(No Working Group)   G. Amringer
Internet-Draft   January 9, 2020
Intended status: Informational
Expires: July 12, 2020

Chacha derived AEAD algorithms in JSON Object Signing and Encryption (JOSE)
draft-amringer-jose-chacha-02

Abstract

This document defines how to use the AEAD algorithms "AEAD_XCHACHA20_POLY1305" and "AEAD_CHACHA20_POLY1305" from [RFC8439] and [I-D.irtf-cfrg-xchacha] in JSON Object Signing and Encryption (JOSE).

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The Internet Research Task Force (IRTF) Crypto Forum Research Group (CFRG) defined the ChaCha20 and Poly1305 algorithms to be used in IETF protocols both independantly and as an AEAD construction ([RFC8439]). It has also been presented with a definition of an eXtended-nonce variant ([I-D.irtf-cfrg-xchacha]) for use in stateless contexts. This document defines how to use those algorithms in JOSE in an interoperable manner.

This document defines the conventions to use in the context of [RFC7516], and [RFC7517].

1.1. Notation and Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

The JOSE key format ("JSON Web Key (JWK)") is defined by [RFC7517] and thumbprints for it ("JSON Web Key (JWK) Thumbprint") in [RFC7638].

2. Key Encryption

2.1. Algorithms

This section defines the specifics of encrypting a JWE Content Encryption Key (CEK) with AEAD_CHACHA20_POLY1305 [RFC8439] and AEAD_XCHACHA20_POLY1305 [I-D.irtf-cfrg-xchacha].

Use of an Initialization Vector (IV) is REQUIRED with this algorithm. The IV is represented in base64url-encoded form as the "iv" (initialization vector) Header Parameter value.

The Additional Authenticated Data value used is the empty octet string.

The JWE Encrypted Key value is the ciphertext output.

The Authentication Tag output is represented in base64url-encoded form as the "tag" (authentication tag) Header Parameter value.

The following "alg" (algorithm) Header Parameter values are used to indicate that the JWE Encrypted Key is the result of encrypting the CEK using the corresponding algorithm and IV size:

Algorithm	IV size	"alg" value
AEAD_CHACHA20_POLY1305	96 bits	C20PKW
AEAD_XCHACHA20_POLY1305	192 bits	XC20PKW

2.2. Header Parameters Used for Key Encryption

The following Header Parameters are used for both algorithms defined for key encryption.

2.2.1. "iv" (Initialization Vector) Header Parameter

The "iv" (initialization vector) Header Parameter value is the base64url-encoded representation of the 96-bit or 192-bit IV value used for the key encryption operation. This Header Parameter MUST be present and MUST be understood and processed by implementations when these algorithms are used.

2.2.2. "tag" (Authentication Tag) Header Parameter

The "tag" (authentication tag) Header Parameter value is the base64url-encoded representation of the 128-bit Authentication Tag value resulting from the key encryption operation. This Header Parameter MUST be present and MUST be understood and processed by implementations when these algorithms are used.

3. Key Agreement with Elliptic Curve Diffie-Hellman Ephemeral Static

This section defines the specifics of key agreement with Elliptic Curve Diffie-Hellman Ephemeral Static [RFC6090], in combination with the Concat KDF, as defined in Section 5.8.2.1 of NIST.800-56A [1] for use as a symmetric key to wrap the CEK with the "C20PKW", or "XC20PKW" algorithms, in the Key Agreement with Key Wrapping mode.

This mode is used exactly as defined in Section 4.6 of RFC7518 [2], except that the combined key wrapping algorithms are the ones indicated in this document. All headers pertaining to both the ECDH-ES and key wrapping components ("iv",' "tag", "epk", "apu", "apv") have the same meaning and requirement as in their original definitions.

The following "alg" (algorithm) Header Parameter values are used to indicate that the JWE Encrypted Key is the result of encrypting the CEK using the corresponding algorithm:

"alg" value	Key Management Algorithm
ECDH-ES+C20PKW	ECDH-ES using Concat KDF and CEK wrapped with C20PKW
ECDH-ES+XC20PKW	ECDH-ES using Concat KDF and CEK wrapped with XC20PKW

4. Content Encryption

4.1. Algorithms

This section defines the specifics of performing authenticated encryption with ChaCha20-Poly1305.

