

Network Working Group
Internet-Draft
Intended status: Informational
Expires: March 18, 2021

J. Schaad
August Cellars
September 14, 2020

CBOR Object Signing and Encryption (COSE): Hash Algorithms
draft-ietf-cose-hash-algs-09

Abstract

The CBOR Object Signing and Encryption (COSE) syntax [I-D.ietf-cose-rfc8152bis-struct] does not define any direct methods for using hash algorithms. There are, however, circumstances where hash algorithms are used, such as indirect signatures where the hash of one or more contents are signed, and X.509 certificate or other object identification by the use of a fingerprint. This document defines a set of hash algorithms that are identified by COSE Algorithm Identifiers.

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

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 March 18, 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Requirements Terminology	3
2. Hash Algorithm Usage	3
2.1. Example CBOR hash structure	4
3. Hash Algorithm Identifiers	5
3.1. SHA-1 Hash Algorithm	6
3.2. SHA-2 Hash Algorithms	6
3.3. SHAKE Algorithms	8
4. IANA Considerations	9
4.1. COSE Algorithm Registry	9
5. Security Considerations	10
6. Normative References	10
7. Informative References	11
Author's Address	12

1. Introduction

The CBOR Object Signing and Encryption (COSE) syntax does not define any direct methods for the use of hash algorithms. It also does not define a structure syntax that is used to encode a digested object structure along the lines of the DigestedData ASN.1 structure in [CMS]. This omission was intentional, as a structure consisting of just a digest identifier, the content, and a digest value does not, by itself, provide any strong security service. Additionally, an application is going to be better off defining this type of structure so that it can include any additional data that needs to be hashed, as well as methods of obtaining the data.

While the above is true, there are some cases where having some standard hash algorithms defined for COSE with a common identifier makes a great deal of sense. Two of the cases where these are going to be used are:

- * Indirect signing of content, and
- * Object identification.

Indirect signing of content is a paradigm where the content is not directly signed, but instead a hash of the content is computed and that hash value, along with an identifier for the hash algorithm, is included in the content that will be signed. Doing indirect signing allows for a signature to be validated without first downloading all of the content associated with the signature. Rather the signature can be validated on all of the hash values and pointers to the associated contents, then those associated parts can be downloaded, the hash value of that part computed, and then compared to the hash value in the signed content. This capability can be of even greater importance in a constrained environment as not all of the content signed may be needed by the device. An example of how this is used can be found in [I-D.ietf-suit-manifest].

The use of hashes to identify objects is something that has been very common. One of the primary things that has been identified by a hash function in a secure message is a certificate. Two examples of this can be found in [ESS] and the COSE equivalents in [I-D.ietf-cose-x509].

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.

2. Hash Algorithm Usage

As noted in the previous section, hash functions can be used for a variety of purposes. Some of these purposes require that a hash function be cryptographically strong. These include direct and indirect signatures. That is, using the hash as part of the signature or using the hash as part of the body to be signed. Other uses of hash functions may not require the same level of strength.

This document contains some hash functions that are not designed to be used for cryptographic operations. An application that is using a hash function needs to carefully evaluate exactly what hash properties are needed and which hash functions are going to provide them. Applications should also make sure that the ability to change hash functions is part of the base design, as cryptographic advances are sure to reduce the strength of a hash function [BCP201].

A hash function is a map from one, normally large, bit string to a second, usually smaller, bit string. As the number of possible input values is far greater than the number of possible output values, it is inevitable that there are going to be collisions. The trick is to make sure that it is difficult to find two values that are going to map to the same output value. A "Collision Attack" is one where an attacker can find two different messages that have the same hash value. A hash function that is susceptible to practical collision attacks, SHOULD NOT be used for a cryptographic purpose. The discovery of theoretical collision attacks against a given hash function SHOULD trigger protocol maintainers and users to do a review of the continued suitability of the algorithm if alternatives are available and migration is viable. The only reason why such a hash function is used is when there is absolutely no other choice (e.g. a Hardware Security Module (HSM) that cannot be replaced), and only after looking at the possible security issues. Cryptographic purposes would include the creation of signatures or the use of hashes for indirect signatures. These functions may still be usable for non-cryptographic purposes.

An example of a non-cryptographic use of a hash is for filtering from a collection of values to find a set of possible candidates; the candidates can then be checked to see if they can successfully be used. A simple example of this is the classic fingerprint of a certificate. If the fingerprint is used to verify that it is the correct certificate, then that usage is a cryptographic one and is subject to the warning above about collision attack. If, however, the fingerprint is used to sort through a collection of certificates to find those that might be used for the purpose of verifying a signature, a simple filter capability is sufficient. In this case, one still needs to confirm that the public key validates the signature (and the certificate is trusted), and all certificates that don't contain a key that validates the signature can be discarded as false positives.

To distinguish between these two cases, a new value in the recommended column of the COSE Algorithms registry is to be added. "Filter Only" indicates that the only purpose of a hash function should be to filter results and it is not intended for applications which require a cryptographically strong algorithm.

2.1. Example CBOR hash structure

[COSE] did not provide a default structure for holding a hash value not only because no separate hash algorithms were defined, but because how the structure is setup is frequently application specific. There are four fields that are often included as part of a hash structure:

- * The hash algorithm identifier.
- * The hash value.
- * A pointer to the value that was hashed. This could be a pointer to a file, an object that can be obtained from the network, or a pointer to someplace in the message, or something very application specific.
- * Additional data; this can be something as simple as a random value (i.e. salt) to make finding hash collisions slightly harder (as the payload handed to the application could have been selected to have a collision), or as complicated as a set of processing instructions that are used with the object that is pointed to. The additional data can be dealt with in a number of ways, prepending or appending to the content, but it is strongly suggested that it either be a fixed known size, or the lengths of the pieces being hashed be included. (Encoding as a CBOR array accomplishes this requirement.)

