OSCORE-capable Proxies

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

Marco Tiloca, RISE
Rikard Höglund, RISE

IETF 111, CoRE WG, July 28th, 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

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
    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]
**Contribution**

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
   - At most 2 OSCORE “layers” in the same message: 1 end-to-end; 1 between two adjacent hops

Focus on OSCORE, but the same applies to Group OSCORE
Leg independence

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

<table>
<thead>
<tr>
<th>Conf. name (b2, b1, b0)</th>
<th>CF-0 (000)</th>
<th>CF-1 (001)</th>
<th>CF-2 (010)</th>
<th>CF-3 (011)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comm. legs using OSCORE</td>
<td>C-P</td>
<td>P-S</td>
<td>C-P</td>
<td>P-S</td>
</tr>
</tbody>
</table>

C=Client, P=Proxy, S=Server

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

<table>
<thead>
<tr>
<th>Conf. name (b2, b1, b0)</th>
<th>CF-4 (100)</th>
<th>CF-5 (101)</th>
<th>CF-6 (110)</th>
<th>CF-7 (111)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comm. legs using OSCORE</td>
<td>C-S</td>
<td>C-S</td>
<td>C-S</td>
<td>C-S</td>
</tr>
<tr>
<td></td>
<td>C-P (*)</td>
<td>P-S (*)</td>
<td>C-P (*)</td>
<td>P-S (*)</td>
</tr>
</tbody>
</table>

C=Client, P=Proxy, S=Server

(*) OSCORE-in-OSCORE

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

Naming convention: CF-X

\[ X = b_0 + (2 \times b_1) + (4 \times b_2) \]

- 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
High-level mechanics

› C request processing
   – (1) If \( b_2 = 1 \), protect with OSCORE \( C \leftrightarrow S \)
   – (2) If \( b_0 = 1 \), (further) protect with OSCORE \( C \leftrightarrow P \)
      › Encrypt options intended to \( P \), e.g., Proxy-Scheme
      › Encrypt the OSCORE option from (1), if any

› P request processing
   – Visible proxy options \( \rightarrow \) Forward to \( S \)
   – Absent proxy options && absent OSCORE option \( \rightarrow \) Deliver to the application
   – Absent proxy options && Visible OSCORE option \( \rightarrow \) Decrypt, as OSCORE \( C \leftrightarrow P \)
      › No proxy options in the decrypted request \( \rightarrow \) Deliver to the application
      › Visible proxy options in the decrypted request \( \rightarrow \) Forward to \( S \)

   – When forwarding to \( S \)
      › If \( b_1 = 1 \), (further) protect with OSCORE \( P \leftrightarrow S \)
      › Encrypt the OSCORE option for \( C \leftrightarrow S \), if any
High-level mechanics

› S request processing
  – Ready to find and process 1 or 2 OSCORE layers

› S response processing
  – (1) If $b_2 = 1$, protect with OSCORE $C \leftrightarrow S$
  – (2) If $b_1 = 1$, (further) protect with OSCORE $P \leftrightarrow S$
    › Encrypt options intended to $P$
    › Encrypt the OSCORE option from (1), if any

› P response processing
  – If $b_1 = 1$, unprotect with OSCORE $P \leftrightarrow S$
  – When forwarding to $C$
    › If $b_0 = 1$, (further) protect with OSCORE $C \leftrightarrow P$
    › Encrypt possible new added options intended to $C$
    › Encrypt the OSCORE option for $C \leftrightarrow S$, if any

› C response processing
  – Reverse of request processing; ready to find and process 1 or 2 OSCORE layers
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”
  – Useful for CoAP group communication, LwM2M external server, transport indication

› Next steps
  – Work on “Response caching” and “Chain of intermediaries”
  – Mention the applicability for the security of transport indication [3]

› The main mechanics are stable – Comments and reviews are welcome!

Thank you!

Comments/questions?

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