OSCORE-capable Proxies

draft-ietf-core-oscore-capable-proxies-02

Marco Tiloca, RISE
Rikard Höglund, RISE

IETF 120 Meeting – Vancouver – July 24th, 2024

Scope: 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. **Define rules to escalate the protection of CoAP options**
   - If possible, encrypt and integrity-protect an option originally defined as Class U or I for OSCORE

3. **Explicitly admit nested OSCORE protection – "OSCORE-in-OSCORE"**
   - For example, 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**

Updates in version -02

- **Submitted before the cut-off for IETF 120**

- **Editorial**
  - Nit fixing and readability improvements
  - Minor clarifications
  - Updated references

- **Source application endpoint X: order of OSCORE protections for outgoing requests**
  - Already said: X first uses the Security Context shared with the destination application endpoint Y
  - Now also explicitly said how X proceeds after that, in general terms
    - X applies one OSCORE layer for each proxy with which it shares an OSCORE Security Context
    - The OSCORE layers are applied in the same order as the proxies are deployed in the chain
      - Starting from the proxy closest to Y
      - Moving backwards towards the proxy closest to X

Updates in version -02

- **Revised escalation of CoAP Option Protection**
  - Same intended rationale: encrypt whenever possible

- **The previous algorithm had a ~~feature~~ bug**
  - Reported by Christian (thanks!) in issue #1, now addressed in version -02
  - The Uri-Host and Uri-Port Options were encrypted as a side-effect, apparently for the better
  - If a reverse-proxy is deployed:
    - The sender endpoint typically does not know about that
    - The proxy can't decrypt those options, and thus can't rely on them to forward the request

- **Revised algorithm: Uri-Host and Uri-Port are encrypted only in one case**
  - They come together with Proxy-Scheme or Proxy-Scheme-Number, and ...
  - ... The intended consumer is a forward-proxy, which the sender endpoint knows about