An example of a structure which permits all of the above fields to exist would look like the following.

```
COSE_Hash_V = (  
  1 : int / tstr, # Algorithm identifier  
  2 : bstr, # Hash value  
  ? 3 : tstr, # Location of object that was hashed  
  ? 4 : any # object containing other details and things  
)
```

Below is an alternative structure that could be used in situations where one is searching a group of objects for a matching hash value. In this case, the location would not be needed and adding extra data to the hash would be counterproductive. This results in a structure that looks like this:

```
COSE_Hash_Find = [  
  hashAlg : int / tstr,  
  hashValue : bstr  
]
```

3. Hash Algorithm Identifiers

3.1. SHA-1 Hash Algorithm

The SHA-1 hash algorithm [RFC3174] was designed by the United States National Security Agency and published in 1995. Since that time a large amount of cryptographic analysis has been applied to this algorithm and a successful collision attack has been created ([SHA-1-collision]). The IETF formally started discouraging the use of SHA-1 with the publishing of [RFC6194].

Despite the above, there are still times where SHA-1 needs to be used and therefore it makes sense to assign a codepoint for the use of this hash algorithm. Some of these situations are with historic HSMS where only SHA-1 is implemented; other situations are where the SHA-1 value is used for the purpose of filtering and thus the collision resistance property is not needed.

Because of the known issues for SHA-1 and the fact that it should no longer be used, the algorithm will be registered with the recommendation of "Filter Only". This provides guidance about when the algorithm is safe for use, while discouraging usage where it is not safe.

The COSE capabilities for this algorithm is an empty array.

Name	Value	Description	Capabilities	Reference	Recommended
SHA-1	-14	SHA-1 Hash	[]	[This Document]	Filter Only

Table 1: SHA-1 Hash Algorithm

3.2. SHA-2 Hash Algorithms

The family of SHA-2 hash algorithms [FIPS-180-4] was designed by the United States National Security Agency and published in 2001. Since that time some additional algorithms have been added to the original set to deal with length extension attacks and some performance issues. While the SHA-3 hash algorithms have been published since that time, the SHA-2 algorithms are still broadly used.

There are a number of different parameters for the SHA-2 hash functions. The set of hash functions which have been chosen for inclusion in this document are based on those different parameters and some of the trade-offs involved.

- * *SHA-256/64* provides a truncated hash. The length of the truncation is designed to allow for smaller transmission size. The trade-off is that the odds that a collision will occur increase proportionally. Use of this hash function needs analysis of the potential problems with having a collision occur, or must be limited to where the function of the hash is non-cryptographic.

The latter is the case for [I-D.ietf-cose-x509]. The hash value is used to select possible certificates and, if there are multiple choices remaining then, each choice can be tested by using the public key.

- * *SHA-256* is probably the most common hash function used currently. SHA-256 is an efficient hash algorithm for 32-bit hardware.
- * *SHA-384* and *SHA-512* hash functions are efficient for 64-bit hardware.
- * *SHA-512/256* provides a hash function that runs more efficiently on 64-bit hardware, but offers the same security levels as SHA-256.

The COSE capabilities array for these algorithms is empty.

Name	Value	Description	Capabilities	Reference	Recommended
SHA-256/64	-15	SHA-2 256-bit Hash truncated to 64-bits	[]	[This Document]	Filter Only
SHA-256	-16	SHA-2 256-bit Hash	[]	[This Document]	Yes
SHA-384	-43	SHA-2 384-bit Hash	[]	[This Document]	Yes
SHA-512	-44	SHA-2 512-bit Hash	[]	[This Document]	Yes
SHA-512/256	-17	SHA-2 512-bit Hash truncated to 256-bits	[]	[This Document]	Yes

Table 2: SHA-2 Hash Algorithms

3.3. SHAKE Algorithms

The family of SHA-3 hash algorithms [FIPS-202] was the result of a competition run by NIST. The pair of algorithms known as SHAKE-128 and SHAKE-256 are the instances of SHA-3 that are currently being standardized in the IETF. This is the reason for including these algorithms in this document.

The SHA-3 hash algorithms have a significantly different structure than the SHA-2 hash algorithms.

Unlike the SHA-2 hash functions, no algorithm identifier is created for shorter lengths. The length of the hash value stored is 256-bits for SHAKE-128 and 512-bits for SHAKE-256.

The COSE capabilities array for these algorithms is empty.

Name	Value	Description	Capabilities	Reference	Recommended
SHAKE128	-18	SHAKE-128 256-bit Hash Value	[]	[This Document]	Yes
SHAKE256	-45	SHAKE-256 512-bit Hash Value	[]	[This Document]	Yes

Table 3: SHAKE Hash Functions

4. IANA Considerations

The IANA actions in [I-D.ietf-cose-rfc8152bis-struct] and [I-D.ietf-cose-rfc8152bis-algs] need to be executed before the actions in this document. Where early allocation of codepoints has been made, these should be preserved.

4.1. COSE Algorithm Registry

IANA is requested to register the following algorithms in the "COSE Algorithms" registry.

- * The SHA-1 hash function found in Table 1.
- * The set of SHA-2 hash functions found in Table 2.
- * The set of SHAKE hash functions found in Table 3.

Many of the hash values produced are relatively long and as such the use of a two byte algorithm identifier seems reasonable. SHA-1 is tagged as 'Filter Only' and thus a longer algorithm identifier is appropriate even though it is a shorter hash value.

IANA is requested to add the value of 'Filter Only' to the set of legal values for the 'Recommended' column. This value is only to be used for hash functions and indicates that it is not to be used for purposes which require collision resistance. IANA is requested to add this document to the reference section for this table due to this addition.

5. Security Considerations

Protocols need to perform a careful analysis of the properties of a hash function that are needed and how they map onto the possible attacks. In particular, one needs to distinguish between those uses that need the cryptographic properties, such as collision resistance, and properties that correspond to possible object identification. The different attacks correspond to who or what is being protected: is it the originator that is the attacker or a third party? This is the difference between collision resistance and second pre-image resistance. As a general rule, longer hash values are "better" than short ones, but trade-offs of transmission size, timeliness, and security all need to be included as part of this analysis. In many cases the value being hashed is a public value and, as such, pre-image resistance is not part of this analysis.

Algorithm agility needs to be considered a requirement for any use of hash functions [BCP201]. As with any cryptographic function, hash functions are under constant attack and the cryptographic strength of hash algorithms will be reduced over time.

6. 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>>.
- [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>>.
- [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-12, August 24, 2020, <<https://tools.ietf.org/html/draft-ietf-cose-rfc8152bis-struct-12>>.
- [FIPS-180-4] National Institute of Standards and Technology, "Secure Hash Standard", FIPS PUB 180-4, August 2015.
- [FIPS-202] National Institute of Standards and Technology, "SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions", FIPS PUB 202, August 2015.

