ACE Working Group
Internet-Draft
Intended status: Informational
Expires: November 21, 2016

E. Wahlstroem
Nexus Technology
M. Jones
Microsoft
H. Tschofenig
ARM Ltd.

May 20, 2016

CBOR Web Token (CWT)
draft-ietf-ace-cbor-web-token-00

Abstract

CBOR Web Token (CWT) is a compact means of representing claims to be transferred between two parties. CWT is a profile of the JSON Web Token (JWT) that is optimized for constrained devices. The claims in a CWT are encoded in the Concise Binary Object Representation (CBOR) and CBOR Object Signing and Encryption (COSE) is used for added application layer security protection. A claim is a piece of information asserted about a subject and is represented as a name/value pair consisting of a claim name and a claim value.

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 http://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 November 21, 2016.

Table of Contents

1. Introduction

The JSON Web Token (JWT) [5] is a standardized security token format that has found use in OAuth 2.0 and OpenID Connect deployments, among other applications. JWT uses JSON Web Signatures (JWS) [3] and JSON Web Encryption (JWE) [4] to secure the contents of the JWT, which is a set of claims represented in JSON [5]. The use of JSON for encoding information is popular for Web and native applications, but it is considered inefficient for some Internet of Things (IoT) systems that use low power radio technologies.

In this document an alternative encoding of claims is defined. Instead of using JSON, as provided by JWTs, this specification uses CBOR [6] and calls this new structure "CBOR Web Token (CWT)", which is a compact means of representing secured claims to be transferred

between two parties. CWT is closely related to JWT. It references the JWT claims and both its name and pronunciation are derived from JWT. To protect the claims contained in CWTs, the CBOR Object Signing and Encryption (COSE) [7] specification is used.

The suggested pronunciation of CWT is the same as the English word "cot".

2. 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 "Key words for use in RFCs to Indicate Requirement Levels" [1].

This document reuses terminology from JWT [5] and COSE [7].

Type3StringOrURI:

> The "Type3StringOrURI" term has the same meaning, syntax, and processing rules as the "StringOrUri" term defined in Section 2 of JWT [5], except that Type3StringOrURI uses CBOR major type 3 instead of a JSON string value.
>
> FIXME: Use tag 32 for URIs?

Type6NumericDate:

> The "Type6NumericDate" term has the same meaning, syntax, and processing rules as the "NumericDate" term defined in Section 2 of JWT [5], except that Type6NumericDate uses CBOR major type 6, with tag value 1, instead of a numeric JSON value.

CBOR encoded claim key:

> The key used to identify a claim value.

3. Claims

The set of claims that a CWT must contain to be considered valid is context dependent and is outside the scope of this specification. Specific applications of CWTs will require implementations to understand and process some claims in particular ways. However, in the absence of such requirements, all claims that are not understood by implementations MUST be ignored.

To keep CWTs as small as possible, the CBOR encoded claim keys are represented using CBOR major type 0. Section 4 summaries all keys used to identity the claims defined in this document.

3.1. Claim Names

None of the claims defined below are intended to be mandatory to use or implement. They rather provide a starting point for a set of useful, interoperable claims. Applications using CWTs should define which specific claims they use and when they are required or optional.

3.1.1. iss (Issuer) Claim

The "iss" (issuer) claim has the same meaning, syntax, and processing rules as the "iss" claim defined in Section 4.1.1 of JWT [5], except that the format MUST be a Type3StringOrURI. The CBOR encoded claim key 1 MUST be used to identify this claim.

3.1.2. sub (Subject) Claim

The "sub" (subject) claim has the same meaning, syntax, and processing rules as the "sub" claim defined in Section 4.1.2 of JWT [5], except that the format MUST be a Type3StringOrURI. The CBOR encoded claim key 2 MUST be used to identify this claim.

3.1.3. aud (Audience) Claim

The "aud" (audience) claim has the same meaning, syntax, and processing rules as the "aud" claim defined in Section 4.1.3 of JWT [5], except that the format MUST be a Type3StringOrURI. The CBOR encoded claim key 3 MUST be used to identify this claim.

