Lightweight AKE for OSCORE

- Problem Statement
- EDHOC as a Solution
- EDHOC Benchmarks

SecDispatch Interim
March 5, 2019

Agenda

2. Problem Statement (20 min)
   - Background (Göran Selander)
   - Motivating use cases for EDHOC (Claes Tidestav, Mališa Vučinić, Jesús Sánchez-Gómez)
   - Requirements of EDHOC use cases (Göran Selander)

3. EDHOC as a Solution (10 min)
   - EDHOC security and non-security objectives (John Mattsson)
   - Protocol design (John Mattsson)

4. Analysis of Alternatives (20 min)
   - Benchmarking current solutions and EDHOC
     - Message sizes (John Mattson)
     - Motivating use cases (Claes Tidestav, Mališa Vučinić, Jesús Sánchez-Gómez)

Problem Statement – Background

- Lack of lightweight AKE for OSCORE (next slide)

- Common setting: CoAP communication where at least one end is constrained
  - E.g. CoAP over multiple hops, last hop(s) over low data rate radio technology
  - OSCORE provides lightweight communication security but lacks a matching AKE

- Enable incremental addition of security
  - PSK (w/o PFS) → PSK with PFS → RPK → Certificate

OSCORE – Background

— draft-ietf-core-object-security
— Extension to CoAP (RFC 7252)
— Protects message exchange between CoAP endpoints
— Uses COSE (RFC 8152) encrypt, sign, HKDF structures
— Small addition to message overhead, memory, code

— IETF WGs
  — CoRE, ACE, 6TiSCH, LPWAN
— Other IoT fora
  — OMA SpecWorks
  — Open Connectivity Foundation
  — Fairhair Alliance

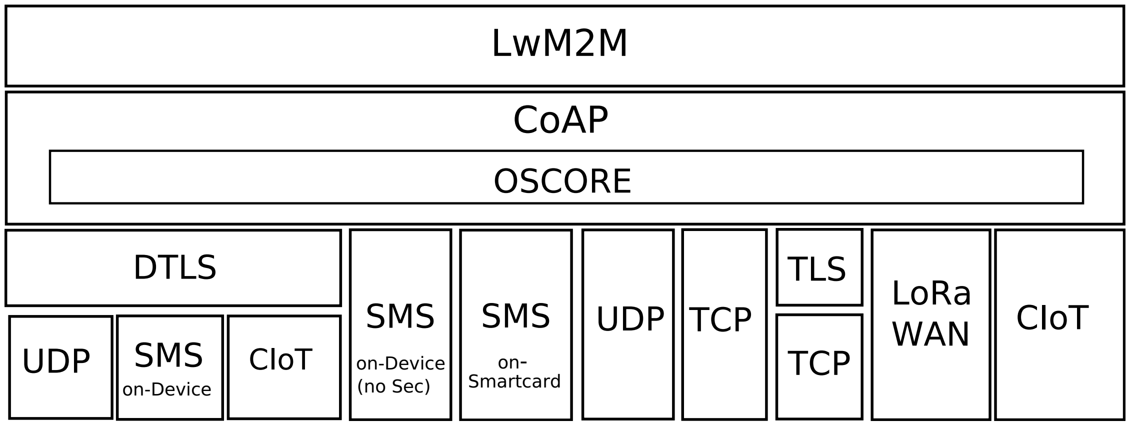


Figure: 4.-1 The Protocol Stack of the LwM2M Enabler

Figure from
OMA SpecWorks
LwM2M
Transport Bindings
Version: 1.1

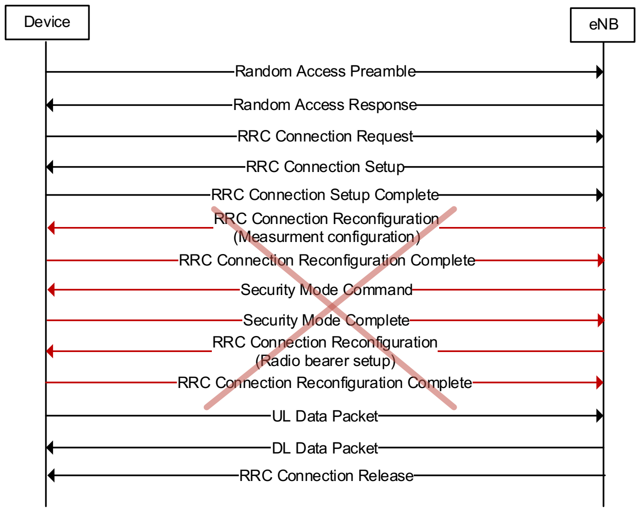
Motivating Use Cases

- Cellular IoT / Narrowband-IoT (NB-IoT)
- 6TiSCH
- LoRaWAN

Next: Overview of these use cases

Later in this slide set: Benchmarking current solutions and EDHOC applied to these use cases

Motivating Use Case – NB-IoT (1/2)



- Low cost and enhanced coverage machine type communication devices
- Cellular licensed spectrum, low data rates

NB-IoT basic design objectives

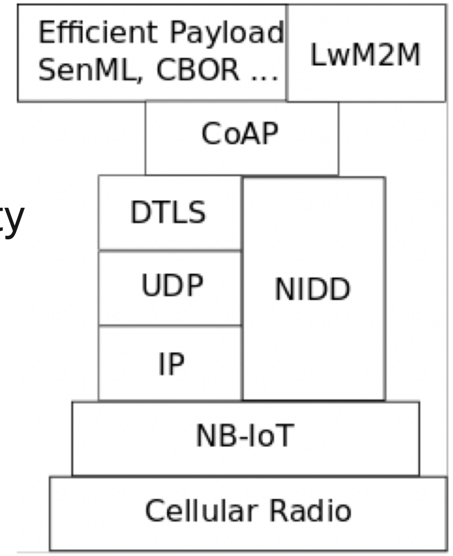
- Support of operation in extreme coverage conditions.
- Support of device battery life of 10 years or more.
- Support of low device complexity and cost.
- Support a high system capacity of thousands of connected devices per square kilometer.