The CEK is used as the encryption key.

Use of an IV is REQUIRED with this algorithm.

The following "enc" (encryption algorithm) Header Parameter values are used to indicate that the JWE Ciphertext and JWE Authentication Tag values have been computed using the corresponding algorithm and IV size:

Algorithm	IV size	"alg" value
AEAD_CHACHA20_POLY1305	96 bits	C20P
AEAD_XCHACHA20_POLY1305	192 bits	XC20P

5. IANA Considerations

The following is added to the "JSON Web Signature and Encryption Algorithms" registry:

o Algorithm Name: "C20PKW"
o Algorithm Description: Key wrapping with ChaCha20-Poly1305
o Algorithm Usage Location(s): "alg"
o JOSE Implementation Requirements: Optional
o Change Controller: IESG
o Specification Document(s): Section 2 of [RFC-THIS]
o Algorithm Analysis Documents(s): [RFC8439]

o Algorithm Name: "XC20PKW"
o Algorithm Description: Key wrapping with XChaCha20-Poly1305
o Algorithm Usage Location(s): "alg"
o JOSE Implementation Requirements: Optional
o Change Controller: IESG
o Specification Document(s): Section 2 of [RFC-THIS]
o Algorithm Analysis Documents(s): [I-D.irtf-cfrg-xchacha]

o Algorithm Name: "ECDH-ES+C20PKW"
o Algorithm Description: ECDH-ES using Concat KDF and "C20PKW" wrapping
o Algorithm Usage Location(s): "alg"
o JOSE Implementation Requirements: Optional
o Change Controller: IESG
o Specification Document(s): Section 3 of [RFC-THIS]
o Algorithm Analysis Documents(s): n/a

o Algorithm Name: "ECDH-ES+XC20PKW"
o Algorithm Description: ECDH-ES using Concat KDF and "XC20PKW" wrapping
o Algorithm Usage Location(s): "alg"
o JOSE Implementation Requirements: Optional
o Change Controller: IESG

o Specification Document(s): Section 3 of [RFC-THIS]
o Algorithm Analysis Documents(s): n/a

o Algorithm Name: "C20P"
o Algorithm Description: ChaCha20-Poly1305
o Algorithm Usage Location(s): "enc"
o JOSE Implementation Requirements: Optional
o Change Controller: IESG
o Specification Document(s): Section 4 of [RFC-THIS]
o Algorithm Analysis Documents(s): [RFC8439]

o Algorithm Name: "XC20P"
o Algorithm Description: ChaCha20-Poly1305
o Algorithm Usage Location(s): "enc"
o JOSE Implementation Requirements: Optional
o Change Controller: IESG
o Specification Document(s): Section 4 of [RFC-THIS]
o Algorithm Analysis Documents(s): [I-D.irtf-cfrg-xchacha]

6. References

6.1. Normative References

[I-D.irtf-cfrg-xchacha]
Arciszewski, S., "XChaCha: eXtended-nonce ChaCha and AEAD_XChaCha20_Poly1305", draft-irtf-cfrg-xchacha-01 (work in progress), July 2019.

[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>.

[RFC6090] McGrew, D., Igoe, K., and M. Salter, "Fundamental Elliptic Curve Cryptography Algorithms", RFC 6090, DOI 10.17487/RFC6090, February 2011, <https://www.rfc-editor.org/info/rfc6090>.

[RFC7516] Jones, M. and J. Hildebrand, "JSON Web Encryption (JWE)", RFC 7516, DOI 10.17487/RFC7516, May 2015, <https://www.rfc-editor.org/info/rfc7516>.

[RFC7517] Jones, M., "JSON Web Key (JWK)", RFC 7517, DOI 10.17487/RFC7517, May 2015, <https://www.rfc-editor.org/info/rfc7517>.

[RFC7638] Jones, M. and N. Sakimura, "JSON Web Key (JWK) Thumbprint", RFC 7638, DOI 10.17487/RFC7638, September 2015, <https://www.rfc-editor.org/info/rfc7638>.

[RFC8439] Nir, Y. and A. Langley, "ChaCha20 and Poly1305 for IETF Protocols", RFC 8439, DOI 10.17487/RFC8439, June 2018, <https://www.rfc-editor.org/info/rfc8439>.

