Workgroup: CoRE Working Group
Internet-Draft:
draft-tiloca-core-oscore-capable-proxies-01
Updates: 8613 (if approved)
Published: 25 October 2021
Intended Status: Standards Track
Expires: 28 April 2022
Authors: M. Tiloca R. Höglund
RISE AB RISE AB

OSCORE-capable Proxies

Abstract

Object Security for Constrained RESTful Environments (OSCORE) can be used to protect CoAP messages end-to-end between two endpoints at the application layer, also in the presence of intermediaries such as proxies. This document defines how OSCORE is used to protect CoAP messages also between an origin application endpoint and an intermediary, or between two intermediaries. Besides, it defines how a CoAP message can be double-protected through "OSCORE-in-OSCORE", i.e., both end-to-end between origin application endpoints, as well as between an application endpoint and an intermediary or between two intermediaries. Thus, this document updates RFC 8613. The same approach applies to Group OSCORE, for protecting CoAP messages when group communication with intermediaries is used.

Discussion Venues

This note is to be removed before publishing as an RFC.

Discussion of this document takes place on the Constrained RESTful Environments Working Group mailing list (core@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/core/.

Source for this draft and an issue tracker can be found at https://gitlab.com/crimson84/draft-tiloca-core-oscore-to-proxies.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents

at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 28 April 2022.

Table of Contents

1. Introduction

The Constrained Application Protocol (CoAP) [RFC7252] supports the presence of intermediaries, such as forward-proxies and reverse-proxies, which assist origin clients by performing requests to

origin servers on their behalf, and forwarding back the related responses.

CoAP supports also group communication scenarios [I-D.ietf-core-groupcomm-bis], where clients can send a one-to-many request targeting all the servers in the group, e.g., by using IP multicast. Like for one-to-one communication, group settings can also rely on intermediaries [I-D.tiloca-core-groupcomm-proxy].

The protocol Object Security for Constrained RESTful Environments (OSCORE) [RFC8613] can be used to protect CoAP messages between two endpoints at the application layer, especially achieving end-to-end security in the presence of (non-trusted) intermediaries. When CoAP group communication is used, the same can be achieved by means of the protocol Group OSCORE [I-D.ietf-core-oscore-groupcomm].

For a number of use cases (see Section 2), it is required and/or beneficial that communications are secured also between an application endpoint (i.e., a CoAP origin client/server) and an intermediary, as well as between two adjacent intermediaries in a chain. This especially applies to the communication leg between the CoAP origin client and the adjacent intermediary acting as next hop towards the origin server.

In such cases, and especially if the origin client already uses OSCORE to achieve end-to-end security with the origin server, it would be convenient that OSCORE is used also to secure communications between the origin client and its next hop. However, the original specification [RFC8613] does not define how OSCORE can be used to protect CoAP messages in such communication leg, i.e., by considering the intermediary as an "OSCORE endpoint".

This document fills this gap, and updates [RFC8613] as follows.

* It defines how OSCORE is used to protect a CoAP message in the communication leg between: i) an origin client/server and an intermediary; or ii) two adjacent intermediaries in an intermediary chain. That is, besides origin clients/servers, it allows also intermediaries to be possible "OSCORE endpoints".

* It admits a CoAP message to be secured by multiple OSCORE protections applied in sequence, as an "OSCORE-in-OSCORE" process. For instance, this is the case when the message is OSCORE-protected end-to-end between the origin client and origin server, and the result is further OSCORE-protected over the leg between the current and next hop (e.g., the origin client and the adjacent intermediary acting as next hop towards the origin server).

The approach defined in this document does not specify any new signaling method to guide the message processing on the different endpoints. In particular, every endpoint is always able to understand what steps to take on an incoming message, depending on the presence of the OSCORE Option, as exclusively included or instead combined together with CoAP options intended for a proxy.

What defined in this document is applicable also when Group OSCORE is used, for protecting CoAP messages in group communication scenarios that rely on intermediaries.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Readers are expected to be familiar with the terms and concepts related to CoAP [RFC7252]; OSCORE [RFC8613] and Group OSCORE [I-D.ietf-core-oscore-groupcomm]. This document especially builds on concepts and mechanics related to intermediaries such as CoAP forward-proxies.

In addition, this document uses to the following terms.

- Source application endpoint: an origin client producing a request, or an origin server producing a response.

- Destination application endpoint: an origin server intended to consume a request, or an origin client intended to consume a response.

- Application endpoint: a source or destination application endpoint.

- Source OSCORE endpoint: an endpoint protecting a message with OSCORE or Group OSCORE.

- Destination OSCORE endpoint: an endpoint unprotecting a message with OSCORE or Group OSCORE.