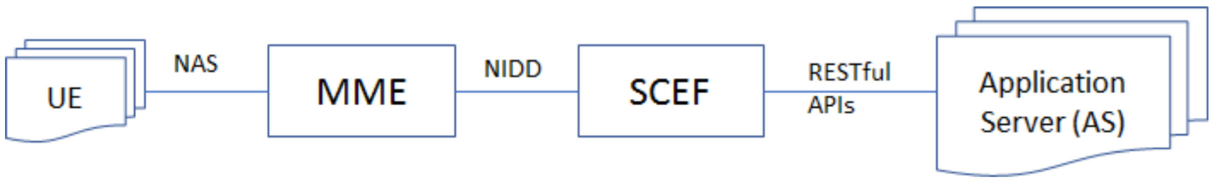
NB-IoT characteristics

- Reduced base band processing, memory and RF enables low complexity device implementation.
- A lightweight setup minimizes control signaling overhead to optimize power consumption.
- In-band, guard band, stand-alone deployment: efficient use of spectrum and NW infrastructure
- Licensed spectrum allows high device transmit power, which in combination with low data rates causes high per-byte energy consumption for uplink transmissions

Motivating Use Case – NB-IoT (2/2)

- CoAP runs over mix of transports (including non-IP)
- OSCORE provides lightweight communication security solution between AS and UE (device)
- Lightweight AKE for OSCORE needed for incremental addition of security





Figures from OMA SpecWorks White Paper
LwM2M 1.1: Managing Non-IP Devices in Cellular IoT Networks

UE = User Equipment
MME = Mobility Management Entity
SCEF = Service Capability Exposure Function
NIDD = Non-IP Data Delivery
NAS = Non Access Stratum

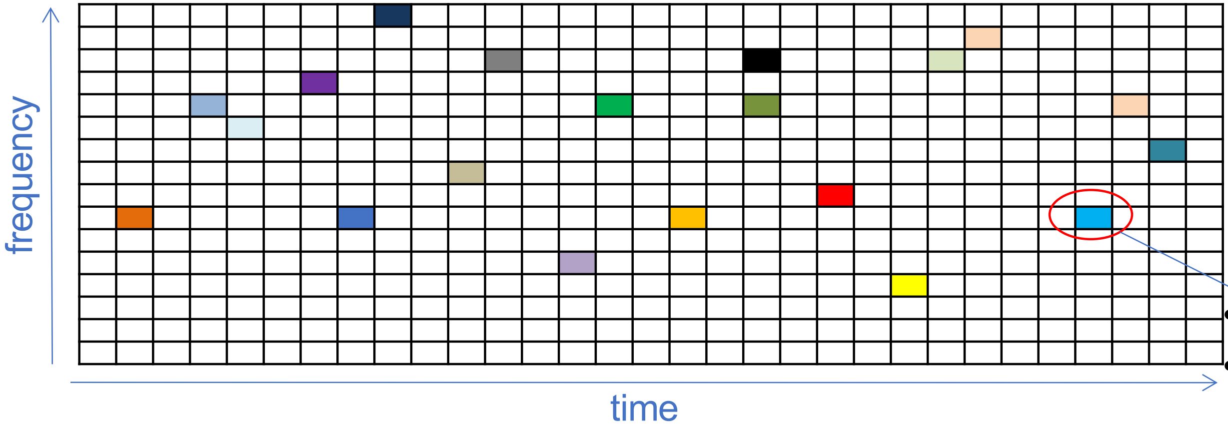
Motivating Use Case – 6TiSCH (1/2)

Figure 3. Toxic Gas Cabinets at Analog Devices' Wafer Fab are Closely Monitored to Ensure Uptime

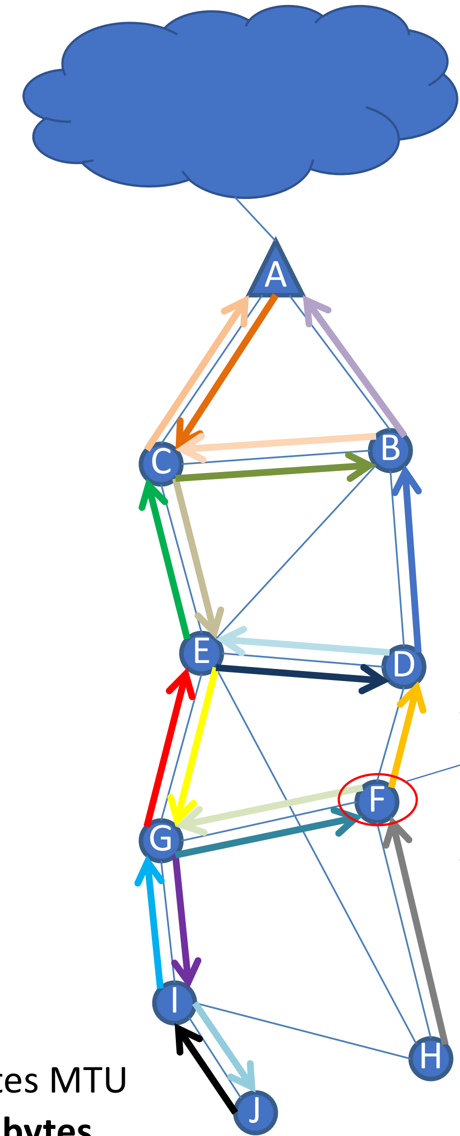
Figure 4. Dense Metal and Concrete—Wireless Nodes Must Perform Reliably Even When Located Among Metal Equipment and Gas Distribution Pipes

Figure 1. Sensors Anywhere—Low Power Wireless Sensor Nodes Powered Perpetually by Harvested Energy, Such as This Thermal-Harvested Wireless Temperature Sensor from ABB, Can Be Placed Optimally to Gain Additional Data in an Industrial Setting

- IPv6-compliant Industrial Internet of Things solution, IETF WG
- Based on Time-slotted Channel Hopping and IEEE 802.15.4 radios
- Non-IP predecessor
  - >76,000 networks deployed,
  - >14 billion operating hours logged