6.2. URIs

[1] https://csrc.nist.gov/publications/detail/sp/800-56a/rev-3/final

[2] https://tools.ietf.org/html/rfc7518#section-4.6

Appendix A. Example using XC20PKW

Considering the payload of "Hello World!" (Base64URL):

SGVsbG8gV29ybGQh

We begin by generating the XChacha20-Poly1309 content encryption key (Base64URL):

la2knCeFPAvUE2IVPm-RNrwj4UrHffLU6Y1tx3d5T1Q

We follow by encrypting the CEK using XChacha20-Poly1309 itself. We generate a new key and a nonce:

KEK (Base64URL)

Rpv7sxPJYeNjKr-L8gPrKtQLHX-1dDuqtJuriVQ0eUY

Nonce (Base64URL)

LuNNS5RAagkOQVewQOLRp9noXET_YsPX

Using those parameters, we end up with the following output from XChacha20-Poly1309:

Ciphertext (Base64URL)

K-kXEFjmSsjKzU91

Tag (Base64URL)

VT2Z9a93JFe2om2gboUz4g

We then construct the following JWE header:

{"alg":"XC20PKW","enc":"XC20P","iv":"LuNNS5RAagkOQVewQOLRp9noXET
_YsPX","tag":"VT2Z9a93JFe2om2gboUz4g"}

The next step is to prepare the content encryption:

AAD (Base64URL)

eyJhbGciOiJYQzIwUEtXIiwiZW5jIjoiWEMyMFAiLCJpdiI6Ikx1Tk5TNVJBYWdr
T1FWZXdRT0xScDlub1hFVF9Zc1BYIiwidGFnIjoiVlQyWjlhOTNKRmUyb20yZ2Jv
VXo0ZyJ9

Key (generated earlier):

la2knCeFPAvUE2IVPm-RNrwj4UrHffLU6Y1tx3d5T1Q

Nonce (Base64URL)

LHs6vru3ggyuAzgT2UJkWyqJuZSv0Gae

We then encrypt the payload with XChacha20-Poly1309 using the previous parameters, which results in the following output:

Ciphertext (Base64URL)

QgxRd4qQrkQNaEK3

Tag (Base64URL)

aQDs_RkdWabvzmxYEnoShg

Lastly, we combine all the previous outputs to form the following JWE:

eyJhbGciOiJYQzIwUEtXIiwiZW5jIjoiWEMyMFAiLCJpdiI6Ikx1Tk5TNVJBYWdr
T1FWZXdRT0xScDlub1hFVF9Zc1BYIiwidGFnIjoiVlQyWjlhOTNKRmUyb20yZ2Jv
VXo0ZyJ9.K-kXEFjmSsjKzU91.LHs6vru3ggyuAzgT2UJkWyqJuZSv0Gae.QgxRd
4qQrkQNaEK3.aQDs_RkdWabvzmxYEnoShg

Appendix B. Example using ECDH-ES+XC20PKW

Considering the payload of "Hello World!" (Base64URL):

SGVsbG8gV29ybGQh

We begin by generating the XChacha20-Poly1309 content encryption key (Base64URL):

O2-TuP5Qz_ab6N61LhVS6asFdN_X5zF0YhJ6Df0vtoE

We follow by encrypting the CEK using XChacha20-Poly1309 itself following a key agreement. We generate a new key pair:

Private X (Base64URL)

OtbkdOjp6SIgQ-TMXlqg48Ds8ycsSCxadJrjCurCcSM

Public X (Base64URL)

xxXXpDLvS0z-Zlx5J6dsVPPVonYufe9zTKfat0dEryM

Using the recipient public key, we generate a shared secret

Recipient PK (Base64URL)

8llBJmFOkoFO8TYhFyDFm90Z8c6ytiD18wUgM5alCHY

Shared Secret (Base64URL)

llX-1dAQU6BiuTDUq4DgRy9Ob-1zoLp-1hvmKa8baGk

We can now derive a KEK:

APU (Base64URL)

Q2tkNDNqSkZNb2FHeGVJZW9FUHgtNF9SYlNmLWd1T19MRHpvbDhrLWFnM2NELXhm
dzdWX1IzM0lXVHRDZ0NqVmhmWTVQa29aT3AyTGwxZTR5ZWZ4d2c

