Workgroup: CoRE Working Group
Internet-Draft:
draft-ietf-core-oscore-key-limits-02
Published: 10 January 2024
Intended Status: Informational
Expires: 13 July 2024
Authors: R. Höglund
RISE AB
M. Tiloca
RISE AB

Key Usage Limits for OSCORE

Abstract

Object Security for Constrained RESTful Environments (OSCORE) uses AEAD algorithms to ensure confidentiality and integrity of exchanged messages. Due to known issues allowing forgery attacks against AEAD algorithms, limits should be followed on the number of times a specific key is used for encryption or decryption. Among other reasons, approaching key usage limits requires updating the OSCORE keying material before communications can securely continue. This document defines how two OSCORE peers can follow these key usage limits and what steps they should take to preserve the security of their communications.

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://github.com/core-wg/oscore-key-limits.

Status of This Memo

This Internet-Draft will expire on 13 July 2024.

Table of Contents

1. Introduction

Object Security for Constrained RESTful Environments (OSCORE) [RFC8613] provides end-to-end protection of CoAP [RFC7252] messages

at the application-layer, ensuring message confidentiality and integrity, replay protection, as well as binding of response to request between a sender and a recipient.

In particular, OSCORE uses AEAD algorithms to provide confidentiality and integrity of messages exchanged between two peers. Due to known issues allowing forgery attacks against AEAD algorithms, limits should be followed on the number of times a specific key is used to perform encryption or decryption [I-D.irtf-cfrg-aead-limits].

The original OSCORE specification [RFC8613] does not consider such key usage limits. However, should they be exceeded, an adversary may break the security properties of the AEAD algorithm, such as message confidentiality and integrity, e.g., by performing a message forgery attack. Among other reasons, approaching the key usage limits requires updating the OSCORE keying material before communications can securely continue. This document defines what steps an OSCORE peer should take to preserve the security of its communications, by stopping to use the OSCORE Security Context shared with another peer when approaching the key usage limits.

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] and OSCORE [RFC8613].

2. AEAD Key Usage Limits in OSCORE

This section details how key usage limits for AEAD algorithms can be considered when using OSCORE. In particular, it discusses specific limits for common AEAD algorithms used with OSCORE; parameters to track associated to an OSCORE Security Context; and additions to the OSCORE message processing.

2.1. Problem Overview

The OSCORE security protocol [RFC8613] uses AEAD algorithms to provide integrity and confidentiality of messages, as exchanged between two peers sharing an OSCORE Security Context.

When processing messages with OSCORE, each peer should follow specific limits as to the number of times it uses a specific key. This applies separately to the Sender Key used to encrypt outgoing

messages, and to the Recipient Key used to decrypt and verify incoming protected messages.

Exceeding these limits may allow an adversary to break the security properties of the AEAD algorithm, such as message confidentiality and integrity, e.g., by performing a message forgery attack.

The following refers to the two parameters 'q' and 'v' introduced in [I-D.irtf-cfrg-aead-limits], to use when deploying an AEAD algorithm.

- 'q': this parameter has as value the number of messages protected with a specific key, i.e., the number of times the AEAD algorithm has been invoked to encrypt data with that key.

- 'v': this parameter has as value the number of alleged forgery attempts that have been made against a specific key, i.e., the amount of failed decryptions that have occurred with the AEAD algorithm for that key.

When a peer uses OSCORE:

- The key used to protect outgoing messages is its Sender Key from its Sender Context.

- The key used to decrypt and verify incoming messages is its Recipient Key from its Recipient Context.

Both keys are derived as part of the establishment of the OSCORE Security Context, as defined in Section 3.2 of [RFC8613].

As mentioned above, exceeding specific limits for the 'q' or 'v' value can weaken the security properties of the AEAD algorithm used, thus compromising secure communication requirements.

Therefore, in order to preserve the security of the used AEAD algorithm, OSCORE has to observe limits for the 'q' and 'v' values, throughout the lifetime of the used AEAD keys.