- 127 bytes MTU
- **60 - 80 bytes** available above UDP

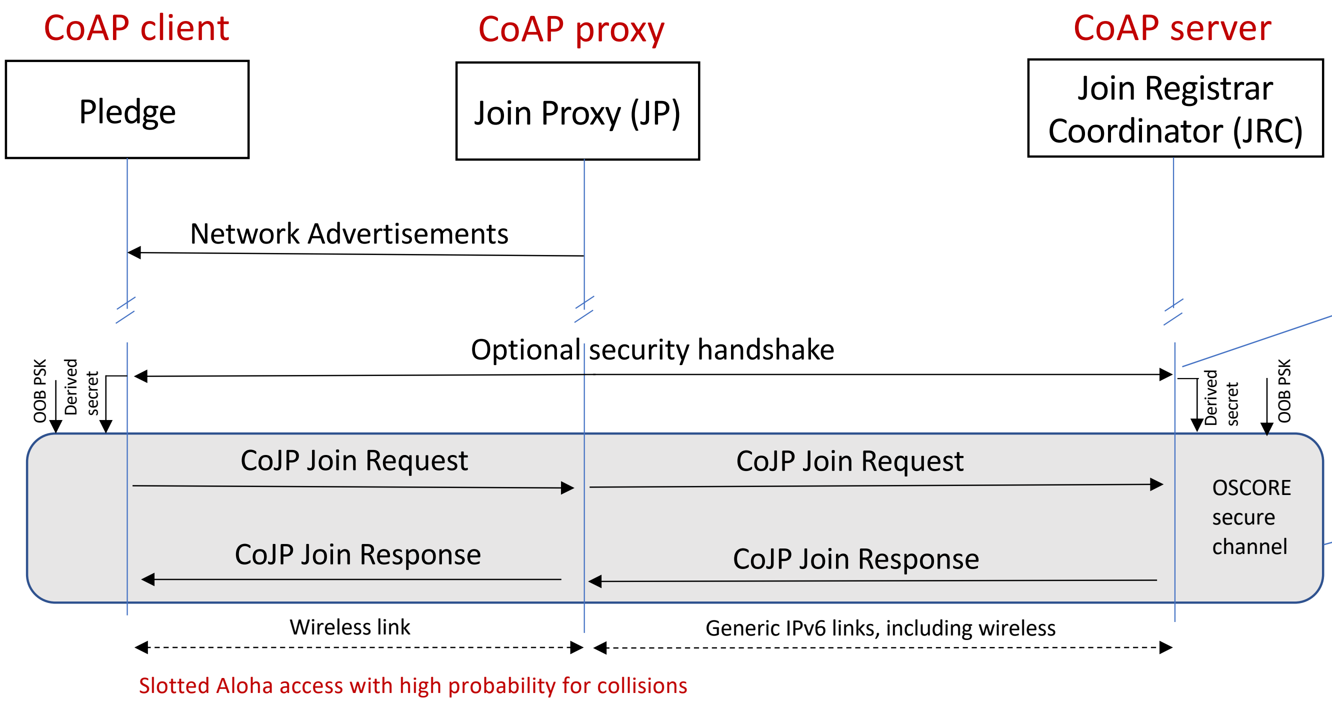


- **10s of KBs** of data memory
- **100s of KBs** of program memory

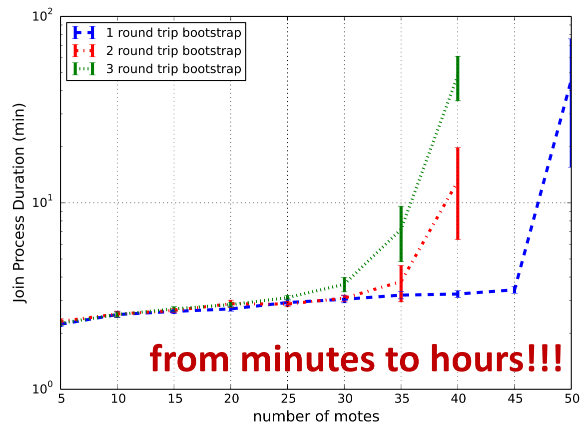
Motivating Use Case – 6TiSCH (2/2)

Network Formation Phase

- 126 B/s of shared bandwidth available for the whole network
- Number of nodes joining
- Number of L2 **frames** exchanged for network access authentication



Time installers need to spend on site



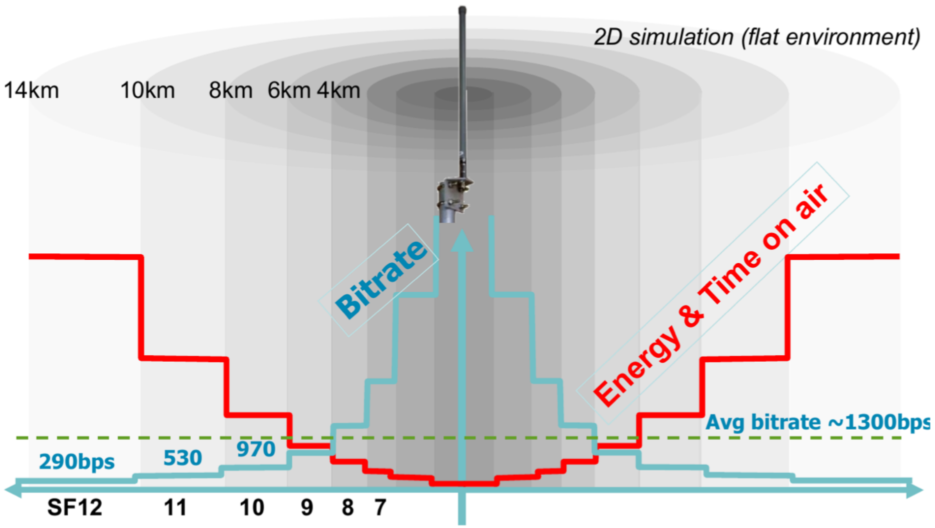
draft-ietf-6tisch-dtsecurity-zerotouch-join
normative dependence on EDHOC

