Guidelines for EDHOC Implementations
(possible new work?)

Marco Tiloca, RISE

LAKE WG Interim Meeting – February 7th, 2023

Motivation

- **As the EDHOC protocol was developed, a number of side topics came up**
  - While reviewing and especially when implementing *draft-ietf-lake-edhoc* [1]

- **Those were rightly considered out of scope for EDHOC itself**
  - Not discussed 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 helpful**

[1] https://datatracker.ietf.org/doc/draft-ietf-lake-edhoc/

Relevant topics (1/3)

- **Most likely, only the application is aware of <u>all</u> the following:**
  - The ongoing and completed EDHOC sessions
  - The authentication credentials of other EDHOC peers
  - The application keys established with other peers from EDHOC (e.g., an OSCORE Security Context)

- **When to invalidate a completed EDHOC session? What does this trigger?**
  - E.g., when learning that the other peer's certificate has been revoked
  - Purge the EDHOC session, then purge the application keys derived from it

- **What to do when application keys become invalid?**
  - E.g., they have reached their expiration or their key usage limit, see [2]
  - Re-run EDHOC? Or update the application keys only, e.g., with KUDOS [2] ?
  - What if EDHOC PRK_out is not persisted yet?
  - What if the EDHOC session is bound to a token for access control? [3]

[2] https://datatracker.ietf.org/doc/draft-ietf-core-oscore-key-update/
[3] https://datatracker.ietf.org/doc/draft-ietf-ace-edhoc-oscore-profile/

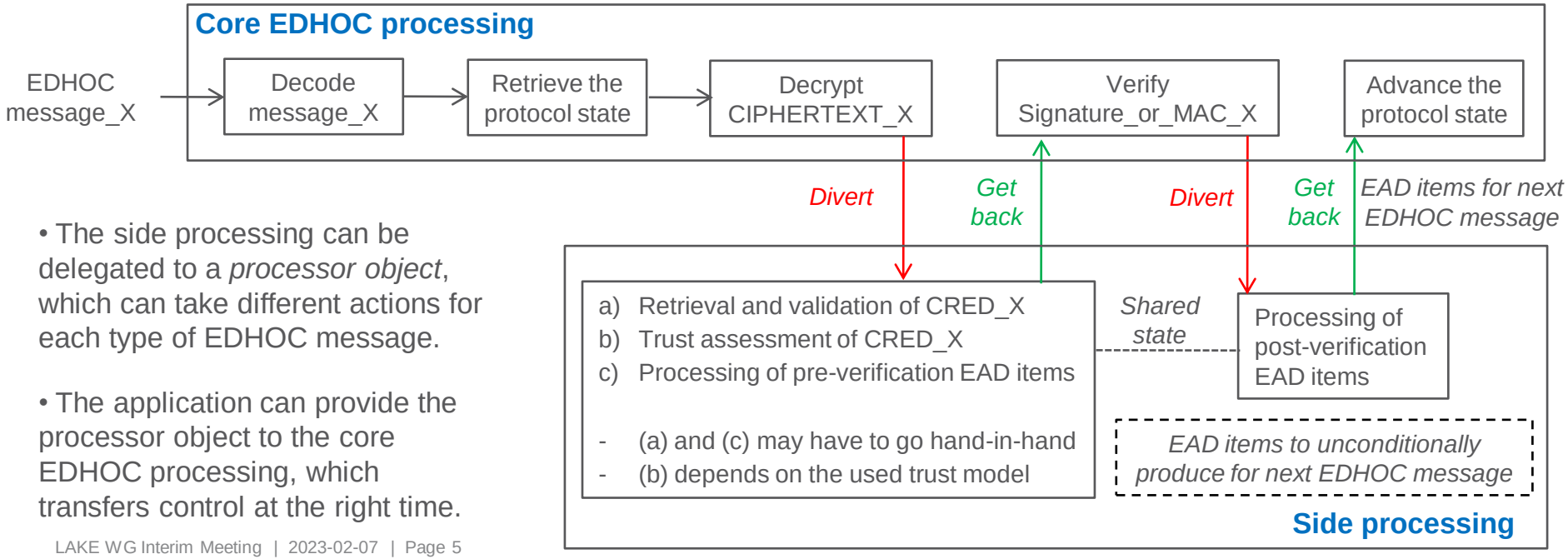
Relevant topics (2/3)

- **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 you trust a new CRED_X while running EDHOC?**
  - Typically, the new CRED_X is transported by value in ID_CRED_X

- **Trust Model 1 – Never trust a new 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
- **Trust Model 2 – Trust and store new CRED_X only if:**
  - It 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
- **Trust Model 3 – Trust and store a new CRED_X as long as it is valid (TOFU)**

Relevant topics (3/3)

- **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”



• The side processing can be delegated to a *processor object*, which can take different actions for each type of EDHOC message.

• The application can provide the processor object to the core EDHOC processing, which transfers control at the right time.

Summary and next steps

- **Guidelines for EDHOC implementations would be helpful on:**
  - Handling of EDHOC sessions become invalid
  - Handling of application keys derived from EDHOC and become invalid
  - Enforcing of different trust models for learning new authentication credentials on-the-fly
  - Branched, side-processing of EDHOC messages
    - Fetching and validation of authentication credentials
    - Processing of EAD items, that may play a role in validating authentication credentials

- **Plan to write an Informational Internet Draft for the LAKE WG to consider**

- **Is this in scope and appropriate? Any further aspects worth covering?**

Thank you!