OSCORE-capable Proxies

draft-tiloca-core-oscore-capable-proxies-07

Marco Tiloca, RISE
Rikard Höglund, RISE

IETF 117 Meeting – San Francisco – July 25th, 2023

Recap

- **A CoAP proxy (P) can be used between client (C) and server (S)**
  - A security association might be required between C and P --- use cases in next slides
- **Good to use OSCORE between C and P**
  - Especially, but not only, if C and S already use OSCORE 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 2021 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/

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 nested 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
   - Possible to seamlessly apply 2 or more OSCORE layers to the same message

- **Focus on OSCORE, but the same applies "as is" to Group OSCORE**

Several use cases

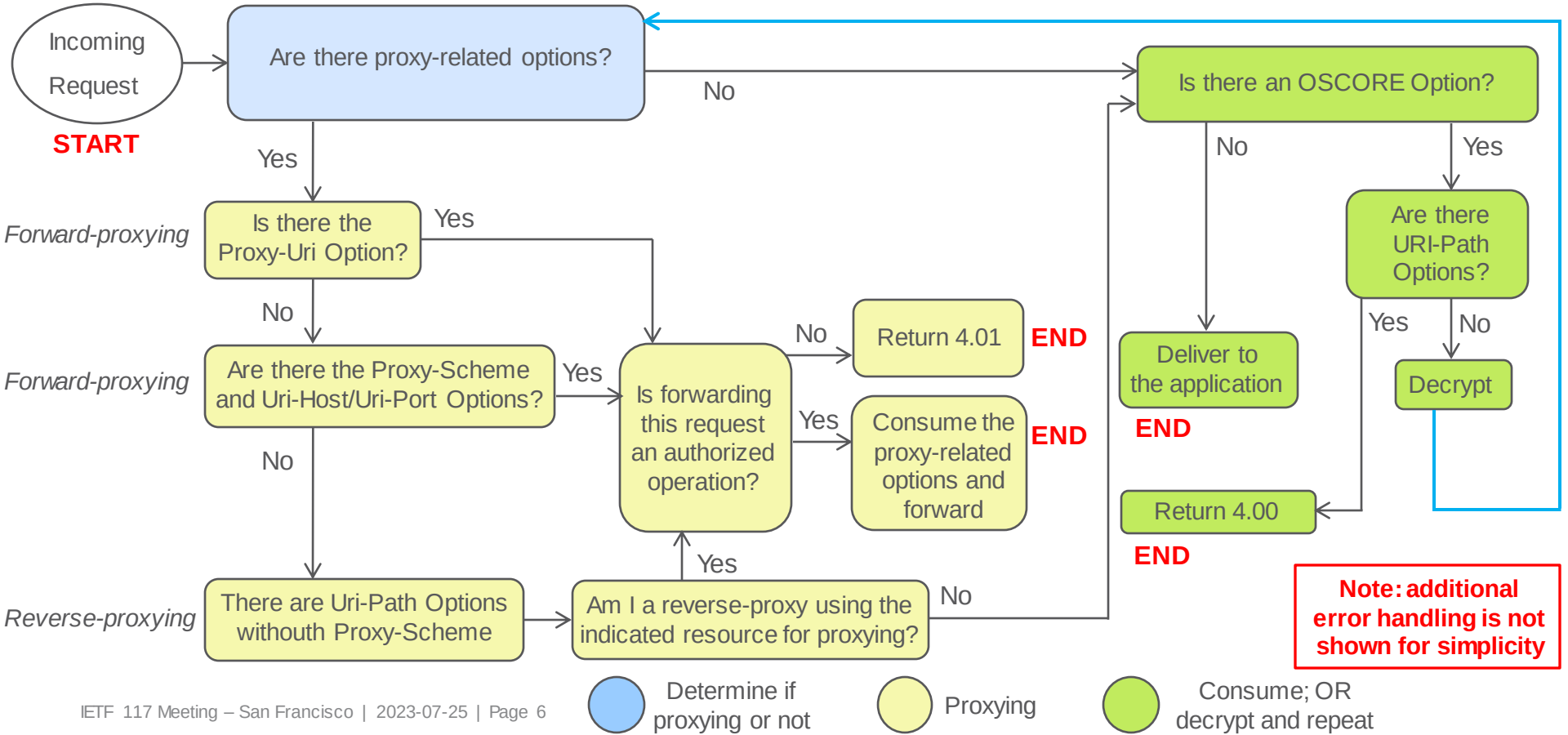
- **Section 2.1, CoAP group communication through a proxy [3]**
  - The proxy identifies the client before forwarding
- **Section 2.2, Observe multicast notifications with Group OSCORE [4]**
  - The client securely provides the Ticket Request to the proxy
- **Sections 2.3 and 2.4, OMA Lightweight Machine-to-Machine (LwM2M)**
  - The LwM2M Client uses the LwM2M Server as proxy towards External Application Servers
  - The LwM2M Server uses the LwM2M Gateway as reverse proxy towards External End Devices
- **Further use cases are listed in Section 2.5**
  - Transport indication through trusted proxies – *draft-ietf-core-transport-indication*
  - CoAP performance measurements involving on-path probes – *draft-ietf-core-coap-pm*
  - EST over OSCORE through a CoAP-to-HTTP proxy – *draft-ietf-ace-coap-est-oscore*
  - OSCORE-protected "onion forwarding", a la TOR – *draft-amsuess-t2trg-onion-coap*
  - Proxies as entry point to a firewalled network