2.1.1. Limits for 'q' and 'v'

Formulas for calculating the security levels, as Integrity Advantage (IA) and Confidentiality Advantage (CA) probabilities, are presented in [I-D.irtf-cfrg-aead-limits]. These formulas take as input specific values for 'q' and 'v' (see section Section 2.1) and for 'l', i.e., the maximum length of each message (in cipher blocks).

For the algorithms shown in Figure 1 that can be used as AEAD Algorithm for OSCORE, the key property to achieve is having IA and CA values which are no larger than p = 2^-64, which will ensure a

safe security level for the AEAD Algorithm. This can be entailed by using the values q = 2^20, v = 2^20, and l = 2^10, that this document recommends to use for these algorithms.

Figure 1 also shows the resulting IA and CA probabilities enjoyed by the considered algorithms, when taking the value of 'q', 'v' and 'l' above as input to the formulas defined in [I-D.irtf-cfrg-aead-limits].

Algorithm name	IA probability	CA probability
AEAD_AES_128_CCM	2^-64	2^-66
AEAD_AES_128_GCM	2^-97	2^-89
AEAD_AES_256_GCM	2^-97	2^-89
AEAD_CHACHA20_POLY1305	2^-73	-

Figure 1: Probabilities for algorithms based on chosen q, v and l values.

When AEAD_AES_128_CCM_8 is used as AEAD Algorithm for OSCORE, the triplet (q, v, l) considered above yields larger values of IA and CA. Hence, specifically for AEAD_AES_128_CCM_8, this document recommends using the triplet (q, v, l) = (2^20, 2^14, 2^8). This is appropriate, since the resulting CA and IA values are not greater than the threshold value of 2^-50 defined in [I-D.irtf-cfrg-aead-limits], and thus yields an acceptable security level. Achieving smaller values of CA and IA would require to inconveniently reduce 'q', 'v' or 'l', with no corresponding increase in terms of security, as further elaborated in Appendix A.

Algorithm name	l=2^6 in bytes	l=2^8 in bytes	l=2^10 in bytes
AEAD_AES_128_CCM	1024	4096	16384
AEAD_AES_128_GCM	1024	4096	16384
AEAD_AES_256_GCM	1024	4096	16384
AEAD_AES_128_CCM_8	1024	4096	16384
AEAD_CHACHA20_POLY1305	4096	16384	65536

Figure 2: Maximum length of each message (in bytes)

With regards to the limit for 'l', the recommended 'l' value for the algorithms shown in Figure 1, and for AEAD_AES_128_CCM_8, is 2^10 (16384 bytes) and 2^8 (4096 bytes) respectively. Considering a

typical MTU size of 1500 bytes, and the fact that the maximum block size when using block-wise transfers with CoAP is 1024 bytes (see Section 2 of [RFC7959]), it is unlikely that a larger size of 'l' than what is recommended makes sense to use in typical network setups.

However, although under typical circumstances an 'l' limit of 2^8 (4096 bytes) is acceptable, exceptional cases can warrant a higher value of 'l'. For instance, Block-wise Extension for Reliable Transport (BERT) extends the CoAP Block-Wise tranfer functionality, enabling use of larger messages over reliable transports such as TCP or WebSockets (see [RFC8323]). In case the OSCORE peers wish to take full advantage of BERT functionality and the large message sizes it allows for, the OSCORE peers must use higher values of 'l'.

An alternative means of allowing for larger values of 'l', while still maintaining the security properties of the used AEAD algorithm, is to adjust the 'q' and 'v' values to compensate. In practice, this means reducing the value of 'q' and 'v' considering the new value of 'l', to ensure an acceptably low value of the IA and CA probabilities. A reasonable target for the IA and CA probability values is the threshold value of 2^-50 defined in [I-D.irtf-cfrg-aead-limits].

2.2. Additional Information in the Security Context

In addition to what is defined in Section 3.1 of [RFC8613], the following parameters associated with a OSCORE Security Context can be used for keeping track of the expiration of that OSCORE Security Context and maintaining key usage below safe limits.

