Workgroup: Network Working Group
Published: 22 May 2020
Intended Status: Informational
Expires: 23 November 2020
Authors: J. Schaad
August Cellars

CBOR Object Signing and Encryption (COSE): Additional Algorithms

Abstract

The CBOR Object Signing and Encryption (COSE) syntax [I-D.ietf-cose-rfc8152bis-struct] allows for adding additional algorithms to the registries. This document adds one additional key wrap algorithm to the registry using the AES Wrap with Padding Algorithm [RFC5649]. This document adds Keccak Message Authentication Code (KMAC) algorithms as well as using KMAC as a Key Derivation Function (KDF).

Contributing to this document

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

The source for this draft is being maintained in GitHub. Suggested changes should be submitted as pull requests at https://github.com/cose-wg/X509 Editorial changes can be managed in GitHub, but any substantial issues need to be discussed on the COSE mailing list.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The CBOR Object Signing and Encryption (COSE) syntax [I-D.ietf-cose-rfc8152bis-struct] is defined to have an object based set of security primitives using CBOR [I-D.ietf-cbor-7049bis] for use in constrained environments. COSE has algorithm agility so that documents like this one can register algorithms which are needed.

In this document we add:

* The AES Wrap with Padding algorithm.

* Keccak Message Authentication Code (KMAC) algorithms.

* KMAC as a Key Derivation Function (KDF) for direct and key agreement algorithms.

1.1. Requirements 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.

1.2. Open Issues

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

* Should 192-bit AES Key Wrap be omitted or just given a large identifier? (John)

* Add the cSHAKE algorithms to the list? (Bob)

* RESOLVED: A desire has been expressed to all for the use of AES Key Wrap with Padding as a content encryption algorithm. This is not compatible with the requirement that all content encryption algorithms "support authentication of both the content and additional data." AES Key Wrap is an AE not an AEAD algorithm. (Jim) Response: Russ said it was ok just to be a key wrap algorithm.

2. Signature Algorithms

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

This document defines no new signature algorithms.

3. Message Authentication Code (MAC) Algorithms

3.1. Keccak Message Authentication Code (KMAC)

As part of the definition of the SHA-3 algorithms, NIST also defined a number of algorithms that are based on SHA-3 [NIST-800-185]. The Keccak Message Authentication Code (KMAC) is defined in that document. KMAC has a big performance advantage when compared to Hash-Based Message Authentication Code (HMAC) [RFC2104] [RFC4231] as it was designed to deal with the length extension attacks that forced the two pass structure of HMAC.

KMAC is parameterized with four inputs:

- *K - the key used for authentication
- *X - the byte string to be authenticated
- *L - the size of the authentication value in bits. This **MUST** be at least 64 and **SHOULD** be at least 128.
- *S - customization string which shall be a zero length byte string.

The algorithm identifier does not encode the length of the authentication tag, unlike the MAC algorithms defined in [I-D.ietf-cose-rfc8152bis-algs]. This is because shortened tags for those algorithms are generated by truncating a longer output. However, KMAC takes the resultant output length as one of the parameters and will generate different outputs depending on the length. The length of the MAC code is therefore chosen by the sender, and the length is inferred from the actual tag by the validator. If an attacker attempts to gain an advantage by shortening the tag, KMAC is not going to generate the correct tag.

Name	Value	Description	Recommended
KMAC 128	TBD4	KMAC w/ SHA-3 128-bits	Yes
KMAC 256	TBD5	KMAC w/ SHA-3 256-bits	Yes

Table 1

When using a COSE key for this algorithm, the following checks are made:

- *The 'kty' field **MUST** be present, and it **MUST** be 'Symmetric'.
- *If the 'alg' field is present, it **MUST** match the KMAC algorithm being used.
- *If the 'key_ops' field is present, it **MUST** include 'MAC create' when creating an KMAC authentication tag.

* If the 'key_ops' field is present, it **MUST** include 'MAC verify' when verifying an KMAC authentication tag.

Implementations creating and validating MAC values **MUST** validate that the key type, key length, and algorithm are correct and appropriate for the entities involved.

