anima Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 6, 2021

M. Richardson
Sandelman Software Works
P. van der Stok
vanderstok consultancy
P. Kampanakis
Cisco Systems

November 02, 2020

Constrained Voucher Artifacts for Bootstrapping Protocols
draft-ietf-anima-constrained-voucher-09

Abstract

This document defines a strategy to securely assign a pledge to an owner, using an artifact signed, directly or indirectly, by the pledge's manufacturer. This artifact is known as a "voucher".

This document builds upon the work in [RFC8366], encoding the resulting artifact in CBOR. Use with two signature technologies are described.

Additionally, this document explains how constrained vouchers may be transported as an extension to the [I-D.ietf-ace-coap-est] protocol.

Table of Contents

1. Introduction

Enrollment of new nodes into constrained networks with constrained nodes present unique challenges.

There are bandwidth and code space issues to contend. A solution such as [I-D.ietf-anima-bootstrapping-keyinfra] may be too large in terms of code space or bandwidth required.

This document defines a constrained version of [RFC8366]. Rather than serializing the YANG definition in JSON, it is serialized into CBOR ([RFC7049]).

This document follows a similar, but not identical structure as [RFC8366] and supplements the brski part to [I-D.ietf-ace-coap-est].

There are three constrained situations described in this document: 1. CMS signed CBOR encoded vouchers transported using CoAP, protected by DTLS (coaps). 2. COSE signed CBOR encoded vouchers transported using CoAP, protected by EDHOC or DTLS. 3. COSE signed CBOR encoded vouchers, integrated into the key exchange as described by [I-D.selander-ace-ake-authz]

Additional sections have been added concerning:

1. Addition of voucher-request specification as defined in [I-D.ietf-anima-bootstrapping-keyinfra],

2. Addition to [I-D.ietf-ace-coap-est] of voucher transport requests over CoAP.

The CBOR definitions for this constrained voucher format are defined using the mechanism describe in [I-D.ietf-core-yang-cbor] using the SID mechanism explained in [I-D.ietf-core-sid]. As the tooling to convert YANG documents into an list of SID keys is still in its infancy, the table of SID values presented here should be considered normative rather than the output of the pyang tool.

Two methods of signing the resulting CBOR object are described in this document:

1. One is CMS [RFC5652].

2. The other is COSE_Sign1 [RFC8152] objects.

2. Terminology

The following terms are defined in [RFC8366], and are used identically as in that document: artifact, imprint, domain, Join Registrar/Coordinator (JRC), Manufacturer Authorized Signing Authority (MASA), pledge, Trust of First Use (TOFU), and Voucher.

3. Requirements Language

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.

4. Survey of Voucher Types

[RFC8366] provides for vouchers that assert proximity, that authenticate the registrar and that include different amounts of anti-replay protection.

This document does not make any extensions to the types of vouchers.

Time based vouchers are included in this definition, but given that constrained devices are extremely unlikely to have accurate time, their use is very unlikely. Most users of these constrained vouchers will be online and will use live nonces to provide anti-replay protection.

[RFC8366] defined only the voucher artifact, and not the Voucher Request artifact, which was defined in [I-D.ietf-anima-bootstrapping-keyinfra].

This document defines both a constrained voucher and a constrained voucher-request. They are presented in the order "voucher-request", followed by a "voucher" response as this is the time order that they occur.

This document defines both CMS-signed voucher requests and responses, and COSE signed voucher requests and responses. The use of CMS signatures implies the use of PKIX format certificates. The pinned-domain-cert present in a voucher, is the certificate of the Registrar.

The constrained voucher and constrained voucher request MUST be signed.

The use of the two signing formats permit the use of both PKIX format certificates, and raw public keys (RPK).

When RPKs are used, the voucher produced by the MASA pins the raw public key of the Registrar: the pinned-domain-subject-public-key-info in a voucher, is the raw public key of the Registrar. This is described in the YANG definition for the constrained voucher.

5. Discovery and URI

This section describes the BRSKI extensions to EST-coaps [I-D.ietf-ace-coap-est] to transport the voucher between registrar, proxy and pledge over CoAP. The extensions are targeted to low-resource networks with small packets. Saving header space is important and the EST-coaps URI is shorter than the EST URI.

The presence and location of (path to) the management data are discovered by sending a GET request to "/.well-known/core" including a resource type (RT) parameter with the value "ace.est" [RFC6690]. Upon success, the return payload will contain the root resource of the EST resources. It is up to the implementation to choose its root resource; throughout this document the example root resource /est is used.

The EST-coaps server URIs differ from the EST URI by replacing the scheme https by coaps and by specifying shorter resource path names:

coaps://www.example.com/est/short-name

Figure 5 in section 3.2.2 of [RFC7030] enumerates the operations and corresponding paths which are supported by EST. Table 1 provides the mapping from the BRSKI extension URI path to the EST-coaps URI path.

BRSKI	EST-coaps
/requestvoucher	/rv
/voucher_status	/vs
/enrollstatus	/es
/requestauditlog	/ra

Table 1: BRSKI path to EST-coaps path

/requestvoucher, /voucher_status and /enrollstatus are needed between pledge and Registrar.

When discovering the root path for the EST resources, the server MAY return the full resource paths and the used content types. This is useful when multiple content types are specified for EST-coaps server. For example, the following more complete response is possible.

[EDNOTE: spell out where voucher artifacts are used in BRSKI flows since the APIs]

[EDNOTE: The /requestauditlog and /voucher-status are exchanged by the Registrar and MASA. The JRC will likely talk to MASA over a normal (not constrained) medium. Do we need /ra and /vs? Do we need to remove them from the example too? Also what happens to the voucher-request and response in this case? Is MASA supposed to support contrained vouchers?]

```
REQ: GET /.well-known/core?rt=brski*

RES: 2.05 Content
</b>; rt="brski"
</b/rv>; rt="brski.rv";ct=TBD2 TBD3
</b/vs>; rt="brski.vs";ct=50 60
</b/es>; rt="brski.es";ct=50 60
```

The return of the content-types allows the client to choose the most appropriate one from multiple content types.

ct=TBD2 stands for Content-Format "application/voucher-cms+cbor, and ct=TBD3 stands for Content-Format "application/voucher-cose+cbor".

Content-Formats TBD2 and TBD3 are defined in this document.

The Content-Format ("application/json") 50 MAY be supported. Content-Formats ("application/cbor") 60, TBD2, and TBD3 MUST be supported by the Registrar.

The Pledge and MASA need to support one or more formats for the voucher. The MASA needds to support whatever formats that the pledge's produced by that manufacturer supports.

6. Artifacts

This section describes the abstract (tree) definition as explained in [I-D.ietf-netmod-yang-tree-diagrams] first. This provides a high-level view of the contents of each artifact.

Then the assigned SID values are presented. These have been assigned using the rules in [I-D.ietf-core-sid], with an allocation that was made via the http://comi.space service.

6.1. Voucher Request artifact

6.1.1. Tree Diagram

The following diagram is largely a duplicate of the contents of [RFC8366], with the addition of proximity-registrar-subject-public-key-info, proximity-registrar-cert, and prior-signed-voucher-request.

prior-signed-voucher-request is only used between the Registrar and the MASA. proximity-registrar-subject-public-key-info replaces proximity-registrar-cert for the extremely constrained cases.

```
   module: ietf-constrained-voucher-request

     grouping voucher-request-constrained-grouping
       +-- voucher
          +-- created-on?
          |       yang:date-and-time
          +-- expires-on?
          |       yang:date-and-time
          +-- assertion
          |       enumeration
          +-- serial-number
          |       string
          +-- idevid-issuer?
          |       binary
          +-- pinned-domain-cert?
          |       binary
          +-- domain-cert-revocation-checks?
          |       boolean
          +-- nonce?
          |       binary
          +-- last-renewal-date?
          |       yang:date-and-time
          +-- proximity-registrar-subject-public-key-info?
          |       binary
          +-- proximity-registrar-sha256-of-subject-public-key-info?
          |       binary
          +-- proximity-registrar-cert?
          |       binary
          +-- prior-signed-voucher-request?
                  binary
```

6.1.2. SID values

```
       SID Assigned to
--------- -------------------------------------------------
     2501 data /ietf-constrained-voucher-request:voucher
     2502 data .../assertion
     2503 data .../created-on
     2504 data .../domain-cert-revocation-checks
     2505 data .../expires-on
     2506 data .../idevid-issuer
     2507 data .../last-renewal-date
     2508 data /ietf-constrained-voucher-request:voucher/nonce
     2509 data .../pinned-domain-cert
     2510 data .../prior-signed-voucher-request
     2511 data .../proximity-registrar-cert
     2512 data mity-registrar-sha256-of-subject-public-key-info
     2513 data .../proximity-registrar-subject-public-key-info
     2514 data .../serial-number

  WARNING, obsolete definitions
```

