Profiling EDHOC for CoAP and OSCORE

draft-ietf-core-oscore-edhoc-05

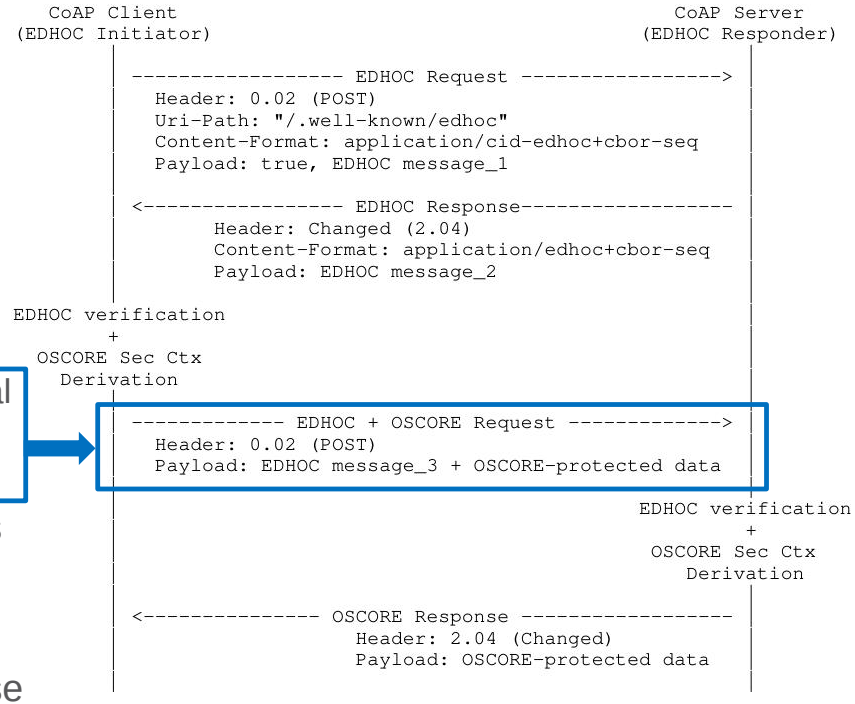
Francesca Palombini, Ericsson
Marco Tiloca, RISE
Rikard Höglund, RISE
Stefan Hristozov, Fraunhofer AISEC
Göran Selander, Ericsson

IETF 115 meeting – London – November 7th, 2022

Recap

- **EDHOC: lightweight authenticated key exchange [1]**
  - Developed in the LAKE Working Group
  - Main use: establish an OSCORE Security Context
  - Normally, two round-trips before using OSCORE

- **Scope of this document**
  - EDHOC for OSCORE, transported over CoAP
  - Optimized key establishment workflow (main item)
    - Single request with EDHOC Option, combining final EDHOC message_3 and first OSCORE-protected application request
  - OSCORE-specific processing of EDHOC messages
  - Consistent extension of EDHOC application profiles
  - Web linking for discovery of EDHOC resources
  - Performance considerations on the use of Block-wise



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

Update since IETF 114

- **Submitted v -05 before the cut-off**
- **No changes to the mechanics of the optimized workflow**
- **IANA considerations – EDHOC CoAP Option**
  - Revised and shortened text, now reasoning only about Option number 21
  - Renewed early registration of Option number 21: new expiration on 2023-11-08

Update since IETF 114

- **Performance considerations on using Block-wise with the optimized workflow**
  - Now moved to Appendix A
  - Same content as in the former Section 6 of v -04
  - Practical point: if the use of Block-wise is triggered exactly by using the optimized workflow, this has no performance advantage anymore, and the client should resort to the original workflow

- **Added security considerations**
  - In general, the server might enforce access control for its resources
  - If so, this must hold also after the EDHOC processing of the EDHOC + OSCORE request
  - Completing EDHOC per se does not grant access to a server resource
  - OSCORE-protected application requests undergo access control like if received stand-alone
  - Access control information to be provided to the server before/during the EDHOC execution