- [RFC3174] Eastlake 3rd, D. and P. Jones, "US Secure Hash Algorithm 1 (SHA1)", RFC 3174, DOI 10.17487/RFC3174, September 2001, <<https://www.rfc-editor.org/info/rfc3174>>.

7. Informative References

- [CMS] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <<https://www.rfc-editor.org/info/rfc5652>>.
- [ESS] Hoffman, P., Ed., "Enhanced Security Services for S/MIME", RFC 2634, DOI 10.17487/RFC2634, June 1999, <<https://www.rfc-editor.org/info/rfc2634>>.
- [I-D.ietf-cose-x509]
Schaad, J., "CBOR Object Signing and Encryption (COSE): Header parameters for carrying and referencing X.509 certificates", Work in Progress, Internet-Draft, draft-ietf-cose-x509-06, March 9, 2020, <<https://tools.ietf.org/html/draft-ietf-cose-x509-06>>.
- [RFC6194] Polk, T., Chen, L., Turner, S., and P. Hoffman, "Security Considerations for the SHA-0 and SHA-1 Message-Digest Algorithms", RFC 6194, DOI 10.17487/RFC6194, March 2011, <<https://www.rfc-editor.org/info/rfc6194>>.
- [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-11, July 1, 2020, <<https://tools.ietf.org/html/draft-ietf-cose-rfc8152bis-algs-11>>.
- [I-D.ietf-suit-manifest]
Moran, B., Tschofenig, H., Birkholz, H., and K. Zandberg, "A Concise Binary Object Representation (CBOR)-based Serialization Format for the Software Updates for Internet of Things (SUIT) Manifest", Work in Progress, Internet-Draft, draft-ietf-suit-manifest-09, July 13, 2020, <<https://tools.ietf.org/html/draft-ietf-suit-manifest-09>>.
- [BCP201] Housley, R., "Guidelines for Cryptographic Algorithm Agility and Selecting Mandatory-to-Implement Algorithms", BCP 201, RFC 7696, November 2015.

<<https://www.rfc-editor.org/info/bcp201>>

[SHA-1-collision]

Stevens, M., Bursztein, E., Karpman, P., Albertini, A.,
and Y. Markov, "The first collision for full SHA-1",
February 2017,
<<https://shattered.io/static/shattered.pdf>>.

[COSE]

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

Author's Address

Jim Schaad
August Cellars

Email: ietf@augustcellars.com

COSE Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 13, 2020

M. Jones
Microsoft
June 11, 2020

COSE and JOSE Registrations for WebAuthn Algorithms
draft-ietf-cose-webauthn-algorithms-08

Abstract

The W3C Web Authentication (WebAuthn) specification and the FIDO Alliance Client to Authenticator Protocol (CTAP) specification use CBOR Object Signing and Encryption (COSE) algorithm identifiers. This specification registers the following algorithms in the IANA "COSE Algorithms" registry, which are used by WebAuthn and CTAP implementations: RSASSA-PKCS1-v1_5 using SHA-256, SHA-384, SHA-512, and SHA-1, and ECDSA using the secp256k1 curve and SHA-256. It registers the secp256k1 elliptic curve in the IANA "COSE Elliptic Curves" registry. Also, for use with JSON Object Signing and Encryption (JOSE), it registers the algorithm ECDSA using the secp256k1 curve and SHA-256 in the IANA "JSON Web Signature and Encryption Algorithms" registry and the secp256k1 elliptic curve in the IANA "JSON Web Key Elliptic Curve" registry.

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 December 13, 2020.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Requirements Notation and Conventions	3
2. RSASSA-PKCS1-v1_5 Signature Algorithm	3
3. Using secp256k1 with JOSE and COSE	4
3.1. JOSE and COSE secp256k1 Curve Key Representations	5
3.2. ECDSA Signature with secp256k1 Curve	5
3.3. Other Uses of the secp256k1 Elliptic Curve	7
4. IANA Considerations	7
4.1. COSE Algorithms Registrations	7
4.2. COSE Elliptic Curves Registrations	8
4.3. JOSE Algorithms Registrations	8
4.4. JSON Web Key Elliptic Curves Registrations	9
5. Security Considerations	9
5.1. RSA Key Size Security Considerations	9
5.2. RSASSA-PKCS1-v1_5 with SHA-2 Security Considerations	9
5.3. RSASSA-PKCS1-v1_5 with SHA-1 Security Considerations	9
5.4. secp256k1 Security Considerations	9
6. References	10
6.1. Normative References	10
6.2. Informative References	11
Acknowledgements	12
Document History	12
Author's Address	14

1. Introduction

This specification defines how to use several algorithms with CBOR Object Signing and Encryption (COSE) [RFC8152] that are used by implementations of the W3C Web Authentication (WebAuthn) [WebAuthn] and FIDO Alliance FIDO2 Client to Authenticator Protocol (CTAP) [CTAP] specifications. This specification registers these algorithms in the IANA "COSE Algorithms" registry [IANA.COSE.Algorithms] and registers an elliptic curve in the IANA "COSE Elliptic Curves" registry [IANA.COSE.Curves]. This specification also registers a corresponding algorithm for use with JSON Object Signing and Encryption (JOSE) [RFC7515] in the IANA "JSON Web Signature and

Encryption Algorithms" registry [IANA.JOSE.Algorithms] and registers an elliptic curve in the IANA "JSON Web Key Elliptic Curve" registry [IANA.JOSE.Curves].

1.1. Requirements Notation and Conventions

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.

2. RSASSA-PKCS1-v1_5 Signature Algorithm

The RSASSA-PKCS1-v1_5 signature algorithm is defined in [RFC8017]. The RSASSA-PKCS1-v1_5 signature algorithm is parameterized with a hash function (h).

A key of size 2048 bits or larger MUST be used with these algorithms. Implementations need to check that the key type is 'RSA' when creating or verifying a signature.

The RSASSA-PKCS1-v1_5 algorithms specified in this document are in the following table.

