OSCORE-capable Proxies

draft-tiloca-core-oscore-capable-proxies-00

Marco Tiloca, RISE
Rikard Höglund, RISE

CoRE Interim Meeting, September 15th, 2021

Motivation

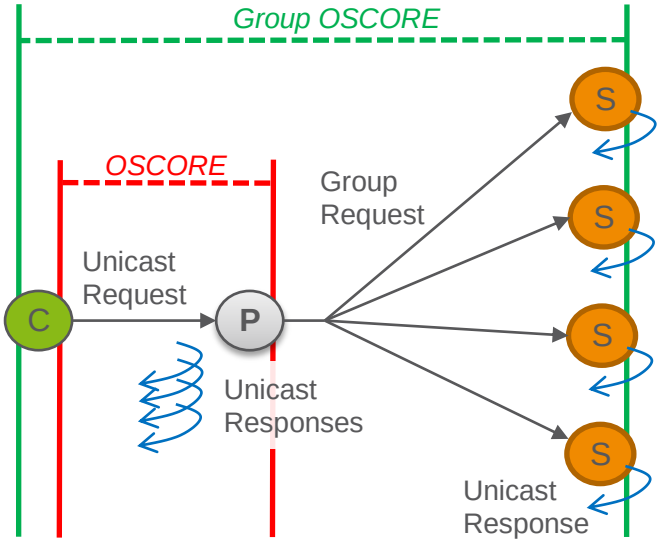
- A CoAP proxy (P) can be used between client (C) and server (S)
  - A security association might be required between C and P --- examples in next slides
- It would be good to use OSCORE between C and P
  - Especially, but not only, if C and S already use OSCORE also end-to-end
- This is not defined and not admitted in OSCORE (RFC 8613)
  - C and S are the only considered “OSCORE endpoints”
  - It is forbidden to double-protect a message, i.e., both over C ↔ S and over C ↔ P
- This started as an Appendix of *draft-tiloca-core-groupcomm-proxy*
  - Agreed at IETF 110 [1] and at the June CoRE interim [2] to have a separate draft

[1] https://datatracker.ietf.org/doc/minutes-110-core-202103081700/

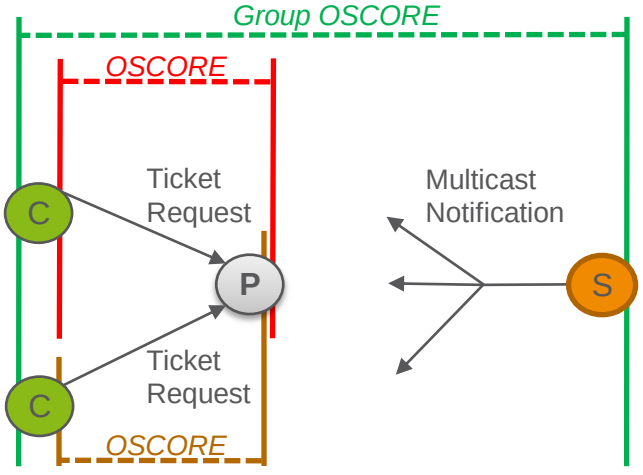
[2] https://datatracker.ietf.org/doc/minutes-interim-2021-core-07-202106091600/

Use cases

- **CoAP Group Communication with Proxies**
  - *draft-tiloca-core-groupcomm-proxy*
  - CoAP group communication through a proxy
  - Possible e2e security with Group OSCORE
  - P must identify C through a security association before forwarding a request to the group



- **CoAP Observe Notifications over Multicast, with Group OSCORE for e2e security**
  - *draft-ietf-core-observe-multicast-notifications*
  - C provides P with a Ticket Request obtained from S
  - This allows P to correctly listen to multicast notifications sent by S
  - The provisioning of the Ticket Request to P should be protected over C ↔ P



Use cases

- **OMA LwM2M Client and External Application Server**
  - *Lightweight Machine to Machine Technical Specification – Transport Binding*

    OSCORE MAY also be used between LwM2M endpoint and non-LwM2M endpoint, e.g., between an Application Server and a LwM2M Client via a LwM2M server. Both the LwM2M endpoint and non-LwM2M endpoint MUST implement OSCORE and be provisioned with an OSCORE Security Context.
  - The LwM2M Client may register to and communicate with the LwM2M Server using OSCORE
  - The LwM2M Client may communicate with an External Application Server, also using OSCORE
  - The LwM2M Server would act as CoAP proxy, forwarding outside the LwM2M domain

- **More generally, a proxy may want an OSCORE Security Context of its own**
  - E.g., it ensures the security of transport indication when OSCORE is used [3][4]

[3] https://datatracker.ietf.org/doc/draft-amsuess-core-transport-indication/

[4] https://mailarchive.ietf.org/arch/msg/core/RZH8pgyksEwtMYVE1MrPkj9opyg/

Contribution

- Twofold update to RFC 8613

1. Define the use of OSCORE in a communication leg including a proxy
   - Between origin client/server and a proxy; or between two proxies in a chain
   - Not only an origin client/server, but also an intermediary can be an “OSCORE endpoint”

2. Explicitly admit double OSCORE protection – “OSCORE-in-OSCORE”
   - E.g., first protect end-to-end over C ↔ S, then further protect the result over C ↔ P
   - Typically, at most 2 OSCORE “layers” for the same message
     - 1 end-to-end + 1 between two adjacent hops
   - Strict limit of 2 layers in v -00; this will be lifted in v -01 (later slide)

- Focus on OSCORE, but the same applies “as is” to Group OSCORE

Leg independence

Helpful during the early design phase.
It will be largely removed in v -01

› Seamless support for different configurations
  – Configurations differ on whether OSCORE is used or not in a certain communication leg

Conf. name (b2, b1, b0)	CF-0 (000)	CF-1 (001)	CF-2 (010)	CF-3 (011)
Comm. legs using OSCORE		C-P	P-S	C-P P-S

C=Client, P=Proxy, S=Server

Figure 1: Configurations without end-to-end security.