- OSCORE endpoint: a source/destination OSCORE endpoint. An OSCORE endpoint is not necessarily also an application endpoint with respect to a certain message.

- Proxy-related option: the Proxy-URI Option, the Proxy-Scheme Option, or any of the Uri-* Options.

*OSCORE-in-OSCORE: the process by which a message protected with (Group) OSCORE is further protected with (Group) OSCORE. This means that, after performing an OSCORE decryption/verification, the resulting message is again an OSCORE-protected message.

2. Use Cases

The approach proposed in this document has been motivated by a number of use cases, which are summarized below.

2.1. CoAP Group Communication with Proxies

CoAP supports also one-to-many group communication, e.g., over IP multicast [I-D.ietf-core-groupcomm-bis], which can be protected end-to-end between origin client and origin servers by using Group OSCORE [I-D.ietf-core-oscore-groupcomm].

This communication model can be assisted by intermediaries such as a CoAP forward-proxy or reverse-proxy, which relays a group request to the origin servers. If Group OSCORE is used, the proxy is intentionally not a member of the OSCORE group. Furthermore, [I-D.tiloca-core-groupcomm-proxy] defines a signaling protocol between origin client and proxy, to ensure that responses from the different origin servers are forwarded back to the origin client within a time interval set by the client, and that they can be distinguished from one another.

In particular, it is required that the proxy identifies the origin client as allowed-listed, before forwarding a group request to the servers (see Section 4 of [I-D.tiloca-core-groupcomm-proxy]). This requires a security association between the origin client and the proxy, which would be convenient to provide with a dedicated OSCORE Security Context between the two, since the client is possibly using also Group OSCORE with the origin servers.

2.2. CoAP Observe Notifications over Multicast

The Observe extension for CoAP [RFC7641] allows a client to register its interest in "observing" a resource at a server. The server can then send back notification responses upon changes to the resource representation, all matching with the original observation request.

In some applications, such as pub-sub [I-D.ietf-core-coap-pubsub], multiple clients are interested to observe the same resource at the same server. Hence, [I-D.ietf-core-observe-multicast-notifications] defines a method that allows the server to send a multicast notification to all the observer clients at once, e.g., over IP multicast. To this end, the server synchronizes the clients, by providing them with a common "phantom observation request".

In case the clients and the server use Group OSCORE for end-to-end security and a proxy is also involved, an additional step is required (see Section 10 of [I-D.ietf-core-observe-multicast-notifications]). That is, clients are in turn required to provide the proxy with the obtained "phantom observation request", thus enabling the proxy to receive the multicast notifications from the server.

Therefore, it is preferable to have a security associations also between each client and the proxy, to especially ensure the integrity of that information provided to the proxy (see Section 13.3 of [I-D.ietf-core-observe-multicast-notifications]). Like for the use case in Section 2.1, this would be conveniently achieved with a dedicated OSCORE Security Context between a client and the proxy, since the client is also using Group OSCORE with the origin server.

2.3. LwM2M Client and External Application Server

The Lightweight Machine-to-Machine (LwM2M) protocol [LwM2M-Core] enables a LwM2M Client device to securely bootstrap and then register at a LwM2M Server, with which it will perform most of its following communication exchanges. As per the transport bindings specification of LwM2M [LwM2M-Transport], the LwM2M Client and LwM2M Server can use CoAP and OSCORE to secure their communications at the application layer, including during the device registration process.

Furthermore, Section 5.5.1 of [LwM2M-Transport] specifies that: "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."

In such a case, the LwM2M Server can practically act as forward-proxy between the LwM2M Client and the external Application Server. At the same time, the LwM2M Client and LwM2M Server must continue protecting communications on their leg using their Security Context. Like for the use case in Section 2.1, this also allows the LwM2M Server to identify the LwM2M Client, before forwarding its request outside the LwM2M domain and towards the external Application Server.

2.4. Further Use Cases

The approach proposed in this document can be useful also in the following use cases relying on a proxy.

* A server aware of a suitable cross proxy can rely on it as a third-party service, in order to indicate transports for CoAP

available to that server (see see Section 4 of [I-D.amsuess-core-transport-indication]).

From a security point of view, it would be convenient if the proxy could provide suitable credentials to the client, as a general trusted proxy for the system. However, in order for OSCORE to be an applicable security mechanism for this, it has to be terminated at the proxy. That is, a dedicated OSCORE Security Context between the client and the proxy would be required.

* A proxy may be deployed to act as an entry point to a firewalled network, which only authenticated clients can join. In particular, authentication can rely on the used secure communication association between a client and the proxy. If the proxy could share a dedicated OSCORE Security Context with each client, the proxy can rely on it to identify the client, before forwarding its messages to any other member of the firewalled network.