[3] https://datatracker.ietf.org/doc/draft-tiloca-core-groupcomm-proxy/
[4] https://datatracker.ietf.org/doc/draft-ietf-core-observe-multicast-notifications/

Message processing

- **Stable and well defined**
  - No need for an explicit signaling method to guide the message processing
  - High-level general algorithm, fitting a client, proxy or server as a message processor
- **Dedicated Section 3.2-3.5 for each message and direction**
  - Outgoing request, incoming request (most delicate), outgoing response, incoming response
- **Recent clarifications suggested by Christian (already in version -06)**
  - A proxy performs authorization checks, before forwarding an incoming request
  - For example, based on the OSCORE Security Context locally used for decryption

Processing an incoming request



Recent updates (already in v -06)

- **OSCORE protection of CoAP options in outgoing messages**
  - If a CoAP option is originally defined as class U or I for OSCORE ...
  - ... when should it be fully protected anyway, as if it was of class E?

- **Improved general rules**
  - Now better covering corner cases and class I options
  - A good sanity check was the Request-Hash option [5]
    - Processed as class I in responses
    - Expected to be elided from responses, but still possible to send it on the wire

- **Current rule formulation – Section 3.1**
  - Three general cases, all phrased as “Any CoAP option such that ...”.
  - The rationale is to encrypt as many options as possible. If there is a match, encrypt!
  - Multiple examples are provided for each case

[5] https://datatracker.ietf.org/doc/draft-amsuess-core-cachable-oscore/

Recent updates (already in v -06)

- **New Section 5 – Guidelines on establishing OSCORE Security Contexts**
  - Generally agnostic of the used establishment method
- **For OSCORE**
  - Guidelines for the client using EDHOC [6], first with the proxy, then with the origin server
  - Reference to the possible to use, optimized EDHOC workflow [7]
- **For Group OSCORE**
  - Expected between origin client and servers, which rely on the Group Manager

[6] https://datatracker.ietf.org/doc/draft-ietf-lake-edhoc/
[7] https://datatracker.ietf.org/doc/draft-ietf-core-oscore-edhoc/

Updates in version -07

- **Section 2.5: mentioned two additional use cases**
  - CoAP performance measurements involving on-path probes – *draft-ietf-core-coap-pm*
  - EST over OSCORE through a CoAP-to-HTTP proxy – *draft-ietf-ace-coap-est-oscore*
  - Both those documents include an informative reference to this document

- **Section 5: clarifications when Group OSCORE is used – Thanks Christian!**
  - New text: no need for a proxy to be in the same OSCORE group of the origin client and server
  - Not forbidden altogether anymore, as it was in version -06; it might be fine and desirable