RPK or PSK with PFS

draft-ietf-6tisch-minimal-security

Motivating Use Case – LoRaWAN (1/2)

- LoRaWAN employs unlicensed radio frequency bands
- Uses the 868 MHz ISM band in Europe regulated by ETSI EN 300 220
- **Time-on-Air**: The amount of time that the antenna is radiating power to transmit a packet
- After every transmission, there is a **Back-off time** period called **Duty Cycle**
  - Typical Duty Cycle in Europe is 1%
- Also, due to the regulations, the **maximum payload size** is limited for each LoRaWAN DataRate configuration

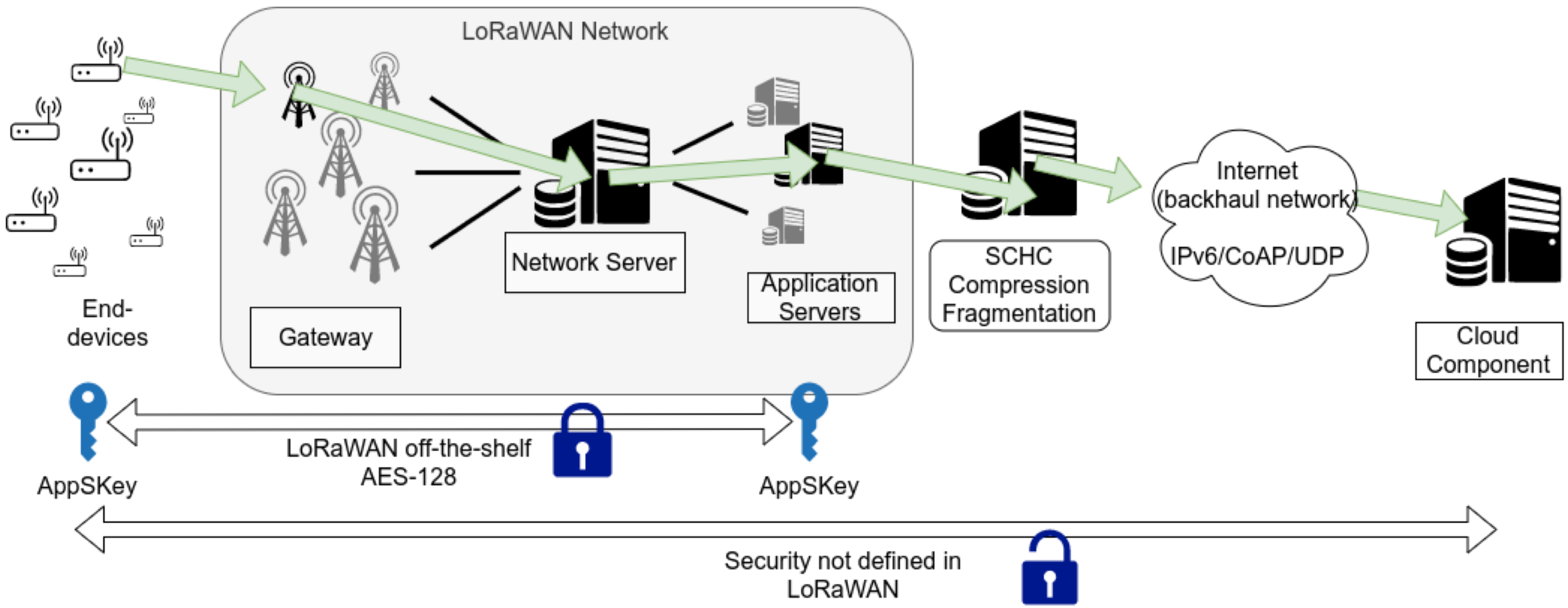


DataRate	*M*	*N*
0	59	51
1	59	51
2	59	51
3	123	115
4	230	222
5	230	222
6	230	222
7	230	222
8:15	Not defined	

Table 7: EU863-870 maximum payload size

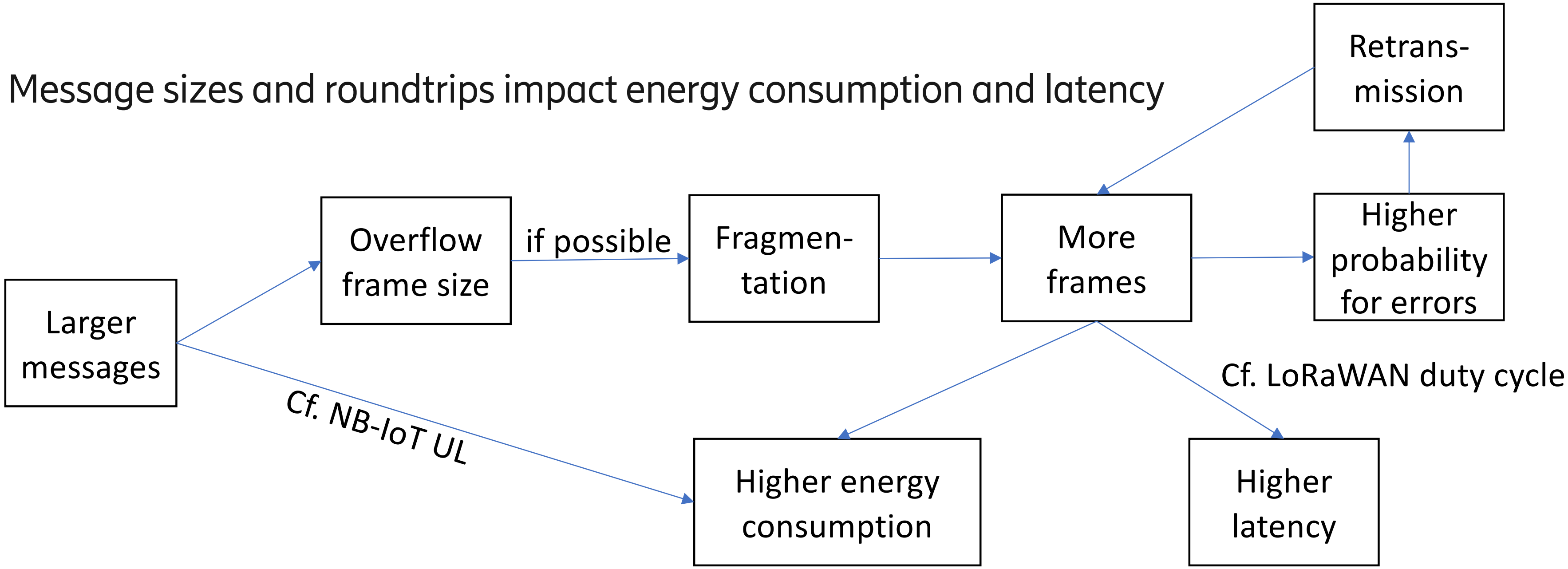
Motivating Use Case – LoRaWAN (2/2)