KEK (Base64URL)

jPC4ybPvJ-FF4qz7hYiHDxr7XGQdQCMDjWaQ-y_MJfQ

We can now perform XChacha20-Poly1309 on the CEK using a new random nonce:

Nonce (Base64URL)

1Ef_Hs3NdFIujh9-uZEYLz4N_b1K1CJl

Ciphertext (Base64URL)

mzHMc5XlqW-jkGP4

Tag (Base64URL)

G8A4JnNmsG2wgvQh6Q5A8g

We then construct the following JWE header:

{"alg":"ECDH-ES+XC20PKW","enc":"XC20P","iv":"1Ef_Hs3NdFIujh9-uZE
YLz4N_b1K1CJl","tag":"G8A4JnNmsG2wgvQh6Q5A8g","apu":"Q2tkNDNqSkZ
Nb2FHeGVJZW9FUHgtNF9SYlNmLWd1T19MRHpvbDhrLWFnM2NELXhmdzdWX1IzM0l
XVHRDZ0NqVmhmWTVQa29aT3AyTGwxZTR5ZWZ4d2c","epk":{"typ":"OKP","cr
v":"X25519","x":"xxXXpDLvS0z-Zlx5J6dsVPPVonYufe9zTKfat0dEryM"}}

The next step is to prepare the content encryption:

AAD (Base64URL)

eyJhbGciOiJFQ0RILUVTK1hDMjBQS1ciLCJlbmMiOiJYQzIwUCIsIml2IjoiMUVm
X0hzM05kRkl1amg5LXVaRVlMejROX2IxSzFDSmwiLCJ0YWciOiJHOEE0Sm5ObXNH
MndndlFoNlE1QThnIiwiYXB1IjoiUTJ0a05ETnFTa1pOYjJGSGVHVkpaVzlGVUhn
dE5GOVNZbE5tTFdkMVQxOU1SSHB2YkRockxXRm5NMk5FTFhobWR6ZFdYMUl6TTBs
WFZIUkRaME5xVm1obVdUVlFhMjlhVDNBeVRHd3haVFI1WldaNGQyYyIsImVwayI6
eyJ0eXAiOiJPS1AiLCJjcnYiOiJYMjU1MTkiLCJ4IjoieHhYWHBETHZTMHotWmx4
NUo2ZHNWUFBWb25ZdWZlOXpUS2ZhdDBkRXJ5TSJ9fQ

Key (generated earlier):

O2-TuP5Qz_ab6N61LhVS6asFdN_X5zF0YhJ6Df0vtoE

Nonce (Base64URL)

okZz0AJz-PfUL4OGjioPLsg6-siwyq2I

We then encrypt the payload with XChacha20-Poly1309 using the previous parameters, which results in the following output:

Ciphertext (Base64URL)

yxpuuXB7DcXBlyVE

Tag (Base64URL)

IwvDEC8hxltfzidjmUKeMg

Lastly, we combine all the previous outputs to form the following JWE:

eyJhbGciOiJFQ0RILUVTK1hDMjBQS1ciLCJlbmMiOiJYQzIwUCIsIml2IjoiMUVm
X0hzM05kRkl1amg5LXVaRVlMejROX2IxSzFDSmwiLCJ0YWciOiJHOEE0Sm5ObXNH
MndndlFoNlE1QThnIiwiYXB1IjoiUTJ0a05ETnFTa1pOYjJGSGVHVkpaVzlGVUhn
dE5GOVNZbE5tTFdkMVQxOU1SSHB2YkRockxXRm5NMk5FTFhobWR6ZFdYMUl6TTBs
WFZIUkRaME5xVm1obVdUVlFhMjlhVDNBeVRHd3haVFI1WldaNGQyYyIsImVwayI6
eyJ0eXAiOiJPS1AiLCJjcnYiOiJYMjU1MTkiLCJ4IjoieHhYWHBETHZTMHotWmx4
NUo2ZHNWUFBWb25ZdWZlOXpUS2ZhdDBkRXJ5TSJ9fQ.mzHMc5XlqW-jkGP4.okZz
0AJz-PfUL4OGjioPLsg6-siwyq2I.yxpuuXB7DcXBlyVE.IwvDEC8hxltfzidjmU
KeMg

Author's Address

Guillaume Amringer
Canada

Email: g.amringer@gmail.com