2.2.1. Common Context

The Common Context has the following associated parameter.

* 'exp': with value the expiration time of the OSCORE Security Context, as a non-negative integer. The parameter contains a numeric value representing the number of seconds from 1970-01-01T00:00:00Z UTC until the specified UTC date/time, ignoring leap seconds, analogous to what is specified for NumericDate in Section 2 of [RFC7519].

  At the time indicated by this parameter, a peer must stop using this Security Context to process any incoming or outgoing message, and is required to establish a new Security Context to continue OSCORE-protected communications with the other peer. That is, the expiration of an OSCORE Security Context means that the current Sender Key must no longer be used for protecting outgoing messages, and the Recipient Key must no longer be used for unprotecting incoming messages.

The value of 'exp' must be set upon installing the OSCORE Security Context, namely at time t_1, considering a lifetime value t_l. In particular, t_l can be a default value (potentially differing between the two peers sharing the OSCORE Security Context), or can alternatively be agreed by the two peers during the establishment of the OSCORE Security Context. For instance, this value may be stored and/or transported in an OSCORE LwM2M object, or specified as part of an EDHOC Application Profile [I-D.ietf-lake-edhoc] used when running EDHOC for establishing the OSCORE Security Context. Regardless of how the lifetime value is determined, the 'exp' parameters is set to indicate the point in time corresponding to t_1 offset by t_l.

2.2.2. Sender Context

The Sender Context has the following associated parameters.

* 'count_q': a non-negative integer counter, keeping track of the current 'q' value for the Sender Key. At any time, 'count_q' has as value the number of messages that have been encrypted using the Sender Key. The value of 'count_q' is set to 0 when establishing the Sender Context.

* 'limit_q': a non-negative integer, which specifies the highest value that 'count_q' is allowed to reach, before stopping using the Sender Key to process outgoing messages.

  The value of 'limit_q' depends on the AEAD algorithm specified in the Common Context, considering the properties of that algorithm. The value of 'limit_q' is determined according to Section 2.1.1.

Note for implementation: it is possible to avoid storing and maintaining the counter 'count_q'. Rather, an estimated value to be compared against 'limit_q' can be computed, by leveraging the Sender Sequence Number of the peer and (an estimate of) the other peer's. A possible method to achieve this is described in Appendix B. While this relieves peers from storing and maintaining the precise 'count_q' value, it results in overestimating the number of encryptions performed with a Sender Key. This in turn results in approaching 'limit_q' sooner and thus in performing a key update procedure more frequently.

2.2.3. Recipient Context

The Recipient Context has the following associated parameters.

* 'count_v': a non-negative integer counter, keeping track of the current 'v' value for the Recipient Key. At any time, 'count_v' has as value the number of failed decryptions occurred on

incoming messages using the Recipient Key. The value of 'count_v' is set to 0 when establishing the Recipient Context.

* 'limit_v': a non-negative integer, which specifies the highest value that 'count_v' is allowed to reach, before stopping using the Recipient Key to process incoming messages.

  The value of 'limit_v' depends on the AEAD algorithm specified in the Common Context, considering the properties of that algorithm. The value of 'limit_v' is determined according to Section 2.1.1.

2.3. OSCORE Message Processing

In order to keep track of the 'q' and 'v' values and ensure that AEAD keys are not used beyond reaching their limits, OSCORE peers protect messages with OSCORE as defined in this section.

A limitation that is introduced is that, in order to not exceed the selected value for 'l', the total size of the COSE plaintext, authentication Tag, and possible cipher padding for a message must not exceed the block size for the selected algorithm multiplied with 'l'. The size of the COSE plaintext is calculated as described in Section 5.3 of [RFC8613].

If OSCORE peers need to transmit messages exceeding the maximum recommended size caclulated from 'l', CoAP Block-Wise transfers [RFC7959] may be used as a means to split the whole, large content into smaller segments. The following steps can be adopted by a client or server to determine whether the usage of block-wise transfer is necessary for the transmission of a specific OSCORE protected message.

