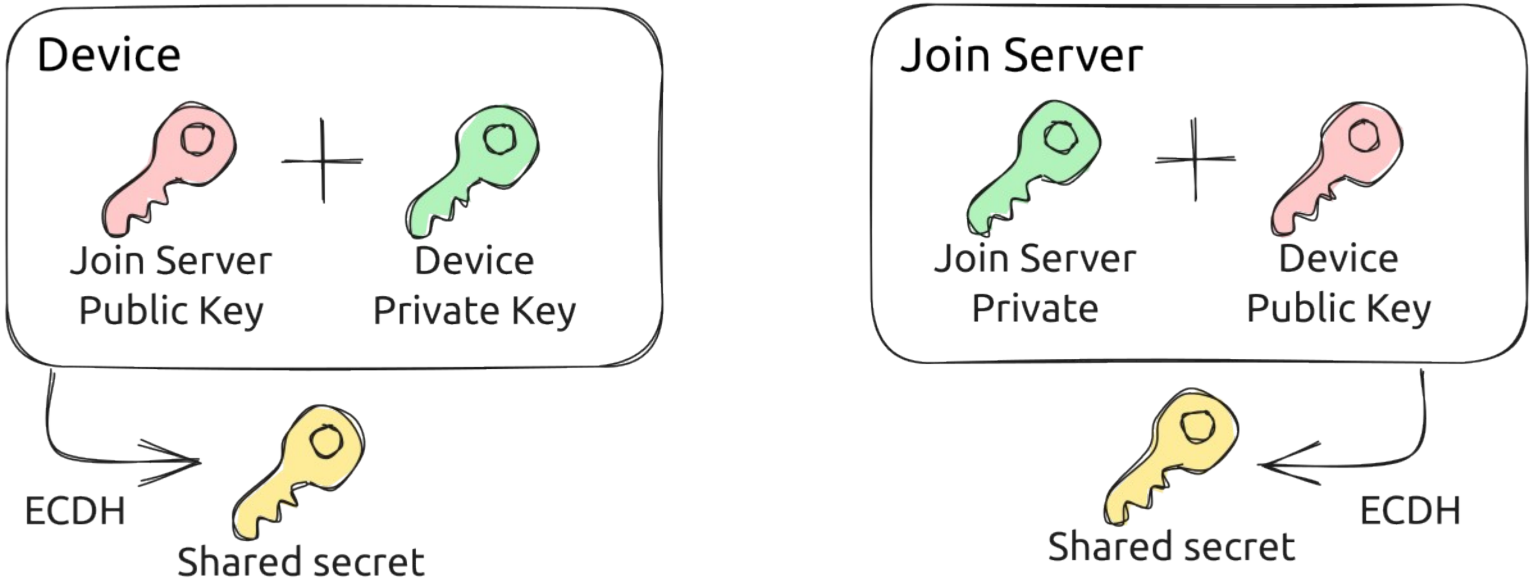
Device join procedure in LoRaWan

- The device is authenticated on the network by a Join Server using a pre-shared key



Device join procedure in LoRaWan

- Use a shared secret
- Why not use asymmetric cryptography ?



- The device need to know the Join Server public key

How to apply DANE / DANCE to LoRa join procedure

The problem

- How does the device authenticate the server as being the one it wants to join?
- How does the server verify that the device is actually the one it says it is?

Presentation of the Proof of Concept

- Use of asymmetric cryptography (elliptic curve)
- Raw public keys in TLSA records
- The device only knows its own private key and a DNS trust anchor

Authenticating the device – Constructing the JoinRequest

Normal JoinRequest

Join EUI (8)	Device EUI (8)	Nonce (2)	MIC (4)

```
CMAC = aes128_cmac(AppKey, MHDR | JoinEUI | DevEUI | DevNonce)
MIC  = CMAC[0..3]
```

Modified JoinRequest

Join EUI (8)	Device EUI (8)	Nonce (2)	Signature (64)

```
Signature = ecdsa_signature(DevicePrivateKey, JoinEUI | DevEUI | DevNonce)
```

Authenticating the device – Validating the JoinRequest

The Join Server:

1. extracts the DeviceEUI from the Join Request
2. queries for the corresponding TLSA record
3. validates the authenticity of the TLSA record using DNSSEC
4. validates the signature with the public key fetched from the TLSA record
5. computes the shared secret using ECDH

```
_lora-join.4.c.b.0.9.8.b.e.3.c.1.b.7.1.8.5.deveuis.iot.rd.nic.fr 1 IN TLSA 3 1 0 d3a289...f01b5d
```