6.1.3. YANG Module

In the constrained-voucher-request YANG module, the voucher is "augmented" within the "used" grouping statement such that one continuous set of SID values is generated for the constrained-voucher-request module name, all voucher attributes, and the constrained-voucher-request attribute. Two attributes of the voucher are "refined" to be optional.

```
<CODE BEGINS> file "ietf-constrained-voucher-request@2019-09-01.yang"
module ietf-constrained-voucher-request {
  yang-version 1.1;

  namespace
    "urn:ietf:params:xml:ns:yang:ietf-constrained-voucher-request";
  prefix "constrained";

  import ietf-restconf {
    prefix rc;
    description
      "This import statement is only present to access
       the yang-data extension defined in RFC 8040.";
    reference "RFC 8040: RESTCONF Protocol";
  }

  import ietf-voucher {
    prefix "v";
  }
```

```
    organization
     "IETF ANIMA Working Group";

    contact
     "WG Web:   <http://tools.ietf.org/wg/anima/>
      WG List:  <mailto:anima@ietf.org>
      Author:   Michael Richardson
                <mailto:mcr+ietf@sandelman.ca>
      Author:   Peter van der Stok
                <mailto: consultancy@vanderstok.org>
      Author:   Panos Kampanakis
                <mailto: pkampana@cisco.com>";
    description
     "This module defines the format for a voucher request,
      which is produced by a pledge to request a voucher.
      The voucher-request is sent to the potential owner's
      Registrar, which in turn sends the voucher request to
      the manufacturer or delegate (MASA).

      A voucher is then returned to the pledge, binding the
      pledge to the owner.  This is a constrained version of the
      voucher-request present in
      draft-ietf-anima-bootstrap-keyinfra.txt.

      This version provides a very restricted subset appropriate
      for very constrained devices.
      In particular, it assumes that nonce-ful operation is
      always required, that expiration dates are rather weak, as no
      clocks can be assumed, and that the Registrar is identified
      by a pinned Raw Public Key.

      The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
      'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'MAY',
      and 'OPTIONAL' in the module text are to be interpreted as
      described in RFC 2119.";

    revision "2019-09-01" {
      description
       "Initial version";
      reference
       "RFC XXXX: Voucher Profile for Constrained Devices";
    }

    rc:yang-data voucher-request-constrained-artifact {
      // YANG data template for a voucher.
      uses voucher-request-constrained-grouping;
    }
```

```
    // Grouping defined for future usage
    grouping voucher-request-constrained-grouping {
      description
        "Grouping to allow reuse/extensions in future work.";

      uses v:voucher-artifact-grouping {

        refine voucher/created-on {
            mandatory  false;
        }

        refine voucher/pinned-domain-cert {
            mandatory  false;
        }


        augment "voucher" {
          description "Base the constrained voucher-request upon the
            regular one";

          leaf proximity-registrar-subject-public-key-info {
            type binary;
            description
              "The proximity-registrar-subject-public-key-info replaces
               the proximit-registrar-cert in constrained uses of
               the voucher-request.
               The proximity-registrar-subject-public-key-info is the
               Raw Public Key of the Registrar. This field is encoded
               as specified in RFC7250, section 3.
               The ECDSA algorithm MUST be supported.
               The EdDSA algorithm as specified in
               draft-ietf-tls-rfc4492bis-17 SHOULD be supported.
               Support for the DSA algorithm is not recommended.
               Support for the RSA algorithm is MAY, but due to
               size is discouraged.";
          }

          leaf proximity-registrar-sha256-of-subject-public-key-info {
            type binary;
            description
              "The proximity-registrar-sha256-of-subject-public-key-info
               is an alternative to
               proximity-registrar-subject-public-key-info.
               and pinned-domain-cert.  In many cases the
               public key of the domain has already been transmitted
               during the key agreement protocol, and it is wasteful
               to transmit the public key another two times.
               The use of a hash of public key info, at 32-bytes for
```

```
            sha256 is a significant savings compared to an RSA
            public key, but is only a minor savings compared to
            a 256-bit ECDSA public-key.
            Algorithm agility is provided by extensions to this
            specifications which define new leaf for other hash
            types.";
          }

          leaf proximity-registrar-cert {
            type binary;
            description
              "An X.509 v3 certificate structure as specified by
               RFC 5280,
               Section 4 encoded using the ASN.1 distinguished encoding
               rules (DER), as specified in ITU-T X.690.

               The first certificate in the Registrar TLS server
               certificate_list sequence  (see [RFC5246]) presented by
               the Registrar to the Pledge. This MUST be populated in a
               Pledge's voucher request if the proximity assertion is
               populated.";
          }

          leaf prior-signed-voucher-request {
            type binary;
            description
              "If it is necessary to change a voucher, or re-sign and
               forward a voucher that was previously provided along a
               protocol path, then the previously signed voucher
               SHOULD be included in this field.

               For example, a pledge might sign a proximity voucher,
               which an intermediate registrar then re-signs to
               make its own proximity assertion.  This is a simple
               mechanism for a chain of trusted parties to change a
               voucher, while maintaining the prior signature
               information.

               The pledge MUST ignore all prior voucher information
               when accepting a voucher for imprinting. Other
               parties MAY examine the prior signed voucher
               information for the purposes of policy decisions.
               For example this information could be useful to a
               MASA to determine that both pledge and registrar
               agree on proximity assertions. The MASA SHOULD
               remove all prior-signed-voucher-request information when
               signing a voucher for imprinting so as to minimize the
               final voucher size.";
```

```
          }
        }
      }
    }
  }
  <CODE ENDS>
```

6.1.4. Example voucher request artifact

Below is a CBOR serialization of the constrained-voucher-request is shown in diagnostic CBOR notation. The enum value of the assertion field is calculated to be zero by following the algorithm described in section 9.6.4.2 of [RFC7950].

```
{
  2501: {
    +2 : "2016-10-07T19:31:42Z", / SID= 2503, created-on /
    +4 : "2016-10-21T19:31:42Z", / SID= 2505, expires-on /
    +1 : 2,                      / SID= 2502, assertion /
                                 /               "proximity" /
    +13: "JADA123456789",        / SID= 2514, serial-number /
    +5 : h'01020D0F',            / SID= 2506, idevid-issuer /
    +10: h'cert.der',            / SID=2511, proximity-registrar-cert/
    +3 : true,                   / SID= 2504, domain-cert
                                                -revocation-checks/
    +6 : "2017-10-07T19:31:42Z", / SID= 2507, last-renewal-date /
    +12: h'key_info'             / SID= 2513, proximity-registrar
                                          -subject-public-key-info /
  }
}
```

6.2. Voucher artifact

The voucher's primary purpose is to securely assign a pledge to an owner. The voucher informs the pledge which entity it should consider to be its owner.

This document defines a voucher that is a CBOR encoded instance of the YANG module defined in Section 5.3 that has been signed with CMS or with COSE.

6.2.1. Tree Diagram

The following diagram is largely a duplicate of the contents of [RFC8366], with only the addition of pinned-domain-subject-public-key-info.

```
module: ietf-constrained-voucher

  grouping voucher-constrained-grouping
    +-- voucher
       +-- created-on?
       |       yang:date-and-time
       +-- expires-on?
       |       yang:date-and-time
       +-- assertion                                  enumeration
       +-- serial-number                              string
       +-- idevid-issuer?                             binary
       +-- pinned-domain-cert?                        binary
       +-- domain-cert-revocation-checks?             boolean
       +-- nonce?                                     binary
       +-- last-renewal-date?
       |       yang:date-and-time
       +-- pinned-domain-subject-public-key-info?     binary
       +-- pinned-sha256-of-subject-public-key-info?  binary
```

6.2.2. SID values

```
      SID Assigned to
--------- -------------------------------------------------
     2451 data /ietf-constrained-voucher:voucher
     2452 data /ietf-constrained-voucher:voucher/assertion
     2453 data /ietf-constrained-voucher:voucher/created-on
     2454 data .../domain-cert-revocation-checks
     2455 data /ietf-constrained-voucher:voucher/expires-on
     2456 data /ietf-constrained-voucher:voucher/idevid-issuer
     2457 data .../last-renewal-date
     2458 data /ietf-constrained-voucher:voucher/nonce
     2459 data .../pinned-domain-cert
     2460 data .../pinned-domain-subject-public-key-info
     2461 data .../pinned-sha256-of-subject-public-key-info
     2462 data /ietf-constrained-voucher:voucher/serial-number

 WARNING, obsolete definitions
```

6.2.3. YANG Module

In the constrained-voucher YANG module, the voucher is "augmented" within the "used" grouping statement such that one continuous set of SID values is generated for the constrained-voucher module name, all voucher attributes, and the constrained-voucher attribute. Two attributes of the voucher are "refined" to be optional.

```
<CODE BEGINS> file "ietf-constrained-voucher@2019-09-01.yang"
module ietf-constrained-voucher {
  yang-version 1.1;

  namespace
    "urn:ietf:params:xml:ns:yang:ietf-constrained-voucher";
  prefix "constrained";

  import ietf-restconf {
    prefix rc;
    description
      "This import statement is only present to access
       the yang-data extension defined in RFC 8040.";
    reference "RFC 8040: RESTCONF Protocol";
  }

  import ietf-voucher {
    prefix "v";
  }

  organization
   "IETF ANIMA Working Group";

  contact
   "WG Web:   <http://tools.ietf.org/wg/anima/>
    WG List:  <mailto:anima@ietf.org>
    Author:   Michael Richardson
              <mailto:mcr+ietf@sandelman.ca>
    Author:   Peter van der Stok
              <mailto: consultancy@vanderstok.org>
    Author:   Panos Kampanakis
              <mailto: pkampana@cisco.com>";
description
  "This module defines the format for a voucher, which is produced
   by a pledge's manufacturer or delegate (MASA) to securely assign
   one or more pledges to an 'owner', so that the pledges may
   establis a secure connection to the owner's network
   infrastructure.

   This version provides a very restricted subset appropriate
```

```
   for very constrained devices.
   In particular, it assumes that nonce-ful operation is
   always required, that expiration dates are rather weak, as no
   clocks can be assumed, and that the Registrar is identified
   by a pinned Raw Public Key.

   The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
   'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'MAY',
   and 'OPTIONAL' in the module text are to be interpreted as
   described in RFC 2119.";

  revision "2019-09-01" {
    description
     "Initial version";
    reference
     "RFC XXXX: Voucher Profile for Constrained Devices";
  }

  rc:yang-data voucher-constrained-artifact {
    // YANG data template for a voucher.
    uses voucher-constrained-grouping;
  }

  // Grouping defined for future usage
  grouping voucher-constrained-grouping {
    description
      "Grouping to allow reuse/extensions in future work.";

    uses v:voucher-artifact-grouping {

      refine voucher/created-on {
          mandatory  false;
      }

      refine voucher/pinned-domain-cert {
          mandatory  false;
      }

      augment "voucher" {
        description "Base the constrained voucher
                                  upon the regular one";
        leaf pinned-domain-subject-public-key-info {
          type binary;
          description
            "The pinned-domain-subject-public-key-info replaces the
             pinned-domain-cert in constrained uses of
             the voucher. The pinned-domain-subject-public-key-info
             is the Raw Public Key of the Registrar.
```

```
              This field is encoded as specified in RFC7250,
              section 3.
              The ECDSA algorithm MUST be supported.
              The EdDSA algorithm as specified in
              draft-ietf-tls-rfc4492bis-17 SHOULD be supported.
              Support for the DSA algorithm is not recommended.
              Support for the RSA algorithm is a MAY.";
          }

          leaf pinned-sha256-of-subject-public-key-info {
            type binary;
            description
              "The pinned-hash-subject-public-key-info is a second
               alternative to pinned-domain-cert.  In many cases the
               public key of the domain has already been transmitted
               during the key agreement process, and it is wasteful
               to transmit the public key another two times.
               The use of a hash of public key info, at 32-bytes for
               sha256 is a significant savings compared to an RSA
               public key, but is only a minor savings compared to
               a 256-bit ECDSA public-key.
               Algorithm agility is provided by extensions to this
               specifications which define new leaf for other hash types";
          }
        }
      }
    }
  }
}
<CODE ENDS>
```

6.2.4. Example voucher artifacts

Below a the CBOR serialization of the constrained-voucher is shown in diagnostic CBOR notation. The enum value of the assertion field is calculated to be zero by following the algorithm described in section 9.6.4.2 of [RFC7950].

```
{
  2451: {
    +2 : "2016-10-07T19:31:42Z", / SID = 2453, created-on /
    +4 : "2016-10-21T19:31:42Z", / SID = 2455, expires-on /
    +1 : 0,                      / SID = 2452, assertion /
                                 /               "verified" /
    +11: "JADA123456789",        / SID = 2462, serial-number /
    +5 : h'01020D0F',            / SID = 2456, idevid-issuer /
    +8 : h'cert.der',            / SID = 2459, pinned-domain-cert/
    +3 : true,                   / SID = 2454, domain-cert
                                                 -revocation-checks /
    +6 : "2017-10-07T19:31:42Z", / SID = 2457, last-renewal-date /
    +9 : h'key-info'             / SID = 2460, pinned-domain
                                           -subject-public-key-info /
  }
}
```

The signing of the example is shown in Appendix B.3.