Post cut-off

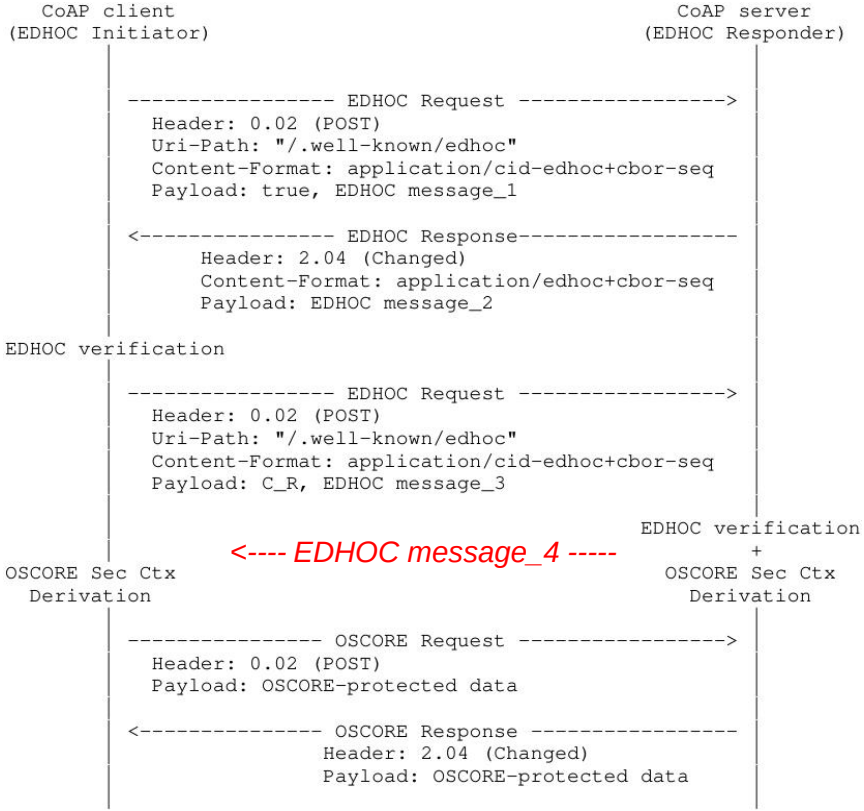
- **Section 6 defines target attributes for EDHOC resources**
- **Now these attributes can be registered in the new IANA registry defined in [2]**
- **Proposal from Carsten, also in order to not delay this document:**
  - In [2], pre-fill the new registry with the target attributes from this document (see PR at [3])
  - Ask for registration of these target attributes also in this document
  - Then, only the document that “wins the race” keeps its text about these registrations
- **Regardless, attribute names can be revised, e.g., to start with an EDHOC-related prefix**

Thoughts? Objections?

[2] https://datatracker.ietf.org/doc/draft-bormann-core-target-attr/
[3] https://github.com/cabo/core-target-attr/pull/4

Post cut-off

- **Proposal from David to extend Figure 1**
  - Also tracked in the PR #7 at [4]
  - In the original workflow, add a response to EDHOC message_3, transporting EDHOC message_4
- **The current omission is building on [5]**
  - *If EDHOC message_4 is used, or in case of an error message, it is sent from the server to the client in the payload of the response to message_3*
  - It should still be ok to not have a response at all
- **Proposal: merge PR #7 and extend the figure caption to also say like in Figure 13 of [5]:**
  - *The optional message_4 is included in this example, without which that message needs no payload.*



