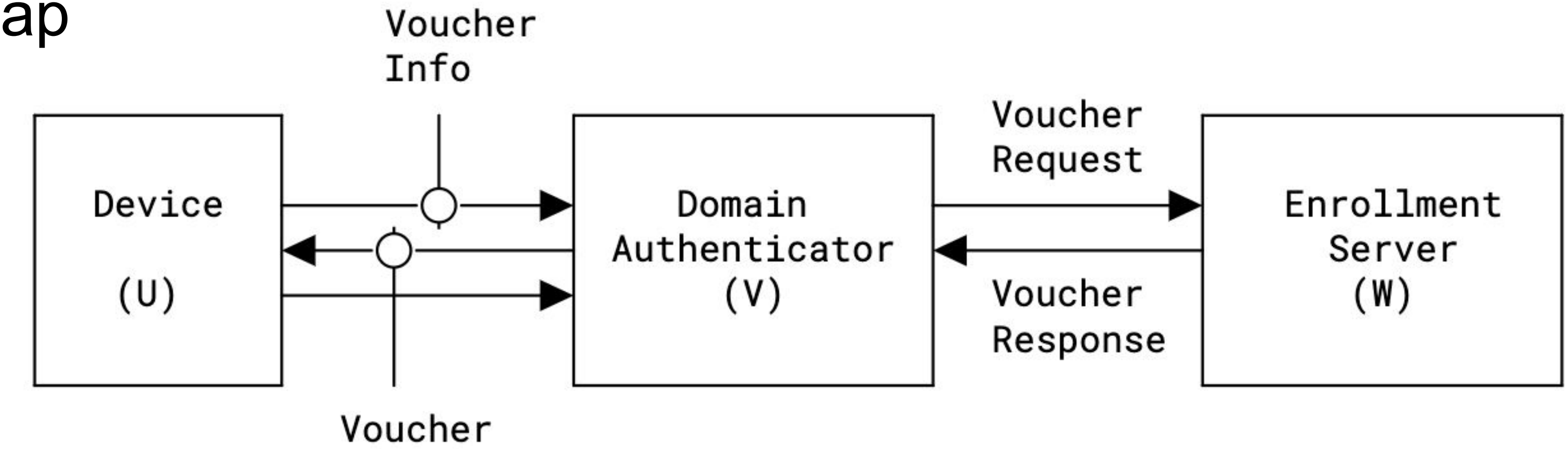
Implementer Feedback: Lightweight Authorization using EDHOC

draft-ietf-**lake-authz** (a.k.a. **zero-touch** authorization)
https://github.com/openwsn-berkeley/edhoc-rs