- LoRaWAN (v1.0) security employs a preprovided root key: *AppKey*. After deployment, a pair of session symmetric keys are derived: *AppSKey* and *NwkSKey*. These keys employ AES-128.
  - Security outside of the LoRaWAN network is not defined in LoRaWAN specification.



Constrained Characteristics

— Message sizes and roundtrips impact energy consumption and latency



— Memory and code footprint (specification complexity) impact suitable device range (cf. 6TiSCH deployed devices)

Requirements on EDHOC Use Cases

Requirements of EDHOC use cases

OSCORE related requirements:

— Agreed shared secret (OSCORE Master Secret) with a good amount of randomness
— Agreed key identifiers (Sender IDs of peer endpoints)
— Support for the same transport as OSCORE (CoAP over foo)

Incremental addition of security:

— Support for authentication based on PSK, RPK, Certificates
— Forward secrecy (ECDHE)
— Crypto agility

Performance and deployment constraints

— Simple protocol, few options
— Given that, as few round trips as possible
— Given that, as small messages as possible
— Small footprint, build on existing OSCORE/COSE code and reuse IETF IoT primitives
— Small memory, fit into low-end chipsets
— Limited processing

EDHOC as a Solution

EDHOC – Security and Non-Security Objectives (now properties)

- Stanislav's CFRG review gives a good overview
  - https://mailarchive.ietf.org/arch/msg/cfrg/2OY2om1FjhNNBmUzwYJroHv7eWQ

- Main security properties from SIGMA-I: PFS, mutual authentication, identity protection, KCI ...
- Credentials under signature, which is good to prevent DSKS-type attacks
- Transcript hashes used in key derivation and external_aad
- When PSK is used session keys are derived from both ECDH Secret and PSK.
- Simple cipher suite negotiation with downgrade protection

- Formal verification by Alessandro Bruni et al. (IT-University of Copenhagen)

- Simplicity: Same COSE algorithms and IANA registries as OSCORE and Group OSCORE.
- Small code footprint: reuses CBOR, COSE encrypt and sign structures, COSE HKDF Context
- Contrained: COSE constructs especially suitable for IoT incl. CCM*, kid, x5t, ...
  - Certificate/RPK do not need to be transported in message

- CoAP for reliable ordered transport, handling message duplication, fragmentation, DoS, ...

EDHOC – Protocol Design

EDHOC with Asymmetric Keys

- EDHOC messages are sequences of CBOR elements.
- The first element of message_1 is an int specifying the method type: asymmetric, symmetric, error

```
Party U                                                          Party V
|          TYPE, C_U, SUITES_U, SUITE_U, X_U, UAD_1                    |
+------------------------------------------------------------------>|
|                             message_1                               |
|                                                                     |
| C_U, C_V, X_V, AEAD(K_2; ID_CRED_V, Sig(V; CRED_V, aad_2), UAD_2) |
|<------------------------------------------------------------------+
|                             message_2                               |
|                                                                     |
|     C_V, AEAD(K_3; ID_CRED_U, Sig(U; CRED_U, aad_3), PAD_3)         |
+------------------------------------------------------------------>|
|                             message_3                               |
```

EDHOC with Asymmetric Keys

- Two explicit connection identifiers C_U and C_V (one for each direction).
- If EDHOC is used for OSCORE, C_U and C_V are reused as identifiers in OSCORE.

```
Party U                                                          Party V
|             TYPE, C_U, SUITES_U, SUITE_U, X_U, UAD_1                 |
+--------------------------------------------------------------------->|
|                              message_1                               |
|                                                                      |
| C_U, C_V, X_V, AEAD(K_2; ID_CRED_V, Sig(V; CRED_V, aad_2), UAD_2)   |
|<---------------------------------------------------------------------+
|                              message_2                               |
|                                                                      |
|      C_V, AEAD(K_3; ID_CRED_U, Sig(U; CRED_U, aad_3), PAD_3)         |
+--------------------------------------------------------------------->|
|                              message_3                               |
```

EDHOC with Asymmetric Keys

- Verification of a common preferred cipher suite
  - (AEAD algorithm, ECDH algorithm, ECDH curve, signature algorithm, signature algorithm parameters)
  - Cipher suites are identified with a pre-defined int or an array of COSE algorithms (0 or [12, -27, 4, -8, 6])

```
Party U                                                      Party V
|              TYPE, C_U, SUITES_U, SUITE_U, X_U, UAD_1             |
+------------------------------------------------------------------>|
|                             message_1                             |
|                                                                   |
| C_U, C_V, X_V, AEAD(K_2; ID_CRED_V, Sig(V; CRED_V, aad_2), UAD_2) |
|<------------------------------------------------------------------+
|                             message_2                             |
|                                                                   |
|     C_V, AEAD(K_3; ID_CRED_U, Sig(U; CRED_U, aad_3), PAD_3)       |
+------------------------------------------------------------------>|
|                             message_3                             |
```

