# Hackathon report

IETF 113, LAKE WG, March 21st, 2022

# Development (1/2)

- **Develop EDHOC implementation for Rust/hacspec [1][2] (Mališa Vučinić)**
  - Provable correctness and memory safety; side-channel resistance
  - Verifiable code, but for microcontrollers
  - Portable to other environments

- **Hackathon goal: interop testing of "minimal" implementation in Rust**
  - Initiator only; Method 3 only (Static-Static); ID CRED: integer 'kid'
  - No dependencies, execute without standard library (i.e., no_std)
  - Rely on hardware acceleration where possible

[1] Denis Merigoux, Franziskus Kiefer, Karthikeyan Bhargavan. *Hacspec: succinct, executable, verifiable specification for high-assurance cryptography embedded in Rust*, https://hal.inria.fr/hal-03176482/document

[2] https://github.com/hacspec/hacspec

# Development (2/2)

- **Available at Hackathon opening**
  - Partial implementation of EDHOC
    - Unit functions and unit tests
  - Hacspec full crypto support
  - CC2538 hardware abstraction layer in Rust
  - nRF52840 hardware abstraction layer in Rust

- **Available at Hackathon closing [3]**
  - API
  - Protocol ”state machine”
  - Multi-target build support
  - AES and SHA-256 hardware-accelerated for CC2538


[3] https://github.com/openwsn-berkeley/edhoc-rs ***branch: ietf113-hackathon***

# Testing

- **Aligned with the Editor's copy of EDHOC → to-be version -13**
  - Mališa Vučinić (INRIA): Rust/hacspec - Initiator
  - Marco Tiloca (RISE): Java (*Eclipse Californium)* - Responder

- **Tested configuration**
  - Cipher suite 2
  - Method 3 (Static-Static)
  - Credential Type (CCS-CCS)
  - ID Credential Type (integer 'kid')
  - Minimum message_2 size (45 bytes)
  - Correctly completed EDHOC execution

```
EDHOC Message 2 (45 bytes):
58 2a 41 80 92 3a 20 81 0e 09
3b f7 de a3 df bb af 3a 98 fa
29 b4 45 66 67 d6 82 c0 3c bd
e0 b7 4c bb 01 84 94 67 a7 da
01 45 6c 1e 00
```

```
Completed processing of EDHOC Message 3

OSCORE Master Secret (16 bytes):
3e e4 e7 9c cf b7 b1 c9 1c 9f
32 09 6f 14 58 a6

OSCORE Master Salt (8 bytes):
92 47 22 27 7c 72 f3 a8
```

```
MP341-PRO:edhoc-rs malishav$ cargo run --example coapclient
   Compiling edhoc v0.1.0 (/Users/malishav/Software/edhoc-rs)
    Finished dev [unoptimized + debuginfo] target(s) in 0.97s
     Running `target/debug/examples/coapclient`
Client request: coap://31.133.128.122:5683/.well-known/edhoc
response_vec = [58, 2a, 41, 80, 92, 3a, 20, 81, 0e, 09, 3b, f
7, de, a3, df, bb, af, 3a, 98, fa, 29, b4, 45, 66, 67, d6, 82
, c0, 3c, bd, e0, b7, 4c, bb, 01, 84, 94, 67, a7, da, 01, 45,
 6c, 1e, 00]
OSCORE_Secret = [3e, e4, e7, 9c, cf, b7, b1, c9, 1c, 9f, 32,
09, 6f, 14, 58, a6]
OSCORE_Salt = [92, 47, 22, 27, 7c, 72, f3, a8]
```

# Next steps

- **More implementations are currently under update**
  - Christian Amsüss: Python (*aiocoap*)
    - Building on the *py-edhoc* from Timothy
  - Stefan Hristozov: C
  - Timothy Claeys: Python (*py-edhoc*) ; C
- **Run more interop tests soon**
  - As more implementations get updated / available
  - Consider use of message_4 and error messages
- **Possible side-testing**
  - (EDHOC message_3) + (first OSCORE-protected request) ➔ Single request on the wire
  - https://datatracker.ietf.org/doc/draft-ietf-core-oscore-edhoc/

Thank you!

# Backup

# Interop test configurations

- **Tested configurations – [Cipher suite, Method, Cred Type, ID Cred Type]**
  - 1 implementation pair: [2, 3, CCS, kid (integer)] → Minimum message_2 size (45 bytes)
  - TBD implementation pairs: (TBD, TBD, TBD)
  - TBD implementation pair:
    - (TBD, TBD, TBD) ; (TBD, TBD, TBD) ; (TBD, TBD, TBD)
    - (TBD, TBD, TBD) ; (TBD, TBD, TBD) ; (TBD, TBD, TBD)
    - (TBD, TBD, TBD) ; (TBD, TBD, TBD)

# Interop test results

- **1 tested implementation pairs – 0 in both directions (*)**
  - Marco with:
    - Mališa (Initiator) → Success
    - TBD
    - TBD (*)
    - TBD (*)
  - TBD with TBD (*)
  - TBD with TBD