* The approach proposed in this document does not pose a limit in the number of OSCORE protections applied to the same CoAP message. This enables more privacy-oriented scenarios based on proxy chains, where the origin endpoint protects a message using each of the OSCORE Security Contexts shared with the different chain hops. Once received at a chain hop, a message would be stripped of the protection layer associated to that hop before being forwarded to the next one.

3. Message Processing

As mentioned in Section 1, this document introduces two main deviations from the original OSCORE specification [RFC8613].

1. An "OSCORE endpoint", i.e., a producer/consumer of an OSCORE option can be not only an application endpoint (i.e., an origin client and server), but also an intermediary such as a proxy.

   Hence, OSCORE can also be used between an origin client/server and a proxy, as well as between two proxies in an intermediary chain.

2. A CoAP message can be protected by multiple OSCORE layers applied in sequence. Therefore, the final result is a message with nested OSCORE layer, as the output of an "OSCORE-in-OSCORE" process. That is, following a decryption, the resulting message may include an OSCORE option, and thus have in turn to be decrypted.

   The most common case is expected to consider a message protected with up to two OSCORE layers, i.e.: i) an inner

layer, protecting the message end-to-end between the origin client and the origin server acting as application endpoints; and ii) an outer layer, protecting the message between a certain OSCORE endpoint and the other OSCORE endpoint adjacent in the intermediary chain.

However, a message can also be protected with a higher arbitrary number of nested OSCORE layers, e.g., in scenarios relying on a longer chain of intermediaries. For instance, the origin client can sequentially apply multiple OSCORE layers to a request, each of which to be consumed and removed by one of the intermediaries in the chain, until the origin server is reached and it consumes the innermost OSCORE layer.

3.1. General Rules on Protecting Options

When a sender endpoint protects an outgoing message by applying the i-th OSCORE layer in the sequence, the following CoAP options are also protected, in addition to the ones already defined to be of class I or class E.

* An OSCORE Option which is present as the result of the j-th OSCORE layer immediately previously applied, i.e., j = (i-1). Such an OSCORE option is protected like an option of class E.

* Any option intended to be protected for and consumed by the other OSCORE endpoint sharing the OSCORE Security Context used for applying the i-th OSCORE layer. These options especially include:

  - The proxy-related options Proxy-Uri, Proxy-Scheme and Uri-*.

  - Listen-To-Multicast-Notifications defined in [I-D.ietf-core-observe-multicast-notifications].

  - Multicast-Signaling, Response-Forwarding and Group-ETag defined in [I-D.tiloca-core-groupcomm-proxy].

3.2. Processing of Outgoing Requests

The rules from Section 3.1 apply when processing an outgoing request message, with the following addition.

When an application endpoint applies multiple OSCORE layers in sequence to protect an outgoing request, and it uses an OSCORE Security Context shared with the other application endpoint, then the first OSCORE layer MUST be applied by using that Security Context.

3.3. Processing of Incoming Requests

The recipient endpoint performs the following actions on the received request REQ, depending on which of the following three conditions apply.

* A - REQ includes visible proxy-related options.

  If the endpoint is not configured to be a proxy, it returns an error.

  Otherwise, the endpoint consumes the proxy-related options and forwards REQ to (the next hop towards) the origin server. This may involve a protection of REQ over that communication leg, as per Section 3.2.

* B - REQ does not include proxy-related options and does not include an OSCORE option.

  If the endpoint does not have an application to handle REQ, it returns an error.

  Otherwise, the endpoint delivers REQ to the application.

* C - REQ does not include proxy-related options and includes an OSCORE option.

  The endpoint decrypts REQ using the OSCORE Security Context indicated by the OSCORE option, i.e. REQ* = dec(REQ). After that, the possible presence of an OSCORE option in the decrypted request REQ* is not treated as an error situation.

  If the OSCORE processing results in an error, the endpoint stops and proceeds with producing an error response, as per Section 3.4.

  Otherwise, REQ takes REQ*, and the endpoint evaluates which of the three conditions (A, B, C) applies to REQ, thus performing again the algorithm defined in this section.

3.4. Processing of Outgoing Responses

The rules from Section 3.1 apply when processing an outgoing response message, with the following additions.

When an application endpoint applies multiple OSCORE layers in sequence to protect an outgoing response, and it uses an OSCORE Security Context shared with the other application endpoint, then the first OSCORE layer MUST be applied by using that Security Context.

The sender endpoint protects the response by applying the same OSCORE layers that it removed from the corresponding incoming request, but in the reverse order than the one they were removed.

In case the response is an error response, the sender endpoint protects it by applying the same OSCORE layers that it successfully removed from the corresponding incoming request, but in the reverse order than the one they were removed.