- **Section 7: added first security considerations**
  - Inherited from OSCORE, Group OSCORE, and the specifically used CoAP options
  - Any OSCORE endpoint in the chain enjoys the same, usual (Group) OSCORE properties

Updates in version -07

- **Added new Section 6, on using SCHC (RFC 8724)**
  - The compression of CoAP headers with SCHC is already defined in RFC 8824
  - When OSCORE is used, an inner and outer compression are performed on a message
  - *draft-tiloca-schc-8824-update* spells out (but does not extend) how this works with proxies
    - The inner compression is end-to-end; the outer compression is hop-by-hop

- **Section 6 generalizes the above for nested OSCORE protections**
  - No changes to the core mechanics of SCHC

- **When a sender endpoint produces an outgoing message, it performs:**
  - One inner compression for each applied OSCORE layer
    - Each for the OSCORE endpoint intended to decrypt and strip that OSCORE layer
  - Exactly one outer compression, after all the inner compressions
    - Intended for the (next-hop towards the) recipient origin endpoint

Updates in version -07

- **Appendix A: improved notation in the 5 examples of message exchange**
  - Easier to see what is encryped, and which OSCORE Security Context is used
  - Easier to follow the sub-steps taken by each endpoint
  - Two examples consider EDHOC [6], one of which with the optimized workflow [7]

- **Removed old Appendix B**
  - It included early specification notes towards “OSCORE-protected onion forwarding”
  - Discussed and agreed to remove it during the CoRE interim meeting on 2023-06-07
  - The removed and already revised content is now an Experimental document in T2TRG [8]

- **Added new Appendix B**
  - State diagram in ASCII art, showing the message processing of incoming requests
  - Based on the earlier slide 6, plus additional error handling; consistent with Section 3.3

[6] https://datatracker.ietf.org/doc/draft-ietf-lake-edhoc/
[7] https://datatracker.ietf.org/doc/draft-ietf-core-oscore-edhoc/
[8] https://datatracker.ietf.org/doc/draft-amsuess-t2trg-onion-coap/

Summary

- **Proposed update to RFC 8613**
  - Define the use of OSCORE in a communication leg including a proxy
  - Explicitly admit nested OSCORE protection: “OSCORE-in-OSCORE” / “Matryoscore”
  - Seamlessly usable also with Group OSCORE

- **Stable approach and mechanics**
  - Signaling-free message processing; CoAP options are protected as much as possible

- **Additional material**
  - On establishing of an OSCORE Security Context with proxies
  - On compressing CoAP headers with SCHC (RFC 8824)
  - Examples of message exchanges, also considering key establishment with EDHOC

- **The authors believe that version -07 is ready for a WG Adoption Call**

Thank you!

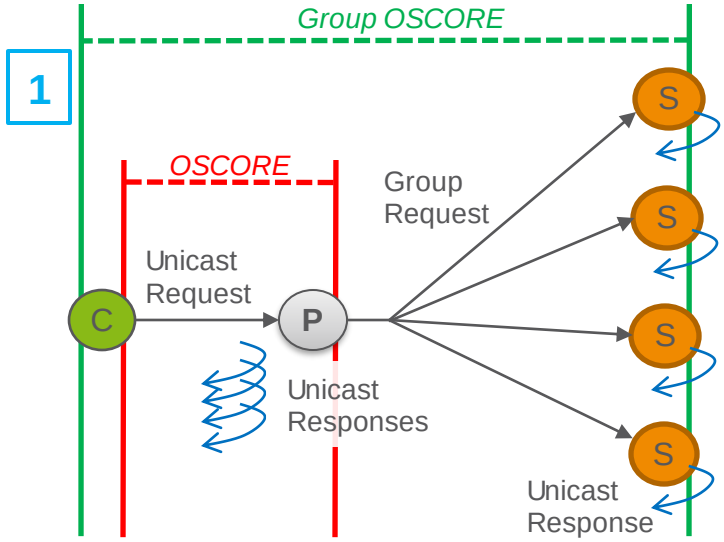
Comments/questions?

