Implementation Considerations for EDHOC

draft-tiloca-lake-edhoc-implem-cons-00

Marco Tiloca, RISE

IETF 118 Meeting – Prague – November 6th, 2023
Motivation

- While developing the EDHOC protocol in [1], a number of side topics came up
  - Those were rightly considered out of scope for EDHOC itself
  - Not elaborated in draft-ietf-lake-edhoc, which rightly focuses on the actual protocol

- Practically, implementors have to deal with those side topics
  - When building an application using EDHOC or an “EDHOC library”
  - Related implementation guidelines would be useful

- A potential Informational document was discussed in previous (interim) meetings

- Also in the latest WG Charter: … the working group will work on an Informational document gathering implementation considerations and guidance for the base protocol specification.

Just released version -00

Focus on three main topics

1. Handling of EDHOC sessions or application keys that have become invalid

2. Different trust models for learning new authentication credentials on-the-fly

3. Side-processing of EDHOC messages
   - Fetching and validation of authentication credentials
   - Processing of EAD items, possibly influencing the validation of authentication credentials
Topic 1 – Purging and cleaning up

› Most likely, only the application is aware of both:
  – The completed EDHOC sessions and the derived application keys (e.g., OSCORE Security Contexts)

› Case 1 – A completed EDHOC session becomes invalid
  – E.g., the other peer’s authentication credential has been revoked
  – Purge the EDHOC session, then purge the derived application keys

› Case 2 – Application keys become invalid (e.g., expiration, too many uses)
  – If the keys are not persisted yet, purge the session and re-run EDHOC
  – Otherwise, if supported, run a key update procedure (e.g., KUDOS [1] for OSCORE)
  – As a last resort, purge the session and re-run EDHOC

› Case 3 – Application keys or bound access rights become invalid
  – Similar to case 2, but the trigger can also be an access token become invalid (e.g., in [2])
  – If so, a new access token is also required before EDHOC can be re-run

Topic 2 – Trusting peer’s credentials

› If already stored, an authentication credential CRED_X is also trusted
  – It is also valid, until its expiration or until a revocation notice says otherwise

› Should a peer trust an unknown CRED_X retrieved from ID_CRED_X?
  – ID_CRED_X conveys CRED_X by value, or a URI from where CRED_X can be retrieved

› #1 PRE-KNOWLEDGE-ONLY (PKO) – Never trust an unknown CRED_X
  – Authentication credentials to use have to be pre-installed by a trusted party
  – ID_CRED_X has to point to an already stored CRED_X

› #2 LEARN-ON-FIRST-USE (LOFU) – Trust and store CRED_X only if:
  – CRED_X is valid AND a compatible, trusted identifier is already stored
  – E.g., ID_CRED_X conveys a certificate by value, and its hash is already stored

› #3 TRUST-ON-FIRST-USE (TOFU) – Always trust an unknown CRED_X
  – Trust and store CRED_X, as long as it is valid
Topic 3 – Side processing of messages

The processing of (especially) EDHOC message_2 and message_3 is not linear

- A big part of it does not pertain to the core EDHOC processing and has several possible incarnations
- Yet, it is something crucial to implement for an application using EDHOC or in an “EDHOC library”

**Core EDHOC processing**

```
EDHOC message_X → Decode message_X → Retrieve the protocol state → Decrypt CIPHERTEXT_X → Verify Signature_or_MAC_X → Advance the protocol state
```

- The application can prepare and instruct a Side-Processor Object (SPO)
- The application provides the SPO to EDHOC
- At the right time, EDHOC gives control to the SPO, which takes different actions for different EDHOC messages

```
a) Retrieval and validation of CRED_X
b) Trust assessment of CRED_X
c) Processing of pre-verification EAD items
   - (a) and (c) may have to go hand-in-hand
   - (b) depends on the used trust model
```

**Shared state**

- Processing of post-verification EAD items
- EAD items to unconditionally produce for next EDHOC message

```
Divert Get back Divert Get back
EAD items for next EDHOC message
```

IETF 118 Meeting – Prague | 2023-11-06 | Page 6
Summary and next steps

Guidelines for EDHOC implementations
- Handling of EDHOC sessions or application keys that have become invalid
- Different trust models for learning new authentication credentials on-the-fly
- Side-processing of EDHOC messages
  - Fetching and validation of authentication credentials
  - Processing of EAD items, possibly influencing the validation of authentication credentials

Plan for the next version
- Add figures, mostly about the side-processing of EDHOC messages
- Add guidelines on using EDHOC with CoAP and Blockwise (RFC 7959)
  - Also together with the EDHOC + OSCORE combined request, see draft-ietf-core-oscore-edhoc
- Add an appendix with public key certificates for testing
  - draft-ietf-lake-traces provides them for Ed25519 keys; good to have also for X25519 and P-256

Comments are welcome! Any further aspects worth covering?
Thank you!

https://gitlab.com/crimson84/draft-tiloca-lake-edhoc-implem-cons