6.3. Signing voucher and voucher-request artifacts

6.3.1. CMS signing

The IETF evolution of PKCS#7 is CMS [RFC5652]. The CMS signed voucher is much like the equivalent voucher defined in [RFC8366].

A different eContentType of TBD1 is used to indicate that the contents are in a different format than in [RFC8366]. The id-ct-animaJSONVoucher allocated by [RFC8366] indicates a voucher and voucher-request encoded in JSON, and the new value TBD1 indicates that the voucher and voucher-request are encoded in CBOR.

The ContentInfo structure contains a payload consisting of the CBOR encoded voucher. The [I-D.ietf-core-yang-cbor] use of delta encoding creates a canonical ordering for the keys on the wire. This canonical ordering is not important as there is no expectation that the content will be reproduced during the validation process.

Normally the recipient is the pledge and the signer is the MASA.

[I-D.ietf-anima-bootstrapping-keyinfra] supports both signed and unsigned voucher requests from the pledge to the JRC. In this specification, voucher-request artifact is not signed from the pledge to the registrar. [EDNOTE: Confirm that voucher requests do not need

to be signed] From the JRC to the MASA, the voucher-request artifact MUST be signed by the domain owner key which is requesting ownership.

The considerations of [RFC5652] section 5.1, concerning validating CMS objects which are really PKCS7 objects (cmsVersion=1) applies.

The CMS structure SHOULD also contain all the certificates leading up to and including the signer's trust anchor certificate known to the recipient. The inclusion of the trust anchor is unusual in many applications, but without it third parties can not accurately audit the transaction.

The CMS structure MAY also contain revocation objects for any intermediate certificate authorities (CAs) between the voucher-issuer and the trust anchor known to the recipient. However, the use of CRLs and other validity mechanisms is discouraged, as the pledge is unlikely to be able to perform online checks, and is unlikely to have a trusted clock source. As described below, the use of short-lived vouchers and/or pledge provided nonce provides a freshness guarantee.

6.3.2. COSE signing

The COSE-Sign1 structure is discussed in section 4.2 of [RFC8152]. The CBOR object that carries the body, the signature, and the information about the body and signature is called the COSE_Sign1 structure. It is used when only one signature is used on the body. Support for ECDSA with sha256 (secp256k1 and prime256v1 curves) is compulsory.

The supported COSE-sign1 object stucture is shown in Figure 1. Support for EdDSA is encouraged. [EDNOTE: Expand and add a reference why.]

```
COSE_Sign1(
  [
    h'A101382E',        # { "alg": EC256K1 }
    {
      "kid" : h'789'  # hash256(public key)
    },
    h'123', #voucher-request binary content
    h'456', #voucher-request binary public signature
  ]
)
```

Figure 1: cose-sign1 example

The [COSE-registry] specifies the integers that replace the strings and the mnemonics in Figure 1. The value of the "kid" parameter is

an example value. Usually a hash of the public key is used to idientify the public key. The public key and its hash are derived from the relevant certificate (Pledge, Registrar or MASA certificate).

In Appendix C a binary cose-sign1 object is shown based on the voucher-request example of Section 6.1.4.

7. Design Considerations

The design considerations for the CBOR encoding of vouchers is much the same as for [RFC8366].

One key difference is that the names of the leaves in the YANG does not have a material effect on the size of the resulting CBOR, as the SID translation process assigns integers to the names.

8. Security Considerations

8.1. Clock Sensitivity

TBD.

8.2. Protect Voucher PKI in HSM

TBD.

8.3. Test Domain Certificate Validity when Signing

TBD.

9. IANA Considerations

9.1. Resource Type Registry

Additions to the sub-registry "CoAP Resource Type", within the "CoRE parameters" registry are specified below. These can be registered either in the Expert Review range (0-255) or IETF Review range (256-9999).

ace.rt.rv needs registration with IANA
ace.rt.vs needs registration with IANA
ace.rt.es needs registration with IANA
ace.rt.ra needs registration with IANA

9.2. The IETF XML Registry

This document registers two URIs in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registration is requested:

```
URI: urn:ietf:params:xml:ns:yang:ietf-constrained-voucher
Registrant Contact: The ANIMA WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-constrained-voucher-request
Registrant Contact: The ANIMA WG of the IETF.
XML: N/A, the requested URI is an XML namespace.
```

9.3. The YANG Module Names Registry

This document registers two YANG modules in the YANG Module Names registry [RFC6020]. Following the format defined in [RFC6020], the the following registration is requested:

```
name:         ietf-constrained-voucher
namespace:    urn:ietf:params:xml:ns:yang:ietf-constrained-voucher
prefix:       vch
reference:    RFC XXXX

name:         ietf-constrained-voucher-request
namespace:    urn:ietf:params:xml:ns:yang:ietf-constrained
                                      -voucher-request
prefix:       vch
reference:    RFC XXXX
```

9.4. The RFC SID range assignment sub-registry

Entry-point	Size	Module name	RFC Number
2450	50	ietf-constrained-voucher	[ThisRFC]
2500	50	ietf-constrained-voucher-request	[ThisRFC}

Warning: These SID values are defined in [I-D.ietf-core-sid], not as an Early Allocation.

9.5. The SMI Security for S/MIME CMS Content Type Registry

This document registers an OID in the "SMI Security for S/MIME CMS Content Type" registry (1.2.840.113549.1.9.16.1), with the value:

Decimal	Description	References
46	id-ct-animaCBORVoucher	[ThisRFC]

9.6. Media-Type Registry

This section registers the 'application/voucher-cms+cbor' media type and the 'application/voucher-cose+cbor' in the "Media Types" registry. These media types are used to indicate that the content is a CBOR voucher either signed with a cms structure or a COSE_Sign1 structure [RFC8152].

9.6.1. application/voucher-cms+cbor

Type name: application
Subtype name: voucher-cms+cbor
Required parameters: none
Optional parameters: none
Encoding considerations: CMS-signed CBOR vouchers are CBOR encoded.
Security considerations: See Security Considerations, Section
Interoperability considerations: The format is designed to be broadly interoperable.
Published specification: THIS RFC.
Applications that use this media type: ANIMA, 6tisch, and other zero-touch imprinting systems
Additional information:
Magic number(s): None
File extension(s): .vch
Macintosh file type code(s): none
Person & email address to contact for further information: IETF ANIMA WG
Intended usage: LIMITED
Restrictions on usage: NONE
Author: ANIMA WG
Change controller: IETF
Provisional registration? (standards tree only): NO

9.6.2. application/voucher-cose+cbor

Type name: application
Subtype name: voucher-cose+cbor
Required parameters: none
Optional parameters: cose-type
Encoding considerations: COSE_Sign1 CBOR vouchers are COSE objects signed with one signer.
Security considerations: See Security Considerations, Section
Interoperability considerations: The format is designed to be broadly interoperable.
Published specification: THIS RFC.
Applications that use this media type: ANIMA, 6tisch, and other zero-touch imprinting systems
Additional information:
Magic number(s): None
File extension(s): .vch
Macintosh file type code(s): none
Person & email address to contact for further information: IETF ANIMA WG
Intended usage: LIMITED
Restrictions on usage: NONE
Author: ANIMA WG
Change controller: IETF
Provisional registration? (standards tree only): NO

9.7. CoAP Content-Format Registry

Additions to the sub-registry "CoAP Content-Formats", within the "CoRE Parameters" registry are needed for two media types. These can be registered either in the Expert Review range (0-255) or IETF Review range (256-9999).

Media type	mime type	Encoding	ID	References
application/voucher-cms+cbor	- -	CBOR	TBD2	[This RFC]
application/voucher-cose+cbor	"COSE-Sign1"	CBOR	TBD3	[This RFC]

10. Acknowledgements

We are very grateful to Jim Schaad for explaining COSE and CMS choices. Also thanks to Jim Schaad for correctinging earlier version of the COSE Sign1 objects.

Michel Veillette did extensive work on pyang to extend it to support the SID allocation process, and this document was among the first users.

We are grateful for the suggestions done by Esko Dijk.

11. Changelog

-08 Examples for cose_sign1 are completed and improved.

-06 New SID values assigned; regenerated examples

-04 voucher and request-voucher MUST be signed examples for signed request are added in appendix IANA SID registration is updated SID values in examples are aligned signed cms examples aligned with new SIDs

-03

Examples are inverted.

-02

Example of requestvoucher with unsigned appllication/cbor is added
attributes of voucher "refined" to optional
CBOR serialization of vouchers improved
Discovery port numbers are specified

-01

application/json is optional, application/cbor is compulsory
Cms and cose mediatypes are introduced

12. References

12.1. Normative References

[I-D.ietf-ace-coap-est]
Stok, P., Kampanakis, P., Richardson, M., and S. Raza, "EST over secure CoAP (EST-coaps)", draft-ietf-ace-coap-est-18 (work in progress), January 2020.

[I-D.ietf-anima-bootstrapping-keyinfra]
Pritikin, M., Richardson, M., Eckert, T., Behringer, M., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructures (BRSKI)", draft-ietf-anima-bootstrapping-keyinfra-44 (work in progress), September 2020.

[I-D.ietf-core-sid]
Veillette, M., Pelov, A., and I. Petrov, "YANG Schema Item iDentifier (YANG SID)", draft-ietf-core-sid-14 (work in progress), July 2020.

[I-D.ietf-core-yang-cbor]
Veillette, M., Petrov, I., and A. Pelov, "CBOR Encoding of Data Modeled with YANG", draft-ietf-core-yang-cbor-13 (work in progress), July 2020.

[I-D.selander-ace-ake-authz]
Selander, G., Mattsson, J., Vucinic, M., Richardson, M., and A. Schellenbaum, "Lightweight Authorization for Authenticated Key Exchange.", draft-selander-ace-ake-authz-01 (work in progress), March 2020.

