Workgroup: LWIG
Internet-Draft: draft-amsuess-lwig-oscore-00
Published: 29 April 2020
Intended Status: Informational
Expires: 31 October 2020
Authors: C. Amsüss

OSCORE Implementation Guidance

Abstract

Object Security for Constrained RESTful Environments (OSCORE) is a means of end-to-end protection of short request/response exchanges for tiny devices, typically transported using the Constrained Application Protocol (CoAP). This document aims to assist implementers in leveraging the optimizations catered for by the combination of CoAP and OSCORE, and by generally providing experience from earlier implementations.

Discussion Venues

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

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

Source for this draft and an issue tracker can be found at https://gitlab.com/chrysn/lwig-oscore/.

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 31 October 2020.

Table of Contents

1. Introduction

2. Context: ACE, LAKE, OSCORE

2.1. Example compositiions

2.1.1. Plain ACE-OSCORE

2.1.2. ACE with opportunistic LAKE

3. Protocol Implementation

3.1. Replay, freshness and safety

3.1.1. Background

3.1.2. Optimization

3.1.3. Implementation

3.1.4. Consequences for replay window recovery using Echo

4. Key IDs

5. HKDFs

6. Security Considerations

6.1. Assessment of idempotency

7. IANA Considerations

8. Informative References

Acknowledgments

Author's Address

1. Introduction

[See abstract for now]

2. Context: ACE, LAKE, OSCORE

[Is this LWIG material? In W3C terminology, this would go into a "primer" document.]

When OSCORE was specified, other parts of the ecosystem in which it is commonly used were already planned, but not to the extent to be fully referernced. This section gives a bigger picture of how surrounding technologies can be combined, with the caveat that some of them are still in development:

*OSCORE ([RFC8613]):

- needs a pre-provisioned key, key identifiers and some other details on two communication parties

- needs to keep sequence numbers on both parties (therfore, the same setup can only be rolled out once, ever)

- provides a secure communication channel between those parties

- does not provide any form of Perfect-Forward Secrecy (PFS)

- has optional provisions for using a secret more than once, with randomness from both parties (Appendix B.2)

*ACE ([I-D.ietf-ace-oauth-authz]):

- needs pre-existing secure channels and pre-established trust between communication parties and an Authorization Server (AS)

- provides tokens that encode some authorization on a Resource Serve (RS) to Clients (C)

- can be started by unprotected resource access (which fails, indicating the AS to get a token from) or from pre-established audiences and scopes

*OSCORE profile for ACE ([I-D.ietf-ace-oscore-profile]):

- needs an ACE token (obtained by the Client at an AS, valid for a particular Resource Server)

-takes randomness from both C and RS

-provides all the data to start OSCORE between C and RS

-ensures to C that RS is the RS it asked a token for from AS

-ensures to RS that C was authorized by the AS for whatever the scope of the token is

*a LAKE (Lightweight Key Exchange), for example EDHOC (Ephemeral Diffie-Hellman Over COSE, [I-D.selander-lake-edhoc]):

-needs any combination of credentials between two parties, not necessarily pre-shared (can be certificates, raw public keys, or pre-shared keys)

-provides a shared set of keys and other details sufficient to start OSCORE between the parties

-provides Perfect-Forward Secrecy (PFS)

-ensures to both parties that the other party has provided the indicated credentials

*BRSKI

-[TBD]

[same could be done for Group OSCORE]

2.1. Example compositiions

[While I'm reasonably sure what I'm writing in this document is correct, the following is wild speculation in the hope that ACE and LAKE authors tell me better]

2.1.1. Plain ACE-OSCORE

A client tries to access a resource over unprotected CoAP, but the server requires credentials (and thus a secure connection).

On the initial unprotected request, the server responds 4.01 Unauthorized, and sends the client off to the AS with information from the payload.

The client, which needs to have a pre-established association with the AS (or establishes one using yet unspecified mechanisms on the fly), obtains a token from it, posts it over the original unprotected CoAP transport to the server, and from then on has an

OSCORE context with the server, over which it can request the resource successfully.

This is illustrated well in [I-D.ietf-ace-oscore-profile] Figure 1.

This combination has the advantage of not requiring any asymmetric cryptography, but has the original request data unprotected, and the AS can decrypt communication between C and RS if it intercepts their first exchanged messages.

2.1.2. ACE with opportunistic LAKE

A client that tries to access a resource but does not want to reveal the request details to passive eavesdroppers can run an EDHOC with the origin server. Nothing else being preconfigured, it runs it on a raw public key, and accepts any credentials from the server. (If a set of root certificates of a public key infrastructure (PKI) is set, it could require a certificate chain to the root certificates).