Name	Value	Hash	Description	Recommended
RS256	TBD (temporary assignment -257 already in place)	SHA-256	RSASSA-PKCS1-v1_5 using SHA-256	No
RS384	TBD (temporary assignment -258 already in place)	SHA-384	RSASSA-PKCS1-v1_5 using SHA-384	No
RS512	TBD (temporary assignment -259 already in place)	SHA-512	RSASSA-PKCS1-v1_5 using SHA-512	No
RS1	TBD (temporary assignment -65535 already in place)	SHA-1	RSASSA-PKCS1-v1_5 using SHA-1	Deprecated

Table 1: RSASSA-PKCS1-v1_5 Algorithm Values

Security considerations for use of the first three algorithms are in Section 5.2. Security considerations for use of the last algorithm are in Section 5.3.

Note that these algorithms are already present in the IANA "JSON Web Signature and Encryption Algorithms" registry [IANA.JOSE.Algorithms], and so these registrations are only for the IANA "COSE Algorithms" registry [IANA.COSE.Algorithms].

3. Using secp256k1 with JOSE and COSE

This section defines algorithm encodings and representations enabling the Standards for Efficient Cryptography Group (SECG) elliptic curve secp256k1 [SEC2] to be used for JOSE [RFC7515] and COSE [RFC8152] messages.

3.1. JOSE and COSE secp256k1 Curve Key Representations

The Standards for Efficient Cryptography Group (SECG) elliptic curve secp256k1 [SEC2] is represented in a JSON Web Key (JWK) [RFC7517] using these values:

- o "kty": "EC"
- o "crv": "secp256k1"

plus the values needed to represent the curve point, as defined in Section 6.2.1 of [RFC7518]. As a compressed point encoding representation is not defined for JWK elliptic curve points, the uncompressed point encoding defined there MUST be used. The "x" and "y" values represented MUST both be exactly 256 bits, with any leading zeros preserved. Other optional values such as "alg" MAY also be present.

It is represented in a COSE_Key [RFC8152] using these values:

- o "kty" (1): "EC2" (2)
- o "crv" (-1): "secp256k1" (TBD - requested assignment 8)

plus the values needed to represent the curve point, as defined in Section 13.1.1 of [RFC8152]. Either the uncompressed or compressed point encoding representations defined there can be used. The "x" value represented MUST be exactly 256 bits, with any leading zeros preserved. If the uncompressed representation is used, the "y" value represented MUST likewise be exactly 256 bits, with any leading zeros preserved; if the compressed representation is used, the "y" value is a boolean value, as specified in Section 13.1.1 of [RFC8152]. Other optional values such as "alg" (3) MAY also be present.

3.2. ECDSA Signature with secp256k1 Curve

The ECDSA signature algorithm is defined in [DSS]. This specification defines the "ES256K" algorithm identifier, which is used to specify the use of ECDSA with the secp256k1 curve and the SHA-256 [DSS] cryptographic hash function. Implementations need to check that the key type is "EC" for JOSE or "EC2" (2) for COSE and that the curve of the key is secp256k1 when creating or verifying a signature.

The ECDSA secp256k1 SHA-256 digital signature is generated as follows:

1. Generate a digital signature of the JWS Signing Input or the COSE Sig_structure using ECDSA secp256k1 SHA-256 with the desired

private key. The output will be the pair (R, S), where R and S are 256-bit unsigned integers.

2. Turn R and S into octet sequences in big-endian order, with each array being 32 octets long. The octet sequence representations MUST NOT be shortened to omit any leading zero octets contained in the values.
3. Concatenate the two octet sequences in the order R and then S. (Note that many ECDSA implementations will directly produce this concatenation as their output.)
4. The resulting 64-octet sequence is the JWS Signature or COSE signature value.

Implementations SHOULD use a deterministic algorithm to generate the ECDSA nonce, *k*, such as [RFC6979]. However, in situations where devices are vulnerable to physical attacks, deterministic ECDSA has been shown to be susceptible to fault injection attacks [Kudelski17] [EuroSP18]. Where this is a possibility, implementations SHOULD implement appropriate countermeasures. Where there are specific certification requirements (such as FIPS approval), implementors should check whether deterministic ECDSA is an approved nonce generation method.

The ECDSA secp256k1 SHA-256 algorithm specified in this document uses these identifiers:

JOSE Alg Name	COSE Alg Value	Description	Recommended
ES256K	TBD (requested assignment -47)	ECDSA using secp256k1 curve and SHA-256	No

Table 2: ECDSA Algorithm Values

When using a JWK or COSE_Key for this algorithm, the following checks are made:

- o The "kty" field MUST be present and it MUST be "EC" for JOSE or "EC2" for COSE.
- o The "crv" field MUST be present and it MUST represent the "secp256k1" elliptic curve.

- o If the "alg" field is present, it MUST represent the "ES256K" algorithm.
- o If the "key_ops" field is present, it MUST include "sign" when creating an ECDSA signature.
- o If the "key_ops" field is present, it MUST include "verify" when verifying an ECDSA signature.
- o If the JWK _use_ field is present, its value MUST be "sig".

3.3. Other Uses of the secp256k1 Elliptic Curve

This specification defines how to use the secp256k1 curve for ECDSA signatures for both JOSE and COSE implementations. While in theory, the curve could also be used for ECDH-ES key agreement, it is beyond the scope of this specification to state whether this is or is not advisable. Thus, whether to recommend its use with ECDH-ES is left for experts to decide in future specifications.

When used for ECDSA, the secp256k1 curve MUST be used only with the "ES256K" algorithm identifier and not any others, including not with the COSE "ES256" identifier. Note that the "ES256K" algorithm identifier needed to be introduced for JOSE to sign with the secp256k1 curve because the JOSE "ES256" algorithm is defined to be used only with the P-256 curve. The COSE treatment of how to sign with secp256k1 is intentionally parallel to that for JOSE, where the secp256k1 curve MUST be used with the "ES256K" algorithm identifier.

4. IANA Considerations

4.1. COSE Algorithms Registrations

This section registers the following values in the IANA "COSE Algorithms" registry [IANA.COSE.Algorithms].