4. AES Key Wrap with Padding

The AES Key Wrap with Padding is defined in [RFC5649]. This algorithm uses an AES key to wrap a value that is a multiple of 8 bits. As such, it can be used to wrap not only the key sizes for the content encryption algorithms, but additionally it can be used to encrypt off size keys that can be used with the keyed hash functions or key derivation functions. The algorithm uses a single fixed parameter, the initial value. This value is fixed in section 3 of [RFC5649], this is a different value from that used for the AES Key Wrap algorithm of [RFC3394]. There are no public parameters that very on a per-invocation bases. This algorithm does not support additional data and thus the protected header field MUST be empty.

Name	Value	Key Size	Description	Recommended
A128KW-Pad	TBD1	128	AES Key Wrap w/padding and a 128-bit key	Yes
A192KW-Pad	TBD2	192	AES Key Wrap w/padding and a 192-bit key	No
A256KW-Pad	TBD3	256	AES Key Wrap w/padding and a 256-bit key	Yes

Table 2: AES Key Wrap Algorithm Values

When using a COSE key for this algorithm, the following checks are made:

* The 'kty' field **MUST** be present, and it **MUST** be 'Symmetric'.

* If the 'alg' field is present, it **MUST** match the AES Key Wrap algorithm being used.

* If the 'key_ops' field is present, it **MUST** include 'encrypt' or 'wrap key' when encrypting.

* If the 'key_ops' field is present, it **MUST** include 'decrypt' or 'unwrap key' when decrypting.

4.1. Security Considerations for AES-KW with Padding

The shared secret needs to have some method to be regularly updated over time. The shared secret is the basis of trust.

5. Key Derivation Functions (KDFs)

5.1. KMAC KDF

KMAC can additionally be used as a key derivation function [NIST-800-56C]. KMAC has a big advantage over the HKDF function, defined in [HKDF], as it executes the hashing function once as opposed to either two or four times for HKDF w/ HMAC SHA-256. This advantage may be offset by having SHA-256 in hardware and KMAC in software, so that should be one consideration in deciding which one to use.

The KMAC-KDF algorithm takes these inputs:

*secret -- a shared value that is secret. Secrets may be either previously shared or derived from operations like a Diffie-Hellman (DH) key agreement.

*salt -- an optional value that is used to change the generation process. The salt value can be either public or private. If the salt is public and carried in the message, then the 'salt' algorithm header parameter defined in Table 9 of [I-D.ietf-cose-rfc8152bis-algs] is used. While [HKDF] suggests that the length of the salt be the same as the length of the underlying hash value, any positive salt length will improve the security as different key values will be generated. This parameter is protected by being included in the key computation and does not need to be separately authenticated. The salt value does not need to be unique for every message sent.

*length -- the number of bytes of output that need to be generated.

*context information -- Information that describes the context in which the resulting value will be used. Making this information specific to the context in which the material is going to be used ensures that the resulting material will always be tied to that usage. The context structure defined in Section 5.2 of [I-D.ietf-cose-rfc8152bis-algs] is used by the KDFs in this document.

Full details of how the key derivation works can be found in Section 4 of [NIST-800-56C]. A quick summary of the details is provided here for simplicity. The KMAC function call is:

```
       Result = KMAC#(salt, x, outputBits, "KDF")
```

where:

*salt is the same parameter as above

* x is built as *counter* || *Z* || *FixedInfo*. Where counter is a 4-byte unsigned integer of 0, Z is the secret, and FixedInfo is the context information.
* outputBits is length * 8

One algorithm parameter is defined for the KMAC-KDF function.

Name	Label	Type	Algorithm	Description
salt	-20	bstr	direct+KMAC-128-KDF, direct+KMAC-256-KDF, ECDH-ES+KMAC-128-KDF, ECDH-ES+KMAC-256-KDF, ECDH-SS+KMAC-128-KDF, ECDH-SS+KMAC-256-KDF ECDH-ES+KMAC-128-KDF+A128KW, ECDH-ES+KMAC-256-KDF+A128KW, ECDH-SS+KMAC-128-KDF+A128KW, ECDH-SS+KMAC-256-KDF+A128KW ECDH-ES+KMAC-256-KDF+A256KW, ECDH-ES+KMAC-256-KDF+A256KW, ECDH-SS+KMAC-256-KDF+A256KW, ECDH-SS+KMAC-256-KDF+A256KW	Random salt