[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>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <https://www.rfc-editor.org/info/rfc5652>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

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

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[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>.

[RFC8366] Watsen, K., Richardson, M., Pritikin, M., and T. Eckert, "A Voucher Artifact for Bootstrapping Protocols", RFC 8366, DOI 10.17487/RFC8366, May 2018, <https://www.rfc-editor.org/info/rfc8366>.

12.2. Informative References

[COSE-registry] IANA, ., "CBOR Object Signing and Encryption (COSE) registry", 2017, <https://www.iana.org/assignments/cose/cose.xhtml>.

[I-D.ietf-netmod-yang-tree-diagrams] Bjorklund, M. and L. Berger, "YANG Tree Diagrams", draft-ietf-netmod-yang-tree-diagrams-06 (work in progress), February 2018.

[RFC6690] Shelby, Z., "Constrained RESTful Environments (CoRE) Link Format", RFC 6690, DOI 10.17487/RFC6690, August 2012, <https://www.rfc-editor.org/info/rfc6690>.

[RFC7030] Pritikin, M., Ed., Yee, P., Ed., and D. Harkins, Ed., "Enrollment over Secure Transport", RFC 7030, DOI 10.17487/RFC7030, October 2013, <https://www.rfc-editor.org/info/rfc7030>.

Appendix A. EST messages to EST-coaps

This section extends the examples from Appendix A of [I-D.ietf-ace-coap-est]. The CoAP headers are only worked out for the enrollstatus example.

A.1. enrollstatus

A coaps enrollstatus message can be :

```
GET coaps://[192.0.2.1:8085]/est/es
```

The corresponding coap header fields are shown below.

```
  Ver = 1
  T = 0 (CON)
  Code = 0x01 (0.01 is GET)
  Options
   Option  (Uri-Path)
     Option Delta = 0xb   (option nr = 11)
     Option Length = 0x3
     Option Value = "est"
   Option  (Uri-Path)
     Option Delta = 0x0   (option nr = 11)
     Option Length = 0x2
     Option Value = "es"
  Payload = [Empty]
```

The Uri-Host and Uri-Port Options are omitted because they coincide with the transport protocol destination address and port respectively.

A 2.05 Content response with an unsigned voucher status (ct=60) will then be:

```
   2.05 Content (Content-Format: application/cbor)
```

With CoAP fields and payload:

```
   Ver=1
   T=2 (ACK)
   Code = 0x45 (2.05 Content)
   Options
     Option1 (Content-Format)
     Option Delta = 0xC  (option nr 12)
     Option Length = 0x2
     Option Value = 60 (application/cbor)

     Payload (CBOR diagnostic) =
     {
     "version":"1",
     "Status": 1,   / 1 = Success, 0 = Fail  /
     "Reason":"Informative human readable message",
     "reason-context": "Additional information"
     }
```

The binary payload is:

```
     A46776657273696F6E6131665374617475730166526561736F6E7822
     496E666F726D6174697665206875D616E2072656164616C65206D
     6573736167656e726561736F6E2D636F6E74657874
     764164646974696F6E616C20696E666F726D6174696F6E
```

The binary payload disassembles to the above CBOR diagnostic code.

A.2. voucher_status

A coaps voucher_status message can be:

```
GET coaps://[2001:db8::2:1]:61616]/est/vs
```

A 2.05 Content response with a non signed CBOR voucher status (ct=60) will then be:

```
2.05 Content (Content-Format: application/cbor)
Payload =
A46776657273696F6E6131665374617475730166526561736F6E7822
496E666F726D6174697665206875D616E207265616461626C65206D
6573736167656e726561736F6E2D636F6E74657874
764164646974696F6E616C20696E666F726D6174696F6E
```

Appendix B. CMS signed messages

Signed request-voucher-request payloads are sent from pledge to Registrar, as explained in Section 5.2 of [I-D.ietf-anima-bootstrapping-keyinfra].

B.1. signed requestvoucher

A CMS signed requestvoucher message from JRC to MASA is shown below. It would be CoAP POSTED to /est/rv.

```
POST coaps://[2001:db8::2:1]:61616]/est/rv
(Content-Format: application/voucher-cms+cbor)
```

The payload would be in binary, but is presented in base64 in this document.

MIIDugYJKoZIhvcNAQcCoIIDqzCCA6cCAQExDTALBglghkgBZQMEAgEwCwYJ
KoZIhvcNAQcBoIICQTCCAj0wggHioAMCAQICCH52Yde1TkYyMAoGCCqGSM49
BAMCMF0xCzAJBgNVBAYTAlVTMQswCQYDVQQIDAJDQTEUMBIGA1UECgwLRXhh
bXBsZSBJbmMxFjAUBgNVBAsMDWNlcnRpZmljYXRpb24xEzARBgNVBAMMCjgw
Mi4xQVIgQ0EwIBcNMTkwMTMxMTEyOTE2WhgPOTk5OTEyMzEyMzU5NTlaMFwx
CzAJBgNVBAYTAlVTMQswCQYDVQQIDAJDQTELMAkGA1UEBwwCTEExFDASBgNV
BAoMC2V4YW1wbGUgSW5jMQwwCgYDVQQLDANJb1QxDzANBgNVBAUTBld0MTIz
NDBZMBMGByqGSM49AgEGCCqGSM49AwEHA0IABMi0IfEcJeR+OsVxI78tn9xJ
TwKLw1HMgMA/FQv1DP+VjXVBnYGmokXf+ueQvpXPdfYC+RUmGPgWorI7Vjjl
n9mjgYowgYcwCQYDVR0TBAIwADAdBgNVHQ4EFgQUlmANhxa/f9DnUtCsdgd3
rWZdAqAwHwYDVR0jBBgwFoAUaNFlUflRv8gqQx0Nnwi8LSBbEWAwDgYDVR0P
AQH/BAQDAgWgMCoGA1UdEQQjMCGgHwYIKwYBBQUHCASgEzARBgkrBgEEAbQ7
CgEEBAECAwQwCgYIKoZIzj0EAwIDSQAwRgIhAMDYGZbSUH1pPzxI6qXulJG9
ptshQJnZgRfGOzYTdM2GAiEAp3SYn0wyGlzyXYMqTTNqCK1n3yDxUGQhGIoK
3m00kjYxggE/MIIBOwIBATBpMF0xCzAJBgNVBAYTAlVTMQswCQYDVQQIDAJD
QTEUMBIGA1UECgwLRXhhbXBsZSBJbmMxFjAUBgNVBAsMDWNlcnRpZmljYXRp
b24xEzARBgNVBAMMCjgwMi4xQVIgQ0ECCH52Yde1TkYyMAsGCWCGSAFlAwQC
AaBpMBgGCSqGSIb3DQEJAzELBgkqhkiG9w0BBwEwHAYJKoZIhvcNAQkFMQ8X
DTE5MDQwODEwNDgzNlowLwYJKoZIhvcNAQkEMSIEILEdCTOLs2Zy7w3LgvSZ
XZEadz3LbznoFBs6FMFN91RaMAoGCCqGSM49BAMCBEcwRQIgASjDsIpr0tW/
n6dRHqvvqsqgZlHbtFnErUbWfhS0KD4CIQDDUEqc5wTmRGf0adEQVQzqmIgh
MEgF10vqXv02gL1jLw==

A 2.04 Changed response returning CBOR voucher signed with a cms structure(ct=TBD2) will then be:

2.04 Changed (Content-Format: application/voucher-cms+cbor)

MIIDuwYJKoZIhvcNAQcCoIIDrDCCA6gCAQExDTALBglghkgBZQMEAgEwCwYJ
KoZIhvcNAQcBoIICQTCCAj0wggHioAMCAQICCH52Yde1TkYyMAoGCCqGSM49
BAMCMF0xCzAJBgNVBAYTAlVTMQswCQYDVQQIDAJDQTEUMBIGA1UECgwLRXhh
bXBsZSBJbmMxFjAUBgNVBAsMDWNlcnRpZmljYXRpb24xEzARBgNVBAMMCjgw
Mi4xQVIgQ0EwIBcNMTkwMTMxMTEyOTE2WhgPOTk5OTEyMzEyMzU5NTlaMFwx
CzAJBgNVBAYTAlVTMQswCQYDVQQIDAJDQTELMAkGA1UEBwwCTEExFDASBgNV
BAoMC2V4YW1wbGUgSW5jMQwwCgYDVQQLDANJb1QxDzANBgNVBAUTBld0MTIz
NDBZMBMGByqGSM49AgEGCCqGSM49AwEHA0IABMi0IfEcJeR+OsVxI78tn9xJ
TwKLw1HMgMA/FQv1DP+VjXVBnYGmokXf+ueQvpXPdfYC+RUmGPgWorI7Vjjl
n9mjgYowgYcwCQYDVR0TBAIwADAdBgNVHQ4EFgQUlmANhxa/f9DnUtCsdgd3
rWZdAqAwHwYDVR0jBBgwFoAUaNFlUflRv8gqQx0Nnwi8LSBbEWAwDgYDVR0P
AQH/BAQDAgWgMCoGA1UdEQQjMCGgHwYIKwYBBQUHCASgEzARBgkrBgEEAbQ7
CgEEBAECAwQwCgYIKoZIzj0EAwIDSQAwRgIhAMDYGZbSUH1pPzxI6qXulJG9
ptshQJnZgRfGOzYTdM2GAiEAp3SYn0wyGlzyXYMqTTNqCK1n3yDxUGQhGIoK
3m00kjYxggFAMIIBPAIBATBpMF0xCzAJBgNVBAYTAlVTMQswCQYDVQQIDAJD
QTEUMBIGA1UECgwLRXhhbXBsZSBJbmMxFjAUBgNVBAsMDWNlcnRpZmljYXRp
b24xEzARBgNVBAMMCjgwMi4xQVIgQ0ECCH52Yde1TkYyMAsGCWCGSAFlAwQC
AaBpMBgGCSqGSIb3DQEJAzELBgkqhkiG9w0BBwEwHAYJKoZIhvcNAQkFMQ8X
DTE5MDQwODA3MzQxMFowLwYJKoZIhvcNAQkEMSIEIP2rKa+J8LVdwYEmB2he
uxsz05As0zoAAYkeyNqsh4fiMAoGCCqGSM49BAMCBEgwRgIhALOd2FKbe9FG
kN4Pg7FIgF+//cQv/N+v7tDZMzGBAFN0AiEAu5BI0oQ4o0wZcrDyKoU2GbeX
hlG/g+OgTUftYMJ32so=