3.5. Processing of Incoming Responses

The recipient endpoint removes the same OSCORE layers that it added when protecting the corresponding outgoing request, but in the reverse order than the one they were removed.

When doing so, the possible presence of an OSCORE option in the decrypted response following the removal of an OSCORE layer is not treated as an error situation, unless it occurs after having removed as many OSCORE layers as were added in the outgoing request.

4. Caching of Responses

TBD

5. Security Considerations

TBD

6. IANA Considerations

This document has no actions for IANA.

7. References

7.1. Normative References

[I-D.ietf-core-oscore-groupcomm] Tiloca, M., Selander, G., Palombini, F., Mattsson, J. P., and J. Park, "Group OSCORE - Secure Group Communication for CoAP", Work in Progress, Internet-Draft, draft-ietf-core-oscore-groupcomm-13, 25 October 2021, <https://www.ietf.org/archive/id/draft-ietf-core-oscore-groupcomm-13.txt>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/

RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <https://www.rfc-editor.org/info/rfc7252>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8613] Selander, G., Mattsson, J., Palombini, F., and L. Seitz, "Object Security for Constrained RESTful Environments (OSCORE)", RFC 8613, DOI 10.17487/RFC8613, July 2019, <https://www.rfc-editor.org/info/rfc8613>.

7.2. Informative References

[I-D.amsuess-core-transport-indication] Amsüss, C., "CoAP Protocol Indication", Work in Progress, Internet-Draft, draft-amsuess-core-transport-indication-01, 10 July 2021, <https://www.ietf.org/archive/id/draft-amsuess-core-transport-indication-01.txt>.

[I-D.ietf-core-coap-pubsub] Koster, M., Keranen, A., and J. Jimenez, "Publish-Subscribe Broker for the Constrained Application Protocol (CoAP)", Work in Progress, Internet-Draft, draft-ietf-core-coap-pubsub-09, 30 September 2019, <https://www.ietf.org/archive/id/draft-ietf-core-coap-pubsub-09.txt>.

[I-D.ietf-core-groupcomm-bis] Dijk, E., Wang, C., and M. Tiloca, "Group Communication for the Constrained Application Protocol (CoAP)", Work in Progress, Internet-Draft, draft-ietf-core-groupcomm-bis-05, 25 October 2021, <https://www.ietf.org/archive/id/draft-ietf-core-groupcomm-bis-05.txt>.

[I-D.ietf-core-observe-multicast-notifications] Tiloca, M., Höglund, R., Amsüss, C., and F. Palombini, "Observe Notifications as CoAP Multicast Responses", Work in Progress, Internet-Draft, draft-ietf-core-observe-multicast-notifications-02, 25 October 2021, <https://www.ietf.org/archive/id/draft-ietf-core-observe-multicast-notifications-02.txt>.

[I-D.tiloca-core-groupcomm-proxy] Tiloca, M. and E. Dijk, "Proxy Operations for CoAP Group Communication", Work in

Progress, Internet-Draft, draft-tiloca-core-groupcomm-proxy-05, 25 October 2021, <https://www.ietf.org/archive/id/draft-tiloca-core-groupcomm-proxy-05.txt>.

[LwM2M-Core] Open Mobile Alliance, "Lightweight Machine to Machine Technical Specification - Core, Approved Version 1.2, OMA-TS-LightweightM2M_Core-V1_2-20201110-A", November 2020, <http://www.openmobilealliance.org/release/LightweightM2M/V1_2-20201110-A/OMA-TS-LightweightM2M_Core-V1_2-20201110-A.pdf>.

[LwM2M-Transport] Open Mobile Alliance, "Lightweight Machine to Machine Technical Specification - Transport Bindings, Approved Version 1.2, OMA-TS-LightweightM2M_Transport-V1_2-20201110-A", November 2020, <http://www.openmobilealliance.org/release/LightweightM2M/V1_2-20201110-A/OMA-TS-LightweightM2M_Transport-V1_2-20201110-A.pdf>.

[RFC7641] Hartke, K., "Observing Resources in the Constrained Application Protocol (CoAP)", RFC 7641, DOI 10.17487/RFC7641, September 2015, <https://www.rfc-editor.org/info/rfc7641>.

Acknowledgments

The authors sincerely thank Christian Amsuess, Peter Blomqvist and Goeran Selander for their comments and feedback.

The work on this document has been partly supported by VINNOVA and the Celtic-Next project CRITISEC; and by the H2020 project SIFIS-Home (Grant agreement 952652).

Authors' Addresses

Marco Tiloca
RISE AB
Isafjordsgatan 22
SE-16440 Kista
Sweden

Email: marco.tiloca@ri.se

Rikard Höglund
RISE AB
Isafjordsgatan 22
SE-16440 Kista
Sweden

Email: rikard.hoglund@ri.se