- o Name: RS256
- o Value: TBD (temporary assignment -257 already in place)
- o Description: RSASSA-PKCS1-v1_5 using SHA-256
- o Reference: Section 2 of this document
- o Recommended: No

- o Name: RS384
- o Value: TBD (temporary assignment -258 already in place)
- o Description: RSASSA-PKCS1-v1_5 using SHA-384
- o Reference: Section 2 of this document
- o Recommended: No

- o Name: RS512
- o Value: TBD (temporary assignment -259 already in place)
- o Description: RSASSA-PKCS1-v1_5 using SHA-512
- o Reference: Section 2 of this document
- o Recommended: No

- o Name: RS1
- o Value: TBD (temporary assignment -65535 already in place)
- o Description: RSASSA-PKCS1-v1_5 using SHA-1
- o Reference: Section 2 of this document
- o Recommended: Deprecated

- o Name: ES256K
- o Value: TBD (requested assignment -47)
- o Description: ECDSA using secp256k1 curve and SHA-256
- o Reference: Section 3.2 of this document
- o Recommended: No

4.2. COSE Elliptic Curves Registrations

This section registers the following value in the IANA "COSE Elliptic Curves" registry [IANA.COSE.Curves].

- o Name: secp256k1
- o Value: TBD (requested assignment 8)
- o Key Type: EC2
- o Description: SECG secp256k1 curve
- o Change Controller: IESG
- o Reference: Section 3.1 of [[this specification]]
- o Recommended: No

4.3. JOSE Algorithms Registrations

This section registers the following value in the IANA "JSON Web Signature and Encryption Algorithms" registry [IANA.JOSE.Algorithms].

- o Algorithm Name: ES256K
- o Algorithm Description: ECDSA using secp256k1 curve and SHA-256
- o Algorithm Usage Locations: alg
- o JOSE Implementation Requirements: Optional
- o Change Controller: IESG
- o Reference: Section 3.2 of [[this specification]]
- o Algorithm Analysis Document(s): [SEC2]

4.4. JSON Web Key Elliptic Curves Registrations

This section registers the following value in the IANA "JSON Web Key Elliptic Curve" registry [IANA.JOSE.Curves].

- o Curve Name: secp256k1
- o Curve Description: SECG secp256k1 curve
- o JOSE Implementation Requirements: Optional
- o Change Controller: IESG
- o Specification Document(s): Section 3.1 of [[this specification]]

5. Security Considerations

5.1. RSA Key Size Security Considerations

The security considerations on key sizes for RSA algorithms from Section 6.1 of [RFC8230] also apply to the RSA algorithms in this specification.

5.2. RSASSA-PKCS1-v1_5 with SHA-2 Security Considerations

The security considerations on the use of RSASSA-PKCS1-v1_5 with SHA-2 hash functions (SHA-256, SHA-384, and SHA-512) from Section 8.3 of [RFC7518] also apply to their use in this specification. For that reason, these algorithms are registered as being "Not Recommended". Likewise, the exponent restrictions described in Section 8.3 of [RFC7518] also apply.

5.3. RSASSA-PKCS1-v1_5 with SHA-1 Security Considerations

The security considerations on the use of the SHA-1 hash function from [RFC6194] apply in this specification. For that reason, the "RS1" algorithm is registered as "Deprecated". Likewise, the exponent restrictions described in Section 8.3 of [RFC7518] also apply.

A COSE algorithm identifier for this algorithm is nonetheless being registered because deployed TPMs continue to use it, and therefore WebAuthn implementations need a COSE algorithm identifier for "RS1" when TPM attestations using this algorithm are being represented. New COSE applications and protocols MUST NOT use this algorithm.

5.4. secp256k1 Security Considerations

Care should be taken that a secp256k1 key is not mistaken for a P-256 [RFC7518] key, given that their representations are the same except for the "crv" value. As described in Section 8.1.1 of [RFC8152], we

currently do not have any way to deal with this attack except to restrict the set of curves that can be used.

The procedures and security considerations described in the [SEC1], [SEC2], and [DSS] specifications apply to implementations of this specification.

Timing side-channel attacks are possible if the implementation of scalar multiplication over the curve does not execute in constant time.

There are theoretical weaknesses with this curve that could result in future attacks. While these potential weaknesses are not unique to this curve, they are the reason that this curve is registered as "Recommended: No".

6. References

6.1. Normative References

- [DSS] National Institute of Standards and Technology (NIST), "Digital Signature Standard (DSS)", FIPS PUB 186-4, July 2013, <<http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf>>.
- [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>>.
- [RFC6194] Polk, T., Chen, L., Turner, S., and P. Hoffman, "Security Considerations for the SHA-0 and SHA-1 Message-Digest Algorithms", RFC 6194, DOI 10.17487/RFC6194, March 2011, <<https://www.rfc-editor.org/info/rfc6194>>.
- [RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <<https://www.rfc-editor.org/info/rfc7515>>.
- [RFC7517] Jones, M., "JSON Web Key (JWK)", RFC 7517, DOI 10.17487/RFC7517, May 2015, <<https://www.rfc-editor.org/info/rfc7517>>.
- [RFC7518] Jones, M., "JSON Web Algorithms (JWA)", RFC 7518, DOI 10.17487/RFC7518, May 2015, <<https://www.rfc-editor.org/info/rfc7518>>.

- [RFC8017] Moriarty, K., Ed., Kaliski, B., Jonsson, J., and A. Rusch, "PKCS #1: RSA Cryptography Specifications Version 2.2", RFC 8017, DOI 10.17487/RFC8017, November 2016, <<https://www.rfc-editor.org/info/rfc8017>>.
- [RFC8152] Schaad, J., "CBOR Object Signing and Encryption (COSE)", RFC 8152, DOI 10.17487/RFC8152, July 2017, <<https://www.rfc-editor.org/info/rfc8152>>.
- [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>>.
- [RFC8230] Jones, M., "Using RSA Algorithms with CBOR Object Signing and Encryption (COSE) Messages", RFC 8230, DOI 10.17487/RFC8230, September 2017, <<https://www.rfc-editor.org/info/rfc8230>>.
- [SEC1] Standards for Efficient Cryptography Group, "SEC 1: Elliptic Curve Cryptography", Version 2.0, May 2009, <<http://www.secg.org/sec1-v2.pdf>>.
- [SEC2] Standards for Efficient Cryptography Group, "SEC 2: Recommended Elliptic Curve Domain Parameters", Version 2.0, January 2010, <<http://www.secg.org/sec2-v2.pdf>>.