B.2. requestauditing

A coaps requestauditing message contains the signed CBOR voucher :

```
    POST coaps://[2001:db8::2:1]:61616]/est/ra
    (Content-Format: application/voucher-cms+cbor)
    Payload =
TO BE FILLED
```

A 2.05 Content response returning a log of the voucher (ct=60) will then be:

```
    2.05 Content (Content-Format: application/cbor)
    Payload =
{
 "version":"1",
 "events":[
   {
    "date":"<date/time of the entry>",
    "domainID":"<domainID extracted from voucher-request>",
    "nonce":"<any nonce if supplied (or the exact string 'NULL')>"
    "assertion":"<the value from the voucher assertion leaf>"
    "truncated":"<the number of domainID entries truncated>"
   },
   {
    "date":"<date/time of the entry>",
    "domainID":"<anotherDomainID extracted from voucher-request>",
    "nonce":"<any nonce if supplied (or the exact string 'NULL')>"
    "assertion":"<the value from the voucher assertion leaf>"
   }
 ],
  "truncation": {
    "nonced duplicates": "<total number of entries truncated>",
    "nonceless duplicates": "<total number of entries truncated>",
    "arbitrary": "<number of domainID entries removed entirely>"
  }
}
```

[EDNOTE: Change JSON to CBOR; Serialize CBOR payload to binary]

B.3. CMS signed voucher-request example

The voucher-request example, visualized in CBOR diagnostic notation in Section 6.1.4 is shown as a hexadecimal dump of the binary file.

```
a11909c5a90274323031362d31302d30375431393a33313a34325a0474323031
362d31302d32315431393a33313a34325a01020d6d4a414441313233343536
373839054401020d0f0a4401020d0f03f50674323031372d31302d30375431
393a33313a34325a0c4401020d0f
```

The voucher-request example has been signed by using the WT1234 certificate and key pair shown in Appendix C of [I-D.ietf-ace-coap-est]. The CMS signing of the binary voucher-request leads to a binary signed voucher-request, shown with a hexadecimal representation shown in the payload of the request part of Appendix B.1 and Appendix B.2.

The breakdown of the CMS signed binary voucher-request file is visualized below:

```
   CMS_ContentInfo:
     contentType: pkcs7-signedData (1.2.840.113549.1.7.2)
     d.signedData:
       version: 1
       digestAlgorithms:
           algorithm: sha256 (2.16.840.1.101.3.4.2.1)
           parameter: <ABSENT>
       encapContentInfo:
         eContentType: pkcs7-data (1.2.840.113549.1.7.1)
         eContent: <ABSENT>
       certificates:
         d.certificate:
           cert_info:
             version: 2
             serialNumber: 9112578475118446130
             signature:
               algorithm: ecdsa-with-SHA256 (1.2.840.10045.4.3.2)
               parameter: <ABSENT>
             issuer: C=US, ST=CA, O=Example Inc, OU=certification,
                                                  CN=802.1AR CA
             validity:
               notBefore: Jan 31 11:29:16 2019 GMT
               notAfter: Dec 31 23:59:59 9999 GMT
             subject: C=US, ST=CA, L=LA, O=example Inc,
                                    OU=IoT/serialNumber=Wt1234
             key:
               algor:
                 algorithm: id-ecPublicKey (1.2.840.10045.2.1)
                 parameter: OBJECT:prime256v1 (1.2.840.10045.3.1.7)
               public_key:  (0 unused bits)
                 0000 - 04 c8 b4 21 f1 1c 25 e4-7e 3a c5 71 23 bf
                 000e - 2d 9f dc 49 4f 02 8b c3-51 cc 80 c0 3f 15
                 001c - 0b f5 0c ff 95 8d 75 41-9d 81 a6 a2 45 df
                 002a - fa e7 90 be 95 cf 75 f6-02 f9 15 26 18 f8
                 0038 - 16 a2 b2 3b 56 38 e5 9f-d9
             issuerUID: <ABSENT>
             subjectUID: <ABSENT>
             extensions:
                 object: X509v3 Basic Constraints (2.5.29.19)
                 critical: BOOL ABSENT
                 value:
                   0000 - 30
                   0002 - <SPACES/NULS>

                 object: X509v3 Subject Key Identifier (2.5.29.14)
                 critical: BOOL ABSENT
                 value:
                   0000 - 04 14 96 60 0d 87 16 bf-7f d0 e7 52 d0
```

```
                000d - ac 76 07 77 ad 66 5d 02-a0

              object: X509v3 Authority Key Identifier (2.5.29.35)
              critical: BOOL ABSENT
              value:
                0000 - 30 16 80 14 68 d1 65 51-f9 51 bf c8 2a
                000d - 43 1d 0d 9f 08 bc 2d 20-5b 11 60

              object: X509v3 Key Usage (2.5.29.15)
              critical: TRUE
              value:
                0000 - 03 02 05 a0

              object: X509v3 Subject Alternative Name (2.5.29.17)
              critical: BOOL ABSENT
              value:
                0000 - 30 21 a0 1f 06 08 2b 06-01 05 05 07 08
                000d - 04 a0 13 30 11 06 09 2b-06 01 04 01 b4
                001a - 3b 0a 01 04 04 01 02 03-04
      sig_alg:
        algorithm: ecdsa-with-SHA256 (1.2.840.10045.4.3.2)
        parameter: <ABSENT>
      signature:  (0 unused bits)
        0000 - 30 46 02 21 00 c0 d8 19-96 d2 50 7d 69 3f 3c
        000f - 48 ea a5 ee 94 91 bd a6-db 21 40 99 d9 81 17
        001e - c6 3b 36 13 74 cd 86 02-21 00 a7 74 98 9f 4c
        002d - 32 1a 5c f2 5d 83 2a 4d-33 6a 08 ad 67 df 20
        003c - f1 50 64 21 18 8a 0a de-6d 34 92 36
  crls:
    <EMPTY>
  signerInfos:
      version: 1
      d.issuerAndSerialNumber:
        issuer: C=US, ST=CA, O=Example Inc, OU=certification,
                                                CN=802.1AR CA
        serialNumber: 9112578475118446130
      digestAlgorithm:
        algorithm: sha256 (2.16.840.1.101.3.4.2.1)
        parameter: <ABSENT>
      signedAttrs:
          object: contentType (1.2.840.113549.1.9.3)
          value.set:
            OBJECT:pkcs7-data (1.2.840.113549.1.7.1)

          object: signingTime (1.2.840.113549.1.9.5)
          value.set:
            UTCTIME:Jul  3 08:53:30 2019 GMT
```

```
            object: messageDigest (1.2.840.113549.1.9.4)
            value.set:
              OCTET STRING:
                0000 - d4 b0 5c dd c8 b4 91 28-4a 18 ca 25 9d
                000d - be d0 60 23 cf ad a0 aa-c2 95 ac e9 3f
                001a - 0b 4f 44 9e 25
                0020 - <SPACES/NULS>
          signatureAlgorithm:
            algorithm: ecdsa-with-SHA256 (1.2.840.10045.4.3.2)
            parameter: <ABSENT>
          signature:
            0000 - 30 46 02 21 00 e5 e1 7f-23 c3 aa 14 9f 35 64
            000f - 1e c4 4a 0f 68 fe b0 16-3b e6 7c 06 51 af bf
            001e - 5a a0 99 59 e0 28 1f 02-21 00 b4 07 2f 7c c4
            002d - f9 26 0c 6d 47 a7 93 56-de b8 da f7 23 f0 af
            003c - 2b 59 16 cc 36 63 e7 91-89 39 df df
          unsignedAttrs:
            <EMPTY>
```

Appendix C. COSE examples

These examples are generated on a pie 4 and a PC running BASH. Keys and Certificates have been generated with openssl with the following shell script:

```
#!/bin/bash
#try-cert.sh
export dir=./brski/intermediate
export cadir=./brski
export cnfdir=./conf
export format=pem
export default_crl_days=30
sn=8

DevID=pledge.1.2.3.4
serialNumber="serialNumber=$DevID"
export hwType=1.3.6.1.4.1.6715.10.1
export hwSerialNum=01020304
export subjectAltName="otherName:1.3.6.1.5.5.7.8.4;SEQ:hmodname"
echo  $hwType - $hwSerialNum
echo $serialNumber

# remove all files
rm -r ./brski/*
#
# initialize file structure
# root level
cd $cadir
```

```
   mkdir certs crl csr newcerts private
   chmod 700 private
   touch index.txt
   touch serial
   echo 11223344556600 >serial
   echo 1000 > crlnumber
   # intermediate level
   mkdir intermediate
   cd intermediate
   mkdir certs crl csr newcerts private
   chmod 700 private
   touch index.txt
   echo 11223344556600 >serial
   echo 1000 > crlnumber
   cd ../..



   # file structure is cleaned start filling

   echo "##############################"
   echo "create registrar keys and certificates "
   echo "##############################"


   echo "create root registrar certificate using ecdsa with sha256"
   openssl ecparam -name prime256v1 -genkey \
     -noout -out $cadir/private/ca-regis.key

   openssl req -new -x509 \
    -key $cadir/private/ca-regis.key \
    -out $cadir/certs/ca-regis.crt \
    -extensions v3_ca\
    -days 365 \
    -subj "/C=NL/ST=NB/L=Helmond/O=vanderstok\
   "/OU=consultancy/CN=registrar.stok.nl"

   # Combine authority certificate and key
   echo "Combine authority certificate and key"
   openssl pkcs12 -passin pass:watnietweet \
    -passout pass:watnietweet \
    -inkey $cadir/private/ca-regis.key \
    -in $cadir/certs/ca-regis.crt -export \
    -out $cadir/certs/ca-regis-comb.pfx

   # converteer authority pkcs12 file to pem
   echo "converteer authority pkcs12 file to pem"
   openssl pkcs12 -passin pass:watnietweet -passout pass:watnietweet\
```

```
      -in $cadir/certs/ca-regis-comb.pfx \\
      -out $cadir/certs/ca-regis-comb.crt -nodes

   #show certificate in registrar combined certificate
   openssl  x509 -in $cadir/certs/ca-regis-comb.crt -text

   #
   # Certificate Authority for MASA
   #
   echo "##############################"
   echo "create MASA keys and certificates "
   echo "##############################"

   echo "create root MASA certificate using ecdsa with sha 256 key"
   openssl ecparam -name prime256v1 -genkey -noout \
    -out $cadir/private/ca-masa.key

   openssl req -new -x509 \
    -days 365 -key $cadir/private/ca-masa.key \
    -out $cadir/certs/ca-masa.crt \
    -extensions v3_ca\
    -subj "/C=NL/ST=NB/L=Helmond/O=vanderstok/\
   OU=manufacturer/CN=masa.stok.nl"

   # Combine authority certificate and key
   echo "Combine authority certificate and key for masa"
   openssl pkcs12 -passin pass:watnietweet \
      -passout pass:watnietweet\
      -inkey $cadir/private/ca-masa.key \
      -in $cadir/certs/ca-masa.crt -export \
      -out $cadir/certs/ca-masa-comb.pfx

   # converteer authority pkcs12 file to pem for masa
   echo "converteer authority pkcs12 file to pem for masa"
   openssl pkcs12 -passin pass:watnietweet \
      -passout pass:watnietweet\
      -in $cadir/certs/ca-masa-comb.pfx \
      -out $cadir/certs/ca-masa-comb.crt -nodes

   #show certificate in pledge combined certificate
   openssl  x509 -in $cadir/certs/ca-masa-comb.crt -text


   #
   # Certificate for Pledge derived from MASA certificate
   #
   echo "##############################"
   echo "create pledge keys and certificates "
```

```
   echo "##############################"


   # Pledge derived Certificate

   echo "create pledge derived certificate using ecdsa with sha256"
   openssl ecparam -name prime256v1 -genkey \
    -noout -out $dir/private/pledge.key

   echo "create pledge certificate request"
   openssl req -nodes -new -sha256 \
     -key $dir/private/pledge.key -out $dir/csr/pledge.csr \
     -subj \
   "/C=NL/ST=NB/L=Helmond/O=vanderstok/OU=manufacturing\
   /CN=uuid:$DevID/$serialNumber"

   # Sign pledge derived Certificate
   echo "sign pledge derived certificate "
   openssl ca -config $cnfdir/openssl-pledge.cnf \
    -extensions 8021ar_idevid\
    -days 365 -in $dir/csr/pledge.csr -out $dir/certs/pledge.crt

   # Add pledge key and pledge certificate to pkcs12 file
   echo "Add pledge key and pledge certificate to pkcs12 file"
   openssl pkcs12  -passin pass:watnietweet\
      -passout pass:watnietweet\
      -inkey $dir/private/pledge.key \
      -in $dir/certs/pledge.crt -export \
      -out $dir/certs/pledge-comb.pfx

   # converteer pledge pkcs12 file to pem
   echo "converteer pledge pkcs12 file to pem"
   openssl pkcs12 -passin pass:watnietweet \
       -passout pass:watnietweet\
      -in $dir/certs/pledge-comb.pfx \
      -out $dir/certs/pledge-comb.crt -nodes

   #show certificate in pledge-comb.crt
   openssl  x509 -in $dir/certs/pledge-comb.crt -text

   #show private key in pledge-comb.crt
   openssl ecparam -name prime256v1 \
     -in $dir/certs/pledge-comb.crt -text

   The xxxx-comb certificates have been generated as required by libcoap
   for the DTLS connection generation.
```