1. The CoAP message to transmit is first produced.

2. The sum of the total size of the COSE plaintext, the length of the authentication tag, and the length of any potential ciphertext padding should be computed to produce a value T. It should be noted that the size of the padding and the length of the authentication tag depend on the used AEAD algorithm.

3. If the value of T exceeds the 'l' value for the used AEAD algorithm, block-wise transfer is to be used with the CoAP message before protecting it with OSCORE.

The processing of CoAP messages with OSCORE follows the steps outlined in Section 8 of [RFC8613], with the additions defined below.

2.3.1. Protecting a Request or a Response

Before encrypting the COSE object using the Sender Key, the 'count_q' counter is incremented.

If 'count_q' exceeds the 'limit_q' limit, the message processing is aborted. From then on, the Sender Key must not be used to encrypt further messages.

2.3.2. Verifying a Request or a Response

If an incoming message is detected to be a replay (see Section 7.4 of [RFC8613]), the 'count_v' counter is not incremented.

If the decryption and verification of the COSE object using the Recipient Key fails, the 'count_v' counter is incremented.

After 'count_v' has exceeded the 'limit_v' limit, incoming messages must not be decrypted and verified using the Recipient Key, and their processing must be aborted.

3. Security Considerations

This document mainly covers security considerations about using AEAD keys in OSCORE and their usage limits, in addition to the security considerations of [RFC8613].

[TODO: Add more considerations.]

4. IANA Considerations

This document has no IANA actions.

5. References

5.1. Normative References

[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>.

5.2. Informative References

[I-D.ietf-lake-edhoc] Selander, G., Mattsson, J. P., and F. Palombini, "Ephemeral Diffie-Hellman Over COSE (EDHOC)", Work in Progress, Internet-Draft, draft-ietf-lake-edhoc-22, 25 August 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-lake-edhoc-22>.

[I-D.irtf-cfrg-aead-limits] Günther, F., Thomson, M., and C. A. Wood, "Usage Limits on AEAD Algorithms", Work in Progress, Internet-Draft, draft-irtf-cfrg-aead-limits-07, 31 May 2023, <https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-aead-limits-07>.

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <https://www.rfc-editor.org/info/rfc7519>.

[RFC7959] Bormann, C. and Z. Shelby, Ed., "Block-Wise Transfers in the Constrained Application Protocol (CoAP)", RFC 7959, DOI 10.17487/RFC7959, August 2016, <https://www.rfc-editor.org/info/rfc7959>.

[RFC8323] Bormann, C., Lemay, S., Tschofenig, H., Hartke, K., Silverajan, B., and B. Raymor, Ed., "CoAP (Constrained Application Protocol) over TCP, TLS, and WebSockets", RFC 8323, DOI 10.17487/RFC8323, February 2018, <https://www.rfc-editor.org/info/rfc8323>.

Appendix A. Detailed considerations for AEAD_AES_128_CCM_8

For the AEAD_AES_128_CCM_8 algorithm when used as AEAD Algorithm for OSCORE, larger IA and CA values are achieved, depending on the value of 'q', 'v' and 'l'. Figure 3 shows the resulting IA and CA probabilities enjoyed by AEAD_AES_128_CCM_8, when taking different values of 'q', 'v' and 'l' as input to the formulas defined in [I-D.irtf-cfrg-aead-limits].

As shown in Figure 3, it is especially possible to achieve the lowest IA = 2^-50 and a good CA = 2^-70 by considering the largest possible value of the (q, v, l) triplet equal to (2^20, 2^10, 2^8), while still keeping a good security level. Note that the value of 'l' does not impact on IA, while CA displays good values for every considered value of 'l'.