6.2. Informative References

- [CTAP] Brand, C., Czeskis, A., Ehrensvaerd, J., Jones, M., Kumar, A., Lindemann, R., Powers, A., and J. Verrept, "Client to Authenticator Protocol (CTAP)", FIDO Alliance Proposed Standard, January 2019, <<https://fidoalliance.org/specs/fido-v2.0-ps-20190130/fido-client-to-authenticator-protocol-v2.0-ps-20190130.html>>.
- [EuroSP18] Poddebniak, D., Somorovsky, J., Schinzel, S., Lochter, M., and P. Roesler, "Attacking Deterministic Signature Schemes using Fault Attacks", IEEE European Symposium on Security and Privacy (EuroS&P) 2018, April 2018, <<https://eprint.iacr.org/2017/1014.pdf>>.
- [IANA.COSE.Algorithms] IANA, "COSE Algorithms", <<https://www.iana.org/assignments/cose/cose.xhtml#algorithms>>.

- [IANA.COSE.Curves]
IANA, "COSE Elliptic Curves",
<<https://www.iana.org/assignments/cose/cose.xhtml#elliptic-curves>>.
- [IANA.JOSE.Algorithms]
IANA, "JSON Web Signature and Encryption Algorithms",
<<https://www.iana.org/assignments/jose/jose.xhtml#web-signature-encryption-algorithms>>.
- [IANA.JOSE.Curves]
IANA, "JSON Web Key Elliptic Curve",
<<https://www.iana.org/assignments/jose/jose.xhtml#web-key-elliptic-curve>>.
- [Kudelski17]
Romailler, Y., "How to defeat Ed25519 and EdDSA using faults", October 2017,
<<https://research.kudelskisecurity.com/2017/10/04/defeating-eddsa-with-faults/>>.
- [RFC6979] Pornin, T., "Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA)", RFC 6979, DOI 10.17487/RFC6979, August 2013, <<https://www.rfc-editor.org/info/rfc6979>>.
- [WebAuthn]
Balfanz, D., Czeskis, A., Hodges, J., Jones, J., Jones, M., Kumar, A., Liao, A., Lindemann, R., and E. Lundberg, "Web Authentication: An API for accessing Public Key Credentials - Level 1", World Wide Web Consortium (W3C) Recommendation, March 2019, <<https://www.w3.org/TR/2019/REC-webauthn-1-20190304/>>.

Acknowledgements

Thanks to Roman Danyliw, Linda Dunbar, Stephen Farrell, John Fontana, Jeff Hodges, Kevin Jacobs, J.C. Jones, Benjamin Kaduk, Murray Kucherawy, Neil Madden, John Mattsson, Matthew Miller, Tony Nadalin, Matt Palmer, Eric Rescorla, Rich Salz, Jim Schaad, Goeran Selander, Wendy Seltzer, Sean Turner, and Samuel Weiler for their roles in registering these algorithm identifiers.

Document History

[[to be removed by the RFC Editor before publication as an RFC]]

- o Addressed IESG review comments by Benjamin Kaduk and Roman Danyliw, primarily completing the edits to register secp256k1 and ES256K as "Recommended: No" for COSE. Some additional security considerations were also added.

-07

- o Addressed editorial SecDir review comment by Linda Dunbar about SHA-2 algorithms.
- o Addressed IETF last call comments by Jim Schaad, Rich Salz, and Eric Rescorla, now registering secp256k1 and ES256K as "Recommended: No" for COSE.

-06

- o Addressed Area Director review comment by Murray Kucherawy (which requested an editorial correction).
- o Changed requested assignment for ES256K from -46 to -47, due to an assignment conflict.

-05

- o Removed unused reference to RFC 7049.

-04

- o Added explanatory comments on design decisions made that were discussed on the mailing list that Jim Schaad requested be added to the draft.

-03

- o Addressed review of -02 by Jim Schaad.

-02

- o Addressed working group last call comments. Thanks to J.C. Jones, Kevin Jacobs, Jim Schaad, Neil Madden, and Benjamin Kaduk for their useful feedback.

-01

- o Changed the JOSE curve identifier from "P-256K" to "secp256k1".
- o Specified that secp256k1 signing is done using the SHA-256 hash function.

-00

- o Created the initial working group draft from draft-jones-cose-additional-algorithms-00, changing only the title, date, and history entry.

Author's Address

Michael B. Jones
Microsoft

Email: mbj@microsoft.com
URI: <http://self-issued.info/>

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 16 June 2021

J. Schaad
August Cellars
13 December 2020

CBOR Object Signing and Encryption (COSE): Header parameters for
carrying and referencing X.509 certificates
draft-ietf-cose-x509-08

Abstract

The CBOR Signing And Encrypted Message (COSE) structure uses references to keys in general. For some algorithms, additional properties are defined which carry parameters relating to keys as needed. The COSE Key structure is used for transporting keys outside of COSE messages. This document extends the way that keys can be identified and transported by providing attributes that refer to or contain X.509 certificates.

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>. Instructions are on that page as well. Editorial changes can be managed in GitHub, but any substantial issues need to be discussed on the COSE mailing list.

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 16 June 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Requirements Terminology	3
2. X.509 COSE Header Parameters	3
3. X.509 certificates and static-static ECDH	7
4. IANA Considerations	8
4.1. COSE Header Parameter Registry	8
4.2. COSE Header Algorithm Parameter Registry	8
5. Security Considerations	9
6. References	9
6.1. Normative References	9
6.2. Informative References	10
Author's Address	11

1. Introduction

In the process of writing [RFC8152], the working group discussed X.509 certificates [RFC5280] and decided that no use cases were presented that showed a need to support certificates. Since that time, a number of cases have been defined in which X.509 certificate support is necessary, and by implication, applications will need a documented and consistent way to handle such certificates. This document defines a set of attributes that will allow applications to transport and refer to X.509 certificates in a consistent manner.

In some of these cases, a constrained device is being deployed in the context of an existing X.509 PKI: for example, in the 6TiSCH environment, [I-D.richardson-enrollment-roadmap] describes a device enrollment solution that relies on the presence of a factory-installed certificate on the device. The [I-D.ietf-lake-edhoc] draft was also written with the idea that long term certificates could be used to provide for authentication of devices, and uses them to establish session keys. Another possible scenario is the use of COSE

as the basis for a secure messaging application. This scenario assumes the presence of long term keys and a central authentication authority. Basing such an application on public key certificates allows it to make use of well established key management disciplines.