C.1. Pledge, Registrar and MASA keys

This first section documents the public and private keys used in the subsequent test vectors below. These keys come from test code and are not used in any production system, and should only be used only to validate implementations.

C.1.1. Pledge private key

```
-----BEGIN PRIVATE KEY-----
MIGHAgEAMBMGByqGSM49AgEGCCqGSM49AwEHBG0wawIBAQQgIpP20ud7muTl460b
xFzupPkaMoaCIIIFwSOf0hvhQByhRANCAASKnIauvAtx6ZFWQniQOqvP0Zpdaudy
Ve6Vrc80AjyWRGnN3oyQ0rnr5dXynfG2xq8+cY+uGwTrAJYp9OyoZCAs
-----END PRIVATE KEY-----
Private-Key: (256 bit)
priv:
    22:93:f6:d2:e7:7b:9a:e4:e5:e3:ad:1b:c4:5c:ee:
    a4:f9:1a:32:86:82:20:82:05:c1:23:9f:d2:1b:e1:
    40:1c
pub:
    04:8a:9c:86:ae:bc:0b:71:e9:91:56:42:78:90:3a:
    ab:cf:d1:9a:5d:6a:e7:72:55:ee:95:ad:cf:34:02:
    3c:96:44:69:cd:de:8c:90:d2:b9:eb:e5:d5:f2:9d:
    f1:b6:c6:af:3e:71:8f:ae:1b:04:eb:00:96:29:f4:
    ec:a8:64:20:2c
ASN1 OID: prime256v1
NIST CURVE: P-256
```

C.1.2. Registrar private key

```
-----BEGIN PRIVATE KEY-----
MIGHAgEAMBMGByqGSM49AgEGCCqGSM49AwEHBG0wawIBAQQgHCCOKLhln+l8pLnx
gWtMUm7zRY4ugkznuFimYDKbrNihRANCAARqJKniS+I00XrUfnYMlLXh3E7hFa2J
ESrUpqZLsb9o+Rd9cOkQnLSMmw3H3yZBGZ0MLb/yHtWEA4rIP0eBvhOO
-----END PRIVATE KEY-----
Private-Key: (256 bit)
priv:
    1c:20:8e:28:b8:65:9f:e9:7c:a4:b9:f1:81:6b:4c:
    52:6e:f3:45:8e:2e:82:4c:e7:b8:58:a6:60:32:9b:
    ac:d8
pub:
    04:6a:24:a9:e2:4b:e2:34:d1:7a:d4:7e:76:0c:94:
    b5:e1:dc:4e:e1:15:ad:89:11:2a:d4:a6:a6:4b:b1:
    bf:68:f9:17:7d:70:e9:10:9c:b4:8c:9b:0d:c7:df:
    26:41:19:9d:0c:2d:bf:f2:1e:d5:84:03:8a:c8:3f:
    47:81:be:13:8e
ASN1 OID: prime256v1
NIST CURVE: P-256
```

C.1.3. MASA private key

```
-----BEGIN PRIVATE KEY-----
MIGHAgEAMBMGByqGSM49AgEGCCqGSM49AwEHBG0wawIBAQQgQODnSgB7xR/aa3Ea
JrPGz9lZhJ1aEc/56OEPiBr86SKhRANCAASB9HLsnEeyjtHrODNBANNi9khQ2gLQ
VrIie8hLgFmVdwfQw1iMPPI8WwCDeVTaDdGwr6HC6M0sO9CGRZ+JcwrL
-----END PRIVATE KEY-----
Private-Key: (256 bit)
priv:
    40:e0:e7:4a:00:7b:c5:1f:da:6b:71:1a:26:b3:c6:
    cf:d9:59:84:9d:5a:11:cf:f9:e8:e1:0f:88:1a:fc:
    e9:22
pub:
    04:81:f4:72:ec:9c:47:b2:8e:d1:eb:38:33:41:00:
    d3:62:f6:48:50:da:02:d0:56:b2:22:7b:c8:4b:80:
    59:95:77:07:d0:c3:58:8c:3c:f2:3c:5b:00:83:79:
    54:da:0d:d1:b0:af:a1:c2:e8:cd:2c:3b:d0:86:45:
    9f:89:73:0a:cb
ASN1 OID: prime256v1
NIST CURVE: P-256
```

C.2. Pledge, Registrar and MASA certificates

Below the certificates that accompany the keys. The certificate description is followed by the hexadecimal DER of the certificate

C.2.1. Pledge IDevID certificate

```
   Certificate:
       Data:
           Version: 3 (0x2)
           Serial Number: 4822678189204992 (0x11223344556600)
           Signature Algorithm: ecdsa-with-SHA256
           Issuer: C=NL, ST=NB, L=Helmond, O=vanderstok,
                   OU=manufacturer, CN=masa.stok.nl
           Validity
               Not Before: Sep  9 07:42:03 2020 GMT
               Not After : Dec 31 23:59:59 9999 GMT
           Subject: C=NL, ST=NB, L=Helmond, O=vanderstok,
                      OU=manufacturing,
          CN=uuid:pledge.1.2.3.4/serialNumber=pledge.1.2.3.4
           Subject Public Key Info:
               Public Key Algorithm: id-ecPublicKey
                   Public-Key: (256 bit)
                   pub:
                       04:8a:9c:86:ae:bc:0b:71:e9:91:56:42:78:90:3a:
                       ab:cf:d1:9a:5d:6a:e7:72:55:ee:95:ad:cf:34:02:
                       3c:96:44:69:cd:de:8c:90:d2:b9:eb:e5:d5:f2:9d:
                       f1:b6:c6:af:3e:71:8f:ae:1b:04:eb:00:96:29:f4:
                       ec:a8:64:20:2c
                   ASN1 OID: prime256v1
                   NIST CURVE: P-256
           X509v3 extensions:
               X509v3 Basic Constraints:
                   CA:FALSE
               X509v3 Subject Key Identifier:
   59:B1:E1:19:F4:68:53:E9:0E:7C:9F:29:D0:FB:5B:1F:AC:C3:82:49
               X509v3 Authority Key Identifier:
                   keyid:
   22:BC:B8:20:D9:C5:6D:2D:5B:B3:BB:64:8B:E0:8B:A7:86:5E:CE:B4

               X509v3 Key Usage: critical
                   Digital Signature, Key Encipherment
       Signature Algorithm: ecdsa-with-SHA256
            30:45:02:20:4d:fd:a8:83:78:31:d2:62:a4:e5:48:a2:e0:a7:
            3b:c5:14:e9:7e:46:13:45:bc:30:fd:1d:e5:d6:63:3e:d8:f4:
            02:21:00:a8:e5:1e:c2:79:77:90:fc:40:a8:7a:bf:b1:bd:81:
            8b:ee:d7:56:1a:04:4d:8f:c8:3d:76:5f:4d:6e:36:a2:c2
```

This is the hexadecimal representation in (request-)voucher examples referred to as pledge-cert-hex.

```
30820254308201faa003020102020711223344556600300a06082a8648ce3d04
0302306f310b300906035504061302e4c310b300906035504080c024e423110
300e06035504070c0748656c6d6f6e643113301106035504a0c0a76616e6465
7273746f6b31153013060355040b0c0c6d616e756661637475726572311530 13
06035504030c0c6d6173612e73746f6b2e6e6c3020170d323030393039303734
3230335a180f393939393132333132333533935395a308190310b300906035504
0613024e4c310b300906035504080c024e423110300e06035504070c0748656c
6d6f6e643113301106035504a0c0a76616e64657273746f6b31163014060355
040b0c0d6d616e75666163747572696e67311c301a06035504030c1375756964
3a706c656467652e312e322e332e343117301506035504051 30e706c65646765
2e312e322e332e343059301306072a8648ce3d020106082a8648ce3d03010703
4200048a9c86aebc0b71e991564278903aabcfd19a5d6ae77255ee95adcf3402
3c964469cdde8c90d2b9ebe5d5f29df1b6c6af3e718fae1b04eb009629f4eca8
64202ca35d305b30090603551d1304023000301d0603551d0e0416041459b1e1
19f46853e90e7c9f29d0fb5b1facc38249301f0603551d2304183016801422bc
b820d9c56d2d5bb3bb648be08ba7865eceb4300e0603551d0f0101ff04040302
05a0300a06082a8648ce3d040302034800304502204dfda8837831d262a4e548
a2e0a73bc514e97e461345bc30fd1de5d6633ed8f4022100a8e51ec2797790fc
40a87abfb1bd818beed7561a044d8fc83d765f4d6e36a2c2
```