Geovane Fedrecheski, Inria

IETF 118, LAKE WG | Prague, November 6, 2023

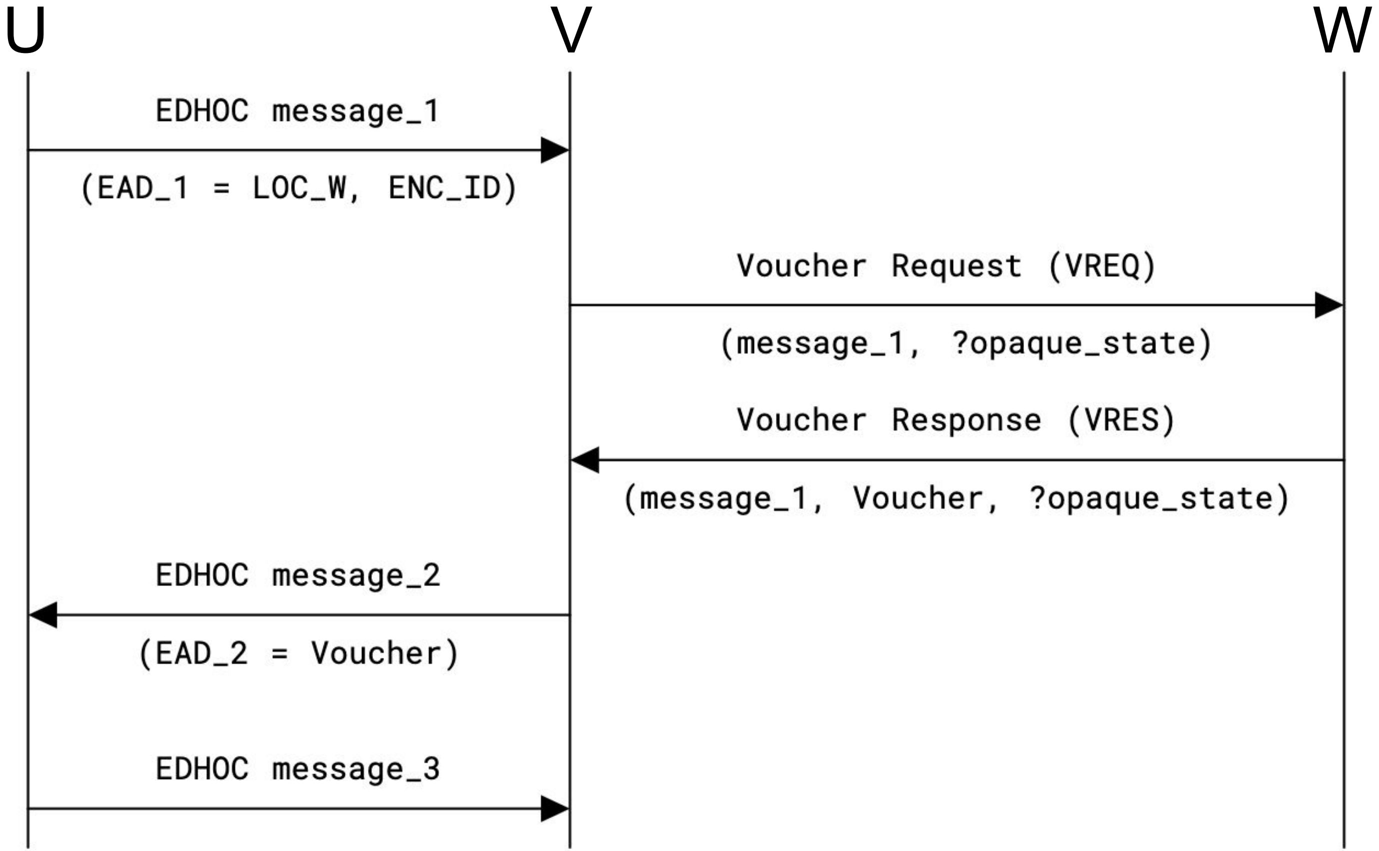
Recap



- The **Device (U)** wants to enroll into a domain over a constrained link
- The Device and **Domain Authenticator (V)** mutually authenticates and authorizes each other
- The procedure is assisted by an **Enrollment Server (W)** located in a non-constrained network

Adapted from: https://datatracker.ietf.org/meeting/117/materials/slides-117-lake-lightweight-authorization-for-edhoc-second-version

Recap


U
V
W
EDHOC message_1
(EAD_1 = LOC_W, ENC_ID)
Voucher Request (VREQ)
(message_1, ?opaque_state)
Voucher Response (VRES)
(message_1, Voucher, ?opaque_state)
EDHOC message_2
(EAD_2 = Voucher)
EDHOC message_3

Implementation: on top of edhoc-rs[1]

- A microcontroller-optimized implementation of EDHOC in Rust
  - no_std, no heap, inline CBOR encoding
- Effort towards formal verification with hax[2]
- Configurable crypto backends
- Skeleton for EAD handlers

[1] https://github.com/openwsn-berkeley/edhoc-rs

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

Status: lake-authz in edhoc-rs

Done ✅

- Preparation and processing of:
  - EAD_1, EAD_2, Voucher_Request, and Voucher_Response
- Validation with test vectors (traces)
- Fields for stateless operation of V (opaque state)
- Have V send CRED_V by value in EDHOC message_2
- Mocked W (runs alongside V)

To-do ➡️

- Implement W, have V communicate with W, authenticate V and W
- Build a demo 🤖

message_2: CRED_V by value

In many cases, **EDHOC** only sends ID_CRED_X by reference

This requires pre-provisioning credentials in I and R

lake-authz proposes “zero touch” network join: avoid pre-provisioning

While lake-authz addresses that CRED_V can be sent over the air

Implementers would benefit from more direct guidance

Possible action:

- add clear requirement that “implementations SHOULD support sending credentials by value”
- add considerations on increased message sizes (60-90 bytes for RPK)

message_2: processing w/ respect to CRED_V

The Voucher is verified by re-computing:

```
    Voucher = bstr .cbor EDHOC-Expand(PRK, info, length)
```

Where `info` contains `CRED_V`

Since U trusts W, and the Voucher (emitted by W) is trusted, then U can trust V

In other words, `CRED_V` is now considered valid, and can be used in the remaining EDHOC processing.

Possible action: make it more clear that the Voucher helps U in trusting CRED_V

Computation of K_1 and IV_1

Draft excerpt:

```
IV_1 = EDHOC-Expand(PRK, info, length*) uses the following input to
the info struct:
- (...)
- length** is length of nonce of the EDHOC AEAD algorithm in bytes
```

Comment: length* happens to have the same value of length**, but the text is only explicit about length**

message_3: EAD handler and ID_CRED_I

EAD handling:

- there is "core" EDHOC handling and EAD handling
- how to trigger EAD handling without an EAD_3? (aka should we have an EAD_3?)

processing ID_CRED_I (usually a reference):

- this is Trust On First Use (TOFU)*
- but given that W trusts U, V should be able to trust U
- however, the Voucher is not bound to CRED_U
- question: should it have such a binding?

* https://www.ietf.org/archive/id/draft-tiloca-lake-edhoc-implem-cons-00.html#name-trust-models-for-learning-n

Final remarks

Comments:

- EDHOC's EAD mechanism works well for extensibility
- Reuse of EDHOC primitives helps a lot
- Some clarifications can be done in the draft
- Questions to discuss regarding message_3

Plans:

- Build a demo (would need a W)
- Interop testing would be cool (idem)