[4] https://github.com/core-wg/oscore-edhoc/pull/7

[5] https://datatracker.ietf.org/doc/html/draft-ietf-lake-edhoc-17#appendix-A.2

Summary and next steps

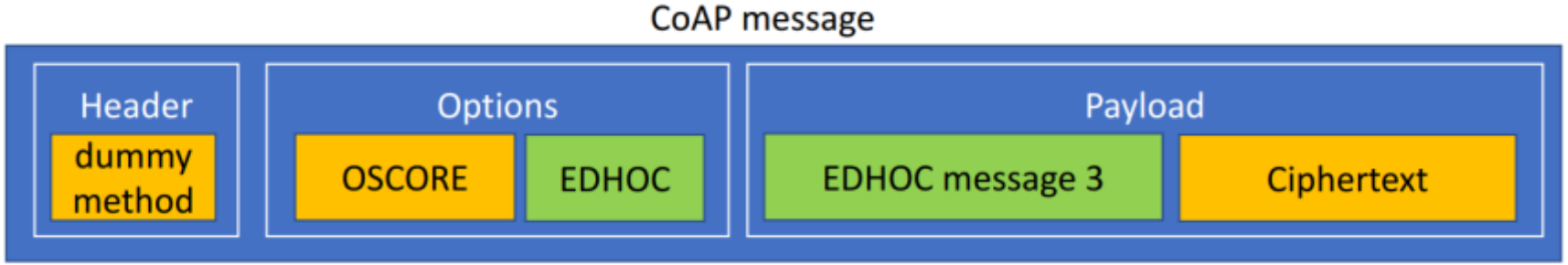
- **This document is stable, and aligned with the latest EDHOC v -17**
- **Agreed to sync with the WGLC of EDHOC in the LAKE Working Group**
  - This concluded on 2022-11-04
- **Start WGLC for this document?**

Thank you!

Comments/questions?

https://github.com/core-wg/oscore-edhoc/

EDHOC + OSCORE request




```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Ver| T |  TKL  |      Code     |          Message ID           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Token (if any, TKL bytes) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| OSCORE Option                                 | EDHOC Option  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Other Options (if any) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 1 1 1 1 1 1 1| Payload
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

On using Block-wise

- **When can the EDHOC + OSCORE request get too big because of EDHOC?**
  - Use of large ID_CRED_I in EDHOC, e.g., as a certificate chain
  - Use of large EAD items in EAD_3 as External Authorization Data
- **Client processing in Section 3.2.1**
  - Only the first inner block conveys EDHOC data and the EDHOC Option
  - Stop if the EDHOC + OSCORE request exceeds MAX_UNFRAGMENTED_SIZE
- **Server processing in Section 3.3.1**
  - Just as per RFC 7959 and RFC 8613: the EDHOC + OSCORE request is rebuilt first
- **Appendix A**
  - Performance guidelines on using Block-wise together with the EDHOC + OSCORE request
  - The Client might use inner Block-wise, but it is assumed to not use also outer Block-wise
    - Possible to fragment the application data, but not the whole EDHOC + OSCORE request

On using Block-wise

- **Client processing (Section 3.2.1)**
  - OSCORE protection of each inner block as usual
  - If the protected block is not the first one (i.e., Block1.NUM ≠ 0)
    - The client MUST NOT add the EDHOC Option, but sends the protected request as is
    - → Only the first inner block can be sent together with EDHOC data
  - If the protected block is the first one (i.e., Block1.NUM = 0) and …
    - … (EDHOC message_3 | OSCORE ciphertext) > MAX_UNFRAGMENTED_SIZE … then
    - … abort and possibly switch to the original vanilla EDHOC workflow
    - No further inner blockwise can happen once the EDHOC + OSCORE request is assembled

- **Server processing (Section 3.3.1)**
  - First re-assemble the full EDHOC + OSCORE, as per RFC 7959 and RFC 8613.

Optimized workflow and Block-wise

- **LIMIT: practical maximum size to exceed before using Block-wise**

- **When is it OK to send the EDHOC + OSCORE request?**
  - Generally, (EDHOC data) <= LIMIT is a requirement
  - If Block-wise is not used, when (Application data + EDHOC data) <= LIMIT
  - If Block-wise is used, when (1 block + EDHOC data) <= LIMIT

- **When using the EDHOC + OSCORE request, use also Block-wise if …**
  - … (Application data) > LIMIT **or** (Application data + EDHOC data) > LIMIT
  - In either case (1 block + EDHOC data) must not exceed LIMIT
  - If both conditions hold, the optimized workflow is always better in terms of RTTs

- **Corner case:** (Application data) <= LIMIT **and** (Application data + EDHOC data) > LIMIT
  - Using the EDHOC + OSCORE request would be the actual cause for using Block-wise!
  - The optimized workflow may still be not worse than the original one, but it may also be just worse
  - Under this case, the Client should not use the EDHOC + OSCORE request, as not worth it