DeviceEUI (reversed) Raw public key

Authenticating the join server – Constructing the JoinAccept

Normal JoinAccept

JoinAcceptEncryptedPayload (16)	MIC (4)

```
EncryptedPayload = aes128_decrypt(
                       AppKey,
                       JoinNonce | NetID | DevAddr | DLSettings | RXDelay | CFList | MIC)
```

Modified JoinAccept

JoinAcceptEncryptedPayload (16)	DnssecChain (Variable)

```
AppKey           = ecdh(JoinServerPrivateKey, DevicePublicKey)
EncryptedPayload = aes128_decrypt(
                       AppKey,
                       JoinNonce | NetID | DevAddr | DLSettings | RXDelay | CFList | 0x00_00_00_00)
```

Authenticating the join server – Constructing the DNSSEC Chain

- Based on RFC 9102 - TLS DNSSEC Chain Extension
- Avoid CNAME to reduce the chain size (only one branch)
- Use a intermediary trust anchor (e.g. lora-alliance.org) to avoid unnecessary records
- Only use DNSSEC algorithm 13 in the chain (ECDSA P256 SHA256) to reduce the chain size (compared to RSA)

```
lora-alliance.org DNSKEY
    RRSIG(lora-alliance.org DNSKEY)
joineuis.lora-alliance.org DS
    RRSIG(joineuis.lora-alliance.org DS)
joineuis.lora-alliance.org DNSKEY
    RRSIG(joineuis.lora-alliance.org DNSKEY)
_lora-join.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.joineuis.lora-alliance.org TLSA
    RRSIG(_lora-join.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.joineuis.lora-alliance.org TLSA)
```

Authenticating the join server – Encoding the DNSSEC Chain

- The chain is encoded in CBOR, with a format loosely inspired by draft-lenders-dns-cbor-04 - A Concise Binary Object Representation (CBOR) of DNS Messages
- Assume “protocol” field is 3 in DNSKEY record
- Compress ECDSA keys in DNSKEY and TLSA resource records (reduce the key size from 64 to 33 bytes)
- Omit the TTL or name if it is the same as the previous RRSet
- Make name relative to the name of the previous RRSet if possible
- Sort records from top to bottom, in a zone the DS comes first followed by the DNSKEY and then other records

Authenticating the join server – Encoding the DNSSEC Chain - CDDL

```
dnskey-data = [
    flags: uint,
    algorithm: uint,
    key: bstr .size 33,
]

tlsa-data = bstr .size 36

rdata = dnskey-data / tlsa-data / bstr

signature = [
    algorithm: uint,
    expiration: uint,
    inception: uint,
    key_tag: uint,
    signature: bstr,
]
```

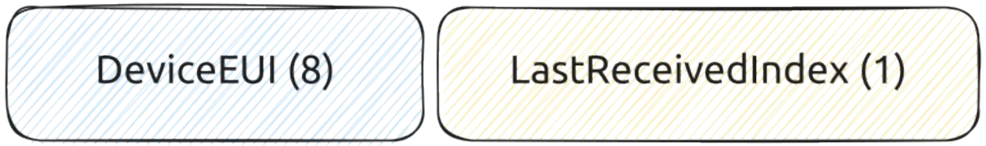
```
rrset = [
    type: uint,
    ? name: tstr,
    ? orig_ttl: uint,
    rdata: [ + rdata ],
    signatures: [ + signature ],
]

chain = [ + rrset ]
```

Authenticating the join server – JoinAccept fragmentation

Need to fragment the JoinAccept due to the large payload size

The next fragment is requested by the device to also work on class A devices



Authenticating the join server – Validating the JoinAccept

The device:

1. validates the DNSSEC chain using the shared trust anchor as a starting point
2. validates that the TLSA record corresponds to the JoinEUI
3. extracts the Join Server public key from the TLSA record in the chain
4. computes the shared secret using ECDH
5. decodes the JoinAccept payload
6. resumes normal LoRa operation (computing session keys, configuring data rate…)

Evaluation (work in progress)

- Device joins in ~30s with highest data rate (200 bytes/packet)
- DNSSEC chain size with intermediary trust anchor :

Uncompressed wire format	1266 B
Uncompressed CBOR	817 B
Compressed CBOR	720 B

Thanks you!

gael.berthaud-muller@afnic.fr

gitlab.rd.nic.fr/iot/lora-dance

Internet made in France

Association Française pour le Nommage Internet en Coopération

Immeuble Stephenson, 1 rue Stephenson, 78180 Montigny-le-Bretonneux, France
Tel. +33 (0)1 39 30 83 00

www.afnic.fr | contact@afnic.fr | Twitter : @AFNIC | Facebook : facebook.com/afnic.fr