Conf. name (b2, b1, b0)	CF-4 (100)	CF-5 (101)	CF-6 (110)	CF-7 (111)
Comm. legs using OSCORE	C-S	C-S C-P (*)	C-S P-S (*)	C-S C-P (*) P-S (*)

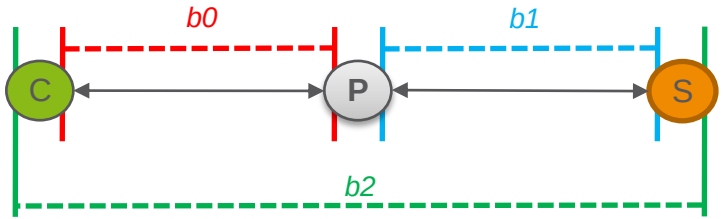
C=Client, P=Proxy, S=Server
(*) OSCORE-in-OSCORE

Figure 2: Configurations with end-to-end security.

Naming convention: CF-X

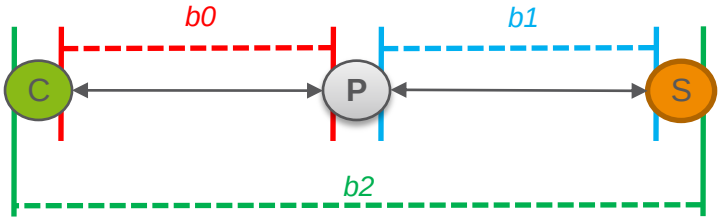
X = b0 + (2 * b1) + (4 * b2)

b0 : 1 if OSCORE over C ↔ P ; 0 otherwise
b1 : 1 if OSCORE over P ↔ S ; 0 otherwise
b2 : 1 if OSCORE over C ↔ S ; 0 otherwise



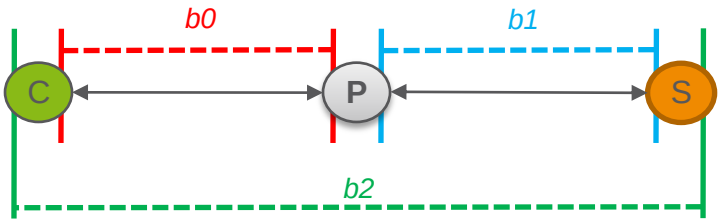
Processing mechanics (1/2)

- C request processing
  - (1) If b2 = 1, protect with OSCORE C ↔ S
  - (2) If b0 = 1, (further) protect with OSCORE C ↔ P
    - Encrypt options intended to P, e.g., Proxy-Scheme
    - Encrypt the OSCORE option from (1), if any



- P request processing
  - Visible proxy options → Forward to S
  - Absent proxy options && absent OSCORE option → Deliver to the application (if any)
  - Absent proxy options && Visible OSCORE option → Decrypt, as OSCORE C ↔ P
    - No proxy options in the decrypted request → Deliver to the application (if any)
    - Visible proxy options in the decrypted request → Forward to S
  - When forwarding to S
    - If b1 = 1, (further) protect with OSCORE P ↔ S
    - Encrypt the OSCORE option for C ↔ S, if any

Processing mechanics (2/2)



- S request processing
  - Ready to find and process 1 or 2 OSCORE layers
- S response processing
  - (1) If b2 = 1, protect with OSCORE C ↔ S
  - (2) If b1 = 1, (further) protect with OSCORE P ↔ S
    - Encrypt options intended to P
    - Encrypt the OSCORE option from (1), if any
- P response processing
  - If b1 = 1, unprotect with OSCORE P ↔ S
  - When forwarding to C
    - If b0 = 1, (further) protect with OSCORE C ↔ P
    - Encrypt possible new added options intended to C
    - Encrypt the OSCORE option for C ↔ S, if any
- C response processing
  - Reverse of request processing; ready to find and process 1 or 2 OSCORE layers

Early reactions

- Got comments from Christian and Göran during the IETF 111 week – Thanks!
  - Use cases are good, and there can be more
  - The mechanics make sense, but it's better to present it in a more general way

1. Abandon the flag-based notation and the 8 configurations
   - Good to have in the early design phases, but cumbersome for the future

2. Change presentation approach
   - OLD: heuristic and detailed step-by-step processing on client, proxy and server
     - It might be good to keep part of this as an appendix with a detailed example
   - NEW: high-level general algorithm, fitting a client, proxy or server as a message processor
     - Sketched algorithm to be finalized; it is easily applicable right-away to a chain of proxies

3. Think about having a message with more than two layers of OSCORE protection
   - From early design discussions, it is possible and very promising

Summary and next steps

- Proposed update to RFC 8613
  - Define the use of OSCORE in a communication leg including a proxy
  - Explicitly admit double OSCORE protection – “OSCORE-in-OSCORE”
  - No need for a new explicit signaling method to guide the message processing
  - Useful for CoAP group communication, external server in LwM2M, transport indication, …

- Next steps for v -01
  - Address early comments, especially present the mechanics in a more general way
  - Mention the applicability for the security of transport indication [3]

- More comments and input are welcome!

[3] https://datatracker.ietf.org/doc/draft-amsuess-core-transport-indication/

Thank you!

Comments/questions?

https://gitlab.com/crimson84/draft-tiloca-core-oscore-to-proxies