Issue #1: https://github.com/core-wg/oscore-capable-proxies/issues/1

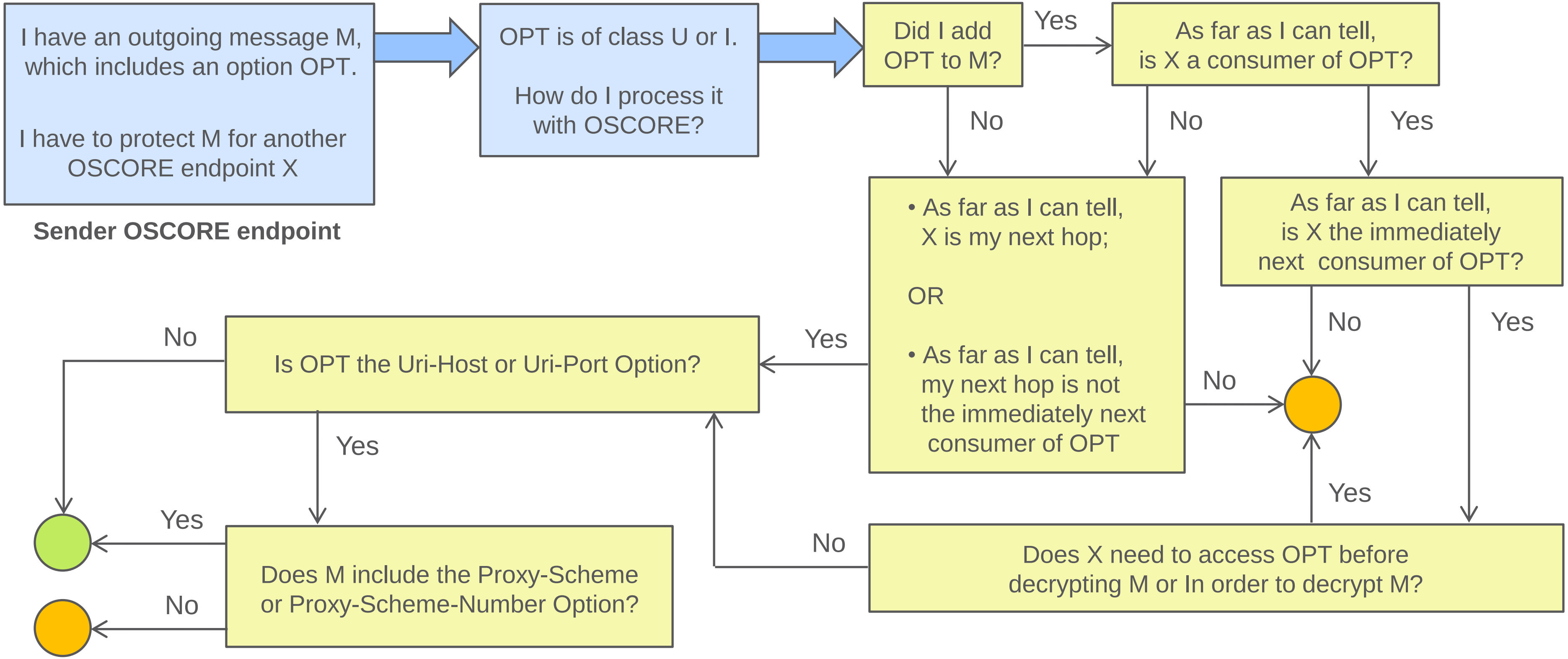
Updates in version -02

- **Section 3.1 – “Protection of CoAP Options” (now revised)**
  - It was written as a set of abstract properties to check for each option
  - Now it is written as an actual sequence of steps, mirroring the state diagram in Appendix B
  - Steps are phrased to reflect the possible presence of reverse-proxies, which the sender endpoint is not expected to know about

- **The algorithm description got further simplified, by merging different cases**

- **Appendix B – “State Diagram: Protection of CoAP Options”**
  - Also updated according to the revised algorithm

Encryption of Class U/I Options



Updates in version -02

- **Section 7 – “CoAP Header Compression with SCHC”**
  - Improved presentation of the steps taken for the Outer or Inner SCHC Decompression
  - Still generalized to the use of (nested) OSCORE also at proxies

- **Appendix A – Added two new examples, specifically with a reverse-proxy**
  - Both examples are aligned with the recent fix about (not) encrypting the Uri-Host Option
  - Appendix A.6:
    - OSCORE between C-S and C-P
    - Typical reverse-proxy, taking forwarding decision based on the Uri-Host Option
  - Appendix A.7:
    - OSCORE between C-S, C-P, and P-S
    - The reverse-proxy operates similar to the LwM2M Gateway (see Section 2.4), and takes forwarding decisions based on the Uri-Path Option

Protection of the Hop-Limit Option

- **Defined in RFC 8768, used for detecting loops in request forwarding**
  - Value set by the first hop in the chain supporting the option
  - Each hop decrements the value; then forwards if value > 0, or returns an error response otherwise

- **RFC 8768 does not define the OSCORE class for Hop-Limit**
  - Thus, the option is by default of Class E for OSCORE, per Section 4.1 of RFC 8613

- **Borderline case: the origin client adds Hop-Limit**
  - The origin client uses OSCORE with the origin server and protects the option end-to-end

- **The proxy chain relies on an outer Hop-Limit Option added by a proxy**
  - Forwarding loops are still detectable, but …
  - … the original intention indicated by the origin Client will not play a role; and …
  - … an inner Hop-Limit Option is pointlessly conveyed throughout each hop, with additional overhead

Protection of the Hop-Limit Option

- **Section 4 – Proposed update to RFC 8768**
  - The Hop-Limit Option is defined as Class U for OSCORE
  - (This is what Section 1 should also say, sorry for the typo!)

- **When using OSCORE as in RFC 8613**
  - The origin client does not protect Hop-Limit end-to-end

- **When using OSCORE as in this document, per the option protection rules:**
  - The origin client does not protect Hop-Limit end-to-end
  - Any two adjacent hops sharing an OSCORE Security Context do protect Hop-Limit with OSCORE

- **Related action for IANA**
  - "CoAP Option Numbers" registry: add a reference to this document in the entry for Hop-Limit Option

Thoughts? Objections?

Next steps

- **Closer look at:**
  - Addition of an outer option, after producing the corresponding, encrypted inner option (e.g., Observe)

- **Handling multiple responses to the same request, if also protected by a proxy**
  - Same rationale and approach as in *draft-ietf-core-oscore-groupcomm*

- **Extend the security considerations**

- **More examples of message exchanges, e.g., with a chain of proxies**

- **Comments and reviews are welcome!**

Thank you!

Comments/questions?

https://github.com/core-wg/oscore-capable-proxies

Backup

Motivation

- **A CoAP proxy (P) can be used between client (C) and server (S)**
  - A security association might be required between C and P

- **Good to use OSCORE between C and P**
  - Especially, <u>but not only</u>, if C and S already use OSCORE end-to-end

- **This is <u>not defined</u> and <u>not admitted</u> 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