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.

2. X.509 COSE Header Parameters

The use of X.509 certificates allows for an existing trust infrastructure to be used with COSE. This includes the full suite of enrollment protocols, trust anchors, trust chaining and revocation checking that have been defined over time by the IETF and other organizations. The key structures that have been defined in COSE currently do not support all of these properties although some may be found in COSE Web Tokens (CWT) [RFC8392].

It is not necessarily expected that constrained devices themselves will evaluate and process X.509 certificates: it is perfectly reasonable for a constrained device to be provisioned with a certificate that it subsequently provides to a relying party - along with a signature or encrypted message - on the assumption that the relying party is not a constrained device, and is capable of performing the required certificate evaluation and processing. It is also reasonable that a constrained device would have the hash of a certificate associated with a public key and be configured to use a public key for that thumbprint, but without performing the certificate evaluation or even having the entire certificate. In any case, there still needs to be an entity that is responsible for handling the possible certificate revocation.

Parties that intend to rely on the assertions made by a certificate obtained from any of these methods still need to validate it. This validation can be done according to the PKIX rules in [RFC5280] or by using a different trust structure, such as a trusted certificate distributor for self-signed certificates. The PKIX validation includes matching against the trust anchors configured for the application. These rules apply when the validation succeeds in a single step as well as when certificate chains need to be built. If the application cannot establish trust in the certificate, the public key contained in the certificate cannot be used for cryptographic operations.

The header parameters defined in this document are:

x5bag: This header parameter contains a bag of X.509 certificates. The set of certificates in this header parameter is unordered and may contain self-signed certificates. Note that there could be duplicating certificates. The certificate bag can contain certificates which are completely extraneous to the message. (An example of this would be where a signed message is being used to transport a certificate containing a key agreement key.) As the certificates are unordered, the party evaluating the signature will need to be capable of building the certificate path as necessary. That party will also have to take into account that the bag may not contain the full set of certificates needed to build any particular chain.

The trust mechanism MUST process any certificates in this parameter as untrusted input. The presence of a self-signed certificate in the parameter MUST NOT cause the update of the set of trust anchors without some out-of-band confirmation. As the contents of this header parameter are untrusted input, the header parameter can be in either the protected or unprotected header bucket.

This header parameter allows for a single X.509 certificate or a bag of X.509 certificates to be carried in the message.

- * If a single certificate is conveyed, it is placed in a CBOR byte string.
- * If multiple certificates are conveyed, a CBOR array of byte strings is used, with each certificate being in its own byte string.

x5chain: This header parameter contains an ordered array of X.509 certificates. The certificates are to be ordered starting with the certificate containing the end-entity key followed by the certificate which signed it and so on. There is no requirement for the entire chain to be present in the element if there is reason to believe that the relying party already has, or can locate the missing certificates. This means that the relying party is still required to do path building, but that a candidate path is proposed in this header parameter.

The trust mechanism MUST process any certificates in this parameter as untrusted input. The presence of a self-signed certificate in the parameter MUST NOT cause the update of the set of trust anchors without some out-of-band confirmation. As the contents of this header parameter are untrusted input, the header parameter can be in either the protected or unprotected header bucket.

This header parameter allows for a single X.509 certificate or a chain of X.509 certificates to be carried in the message.

- * If a single certificate is conveyed, it is placed in a CBOR byte string.
- * If multiple certificates are conveyed, a CBOR array of byte strings is used, with each certificate being in its own byte string.

x5t: This header parameter provides the ability to identify an X.509 certificate by a hash value (a thumbprint). The 'x5t' header parameter can be represented as an array of two elements. The first element is an algorithm identifier which is an integer or a string containing the hash algorithm identifier corresponding to either the Value (integer) or Name (string) column of the algorithm registered in the "COSE Algorithms" registry. The second element is a binary string containing the hash value computed over the DER encoded certificate.

As this header parameter does not provide any trust, the header parameter can be in either a protected or unprotected header bucket.

For interoperability, applications which use this header parameter MUST support the hash algorithm 'SHA-256', but can use other hash algorithms. This requirement allows for different implementations to be configured to use an interoperable algorithm, but does not preclude the use (by prior agreement) of other algorithms.

RFC Editor please remove the following two paragraphs:

During AD review, a question was raised about how effective the previous statement is in terms of dealing with a MTI algorithm. There needs to be some type of arrangement between the parties to agree that a specific hash algorithm is going to be used in computing the thumbprint. Making it a MUST use would make that true, but it then means that agility is going to be very difficult.

The worry is that while SHA-256 may be mandatory, if a sender supports SHA-256 but only sends SHA-512 then the recipient which only does SHA-256 would not be able to use the thumbprint. In that case both applications would conform to the specification, but still not be able to inter-operate.

x5u: This header parameter provides the ability to identify an X.509 certificate by a URI [RFC3986]. It contains a CBOR text string. The referenced resource can be any of the following media types:

- * application/pkix-cert [RFC2585]
- * application/pkcs7-mime; smime-type="certs-only" [RFC8551]

As this header parameter implies a trust relationship between the party generating the x5u parameter and the party hosting the referred-to resource, this header parameter MUST be in the protected attribute bucket.

The URI provided MUST provide integrity protection and server authentication. For example, an HTTP or CoAP GET request to retrieve a certificate MUST use TLS [RFC8446] or DTLS [I-D.ietf-tls-dtls13]. If the retrieved certificate does not chain to an existing trust anchor, the certificate MUST NOT be trusted unless the server is configured as trusted to provide new trust anchors or if an out-of-band confirmation can be received for trusting the retrieved certificate.

The header parameters are used in the following locations:

- * COSE_Signature and COSE_Sign1 objects: in these objects they identify the certificate to be used for validating the signature.
- * COSE_recipient objects: in this location they identify the certificate for the recipient of the message.

The labels assigned to each header parameter can be found in the following table.