3.1.4. exp (Expiration Time) Claim

The "exp" (expiration time) claim has the same meaning, syntax, and processing rules as the "exp" claim defined in Section 4.1.4 of JWT [5], except that the format MUST be a Type6NumericDate. The CBOR encoded claim key 4 MUST be used to identify this claim.

3.1.5. nbf (Not Before) Claim

The "nbf" (not before) claim has the same meaning, syntax, and processing rules as the "nbf" claim defined in Section 4.1.5 of JWT [5], except that the format MUST be a Type6NumericDate. The CBOR encoded claim key 5 MUST be used to identify this claim.

3.1.6. iat (Issued At) Claim

The "iat" (issued at) claim has the same meaning, syntax, and processing rules as the "iat" claim defined in Section 4.1.6 of JWT [5], except that the format MUST be a Type6NumericDate. The CBOR encoded claim key 6 MUST be used to identify this claim.

3.1.7. cti (CWT ID) Claim

The "cti" (CWT ID) claim has the same meaning, syntax, and processing rules as the "jti" claim defined in Section 4.1.7 of JWT [5], except that the format MUST be of major type 3 with a case-sensitive string value. The CBOR encoded claim key 7 MUST be used to identify this claim.

4. Summary of the values, CBOR major types and encoded claim keys

Claim	CBOR encoded claim key	CBOR major type of value
iss	1	3
sub	2	3
aud	3	3
exp	4	6 tag value 1
nbf	5	6 tag value 1
iat	6	6 tag value 1
cti	7	3

Figure 1: Summary of the values, CBOR major types and encoded claim keys.

Note: Claims defined by the OpenID Foundation have not yet been included in the table above.

5. Security Considerations

The security of the CWT is dependent on the protection offered by COSE. Without protecting the claims contained in a CWT an adversary is able to modify, add or remove claims. Since the claims conveyed in a CWT are used to make authorization decisions it is not only important to protect the CWT in transit but also to ensure that the recipient is able to authenticate the party that collected the claims and created the CWT. Without trust of the recipient in the party that created the CWT no sensible authorization decision can be made. Furthermore, the creator of the CWT needs to carefully evaluate each claim value prior to including it in the CWT so that the recipient can be assured about the correctness of the provided information.

6. IANA Considerations

This section will create a registry for CWT claims, possibly relating them to the JWT Claims Registry.

7. Normative References

[1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[2] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <http://www.rfc-editor.org/info/rfc7159>.

[3] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <http://www.rfc-editor.org/info/rfc7515>.

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

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

[6] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <http://www.rfc-editor.org/info/rfc7049>.

[7] Schaad, J., "CBOR Encoded Message Syntax", draft-ietf-cose-msg-12 (work in progress), May 2016.

[8] Seitz, L., Selander, G., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "Authorization for the Internet of Things using OAuth 2.0", draft-seitz-ace-oauth-authz-00 (work in progress), October 2015.

Appendix A. Examples

Three examples of CWTs follow.

A.1. CWT with "aud" and symmetric key

A CWT used in the context of ACE requires at least the "aud" and a "cks" claim (defined elsewhere). This means that "iss", "alg", "key_ops" and others are pre-established and assumed. This would look like this non-normative JSON.

```
{
  "aud":"coap://light.example.com",
  "cks":
    [                       // COSE_Key is a CBOR map with an array of keys
      {
        "kty":4,            // symmetric key is indicated using kty 4
        "k": "loremipsum"   // the symmetric key
      }
    ]
}
```

Figure 2: "aud" claim and symmetric key in non-normative JSON

Using the CBOR encoded claim keys according to Section 4 and COSE [7] makes a CWT with "aud" and a symmetric key look like this in CBOR diagnostic notation:

```
{
  3: "coap://light.example.com",
  8:
  [
    {
      1: 4,
      -1: "loremipsum"
    }
  ]
}
```

Figure 3: CWT in CBOR diagnostic notation

Defined in CBOR.

```
a2                                      # map(2)
   03                                   # unsigned(3)
   78 18                                # text(24)
      636f61703a2f2f6c696768742e6578616d706c652e636f6d # "coap://
light.example.com"
   08                                   # unsigned(8)
   81                                   # array(1)
      a2                                # map(2)
         01                             # unsigned(1)
         04                             # unsigned(4)
         20                             # negative(0)
         6a                             # text(10)
            6c6f72656d697073756d        # "loremipsum"
```

Figure 4: CWT with "aud" and symmetric key in CBOR

Size of the CWT with a symmetric key of 10 bytes is 45 bytes. This is then packaged signed and encrypted using COSE.