EDHOC with Asymmetric Keys

— Two ephemeral public keys X_U and X_V (x-coordinates only)

```
Party U                                                        Party V
|            TYPE, C_U, SUITES_U, SUITE_U, X_U, UAD_1                |
+------------------------------------------------------------------->|
|                            message_1                               |
|                                                                    |
| C_U, C_V, X_V, AEAD(K_2; ID_CRED_V, Sig(V; CRED_V, aad_2), UAD_2) |
|<-------------------------------------------------------------------+
|                            message_2                               |
|                                                                    |
|     C_V, AEAD(K_3; ID_CRED_U, Sig(U; CRED_U, aad_3), PAD_3)        |
+------------------------------------------------------------------->|
|                            message_3                               |
```

EDHOC with Asymmetric Keys

- Unprotected application data (UAD_1, UAD_2) can be used e.g. to transfer authorization tokens.
- Protected application data (PAD_3) can be used to transfer application data.

```
Party U                                                          Party V
|             TYPE, C_U, SUITES_U, SUITE_U, X_U, UAD_1              |
+------------------------------------------------------------------>|
|                             message_1                             |
|                                                                   |
| C_U, C_V, X_V, AEAD(K_2; ID_CRED_V, Sig(V; CRED_V, aad_2), UAD_2) |
|<------------------------------------------------------------------+
|                             message_2                             |
|                                                                   |
|      C_V, AEAD(K_3; ID_CRED_U, Sig(U; CRED_U, aad_3), PAD_3)      |
+------------------------------------------------------------------>|
|                             message_3                             |
```

EDHOC with Asymmetric Keys

- Two COSE Encrypt0 objects protected with two different keys
  - K_2 and K_3 derived from the Diffie-Hellman secret and transcript hashes

```
Party U                                                          Party V
|              TYPE, C_U, SUITES_U, SUITE_U, X_U, UAD_1                |
+--------------------------------------------------------------------->|
|                              message_1                               |
|                                                                      |
| C_U, C_V, X_V, AEAD(K_2; ID_CRED_V, Sig(V; CRED_V, aad_2), UAD_2)    |
|<---------------------------------------------------------------------+
|                              message_2                               |
|                                                                      |
|      C_V, AEAD(K_3; ID_CRED_U, Sig(U; CRED_U, aad_3), PAD_3)         |
+--------------------------------------------------------------------->|
|                              message_3                               |
```

EDHOC with Asymmetric Keys

- Certificates or RPK identifiers are sent in ID_CRED_V and ID_CRED_U.
- RPK identified with a COSE kid
- Makes use of draft-ietf-cose-x509
- Certificates are identified with x5t, x5u, x5chain, x5bag

```
Party U                                                          Party V
|             TYPE, C_U, SUITES_U, SUITE_U, X_U, UAD_1             |
+----------------------------------------------------------------->|
|                               message_1                           |
|                                                                   |
| C_U, C_V, X_V, AEAD(K_2; ID_CRED_V, Sig(V; CRED_V, aad_2), UAD_2) |
|<------------------------------------------------------------------+
|                               message_2                           |
|                                                                   |
|      C_V, AEAD(K_3; ID_CRED_U, Sig(U; CRED_U, aad_3), PAD_3)      |
+----------------------------------------------------------------->|
|                               message_3                           |
```

EDHOC with Asymmetric Keys

- Two COSE_Sign1 objects, signed by Party V and Party U.
- The signatures covers the Certificate or RPK (CRED_V, CRED_U)

```
Party U                                                       Party V
|          TYPE, C_U, SUITES_U, SUITE_U, X_U, UAD_1                  |
+------------------------------------------------------------------>|
|                         message_1                                  |
|                                                                    |
| C_U, C_V, X_V, AEAD(K_2; ID_CRED_V, Sig(V; CRED_V, aad_2), UAD_2)  |
|<-------------------------------------------------------------------+
|                         message_2                                  |
|                                                                    |
|      C_V, AEAD(K_3; ID_CRED_U, Sig(U; CRED_U, aad_3), PAD_3)       |
+------------------------------------------------------------------>|
|                         message_3                                  |
```

EDHOC with Asymmetric Keys

- Signatures, MACs, and key derivation bound to all previous messages and data (aad_2).
- Transcripts of earlier messages and data are hashed to save memory.

```
Party U                                                          Party V
|            TYPE, C_U, SUITES_U, SUITE_U, X_U, UAD_1                  |
+--------------------------------------------------------------------->|
|                              message_1                               |
|                                                                      |
| C_U, C_V, X_V, AEAD(K_2; ID_CRED_V, Sig(V; CRED_V, aad_2), UAD_2)    |
|<---------------------------------------------------------------------+
|                              message_2                               |
|                                                                      |
|       C_V, AEAD(K_3; ID_CRED_U, Sig(U; CRED_U, aad_3), PAD_3)        |
+--------------------------------------------------------------------->|
|                              message_3                               |
```

EDHOC with Asymmetric Keys

- Signatures, MACs, and key derivation bound to all previous messages and data (aad_3).
- Transcripts of earlier messages and data are hashed to save memory.

```
Party U                                                          Party V
|             TYPE, C_U, SUITES_U, SUITE_U, X_U, UAD_1              |
+------------------------------------------------------------------>|
|                             message_1                             |
|                                                                   |
| C_U, C_V, X_V, AEAD(K_2; ID_CRED_V, Sig(V; CRED_V, aad_2), UAD_2) |
|<------------------------------------------------------------------+
|                             message_2                             |
|                                                                   |
|     C_V, AEAD(K_3; ID_CRED_U, Sig(U; CRED_U, aad_3), PAD_3)       |
+------------------------------------------------------------------>|
|                             message_3                             |
```

EDHOC with Symmetric Keys

- Very similar to the asymmetric case but with a different TYPE and without COSE_Sign1
- Key identifier KID in message_1
- Keys K_2 and K_3 derived from both PSK and the Diffie-Hellman secret.

```
Party U                                                       Party V
|          TYPE, C_U, SUITES_U, SUITE_U, X_U, KID, UAD_1           |
+----------------------------------------------------------------->|
|                            message_1                             |
|                                                                  |
|                 C_U, C_V, X_V, AEAD(K_2; aad_2, UAD_2)           |
|<-----------------------------------------------------------------+
|                            message_2                             |
|                                                                  |
|                     C_V, AEAD(K_3; aad_3, PAD_3)                 |
+----------------------------------------------------------------->|
|                            message_3                             |
```

EDHOC, COAP, AND OSCORE

- EDHOC are transferred as CoAP payloads.

