Profiling EDHOC for CoAP and OSCORE

draft-ietf-core-oscore-edhoc-04

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

IETF 114 meeting – Philadelphia – July 26th, 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 EDHOC resources and their application profiles (through target attributes)

Update since IETF 113

Presented at the CoRE interim meeting on 2022-04-27

Updates in this slide are due to changes in EDHOC (now in its version -15)

› No more special conversion of identifiers
  – OSCORE Recipient/Sender IDs → EDHOC Connection Identifiers
  – Simple "identity" relation like in the opposite direction (defined by EDHOC)
  – When receiving the EDHOC + OSCORE request ...
    › … the server retrieves the value of 'kid' from the OSCORE Option
    › The ‘kid’ value is both the Server’s OSCORE Recipient ID and EDHOC Connection Identifier C_R

› Text and examples using the new Content-Formats
  – application/cid-edhoc+cbor-seq and application/edhoc+cbor-seq
  – The combined EDHOC + OSCORE request has still unnamed media-type

▶ “EDHOC Applicability Statement” → “EDHOC Application Profile”
Update since IETF 113

› On the “good behavior” expected from the Client
  – “With the same Server, the Client SHOULD NOT have multiple simultaneous outstanding interactions (see Section 4.7 of [RFC7252]) such that: they consist of an EDHOC + OSCORE request; and their EDHOC data pertain to the EDHOC session with the same connection identifier C_R.”
    – Changed from "MUST NOT“, based on feedback during the CoRE interim meeting in April [2].

› Revised and simplified processing of EDHOC messages
  – Selection of own EDHOC Connection Identifier (offered as own OSCORE Recipient ID).
  – Related consistency checks on incoming EDHOC messages.
  – Consistent with requirements from Section 3.3 of RFC 8613.

Update since IETF 113

› Simplified extension/consistency of EDHOC Application Profile template
  – Nothing to say anymore about conversion of OSCORE/EDHOC identifiers
  – If the EDHOC + OSCORE request is supported, the application profile of an EDHOC resource:
    › SHOULD signal the support of the EDHOC + OSCORE request
    › MUST NOT signal the support of message_4

› Revised use of web-linking to signal EDHOC Application Profiles
  – Removed target attribute related to conversion of EDHOC/OSCORE identifiers
  – Admitted multiple instances of an "ead_X" target attribute, with value the ead_label of a supported External Authorization Data (EAD) item for EAD_X in EDHOC message_X.

› Added security considerations
  – Flooding the Server with EDHOC + OSCORE combined requests is not a security problem.
    › The server does not process the same EDHOC message_3 multiple times
    › The server performs replay checks on the OSCORE-protected application request
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

- **New Section 6**
  - Guidelines on (not) 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
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) \(\leq\) LIMIT is a requirement
  – If Block-wise is not used, when (Application data + EDHOC data) \(\leq\) LIMIT
  – If Block-wise is used, when (1 block + EDHOC data) \(\leq\) 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) \(\leq\) 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
Next steps

› Add more security considerations, e.g.:
  – When using the EDHOC + OSCORE combined request, the OSCORE-protected application request has to undergo access control enforcement, like if it was received stand-alone.

› We have running code built for Eclipse Californium (Java)
  – Aligned to the latest EDHOC v -15
    › https://github.com/rikard-sics/californium/tree/edhoc-dev

› TODO: Renew early registration of EDHOC CoAP Option number (21)
  – Expiration on 2022-11-08
  – IANA: is it needed to register also the other suggested number 13? \(\rightarrow\) No need to

› Absent big issues or EDHOC changes, the next version might be good for WGLC
  – Maybe we should synch with the LAKE WG, and have it in parallel with the WGLC of EDHOC?

› Comments are reviews are welcome!
Thank you!

Comments/questions?

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