C.2.2. Registrar Certificate

```
 Certificate:
     Data:
         Version: 3 (0x2)
         Serial Number:
             39:73:74:f3:fa:81:2a:0d:37:10:3b:68:c1:84:81:c5:01:bc:7c:fe
         Signature Algorithm: ecdsa-with-SHA256
         Issuer: C=NL, ST=NB, L=Helmond, O=vanderstok,
                 OU=consultancy, CN=registrar.stok.nl
         Validity
             Not Before: Sep  9 07:42:03 2020 GMT
             Not After : Sep  9 07:42:03 2021 GMT
         Subject: C=NL, ST=NB, L=Helmond, O=vanderstok,
               OU=consultancy, CN=registrar.stok.nl
         Subject Public Key Info:
             Public Key Algorithm: id-ecPublicKey
                 Public-Key: (256 bit)
                 pub:
                     04:6a:24:a9:e2:4b:e2:34:d1:7a:d4:7e:76:0c:94:
                     b5:e1:dc:4e:e1:15:ad:89:11:2a:d4:a6:a6:4b:b1:
                     bf:68:f9:17:7d:70:e9:10:9c:b4:8c:9b:0d:c7:df:
                     26:41:19:9d:0c:2d:bf:f2:1e:d5:84:03:8a:c8:3f:
                     47:81:be:13:8e
                 ASN1 OID: prime256v1
                 NIST CURVE: P-256
         X509v3 extensions:
             X509v3 Subject Key Identifier:
 25:CD:93:71:B5:A1:5F:6D:1E:E8:C3:7A:51:13:BE:0B:8F:13:2C:C2
             X509v3 Authority Key Identifier:
                 keyid:
 25:CD:93:71:B5:A1:5F:6D:1E:E8:C3:7A:51:13:BE:0B:8F:13:2C:C2

             X509v3 Basic Constraints:
                 CA:TRUE
     Signature Algorithm: ecdsa-with-SHA256
          30:46:02:21:00:a6:6d:9e:24:f9:de:08:b7:f0:cf:43:c3:c0:
          ee:57:cc:b6:60:de:ae:2e:70:cc:61:a1:a2:b3:35:35:02:5b:
          ba:02:21:00:bf:fd:74:6a:99:eb:da:01:77:fc:6c:37:95:75:
          8a:f4:a0:9f:99:8e:bc:4a:90:62:49:f0:7a:c9:65:96:dc:75
```

This the hexadecimal representation, in (request-)voucher examples referred to as regis-cert-hex

```
30820239308201dea003020102021439737 4f3fa812a0d37103b68c18481c501
bc7cfe300a06082a8648ce3d0403023073310b30090603550406 13024e4c310b
300906035504080c024e423110300e06035504070c0748656c6d6f6e64311330
11060355040a0c0a76616e64657273746f6b31143012060355040b0c0b636f6e
73756c74616e6379311a301806035504030c11726567697374726172e73746f
6b2e6e6c301e170d323030393039303734323033 5a170d323130393039303734
3230335a3073310b300906035504061302e4c310b300906035504080c024e42
3110300e06035504070c0748656c6d6f6e643113301106035504 0a0c0a76616e
64657273746f6b31143012060355040b0c0b636f6e73756c74616e6379311a30
1806035504030c11726567697374726172e73746f6b2e6e6c3059301306072a
8648ce3d020106082a8648ce3d030107034200046a24a9e24be234d17ad47e76
0c94b5e1dc4ee115ad89112ad4a6a64bb1bf68f9177d70e9109cb48c9b0dc7df
2641199d0c2dbff21ed584038ac83f4781be138ea350304e301d0603551d0e04
16041425cd9371b5a15f6d1ee8c37a5113be0b8f132cc2301f0603551d230418
3016801425cd9371b5a15f6d1ee8c37a5113be0b8f132cc2300c0603551d1304
0530030101ff300a06082a8648ce3d040302034900 3046022100a66d9e24f9de
08b7f0cf43c3c0ee57ccb660deae2e70cc61a1a2b33535025bba022100bffd74
6a99ebda0177fc6c3795758af4a09f998ebc4a906249f07ac96596dc75
```

C.2.3. MASA Certificate

```
 Certificate:
     Data:
         Version: 3 (0x2)
         Serial Number:
             70:5a:34:7e:67:d2:4d:70:b0:c6:ca:60:ff:fb:75:d9:46:82:e6:0e
         Signature Algorithm: ecdsa-with-SHA256
         Issuer: C=NL, ST=NB, L=Helmond, O=vanderstok,
                 OU=manufacturer, CN=masa.stok.nl
         Validity
             Not Before: Sep  9 07:42:03 2020 GMT
             Not After : Sep  9 07:42:03 2021 GMT
         Subject: C=NL, ST=NB, L=Helmond, O=vanderstok,
                   OU=manufacturer, CN=masa.stok.nl
         Subject Public Key Info:
             Public Key Algorithm: id-ecPublicKey
                 Public-Key: (256 bit)
                 pub:
                     04:81:f4:72:ec:9c:47:b2:8e:d1:eb:38:33:41:00:
                     d3:62:f6:48:50:da:02:d0:56:b2:22:7b:c8:4b:80:
                     59:95:77:07:d0:c3:58:8c:3c:f2:3c:5b:00:83:79:
                     54:da:0d:d1:b0:af:a1:c2:e8:cd:2c:3b:d0:86:45:
                     9f:89:73:0a:cb
                 ASN1 OID: prime256v1
                 NIST CURVE: P-256
         X509v3 extensions:
             X509v3 Subject Key Identifier:
 22:BC:B8:20:D9:C5:6D:2D:5B:B3:BB:64:8B:E0:8B:A7:86:5E:CE:B4
             X509v3 Authority Key Identifier:
                 keyid:
 22:BC:B8:20:D9:C5:6D:2D:5B:B3:BB:64:8B:E0:8B:A7:86:5E:CE:B4

             X509v3 Basic Constraints:
                 CA:TRUE
     Signature Algorithm: ecdsa-with-SHA256
          30:45:02:20:04:ac:8d:48:62:a2:a5:04:4f:61:fd:38:83:53:
          9f:00:e7:d6:4b:4d:30:1b:84:29:d4:2d:35:58:b0:a0:0c:7d:
          02:21:00:8c:f1:f4:f9:a2:11:fe:64:46:a9:87:9f:58:ca:ea:
          da:4f:0a:42:32:c2:6a:e8:c5:9d:62:c0:67:f0:b8:44:43
```

This is the hexadecimal representation, in (request-)voucher examples referred to as masa-cert-hex.

```
30820230308201d6a0030201020214705a347e67d24d70b0c6ca60fffb75d946
82e60e300a06082a8648ce3d040302306f310b300906035504061302e4c310b
300906035504080c024e423110300e06035504070c0748656c6d6f6e64311330
11060355040a0c0a76616e64657273746f6b31153013060355040b0c0c6d616e
756661637475726572311530130603550403 0c0c6d6173612e73746f6b2e6e6c
301e170d3230303930393037343230335a170d3231303930393037343230335a
306f310b300906035504061302e4c310b300906035504080c024e423110300e
06035504070c0748656c6d6f6e643113301106035504 0a0c0a76616e64657273
746f6b31153013060355040b0c0c6d616e756661637475726572311530130603
5504030c0c6d6173612e73746f6b2e6e6c3059301306072a8648ce3d02010608
2a8648ce3d030107034200481f472ec9c47b28ed1eb38334100d362f64850da
02d056b2227bc84b8059957707d0c3588c3cf23c5b00837954da0dd1b0afa1c2
e8cd2c3bd086459f89730acba350304e301d0603551d0e0416041422bcb820d9
c56d2d5bb3bb648be08ba7865eceb4301f0603551d2304183016801422bcb820
d9c56d2d5bb3bb648be08ba7865eceb4300c0603551d13040530030101ff300a
06082a8648ce3d0403020348003045022004ac8d4862a2a5044f61fd3883539f
00e7d64b4d301b8429d42d3558b0a00c7d0221008cf1f4f9a211fe6446a9879f
58caeada4f0a4232c26ae8c59d62c067f0b84443
```

C.3. COSE signed voucher request from pledge to Registrar

In this example the voucher request has been signed by the pledge, and has been sent to the JRC over CoAPS. This example uses the proximity-registrar-cert mechanism to request a voucher that pins the certificate of the registrar.

```
POST coaps://registrar.example.com/est/rv
(Content-Format: application/voucher-cose+cbor)
signed_request_voucher
```

The payload signed_request_voucher is shown as hexadecimal dump (with lf added):

```
d28444a101382ea1045820f8926f5ba385b7bccf23592b97a73c1b00bffc01023
0f647f06960870b1fd6ee5902aca11909c5a61909c77818323032302d31302d35
5431333a34363a31332d30303a30301909c97818323032322d31302d355431333
a34363a31332d30303a30301909c6021909cc5029c7bafb81a2c6160d3357d229
11f5101909d26e706c656467652e312e322e332e341909cf59023d30820239308
201dea003020102021439737 4f3fa812a0d37103b68c18481c501bc7cfe300a06
082a8648ce3d0403023073310b30090603550406 13024e4c310b3009060355040
80c024e423110300e06035504070c0748656c6d6f6e643113301106035504 0a0c
0a76616e64657273746f6b31143012060355040b0c0b636f6e73756c74616e637
9311a301806035504030c11726567697374726172e73746f6b2e6e6c301e170d
323030393039303734323033 5a170d323130393039303734323033 5a3073310b3
009060355040613024e4c310b300906035504080c024e423110300e0603550407
0c0748656c6d6f6e643113301106035504 0a0c0a76616e64657273746f6b31143
012060355040b0c0b636f6e73756c74616e6379311a301806035504030c117265
676973747261722e73746f6b2e6e6c3059301306072a8648ce3d020106082a864
8ce3d030107034200046a24a9e24be234d17ad47e760c94b5e1dc4ee115ad8911
2ad4a6a64bb1bf68f9177d70e9109cb48c9b0dc7df2641199d0c2dbff21ed5840
38ac83f4781be138ea350304e301d0603551d0e0416041425cd9371b5a15f6d1e
e8c37a5113be0b8f132cc2301f0603551d2304183016801425cd9371b5a15f6d1
ee8c37a5113be0b8f132cc2300c0603551d13040530030101ff300a06082a8648
ce3d040302034900304602210 0a66d9e24f9de08b7f0cf43c3c0ee57ccb660dea
e2e70cc61a1a2b33535025bba022100bffd746a99ebda0177fc6c3795758af4a0
9f998ebc4a906249f07ac96596dc7558473045022100fc28be418e5f25152590e
872b4bbdbe334cd31d1ebb0a806e7a172cad5cff604022056ee414ddac438e7f5
1dda9ddf6ec6e31a78cdde6574717fe46dd3a7c60f5bb5
```