Table 3: KMAC-KDF Algorithm Parameters

6. Content Key Distribution Methods

6.1. Direct Key with KDF

These recipient algorithms take a common shared secret between the two parties and applies the KMAC-KDF function (Section 5.1), using the context structure defined in Section 5.2 of [I-D.ietf-cose-rfc8152bis-algs] to transform the shared secret into the CEK. The 'protected' field can be of non-zero length. Either the 'salt' parameter of KMAC-KDF or the 'PartyU nonce' parameter of the context structure **MUST** be present. The salt/nonce parameter can be generated either randomly or deterministically. The requirement is that it be a unique value for the shared secret in question.

If the salt/nonce value is generated randomly, then it is suggested that the length of the random value be the same length as the KMAC-KDF. While there is no way to guarantee that it will be unique, there is a high probability that it will be unique. If the salt/nonce value is generated deterministically, it can be guaranteed to be unique, and thus there is no length requirement.

A new IV must be used for each message if the same key is used. The IV can be modified in a predictable manner, a random manner, or an unpredictable manner (i.e., encrypting a counter).

The IV used for a key can also be generated from the same KMAC-KDF functionality as the key is generated. If KMAC-KDF is used for generating the IV, the algorithm identifier is set to "IV-GENERATION". Doing this requires that the context be modified for every IV generated to ensure that it is unique.

When these algorithms are used, the key type **MUST** be 'symmetric'.

The set of algorithms defined in this document can be found in Table 4.

Name	Value	KDF	Description
direct+KMAC-128	TBD6	KMAC-128	Shared secret w/ KMAC-128
direct+KMAC-256	TBD7	KMAC-256	Shared secret w/ KMAC-128

Table 4: Direct Key with KDF

When using a COSE key for this algorithm, the following checks are made:

*The 'kty' field **MUST** be present, and it **MUST** be 'Symmetric'.

*If the 'alg' field is present, it **MUST** match the algorithm being used.

*If the 'key_ops' field is present, it **MUST** include 'deriveKey' or 'deriveBits'.

6.1.1. Security Considerations

The shared secret needs to have some method to be regularly updated over time. The shared secret forms the basis of trust. Although not used directly, it should still be subject to scheduled rotation.

While these methods do not provide for perfect forward secrecy, as the same shared secret is used for all of the keys generated, if the key for any single message is discovered, only the message (or series of messages) using that derived key are compromised. A new key derivation step will generate a new key that requires the same amount of work to get the key.

6.2. Direct ECDH

This document adds to the set of Direct ECDH algorithms which were defined in Section 6.3 of [I-D.ietf-cose-rfc8152bis-algs]. This is done by adding a changing the KDF used to derive the shared secret.

Name	Value	KDF	Ephemeral-Static	Key Wrap	Description
ECDH-ES + KMAC-128	TBD8	KMAC-128	yes	none	ECDH ES w/ KMAC - generate key directly
ECDH-ES + KMAC-256	TBD9	KMAC-256	yes	none	ECDH ES w/ KMAC - generate key directly

Table 5: ECDH Algorithm Values

Both of these algorithms use the same set of the ECDH Algorithm Parameters as their HKDF counterparts.

This document defines these algorithms to be used with the curves P-256, P-384, P-521, X25519, and X448. Implementations **MUST** verify that the key type and curve are correct. Different curves are restricted to different key types. Implementations **MUST** verify that the curve and algorithm are appropriate for the entities involved.

When using a COSE key for this algorithm, the following checks are made:

*The 'kty' field **MUST** be present, and it **MUST** be 'EC2' or 'OKP'.

*If the 'alg' field is present, it **MUST** match the key agreement algorithm being used.

*If the 'key_ops' field is present, it **MUST** include 'derive key' or 'derive bits' for the private key.

*If the 'key_ops' field is present, it **MUST** be empty for the public key.

