Implementation Considerations for Ephemeral Diffie-Hellman Over COSE (EDHOC)

*draft-tiloca-lake-edhoc-implem-cons-01*

Marco Tiloca, RISE

LAKE WG Interim Meeting – February 13th, 2024
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

› This draft is a proposal for an Informational document
  – Early idea presented at past interim meetings, and at IETF 118 as version -00

› 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.

Recently submitted version -01
- Updated title, now expanding the EDHOC acronym
- Adopted feedback from John, Göran, and Geovane – Thanks!
- Added ASCII art diagrams
- Editorial improvements and fixes

Focus on the same three topics
1. Handling of EDHOC sessions or application keys that have become invalid
2. Trust models for learning new authentication credentials on-the-fly
3. Branched, side-processing of incoming EDHOC messages
   - Fetching and validation of authentication credentials
   - Processing of EAD items, possibly influencing the validation of authentication credentials

Since IETF 118
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- E.g., the other peer’s authentication credential has been revoked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 session and the derived application keys, then re-run EDHOC

Case 2 – Application keys become invalid (e.g., expiration, too many uses)
- If the keys are not persisted, purge the session and the keys, then re-run EDHOC
- Otherwise, if supported, run a key update procedure (e.g., KUDOS [2] for OSCORE)
- If key update is not supported or fails, purge the session and the keys, then 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 [3])
- If so, a new access token is also required before EDHOC can be re-run

**Topic 1 – Purging and cleaning up**

- Updated Section 2 **NEW**
  - Clarified steps for the three different cases
  - Added ASCII art diagrams
  - No breaking changes

---

**Case 1: EDHOC session become invalid**

Invalid EDHOC and the application keys → EDHOC → install new application keys

**Case 2: Application keys become invalid**

Invalid application keys

Are the application keys persisted?

- **YES**
  - Is KUDOS supported?
    - **NO** → Install the updated application keys
    - **YES** → Run KUDOS

- **NO** → Delete the application keys and the EDHOC session → Rerun EDHOC

**Case 3: Application keys or access token become invalid**

Invalid token specifying CREI_A, or invalid application keys

Is the token still valid?

- **NO** → Delete the associated EDHOC sessions and the application keys derived from those → Obtain and upload a new token → Rerun EDHOC
- **YES** → The application keys are not valid anymore

Are the application keys persisted?

- **YES** → Derive and install new application keys
- **NO** → Delete the application keys and the associated EDHOC session

**Run KUDOS**

Has KUDOS succeeded?

- **YES**
  - Install the updated application keys
- **NO**
Topic 2 – Trusting peers’ 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 specified in an EDHOC message?
  – CRED_X can be specified in, or retrieved through, ID_CRED_X or an EAD item

› “NO-LEARNING” policy – Never trust an unknown CRED_X
  – Authentication credentials have to be already stored at message reception time

› “LEARNING” policy – Always trust an unknown CRED_X
  – Trust and store CRED_X, provided that it is valid

› Updated Section 3  **NEW**
  – Removed middle-ground policy, based on received feedback
  – Renamed policies as “NO-LEARNING” and “LEARNING”
  – Revised policy descriptions to be simpler and neat
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

- Decode message_2
- Retrieve the protocol state
- Decrypt CIPHERTEXT_2
- Verify Signature_or_MAC_2
- Advance the protocol state

### Topic 3 – Side processing

- 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_2
b) Trust assessment of CRED_2
c) Processing of pre-verification EAD items

- (a) and (c) may have to go hand-in-hand
- (b) depends on the used trust model

### Processing of post-verification EAD items

- EAD items to unconditionally produce for next EDHOC message

### EAD items for next EDHOC message
Topic 3 – Side processing

› Updated Section 4 **NEW**
  – Simpler processing steps in the SPO
  – Only two possible trust policies
  › “NO-LEARNING” or “LEARNING”
  – Added ASCII art diagrams

Overview of high-level interaction

Pre-Verification Side Processing
Summary and next steps

Guidelines for EDHOC implementations about:
- Handling of EDHOC sessions and derived applications keys, if become invalid
- Trust models for learning new authentication credentials on-the-fly
- Branched, side-processing of incoming EDHOC messages
  - Fetching and validation of authentication credentials
  - Processing of EAD items, which may play a role in validating authentication credentials

Possible further content to include
- Considerations on using CoAP Blockwise (RFC 7959)
  - Also together with a combined EDHOC + OSCORE request (draft-ietf-core-oscore-edhoc)
- Appendix with example certificates to plug-in for testing

Ready for a WG Adoption Call?
Thank you!

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