- OSCORE parameters can be obtained from EDHOC:
  - Master Secret
  - Master Salt
  - Identifiers
  - Algorithms

- OSCORE Master Secret derived from
  - ECDH secret
  - Transcript hash
  - PSK

```
Client    Server
  |          |
  +--------->| Header: POST (Code=0.02)
  |   POST   | Uri-Path: "/.well-known/edhoc"
  |          | Content-Format: application/edhoc
  |          | Payload: EDHOC message_1
  |          |
  |<---------+ Header: 2.04 Changed
  |   2.04   | Content-Format: application/edhoc
  |          | Payload: EDHOC message_2
  |          |
  +--------->| Header: POST (Code=0.02)
  |   POST   | Uri-Path: "/.well-known/edhoc"
  |          | Content-Format: application/edhoc
  |          | Payload: EDHOC message_3
  |          |
  |<---------+ Header: 2.04 Changed
  |   2.04   |
  |          |
```

Benchmarking current solutions and EDHOC

Message Size Comparison

Comparison of message sizes of EDHOC with DTLS 1.3 handshake with connection ID.

Assumptions used for the energy measurements:

— A minimum number of extensions and offered algorithms/cipher suites

— 4 bytes key identifiers

— 1 byte connection IDs

— no DTLS message fragmentation

— DTLS RPK SubjectPublicKeyInfo with point compression.

Message Size Comparison

Message sizes in bytes

- **PSK ECDHE:**

  Factor > 4

PSK ECHDE	EDHOC-12	DTLS 1.3	EDHOC-13
Flight1	44	187	41
Flight2	46	190	46
Flight3	11	57	11
Total	**101**	**434**	**98**

- **RPK ECHDE:**

  Factor 3

RPK ECHDE	EDHOC-12	DTLS 1.3	EDHOC-13
Flight1	39	150	39
Flight2	120	373	115
Flight3	85	213	80
Total	**244**	**736**	**234**

- **Repeating question**: "is it possible to optimize a little bit more?"
- **Target size:** "as small as possible"

MTU size examples

MTU size (bytes)	Technology
12	Sigfox
16	CoAP Blockwise
32	CoAP Blockwise
47 (UL) / 49 (DL)	6TiSCH join protocol over proxy
51	LoRaWAN DR0-2 (excl. HC)
64	CoAP Blockwise
102	IEEE 802.15.4 (incl. frame overhead)
115	LoRaWAN DR3 (excl. HC)
128	CoAP Blockwise
140	SMS
. . .	. . .
222	LoRaWAN DR4- (excl. HC)

EDHOC PSK ECDHE

EDHOC RPK ECDHE

DTLS 1.3 PSK ECDHE

DTLS 1.3 RPK ECDHE

NB-IoT Energy Consumption – Assumptions

Performance for key exchange protocol is calculated for good / low coverage

Assumptions

- Power consumption 500mW (transmission), 80mW (reception)
  - Omitted power consumptions for ”light sleep” (~ 3mW) and ”deep sleep” (~ 0.015mW)
- Bitrates UL/DL: 28/170 kbps (good coverage); 0,37/2,5 kbps (low coverage)
- Energy consumption estimate includes RRC Resume procedure for transition from RRC Inactive to RRC Connected, perform operation and returning RRC Inactive

Table in next slide supported by calculations in:
https://github.com/EricssonResearch/EDHOC/blob/master/docs/NB%20IoT%20power%20consumption.xlsx

NB-IoT Energy Consumption – Estimates

Energy in mJ

- PSK ECDHE:

  Factor 2.5-3.3

- RPK ECHDE:

  Factor 2.2-2.6

Normal coverage

PSK ECHDE	EDHOC-12	DTLS 1.3
Flight1	6.3	26.7
Flight2	0.2	0.7
Flight3	1.6	8.1
Total	**19**	**47**

RPK ECHDE	EDHOC-12	DTLS 1.3
Flight1	5.6	21.4
Flight2	0.5	1.4
Flight3	12.1	30.4
Total	**29**	**64**

Normal coverage: 11 mJ to get connected

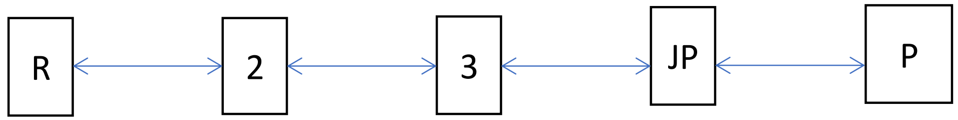
Low coverage

PSK ECHDE	EDHOC-12	DTLS 1.3
Flight1	475.7	2021.6
Flight2	11.8	48.6
Flight3	118.9	616.2
Total	**912**	**2992**

RPK ECHDE	EDHOC-12	DTLS 1.3
Flight1	421.6	1621.6
Flight2	30.7	95.5
Flight3	918.9	2302.7
Total	**1677**	**4326**

Low coverage: 306 mJ to get connected

6TiSCH Message Overhead – Assumptions



NETWORK TOPOLOGY

— R stands for DAG root
— JP stands for Join Proxy
— P stands for Pledge
— 2 and 3 are IPv6 routers that just forward packets at IPv6 layer

— L2SEC = 6 (2 bytes for signaling + 4-byte MIC)
— EUI64_SOURCE_ENCODING = 5 (Assuming nodes 2 and 3 are from the same vendor)
— N = 2 (when R sends a packet to JP, it needs to include addresses of 2 and 3 in the packet)

— 4 byte COAP HEADER OVERHEAD W/O TOKEN
— 12 byte COAP-URI-HOST 6TISCH.ARPA
— 6 byte COAP-PROXY-SCHEME
— 2 byte COAP-1B-URIPATH
— 1 byte COAP-PAYLOAD-MARKER
— 10 byte COAP-STATELESS-PROXY