Inside that opportunistically encrypted channel, the client sends a first request to the resource. If the server, as in the ACE-OSCORE example, requires authorization, it can still reject the request and send the client off to the AS with the same response.

The client obtains a token from the AS as before, but does not need to generate a new OSCORE context from it (and thus does not use the OSCORE profile for ACE). Instead, it can post the token to the server in the existing EDHOC-created OSCORE context, and thus upgrades the authorization set of that context.

This combination has the advantage of not sending any actual request unprotected, but does not ensure to the client that the server has any association with the AS.

Alternatively, it can use ACE-OSCORE when obtaining the token and when posting it to the RS over the EDHOC-created OSCORE context to obtain a new OSCORE context.

This combination has similar properties to the plain ACE-OSCORE, at the cost of an asymmetric cryptography step, but protecting the original request from passive eavesdroppers.

3. Protocol Implementation

3.1. Replay, freshness and safety

3.1.1. Background

[RFC8613] Section 7.4 says that the server "SHALL stop processing the message" if that fails, [and I'm in quite a pickle here because I'd like to tell implementers that they can partially ignore that].

In OSCORE, replay protection serves two distinct purposes:

- To ensure that only one response is sent with no Partial IV present to any given request Partial IV on a context - i.e., to curb nonce reuse.

- To ensure that any action authorized by OSCORE protection on the request is only executed once.

For the first purpose, that mandate is absolute: processing any request a second time and responding with an absent Partial IV is a severe security violation. It does not apply, however, if the server chooses to encode an own Partial IV in the response in step 3 of RFC8613 Section 8.3.

The relevance of the second purpose depends on the request and the implememtation of the resource backing it. If the request is safe (in the sense of [RFC7231] Section 4.2.1, i.e. it is a GET or FETCH), or at least idempotent (i.e. PUT, DELETE or iPATCH), processing a replay of it has no side effect on the server, and the only thing an attacking replaying party could learn from another response is the current content's length.

[TBD: Talk about how this interacts with freshness.]

3.1.2. Optimization

Combined, these open some space for legitimate processing of replays. Opening up to such processing is beneficial to the server, as it allows a common optimization to happen even on OSCORE messages: [RFC7252] Section 4.5 allows relaxation on message deduplication for idempotent requests, to the point where some implementations of CoAP do not perform message deduplication at all and demand of their applications to only implement idempotent behavior.

On platforms that perform selective optimizations, these optimizations can free up memory otherwised used for deduplication and retransmission, provided the operation's idempotency is communicated to the OSCORE and CoAP implementation (which would, in

general, not be allowed to enact that optimization for the POST requests OSCORE requests appear as).

On platforms that do not perform deduplication at all, this enables the implementation of OSCORE in the first place. (Otherwise, any lost response message results in an otherwise unactionable 4.01 Unauthorized error).

Intermediaries (proxies) have no justification for treating the POST requests they see most OSCORE request as as idempotent; however [and this can not really be called "moving on the fringe of the specification" because it's clearly exceeding it], clients of a very constrained proxy (which might not even be able to forward non-idempotent requests at all) might still appreciate a presumed-idempotent forwarding of OSCORE messages over a 5.05 Proxying Not Supported.

3.1.3. Implementation

Ensuring that only one response without a Partial IV is ever sent to a given request is of utmost importance when implementing this optimization.

One way of doing this is to annotate the received nonce or partial IV with a marker that indicates the usability as an elided response Partial IV. That marker is originally unset when the Partial IV is extracted from the request, and only ever gets set the very time that sequence number gets removed from the replay window. The marker is removed from the data structure when it is used in the encryption of a message. Care must be taken when a nonce / Partial IV is copied to only let the marker stay with one copy, and to unset it on the other one.

Note that this aligns well with the typical other cases when responses use an own Partial IV:

* In observations, the first response can be sent without a Partial IV. Later notifications are built with AAD linked to the original request's Partial IV, but that was copied over from the original request's Partial IV and thus does not carry a marker any more.

* In responses that serve to recover the replay window as in [RFC8613] Appendix B.1.2, the replay window is invalid when the first response is generated, thus there is no marker to respond without Partial IV.

3.1.4. Consequences for replay window recovery using Echo

[This is maybe more corr-clar or even my own wishful thinking than actual implementation guidance.]

For the first response to the replay window recovery of [RFC8613] Appendix B.1.2, applying the above considerations means that the server may not need to recover its replay window right away.

If the initial request that triggers the recovery process is idempotent (for example, the typical initial GET to /.well-known/core), the server can accept the request as possible duplicate, and send a successful response righ away.