A.2. CWT with "aud" and EC key

Token with "aud" set to "coap://light.example.com" and an EC key with "kid" set to "11".

```
{
  "aud": "coap://light.example.com",
  "cks":
    [                       // COSE_Key is a CBOR map with an array of keys
      {
        "kty": "EC",
        "kid": "11",
        "crv": 1, // using P-384
        "x":
h'bac5b11cad8f99f9c72b05cf4b9e26d244dc189f745228255a219a86d6a09eff',
        "y":
h'20138bf82dc1b6d562be0fa54ab7804a3a64b6d72ccfed6b6fb6ed28bbfc117e'
      }
    ]
}
```

Figure 5: "aud" claim and EC key in non-normative JSON

Using the CBOR encoded claim keys according to Section 4 and COSE [7] makes a CWT with "aud" and an EC key look like this in CBOR diagnostic notation:

```
{
  3: "coap://light.example.com",
  8:
  [
    {
      1: 2,
      2: "11",
      -1: 1,
      -2: h'bac5b11cad8f99f9c72b05cf4b9e26d244dc189f745228255a219a86d6a09eff',
      -3: h'20138bf82dc1b6d562be0fa54ab7804a3a64b6d72ccfed6b6fb6ed28bbfc117e'
    }
  ]
}
```

Figure 6: CWT with EC key in CBOR diagnostic notation

Defined in CBOR.

```
a2                                      # map(2)
   03                                   # unsigned(3)
   78 18                                # text(24)
      636f61703a2f2f6c696768742e6578616d706c652e636f6d # "coap://
light.example.com"
   08                                   # unsigned(8)
   81                                   # array(1)
      a5                                # map(5)
         01                             # unsigned(1)
         02                             # unsigned(2)
         02                             # unsigned(2)
         62                             # text(2)
            3131                        # "11"
         20                             # negative(0)
         01                             # unsigned(1)
         21                             # negative(1)
         58 20                          # bytes(32)
            bac5b11cad8f99f9c72b05cf4b9e26d244dc189f745228255a219a86d6a09eff #
"\xBA\xC5\xB1\x1C\xAD\x8F\x99\xF9\xC7+\x05\xCFK\x9E&\xD2D\xDC\x18\x9FtR(%Z!
\x9A\x86\xD6\xA0\x9E\xFF"
         22                             # negative(2)
         58 20                          # bytes(32)
            20138bf82dc1b6d562be0fa54ab7804a3a64b6d72ccfed6b6fb6ed28bbfc117e #
"\x13\x8B\xF8-
\xC1\xB6\xD5b\xBE\x0F\xA5J\xB7\x80J:d\xB6\xD7,\xCF\xEDko\xB6\xED(\xBB\xFC\x11~"
```

Figure 7: CWT with EC in CBOR

Size of the CWT with an EC key is 109 bytes. This is then packaged signed and encrypted using COSE.

A.3. Full CWT

CWT using all claims defined by this specification, plus extensions for AIF and an EC key.

```
{
   "iss": "coap://as.example.com",
   "aud": "coap://light.example.com",
   "sub": "erikw",
   "exp": 1444064944,
   "nbf": 1443944944,
   "iat": 1443944944,
   "cti": 2929,
   "cks":
     [                      // COSE_Key is a CBOR map with an array of keys
       {
         "kty": "EC",
         "kid": "11",
         "crv": 1, // using P-384
         "x":
h'bac5b11cad8f99f9c72b05cf4b9e26d244dc189f745228255a219a86d6a09eff',
         "y":
h'20138bf82dc1b6d562be0fa54ab7804a3a64b6d72ccfed6b6fb6ed28bbfc117e'
       }
     ],
   "aif": [["/s/light", 1], ["/a/led", 5], ["/dtls", 2]]
 }
```

