Workgroup: anima Working Group
Internet-Draft:
draft-ietf-anima-constrained-voucher-17
Updates: 8366, 8995 (if approved)
Published: 7 April 2022
Intended Status: Standards Track
Expires: 9 October 2022
Authors: M. Richardson Sandelman Software Works
P. van der Stok vanderstok consultancy
P. Kampanakis Cisco Systems
E. Dijk IoTconsultancy.nl

Constrained Bootstrapping Remote Secure Key Infrastructure (BRSKI)

Abstract

This document defines the Constrained Bootstrapping Remote Secure Key Infrastructure (Constrained BRSKI) protocol, which provides a solution for secure zero-touch bootstrapping of resource-constrained (IoT) devices into the network of a domain owner. This protocol is designed for constrained networks, which may have limited data throughput or may experience frequent packet loss. Constrained BRSKI is a variant of the BRSKI protocol, which uses an artifact signed by the device manufacturer called the "voucher" which enables a new device and the owner's network to mutually authenticate. While the BRSKI voucher is typically encoded in JSON, Constrained BRSKI defines a compact CBOR-encoded voucher. The BRSKI voucher is extended with new data types that allow for smaller voucher sizes. The Enrollment over Secure Transport (EST) protocol, used in BRSKI, is replaced with EST-over-CoAPS; and HTTPS used in BRSKI is replaced with CoAPS.

Status of This Memo

Table of Contents

1. Introduction

Secure enrollment of new nodes into constrained networks with constrained nodes presents unique challenges. As explained in [RFC7228], the networks are challenged and the nodes are constrained by energy, memory space, and code size.

The Bootstrapping Remote Secure Key Infrastructure (BRSKI) protocol described in [RFC8995] provides a solution for secure zero-touch (automated) bootstrap of new (unconfigured) devices. In it, new devices, such as IoT devices, are called "pledges", and equipped with a factory-installed Initial Device Identifier (IDevID) (see [ieee802-1AR]), are enrolled into a network.

The BRSKI solution described in [RFC8995] was designed to be modular, and this document describes a version scaled to the constraints of IoT deployments.

Therefore, this document defines a constrained version of the voucher artifact (described in [RFC8366]), along with a constrained version of BRSKI. This constrained-BRSKI protocol makes use of the constrained CoAP-based version of EST (EST-coaps from [I-D.ietf-ace-coap-est]) rather than the EST over HTTPS [RFC7030]. Constrained-BRSKI is itself scalable to multiple resource levels through the definition of optional functions. Appendix E illustrates this.

In BRSKI, the [RFC8366] voucher is by default serialized to JSON with a signature in CMS [RFC5652]. This document defines a new voucher serialization to CBOR [RFC8949] with a signature in COSE [I-D.ietf-cose-rfc8152bis-struct].

This COSE-signed CBOR-encoded voucher is transported using both secured CoAP and HTTPS. The CoAP connection (between Pledge and Registrar) is to be protected by either OSCORE+EDHOC [I-D.ietf-lake-edhoc] or DTLS (CoAPS). The HTTP connection (between Registrar and MASA) is to be protected using TLS (HTTPS).

This document specifies a constrained voucher-request artifact based on Section 3 of [RFC8995], and voucher(-request) transport over CoAP based on Section 3 of [RFC8995] and on [I-D.ietf-ace-coap-est].

The CBOR definitions for the constrained voucher format are defined using the mechanism described 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 a list of SID keys is still in its infancy, the table of SID values presented here MUST be considered normative rather than the output of the tool specified in [I-D.ietf-core-sid].

2. Terminology

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

The following terms from [RFC8995] are used identically as in that document: Domain CA, enrollment, IDevID, Join Proxy, LDevID, manufacturer, nonced, nonceless, PKIX.

The term Pledge Voucher Request, or acronym PVR, is introduced to refer to the voucher request between the pledge and the Registrar.

The term Registrar Voucher Request, or acronym RVR, is introduced to refer to the voucher request between the Registrar and the MASA.

In code examples, the string "<CODE BEGINS>" denotes the start of a code example and "<CODE ENDS>" the end of the code example. Four dots ("....") in a CBOR diagnostic notation byte string denotes a further sequence of bytes that is not shown for brevity.