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

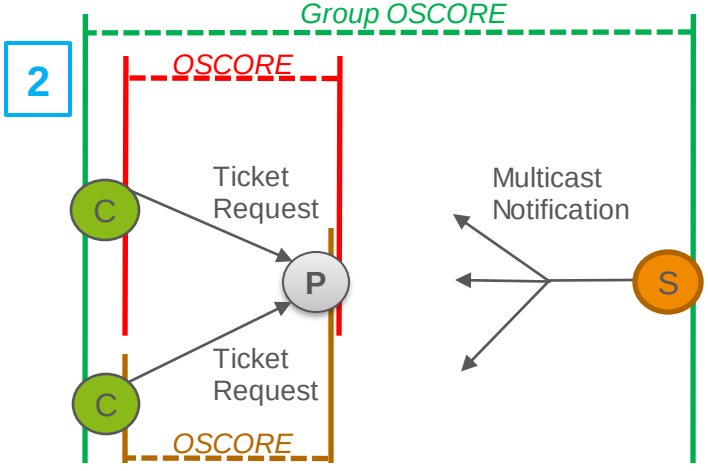
Backup

Use cases

1. **CoAP Group Communication with Proxies**
   - *draft-tiloca-core-groupcomm-proxy*
   - CoAP group communication through a proxy
   - P must identify C through a security association



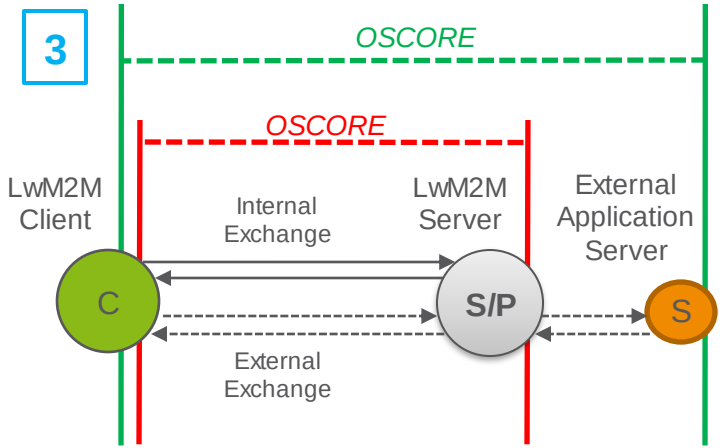
2. **CoAP Observe Notifications over Multicast**
   - *draft-ietf-core-observe-multicast-notifications*
   - If Group OSCORE is used for end-to-end security …
   - … C provides P with a Ticket Request obtained from S
   - That provisioning should be protected over C ↔ P



Use cases

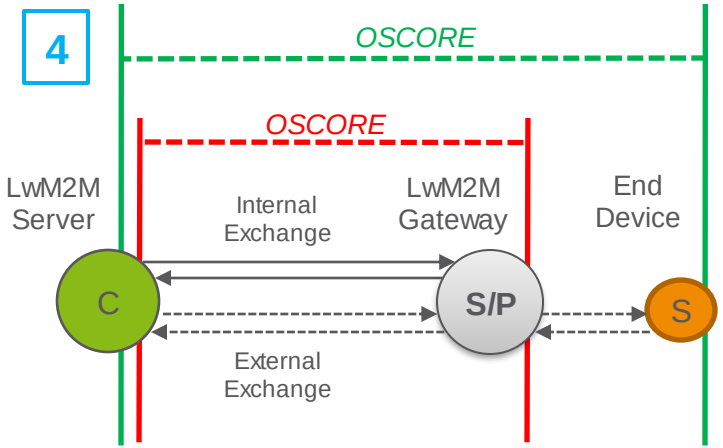
3. LwM2M Client and external Application Server

- From the *L2wM2M Transport Binding* specification:
  - OSCORE can be used between a LwM2M endpoint and a non-LwM2M endpoint, via the LwM2M Server
- The LwM2M Client may use OSCORE to interact:
  - With the LwM2M Server (LS), as usual; and
  - With an external Application Server, via LS acting as proxy



4. Use of the LwM2M Gateway (from David Navarro)

- It provides the LwM2M Server with access to:
  - a) Resources at the LwM2M Gateway
  - b) Resources at external End Devices, through the LwM2M Gateway, via dedicated URI paths
- In case (b), the LwM2M Gateway acts, at its core, as a reverse-proxy