Figure 8: All claims, "aif" and EC key in non-normative JSON

Using the CBOR encoded claim keys according to Section 4 and COSE [7] makes a full CWT look like this in CBOR diagnostic notation:

```
{
  1: "coap://as.example.com",
  3: "coap://light.example.com",
  2: "erikw",
  4: 1(1444064944),
  5: 1(1443944944),
  6: 1(1443944944),
  7: 2929,
  8: [
    {
      1: 2,
      2: "11",
      -1: 1,
      -2: h'bac5b11cad8f99f9c72b05cf4b9e26d244dc189f745228255a219a86d6a09eff',
      -3: h'20138bf82dc1b6d562be0fa54ab7804a3a64b6d72ccfed6b6fb6ed28bbfc117e'
    }
    ],
  9: [["/s/light", 1], ["/a/led", 5], ["/dtls", 2]]
}
```

Figure 9: Full CWT with EC key in CBOR diagnostic notation

Defined in CBOR.

```
a9                                      # map(9)
   01                                   # unsigned(1)
   75                                   # text(21)
      636f61703a2f2f61732e6578616d706c652e636f6d # "coap://as.example.com"
   03                                   # unsigned(3)
   78 18                                # text(24)
      636f61703a2f2f6c696768742e6578616d706c652e636f6d # "coap://
light.example.com"
   02                                   # unsigned(2)
   65                                   # text(5)
      6572696b77                        # "erikw"
   04                                   # unsigned(4)
   c1                                   # tag(1)
      1a 5612aeb0                       # unsigned(1444064944)
   05                                   # unsigned(5)
   c1                                   # tag(1)
      1a 5610d9f0                       # unsigned(1443944944)
   06                                   # unsigned(6)
   c1                                   # tag(1)
      1a 5610d9f0                       # unsigned(1443944944)
   07                                   # unsigned(7)
   19 0b71                              # unsigned(2929)
```

```
   08                                  # unsigned(8)
```

```
   81                                  # array(1)
      a5                               # map(5)
         01                            # unsigned(1)
         02                            # unsigned(2)
         02                            # unsigned(2)
         62                            # text(2)
            3131                       # "11"
         20                            # negative(0)
         01                            # unsigned(1)
         21                            # negative(1)
         58 20                         # bytes(32)
            bac5b11cad8f99f9c72b05cf4b9e26d244dc189f745228255a219a86d6a09eff #
"\xBA\xC5\xB1\x1C\xAD\x8F\x99\xF9\xC7+\x05\xCFK\x9E&\xD2D\xDC\x18\x9FtR(%Z!
\x9A\x86\xD6\xA0\x9E\xFF"
         22                            # negative(2)
         58 20                         # bytes(32)
            20138bf82dc1b6d562be0fa54ab7804a3a64b6d72ccfed6b6fb6ed28bbfc117e #
"\x13\x8B\xF8-
\xC1\xB6\xD5b\xBE\x0F\xA5J\xB7\x80J:d\xB6\xD7,\xCF\xEDko\xB6\xED(\xBB\xFC\x11~"
   09                                  # unsigned(9)
   83                                  # array(3)
      82                               # array(2)
         68                            # text(8)
            2f732f6c69676874           # "/s/light"
         01                            # unsigned(1)
      82                               # array(2)
         66                            # text(6)
            2f612f6c6564               # "/a/led"
         05                            # unsigned(5)
      82                               # array(2)
         65                            # text(5)
            2f64746c73                 # "/dtls"
         02                            # unsigned(2)
```

Figure 10: Full CWT with EC in CBOR

Size of the CWT with an EC key is 194 bytes. This is then packaged signed and encrypted using COSE.

Appendix B. Acknowledgements

A straw man proposal of CWT was written in the draft "Authorization for the Internet of Things using OAuth 2.0" [8] with the help of Ludwig Seitz, Goeran Selander, and Samuel Erdtman.

Appendix C. Document History

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

-00

- Created the initial working group version based on draft-wahlstroem-ace-cbor-web-token-00.

Authors' Addresses

Erik Wahlstroem
Nexus Technology
Sweden

Email: erik.wahlstrom@nexusgroup.com
URI: https://www.nexusgroup.com

Michael B. Jones
Microsoft

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

Hannes Tschofenig
ARM Ltd.
Hall in Tirol 6060
Austria

Email: Hannes.Tschofenig@arm.com