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. Overview of Protocol

[RFC8366] provides for vouchers that assert proximity, authenticate the Registrar, and can offer varying levels of anti-replay protection.

The proximity proof provided for in [RFC8366], is an assertion that the Pledge and the Registrar are believed to be close together, from a network topology point of view. Like in [RFC8995], proximity is shown by making TLS connections between the Pledge and Registrar using IPv6 Link-Local addresses.

The TLS connection is used to make a Voucher Request. This request is verified by an agent of the Pledge's manufacturer, which then issues a voucher. The voucher provides an authorization statement from the manufacturer indicating that the Registrar is the intended owner of the device. The voucher refers to the Registrar through pinning of the Registrar's identity.

This document does not make any extensions to the semantic meaning of vouchers, only the encoding has been changed to optimize for constrained devices and networks. The two main parts of the BRSKI protocol are named separately in this document: BRSKI-EST for the protocol between Pledge and Registrar, and BRSKI-MASA for the protocol between the Registrar and the MASA.

Time-based vouchers are supported in this definition, but given that constrained devices are extremely unlikely to have accurate time, their use will be uncommon. Most Pledges using constrained vouchers will be online during enrollment and will use live nonces to provide anti-replay protection rather than expiry times.

[RFC8366] defines the voucher artifact, while the Voucher Request artifact was defined in [RFC8995]. 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 order that they occur in the protocol.

The constrained voucher request MUST be signed by the Pledge. It signs using the private key associated with its IDevID X.509 certificate, or if an IDevID is not available, then the private key associated with its manufacturer-installed raw public key (RPK). Section 12 provides additional details on PKIX-less operations.

The constrained voucher MUST be signed by the MASA.

For the constrained voucher request this document defines two distinct methods for the Pledge to identify the Registrar: using either the Registrar's X.509 certificate, or using a raw public key (RPK) of the Registrar.

For the constrained voucher both methods are supported to indicate (pin) a trusted domain identity: using either a pinned domain X.509 certificate, or a pinned raw public key (RPK).

The BRSKI architectures mandates that the MASA be aware of the capabilities of the pledge. This is not a drawback as the pledges are constructed by a manufacturer which also arranges for the MASA to be aware of the inventory of devices.

The MASA therefore knows if the pledge supports PKIX operations, PKIX format certificates, or if the pledge is limited to Raw Public Keys (RPK). Based upon this, the MASA can select which attributes to use in the voucher for certain operations, like the pinning of the Registrar identity. This is described in more detail in Section 9.2.3, Section 8 and Section 8.3 (for RPK specifically).

5. Updates to RFC8366 and RFC8995

This section details the ways in which this document updates other RFCs. The terminology for Updates is taken from [I-D.kuehlewind-update-tag].

This document Updates [RFC8366]. It Extends [RFC8366] by creating a new serialization format, and creates a mechanism to pin Raw Public Key (RPK).

This document Updates [RFC8995]. It Amends [RFC8995]

- by clarifying how pinning is done,
- adopts clearer explanation of the TLS Server Name Indicator (SNI), see Section 6.1 and Section 7.3
- clarifies when new trust anchors should be retrieved (Section 6.6.1),
- clarified what kinds of Extended Key Usage attributes are appropriate for each certificate (Section 7.3.1)

It Extends [RFC8995] as follows: * defines the CoAP version of the BRSKI protocol

- makes some messages optional if the results can be inferred from other validations (Section 6.6),

* provides the option to return trust anchors in a simpler format (Section 6.6.3)

* extends the BRSKI-MASA protocol to carry the new voucher-cose+cbor format.

6. BRSKI-EST Protocol

This section describes the constrained BRSKI extensions to EST-coaps [I-D.ietf-ace-coap-est] to transport the voucher between Registrar and Pledge (optionally via a Join Proxy) over CoAP. The extensions are targeting low-resource networks with small packets.

The constrained BRSKI-EST protocol described in this section is between the Pledge and the Registrar only.