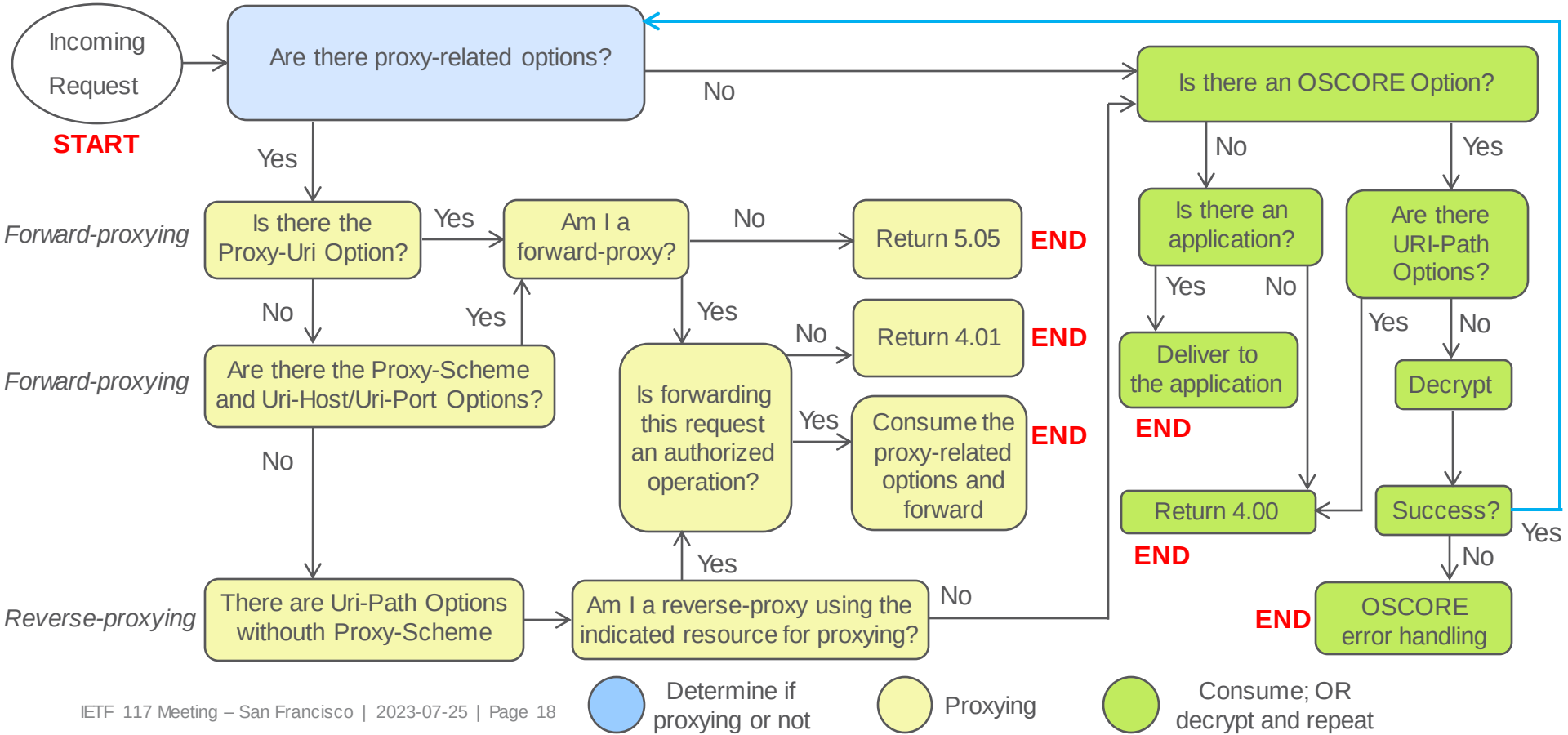
Use case 3 – LwM2M

- **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 traffic outside the LwM2M domain

Processing an incoming request



Encryption of Class U/I Options