'q', 'v' and 'l'	IA probability	CA probability
q=2^20, v=2^20, l=2^8	2^-44	2^-70
q=2^15, v=2^20, l=2^8	2^-44	2^-80
q=2^10, v=2^20, l=2^8	2^-44	2^-90
q=2^20, v=2^15, l=2^8	2^-49	2^-70
q=2^15, v=2^15, l=2^8	2^-49	2^-80
q=2^10, v=2^15, l=2^8	2^-49	2^-90
q=2^20, v=2^14, l=2^8	2^-50	2^-70
q=2^15, v=2^14, l=2^8	2^-50	2^-80
q=2^10, v=2^14, l=2^8	2^-50	2^-90
q=2^20, v=2^10, l=2^8	2^-54	2^-70
q=2^15, v=2^10, l=2^8	2^-54	2^-80
q=2^10, v=2^10, l=2^8	2^-54	2^-90
q=2^20, v=2^20, l=2^6	2^-44	2^-74
q=2^15, v=2^20, l=2^6	2^-44	2^-84
q=2^10, v=2^20, l=2^6	2^-44	2^-94
q=2^20, v=2^15, l=2^6	2^-49	2^-74
q=2^15, v=2^15, l=2^6	2^-49	2^-84
q=2^10, v=2^15, l=2^6	2^-49	2^-94
q=2^20, v=2^14, l=2^6	2^-50	2^-74
q=2^15, v=2^14, l=2^6	2^-50	2^-84
q=2^10, v=2^14, l=2^6	2^-50	2^-94
q=2^20, v=2^10, l=2^6	2^-54	2^-74
q=2^15, v=2^10, l=2^6	2^-54	2^-84
q=2^10, v=2^10, l=2^6	2^-54	2^-94

Figure 3: Probabilities for AEAD_AES_128_CCM_8 based on chosen q, v and l values.

Appendix B. Estimation of 'count_q'

This section defines a method to compute an estimate of the counter 'count_q' (see Section 2.2.2), hence not requiring a peer to store it in its own Sender Context.

This method relies on the fact that, at any point in time, a peer has performed *at most* ENC = (SSN + SSN*) encryptions using its own Sender Key, where:

*SSN is the current value of this peer's Sender Sequence Number.

SSN is the current value of other peer's Sender Sequence Number. That is, SSN* is an overestimation of the responses without Partial IV that this peer has sent.

Thus, when protecting an outgoing message (see Section 2.3.1), the peer aborts the message processing if the estimated est_q > limit_q, where est_q = (SSN + X) and X is determined as follows.

- If the outgoing message is a response, X is the Partial IV specified in the corresponding request that this peer is responding to. Note that X < SSN* always holds.

- If the outgoing message is a request, X is the highest Partial IV value marked as received in this peer's Replay Window plus 1, or 0 if it has not accepted any protected message from the other peer yet. That is, X is the highest Partial IV specified in message received from the other peer, i.e., the highest seen Sender Sequence Number of the other peer. Note that, also in this case, X < SSN* always holds.

Appendix C. Document Updates

RFC EDITOR: PLEASE REMOVE THIS SECTION.

C.1. Version -01 to -02

- Updated references

C.2. Version -00 to -01

- Extended discussion on setting the lifetime of OSCORE Security Contexts.

- Mention adjusting the 'q' and 'v' values to compensate for a larger 'l' value.

- Specify how to perform pre-calculation of message size to determine need for block-wise.

- Cover exceptional cases where the 'l' value needs to be larger than 2^8.

- Note on relevance of 'l' limit considering maximum block size and typical MTU.

C.3. Version -00

- Editorial improvements.

- Extended terminology.

- Recommendation on limits for CCM_8. Details in Appendix.

- Example of method to estimate and not store 'count_q'.

*Split out material from Key Update for OSCORE draft into this new document.

Acknowledgments

The authors sincerely thank Christian Amsüss, Carsten Bormann, John Preuß Mattsson, Göran Selander and Rafa Marin-Lopez for their feedback and comments.

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

Rikard Höglund
RISE AB
Isafjordsgatan 22
SE-16440 Stockholm Kista
Sweden

Email: rikard.hoglund@ri.se

Marco Tiloca
RISE AB
Isafjordsgatan 22
SE-16440 Stockholm Kista
Sweden

Email: marco.tiloca@ri.se