6TiSCH Message Overhead – No of Frames

No. of frames (bytes)

– **PSK ECDHE:**

PSK ECHDE	EDHOC-12	DTLS 1.3
Flight1	1 (44)	4 (187)
Flight2	1 (46)	4 (190)
Flight3	1 (11)	2 (57)
Total	**3**	**10**

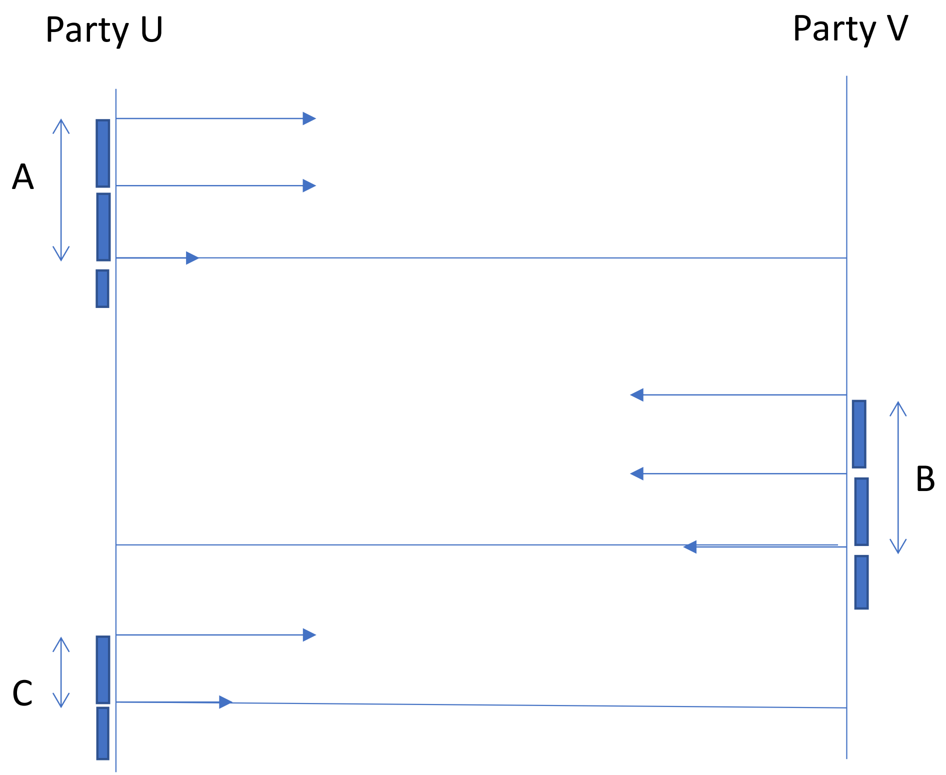
Limit for no fragmentation
Uplink: 47 bytes
Downlink: 51 bytes

– **RPK ECHDE:**

RPK ECHDE	EDHOC-12	DTLS 1.3
Flight1	1 (39)	4 (150)
Flight2	3 (120)	8 (373)
Flight3	2 (85)	5 (213)
Total	**6**	**17**

Factor 3

LoRaWAN Backoff Time Estimates



Lower limit for backoff time is estimated by A + B + C

Tables in next slide supported by calculations in:
https://github.com/EricssonResearch/EDHOC/blob/master/docs/LoRaWAN_ToA.xlsx
https://github.com/EricssonResearch/EDHOC/blob/master/docs/LoRaWAN-Backoff-Time-Lower-Bound.xls

LoRaWAN Time-on-Air and Backoff Time Estimates

Assumption: SF12 (DR0)
Fragmentation into 51 byte packets, neglecting additional headers

– **PSK ECDHE:**

Time-on-Air (s)

PSK ECHDE	EDHOC-12	DTLS 1.3
Flight1	2.6	10.7
Flight2	2.6	10.7
Flight3	1.5	4.1
Total	**6.7 s**	**25.5 s**

Duty Cycle backoff time estimates (min)

PSK ECHDE	EDHOC-12	DTLS 1.3
Flight1	4.3*)	13.8
Flight2	0*)	13.8
Flight3	0*)	4.6
Total	**4.3 min**	**32.3 min**

– **RPK ECHDE:**

RPK ECDHE	EDHOC-12	DTLS 1.3
Flight1	2.5	8.4
Flight2	7.1	21.2
Flight3	4.9	12.7
Total	**14.5 s**	**42.2 s**

RPK ECHDE	EDHOC-12	DTLS 1.3
Flight1	0*)	9.2
Flight2	8.7	32.3
Flight3	4.3	18.4
Total	**13.0 min**	**59.9 min**

*) Since no fragmentation, the duty cycle overlaps with waiting for the next message

Backup

EDHOC vs re-encoded profile of TLS 1.3 handshake

Why not a re-encoded profile of the TLS 1.3 handshake?

- A reduced TLS 1.3 handshake on par with EDHOC is most likely a new security protocol (or EDHOC!)
  - New specification needed
  - New security analysis needed
  - Not compatible with TLS 1.3
  - New code needed

  Most benefits of reuse are lost

- A TLS 1.3 profile has larger messages
- Does not fit into same MTUs as EDHOC, hence larger energy consumption and latency
  - Cf. LoRaWAN DR0-2 packet size
  - Cf. 6TiSCH join protocol over proxy
- Does not reuse COSE structures from the existing OSCORE implementation
  - Negatively impact code footprint
  - Misses out on COSE supported IoT features

6TiSCH Network Formation Time Example

- Simulation of network formation time for key exchange and join procedure in 6TiSCH network (fully-meshed) by Yasuyuki Tanaka, INRIA Paris
  https://bitbucket.org/6tisch/simulator/

- Simulation omitting CoAP and join protocol overhead
  - EDHOC-10 RPK: (1, 2, 2, 1)
  - TLS 1.3 RPK: (2, 4, 4, 1)
    - Last message is CoAP response without payload