The representiation of signed_voucher_request in CBOR diagnostic format is:

```
Diagnose(signed_request_voucher) =
18([
h'A101382E',     # {"alg": -47}
{4:h'F8926F5BA385B7BCCF23592B97A73C1B00BFFC010230F647F06960870B1F
D6EE'},
h'request_voucher'
h'3045022100FC28BE418E5F25152590E872B4BBDBE334CD31D1EBB0A806E7A17
2CAD5CFF604022056EE414DDAC438E7F51DDA9DDF6EC6E31A78CDDE6574717FE4
6DD3A7C60F5BB5'])

Diagnose(request_voucher) =
{2501: {2503: "2020-10-5T13:46:13-00:00",
        2505: "2022-10-5T13:46:13-00:00",
        2502: 2,
        2508: h'29C7BAFB81A2C6160D3357D22911F510',
        2514: "pledge.1.2.3.4",
        2511: h'regis-cert-hex'}},
```

C.4. COSE signed voucher request from Registrar to MASA

In this example the voucher request has been signed by the JRC using the private key from Appendix C.1.2. Contained within this voucher request is the voucher request from the pledge to JRC.

```
POST coaps://masa.example.com/est/rv
(Content-Format: application/voucher-cose+cbor)
signed_masa_request_voucher
```

The payload signed_masa_voucher_request is shown as hexadecimal dump (with lf added):

d28444a101382ea1045820b86ae808f79af17e5948cbda731f158d04bd091c73f
485f2409eac08ee7ddb5c5903fea11909c5a61909c77818323032302d31302d35
5431333a34363a31332d30303a30301909c97818323032322d31302d355431333
a34363a31332d30303a30301909cc5029c7bafb81a2c6160d3357d22911f51019
09d26e706c656467652e312e322e332e341909ca586b433d4e4c2c2053543d4e4
22c204c3d48656c6d6f6e642c204f3d76616e64657273746f6b2c204f553d6d61
6e75666163747572696e672c20434e3d757569643a706c656467652e312e322e3
32e342c2073657269616c4e756d6265723d706c656467652e312e322e332e3419
09ce590323d28444a101382ea1045820f8926f5ba385b7bccf23592b97a73c1b0
0bffc010230f647f06960870b1fd6ee5902aca11909c5a61909c7781832303230
2d31302d355431333a34363a31332d30303a30301909c97818323032322d31302
d355431333a34363a31332d30303a30301909c6021909cc5029c7bafb81a2c616
0d3357d22911f5101909d26e706c656467652e312e322e332e341909cf59023d3
0820239308201dea003020102021439737 4f3fa812a0d37103b68c18481c501bc
7cfe300a06082a8648ce3d0403023073310b30090603550406 13024e4c310b300
906035504080c024e423110300e06035504070c0748656c6d6f6e643113301106
0355040a0c0a76616e64657273746f6b31143012060355040b0c0b636f6e73756
c74616e6379311a301806035504030c11726567697374726 1722e73746f6b2e6e
6c301e170d323030393039303734323033 5a170d323130393039303734323033 5
a3073310b30090603550406 13024e4c310b300906035504080c024e423110300e
06035504070c0748656c6d6f6e643113301106 0355040a0c0a76616e646572737
46f6b31143012060355040b0c0b636f6e73756c74616e6379311a301806035504
030c11726567697374726 1722e73746f6b2e6e6c3059301306072a8648ce3d020
106082a8648ce3d030107034200046a24a9e24be234d17ad47e760c94b5e1dc4e
e115ad89112ad4a6a64bb1bf68f9177d70e9109cb48c9b0dc7df2641199d0c2db
ff21ed584038ac83f4781be138ea350304e301d0603551d0e0416041425cd9371
b5a15f6d1ee8c37a5113be0b8f132cc2301f0603551d2304183016801425cd937
1b5a15f6d1ee8c37a5113be0b8f132cc2300c0603551d13040530030101ff300a
06082a8648ce3d040302034900304602210 0a66d9e24f9de08b7f0cf43c3c0ee5
7ccb660deae2e70cc61a1a2b33535025bba022100bffd746a99ebda0177fc6c37
95758af4a09f998ebc4a906249f07ac96596dc7558473045022100fc28be418e5
f25152590e872b4bbdbe334cd31d1ebb0<<a806e7a172cad5cff604022056ee41
4ddac438e7f51dda9ddf6ec6e31a78cdde6574717fe46dd3a7c60f5bb55847304
5022047b5314c72cbb2d1212e51198061167c79e1002874cd2665a5b643fa6436
3c30022100ce49ac309f760bd0e75660a7e29edee82f0251724c124150f5326b9
b2654927c

The representiation of signed_masa_voucher_request in CBOR diagnostic format is:

```
Diagnose(signed_masa_request_voucher) =

18([
h'A101382E',     # {"alg": -47}
{4:h'B86AE808F79AF17E5948CBDA731F158D04BD091C73F485F2409EAC08EE7D
DB5C'},
h'masa_request_voucher',
h'3045022047B5314C72CBB2D1212E51198061167C79E1002874CD2665A5B643F
A64363C30022100CE49AC309F760BD0E75660A7E29EDEE82F0251724C124150F5
326B9B2654927C'])

Diagnose(masa_request_voucher) =
{2501:
   {2503: "2020-10-5T13:46:13-00:00",
    2505: "2022-10-5T13:46:13-00:00",
    2508: h'29C7BAFB81A2C6160D3357D22911F510',
    2514: "pledge.1.2.3.4",
    2506:h'433D4E4C2C2053543D4E422C204C3D48656C6D6F6E642C
204F3D76616E64657273746F6B2C204F553D6D616E75666163747572696E672C2
0434E3D757569643A706C656467652E312E322E332E342C2073657269616C4E75
6D6265723D706C656467652E312E322E332E34',
    2510: h'request_voucher'}},
```

C.5. COSE signed voucher from MASA to Pledge via Registrar

The resulting voucher is created by the MASA and returned via the JRC to the Pledge. It is signed by the MASA's private key Appendix C.1.3 and can be verified by the pledge using the MASA's public key contained within the MASA certificate.

This is the raw binary signed_voucher, encoded in hexadecimal (with lf added):

```
   d28444a101382ea1045820ab59b0679fcf65d5223d4ce4266a27a9c7432702466
   ff5f3648e822a64d61b145902b0a1190993a71909957818323032302d31302d35
   5431333a34363a31342d30303a30301909977818323032322d31302d355431333
   a34363a31342d30303a30301909940319099a5029c7bafb81a2c6160d3357d229
   11f51019099e6e706c656467652e312e322e332e3419099b59023d30820239308
   201dea003020102021439737 4f3fa812a0d37103b68c18481c501bc7cfe300a06
   082a8648ce3d0403023073310b30090603550406 13024e4c310b3009060355040
   80c024e423110300e06035504070c0748656c6d6f6e643113301106035504 0a0c
   0a76616e64657273746f6b31143012060355040b0c0b636f6e73756c74616e637
   9311a301806035504030c11726567697374726172e73746f6b2e6e6c301e170d
   323030393039303734323033 5a170d323130393039303734323033 5a3073310b3
   009060355040613024e4c310b300906035504080c024e423110300e0603550407
   0c0748656c6d6f6e643113301106035504 0a0c0a76616e64657273746f6b31143
   012060355040b0c0b636f6e73756c74616e6379311a301806035504030c117265
   676973747261722e73746f6b2e6e6c3059301306072a8648ce3d020106082a864
   8ce3d030107034200046a24a9e24be234d17ad47e760c94b5e1dc4ee115ad8911
   2ad4a6a64bb1bf68f9177d70e9109cb48c9b0dc7df2641199d0c2dbff21ed5840
   38ac83f4781be138ea350304e301d0603551d0e0416041425cd9371b5a15f6d1e
   e8c37a5113be0b8f132cc2301f0603551d2304183016801425cd9371b5a15f6d1
   ee8c37a5113be0b8f132cc2300c0603551d13040530030101ff300a06082a8648
   ce3d040302034900304602210 0a66d9e24f9de08b7f0cf43c3c0ee57ccb660dea
   e2e70cc61a1a2b33535025bba022100bffd746a99ebda0177fc6c3795758af4a0
   9f998ebc4a906249f07ac96596dc7519099600584830460221 00d07cadc5c2836
   e7845d6d2e2652527386bd40258d20ab24b6bbce5515df915e9022100aba68a07
   b2295c4b49d53f73ea370ca66f761ad5d8d8c11c19a2d505729285cb

   The representiation of signed_voucher in CBOR diagnostic format is:

   Diagnose (signed_voucher) =
   18([
   h'A101382E',     # {"alg": -47}
   {4: h'AB59B0679FCF65D5223D4CE4266A27A9C7432702466FF5F3648E822A64D61
   B14'},
   h'voucher',
   h'3046022100D07CADC5C2836E7845D6D2E2652527386BD40258D20AB24B6BBCE
   5515DF915E9022100ABA68A07B2295C4B49D53F73EA370CA66F761AD5D8D8C11C
   19A2D505729285CB'])

   Diagnose (voucher) =

   {2451:
       {2453: "2020-10-5T13:46:14-00:00",
        2455: "2022-10-5T13:46:14-00:00",
        2452: 3,
        2458: h'29C7BAFB81A2C6160D3357D22911F510',
        2462: "pledge.1.2.3.4",
        2459: h'regis-cert-hex',
         2454: 0}}
```

Authors' Addresses

Michael Richardson
Sandelman Software Works

Email: mcr+ietf@sandelman.ca

Peter van der Stok
vanderstok consultancy

Email: consultancy@vanderstok.org

Panos Kampanakis
Cisco Systems

Email: pkampana@cisco.com