Name	Label	Value Type	Description
x5bag	TBD4	COSE_X509	An unordered bag of X.509 certificates
x5chain	TBD3	COSE_X509	An ordered chain of X.509 certificates
x5t	TBD1	COSE_CertHash	Hash of an X.509 certificate
x5u	TBD2	uri	URI pointing to an X.509 certificate

Table 1: X.509 COSE Header Parameters

Below is an equivalent CDDL [RFC8610] description of the text above.

```
COSE_X509 = bstr / [ 2*certs: bstr ]
COSE_CertHash = [ hashAlg: (int / tstr), hashValue: bstr ]
```

The content of the bstr are the bytes of a DER encoded certificate.

3. X.509 certificates and static-static ECDH

The header parameters defined in the previous section are used to identify the recipient certificates for the ECDH key agreement algorithms. In this section we define the algorithm specific parameters that are used for identifying or transporting the sender's key for static-static key agreement algorithms.

These attributes are defined analogously to those in the previous section. There is no definition for the certificate bag, as the same attribute would be used for both the sender and recipient certificates.

x5chain-sender: This header parameter contains the chain of certificates starting with the sender's key exchange certificate. The structure is the same as 'x5chain'.

x5t-sender: This header parameter contains the hash value for the sender's key exchange certificate. The structure is the same as 'x5t'.

x5u-sender: This header parameter contains a URI for the sender's

key exchange certificate. The structure and processing are the same as 'x5u'.

Name	Label	Type	Algorithm	Description
x5t-sender	TBD	COSE_CertHash	ECDH-SS+HKDF-256, ECDH-SS+HKDF-512, ECDH-SS+A128KW, ECDH-SS+A192KW, ECDH-SS+A256KW	Thumbprint for the senders X.509 certificate
x5u-sender	TBD	uri	ECDH-SS+HKDF-256, ECDH-SS+HKDF-512, ECDH-SS+A128KW, ECDH-SS+A192KW, ECDH-SS+A256KW	URI for the senders X.509 certificate
x5chain-sender	TBD	COSE_X509	ECDH-SS+HKDF-256, ECDH-SS+HKDF-512, ECDH-SS+A128KW, ECDH-SS+A192KW, ECDH-SS+A256KW	static key X.509 certificate chain

Table 2: Static ECDH Algorithm Values

4. IANA Considerations

4.1. COSE Header Parameter Registry

IANA is requested to register the new COSE Header parameters in Table 1 in the "COSE Header Parameters" registry. The "Value Registry" field is empty for all of the items. For each item, the 'Reference' field points to this document.

4.2. COSE Header Algorithm Parameter Registry

IANA is requested to register the new COSE Header Algorithm parameters in Table 2 in the "COSE Header Algorithm Parameters" registry. For each item, the 'Reference' field points to this document.

5. Security Considerations

Establishing trust in a certificate is a vital part of processing. A major component of establishing trust is determining what the set of trust anchors are for the process. A new self-signed certificate appearing on the client cannot be a trigger to modify the set of trust anchors, because a well defined trust-establishment process is required. One common way for a new trust anchor to be added (or removed) from a device is by doing a new firmware upgrade.

In constrained systems, there is a trade-off between the order of checking the signature and checking the certificate for validity. Validating certificates can require that network resources be accessed in order to get revocation information or retrieve certificates during path building. The resulting network access can consume power and network bandwidth. On the other hand, if the certificates are validated after the signature is validated, an oracle can potentially be built based on detecting the network resources which is only done if the signature validation passes. In any event, both the signature and certificate validation **MUST** be completed successfully before acting on any requests.

Before using the key in a certificate, the key **MUST** be checked against the algorithm to be used and any algorithm specific checks need to be made. These checks can include validating that points are on curves for elliptical curve algorithms, and that sizes of RSA keys are of an acceptable size. The use of unvalidated keys can lead either to loss of security or excessive consumption of resources (for example using a 200K RSA key).

When processing x5u header parameter the security considerations of [RFC3986] and specifically those defined in Section 7.1 also apply.

Regardless of the source, certification path validation is an important part of establishing trust in a certificate. Section 6 of [RFC5280] provides guidance for the path validation. The security considerations of [RFC5280] are also important for the correct usage of this document.

The security of the algorithm used for 'x5t' does not affect the security of the system as this header parameter selects which certificate that is already present on the system should be used, but it does not provide any trust.

6. References

6.1. 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>>.
- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <<https://www.rfc-editor.org/info/rfc5280>>.
- [RFC8152] Schaad, J., "CBOR Object Signing and Encryption (COSE)", RFC 8152, DOI 10.17487/RFC8152, July 2017, <<https://www.rfc-editor.org/info/rfc8152>>.
- [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>>.

6.2. Informative References

- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.
- [I-D.ietf-tls-dtls13]
Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", Work in Progress, Internet-Draft, draft-ietf-tls-dtls13-39, 2 November 2020, <<https://tools.ietf.org/html/draft-ietf-tls-dtls13-39>>.
- [RFC8551] Schaad, J., Ramsdell, B., and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Message Specification", RFC 8551, DOI 10.17487/RFC8551, April 2019, <<https://www.rfc-editor.org/info/rfc8551>>.
- [RFC2585] Housley, R. and P. Hoffman, "Internet X.509 Public Key Infrastructure Operational Protocols: FTP and HTTP", RFC 2585, DOI 10.17487/RFC2585, May 1999, <<https://www.rfc-editor.org/info/rfc2585>>.
- [I-D.ietf-lake-edhoc]
Selandier, G., Mattsson, J., and F. Palombini, "Ephemeral Diffie-Hellman Over COSE (EDHOC)", Work in Progress, Internet-Draft, draft-ietf-lake-edhoc-02, 2 November 2020, <<https://tools.ietf.org/html/draft-ietf-lake-edhoc-02>>.

- [RFC8392] Jones, M., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "CBOR Web Token (CWT)", RFC 8392, DOI 10.17487/RFC8392, May 2018, <<https://www.rfc-editor.org/info/rfc8392>>.
- [RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <<https://www.rfc-editor.org/info/rfc8610>>.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <<https://www.rfc-editor.org/info/rfc3986>>.
- [I-D.richardson-enrollment-roadmap]
Richardson, M., "Device Enrollment in IETF protocols -- A Roadmap", Work in Progress, Internet-Draft, draft-richardson-enrollment-roadmap-03, 7 October 2020, <<https://tools.ietf.org/html/draft-richardson-enrollment-roadmap-03>>.

Author's Address

Jim Schaad
August Cellars

Email: ietf@augustcellars.com