6.3. ECDH with Key Wrap

This document adds to the set of Direct ECDH algorithms which were defined in Section 6.4 of [I-D.ietf-cose-rfc8152bis-algs]. This is done by adding a changing the KDF used to derive the shared secret.

Name	Value	KDF	Ephemeral-Static	Key Wrap	Description
ECDH-ES + KMAC-128 + A128KW	TBD10	KMAC-128	yes	A128KW	ECDH ES w/ KMAC-128 and AES Key Wrap w/ 128-bit key
ECDH-ES + KMAC-256 + A256KW	TBD11	KMAC-256	yes	A256KW	ECDH ES w/ KMAC-256 and AES

Name	Value	KDF	Ephemeral-Static	Key Wrap	Description
					Key Wrap w/ 256-bit key
ECDH-SS + KMAC-128 + A128KW	TBD12	KMAC-128	yes	A128KW	ECDH SS w/ KMAC-128 and AES Key Wrap w/ 128-bit key
ECDH-SS + KMAC-256 + A256KW	TBD13	KMAC-256	yes	A256KW	ECDH SS w/ KMAC-256 and AES Key Wrap w/ 256-bit key

Table 6: ECDH Algorithm Values with Key Wrap

When using a COSE key for this algorithm, the following checks are made:

* The 'kty' field **MUST** be present, and it **MUST** be 'EC2' or 'OKP'.

* If the 'alg' field is present, it **MUST** match the key agreement algorithm being used.

* If the 'key_ops' field is present, it **MUST** include 'derive key' or 'derive bits' for the private key.

* If the 'key_ops' field is present, it **MUST** be empty for the public key.

7. Security Considerations

Decide on this - TBD

8. IANA Considerations

8.1. Changes to the Algorithm Table

IANA is requested to add new items to the "COSE Algorithms" registry. The content to be added can be found in Table 2. For all items to be added, the Reference column should be set to this document.

9. References

9.1. Normative References

[I-D.ietf-cose-rfc8152bis-struct]
Schaad, J., "CBOR Object Signing and Encryption (COSE): Structures and Process", Work in Progress, Internet-Draft, draft-ietf-cose-rfc8152bis-struct-08, 9 March

2020, <https://tools.ietf.org/html/draft-ietf-cose-rfc8152bis-struct-08>.

[I-D.ietf-cose-rfc8152bis-algs]
Schaad, J., "CBOR Object Signing and Encryption (COSE): Initial Algorithms", Work in Progress, Internet-Draft, draft-ietf-cose-rfc8152bis-algs-07, 9 March 2020, <https://tools.ietf.org/html/draft-ietf-cose-rfc8152bis-algs-07>.

[I-D.ietf-cbor-7049bis]
Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", Work in Progress, Internet-Draft, draft-ietf-cbor-7049bis-13, 8 March 2020, <https://tools.ietf.org/html/draft-ietf-cbor-7049bis-13>.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <https://www.rfc-editor.org/info/rfc2104>.

[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>.

[RFC4231] Nystrom, M., "Identifiers and Test Vectors for HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512", RFC 4231, DOI 10.17487/RFC4231, December 2005, <https://www.rfc-editor.org/info/rfc4231>.

[HKDF] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010, <https://www.rfc-editor.org/info/rfc5869>.

[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>.

[RFC5649] Housley, R. and M. Dworkin, "Advanced Encryption Standard (AES) Key Wrap with Padding Algorithm", RFC 5649, DOI 10.17487/RFC5649, September 2009, <https://www.rfc-editor.org/info/rfc5649>.

[RFC3394] Schaad, J. and R. Housley, "Advanced Encryption Standard (AES) Key Wrap Algorithm", RFC 3394, DOI 10.17487/RFC3394, September 2002, <https://www.rfc-editor.org/info/rfc3394>.

[NIST-800-185]
Kelsey, J., Change, S., and R. Perlner, "SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash, ParallelHash", December 2016, <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-185.pdf>.

[NIST-800-56C] Barker, E., Chen, L., and R. Davis, "Recommendation for Key-Derivation Methods in Key-Establishment Schemes"", March 2020, <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Cr2-draft.pdf>.

Author's Address

Jim Schaad
August Cellars

Email: ietf@augustcellars.com