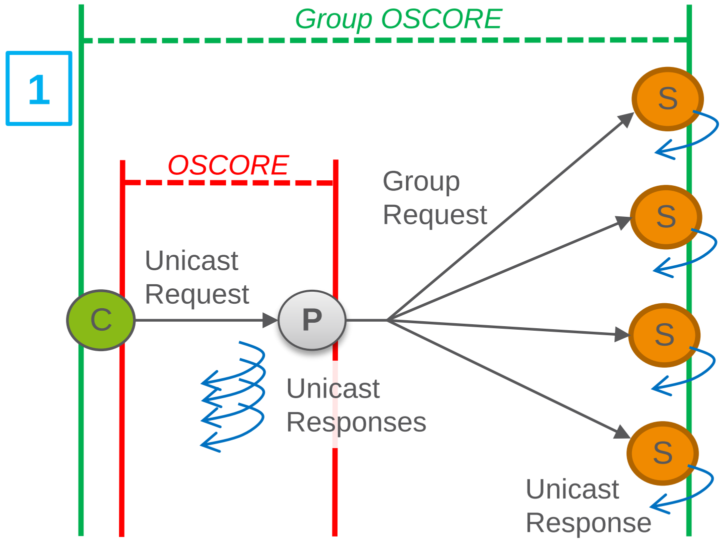
Use cases

- **Section 2.1, CoAP group communication through a proxy [4]**
  - The proxy identifies the client before forwarding
- **Section 2.2, Observe multicast notifications with Group OSCORE [5]**
  - 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 a proxy towards External Application Servers
  - The LwM2M Server uses the LwM2M Gateway as a 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

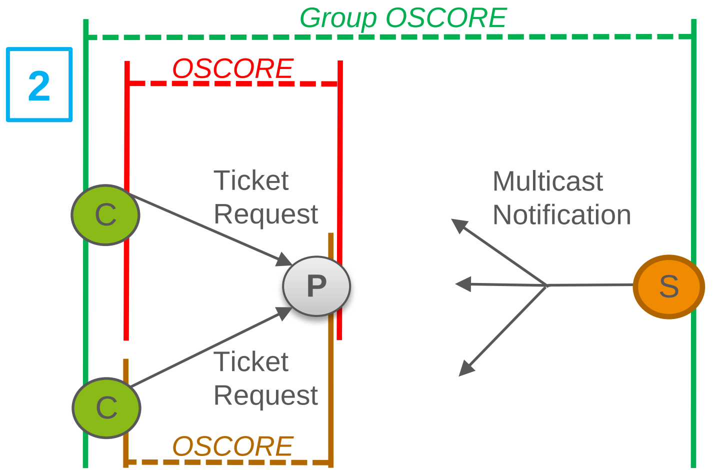
[4] https://datatracker.ietf.org/doc/draft-ietf-core-groupcomm-proxy/
[5] https://datatracker.ietf.org/doc/draft-ietf-core-observe-multicast-notifications/

Use cases

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

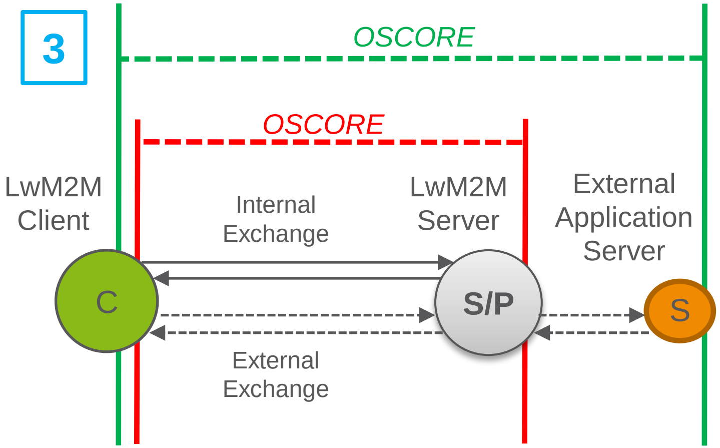


2. **CoAP Observe Notifications over Multicast**
   - *draft-ietf-core-observe-multicast-notifications*
   - If Group OSCORE is used for end-to-end security …
   - … C provides P with a Ticket Request obtained from S
   - That provisioning should be protected over C ↔ P

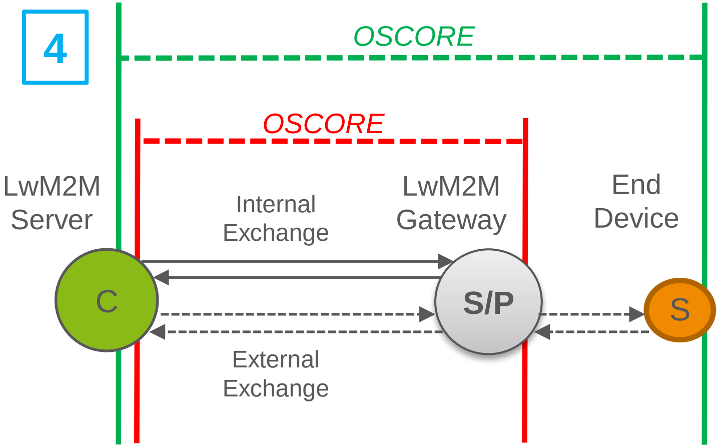


Use cases

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**
   - 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