The response should still contain an Echo value (and the client should send the same Echo with the next request) for the benefit of future non-idempotent operations and to eventually allow the server to send shorter responses (without a Partial IV).

Only when a resource with freshness requirements is accessed, the client needs to have included the Echo value in at latest that very response. A 4.01 response (which incurs the additional round-trip) thus only needs to happen if the first request that triggers recovery is not idempotent.

4. Key IDs

The naming of Keys is a difficult matter,
it's not just one of your entropy games;
You may think at first I'm as mad as a hatter
When I tell you, a context needs space for its names.

[

TBD. Topics:

* KID contexts are not strictly hierarchically above KIDs, that is up to the KID: A device may have some KIDs that are KID-context namespaced and will (at least initially) need a KID context, and some that are B.2 and the KID context is more of a detail under the KID than a namespace above.

* When ACE'ing with different ASs, or combining ACE with EDHOC, or any of that with preconfigured keys, consider namespacing (eg. dealing out a prefix-based group of KIDs to an AS)

  - Generally recommend treating KIDs like URI paths - it's always up to the server?

* Eventually avoid ever having to try different contexts, show it can be done

]

5. HKDFs

[This is more of a corr-clar topic than a LWIG - still fits here?]

[TBD: Does it need to be HKDF? HMAC-based HKDF? What about other direct+HKDFs in COSE ([I-D.ietf-core-oscore-groupcomm] may do some updates here)?]

[Why and how do direct+ KDFs of COSE apply at all? My current impression is:]

When KDFs are referred to by identifier, they are usually identified with the direct+HKDF-... COSE Algorithms. This makes sense as those algorithms specify a particular key derivation function, but also slightly misleading: What is meant by that usage is not to actually apply the "Direct Key with KDF" algorthm of [RFC8152] ([what goes in as info there?]) but to apply the OSCORE key derivation process (with [id, id_context, aead_alg, type, L] as info).

6. Security Considerations

6.1. Assessment of idempotency

Getting all implications of processing an idempotent request without an indication of freshness is hard. The topic has been discussed at length in the context of TLS and HTTP/2 zero-round-trip actions [but I couldn't be bothered to find precise references for that yet].

[Applications probably also need guidance as to what are the temporal aspects or idempotency (a request that has the same effect when processed twice in the same second, and different ones if processed a day later - is it still idempotent then?), and on the re-ordering of requests permitted by idempotency ("Just because you received confirmation does not mean it can't be reordered with a request you send later ... unless you put in a memory barrier?")]

7. IANA Considerations

This document has no IANA actions.

8. Informative References

[I-D.ietf-ace-oauth-authz]
Seitz, L., Selander, G., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "Authentication and Authorization for Constrained Environments (ACE) using the OAuth 2.0 Framework (ACE-OAuth)", Work in Progress, Internet-Draft, draft-ietf-ace-oauth-authz-33, 6 February 2020, <http://

www.ietf.org/internet-drafts/draft-ietf-ace-oauth-authz-33.txt>.

[I-D.ietf-ace-oscore-profile]
Palombini, F., Seitz, L., Selander, G., and M. Gunnarsson, "OSCORE profile of the Authentication and Authorization for Constrained Environments Framework", Work in Progress, Internet-Draft, draft-ietf-ace-oscore-profile-10, 9 March 2020, <http://www.ietf.org/internet-drafts/draft-ietf-ace-oscore-profile-10.txt>.

[I-D.ietf-core-oscore-groupcomm]
Tiloca, M., Selander, G., Palombini, F., and J. Park, "Group OSCORE - Secure Group Communication for CoAP", Work in Progress, Internet-Draft, draft-ietf-core-oscore-groupcomm-08, 6 April 2020, <http://www.ietf.org/internet-drafts/draft-ietf-core-oscore-groupcomm-08.txt>.

[I-D.selander-lake-edhoc]
Selander, G., Mattsson, J., and F. Palombini, "Ephemeral Diffie-Hellman Over COSE (EDHOC)", Work in Progress, Internet-Draft, draft-selander-lake-edhoc-01, 9 March 2020, <http://www.ietf.org/internet-drafts/draft-selander-lake-edhoc-01.txt>.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <https://www.rfc-editor.org/info/rfc7231>.

[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>.

[RFC8152] Schaad, J., "CBOR Object Signing and Encryption (COSE)", RFC 8152, DOI 10.17487/RFC8152, July 2017, <https://www.rfc-editor.org/info/rfc8152>.

[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>.

Acknowledgments

[TBD put into text]

OSCORE authors in general: discussion leading up to most of this during OSCORE dev'mt FP: HKDF input

Author's Address

Christian Amsüss

Email: christian@amsuess.com