6.1. Registrar and the Server Name Indicator (SNI)

A DTLS connection is established between the Pledge and the Registrar, similar to the TLS connection described in Section 5.1 of [RFC8995]. This may occur via a Join Proxy as described in Section 6.4. Regardless of the Join Proxy mechanism, the DTLS connection should operate identically.

The SNI issue described below affects [RFC8995] as well, and is reported in errata: https://www.rfc-editor.org/errata/eid6648

As the Registrar is discovered by IP address, and typically connected via a Join Proxy, the name of the Registrar is not known to the Pledge. The Pledge will not know what the hostname for the Registrar is, so it cannot do RFC6125 DNS-ID validation on the Registrar's certificate. Instead, it must do validation using the RFC8366 voucher.

As the Pledge does not know the name of the Registrar, the Pledge cannot put any reasonable value into the [RFC6066] Server Name Indicator (SNI). Threfore the Pledge SHOULD omit the SNI extension as per Section 9.2 of [RFC8446].

In some cases, particularly while testing BRSKI, a Pledge may be given the hostname of a particular Registrar to connect to directly. Such a bypass of the discovery process may result in the Pledge taking a different code branch to establish a DTLS connection, and may result in the SNI being inserted by a library. The Registrar MUST ignore any SNI seen.

A primary motivation for making the SNI ubiquitous in the public web is because it allows for multi-tenant hosting of HTTPS sites on a single (scarce) IPv4 address. This consideration does not apply to the server function in the Registrar because:

* it uses DTLS and CoAP, not HTTPS

* it typically uses IPv6, often [RFC4193] Unique Local Address, which are plentiful

* the server port number is typically discovered, so multiple tenants can be accomodated via unique port numbers.

As per Section 3.6.1 of [RFC7030], the Registrar certificate MUST have the Extended Key Usage (EKU) id-kp-cmcRA. This certificate is also used as a TLS Server Certificate, so it MUST also have the EKU id-kp-serverAuth.

6.2. TLS Client Certificates: IDevID authentication

As described in Section 5.1 of [RFC8995], the Pledge makes a connection to the Registrar using a TLS Client Certificate for authentication.

Subsequently the Pledge will send a Pledge Voucher Request (PVR).

As explained below in Section 8.1, the "x5bag" element may be used in the RVR to communicate identity of the Registrar to MASA. The Pledge SHOULD NOT use the x5bag attribute in this way in the PVR. A Registrar that processes a PVR with an x5bag attribute MUST ignore it, and MUST use only the TLS Client Certificate extension for authentication of the Pledge.

A situation where the Pledge MAY use the x5bag is for communication of certificate chains to the MASA. This would arise in some vendor-specific situations involving outsourcing of MASA functionality, or rekeying of the IDevID certification authority.

6.3. Discovery, URIs and Content Formats

To keep the protocol messages small the EST-coaps and constrained-BRSKI URIs are shorter than the respective EST and BRSKI URIs.

The EST-coaps server URIs differ from the EST URIs by replacing the scheme https by coaps and by specifying shorter resource path names. Below are some examples; the first two using a discovered short path name and the last one using the well-known URI of EST which requires no discovery.

```
coaps://server.example.com/est/<short-name>
coaps://server.example.com/e/<short-name>
coaps://server.example.com/.well-known/est/<short-name>
```

Similarly the constrained BRSKI server URIs differ from the BRSKI URIs by replacing the scheme https by coaps and by specifying shorter resource path names. Below are some examples; the first two using a discovered short path name and the last one using the well-known URI prefix which requires no discovery. This is the same "/.well-known/brski" prefix as defined in Section 5 of [RFC8995].

```
coaps://server.example.com/brski/<short-name>
coaps://server.example.com/b/<short-name>
coaps://server.example.com/.well-known/brski/<short-name>
```

Figure 5 in Section 3.2.2 of [RFC7030] enumerates the operations supported by EST, for which Table 1 in Section 5.1 of [I-D.ietf-ace-coap-est] enumerates the corresponding EST-coaps short path names. Similarly, Table 1 below provides the mapping from the supported BRSKI extension URI paths to the constrained-BRSKI URI paths.

BRSKI resource	constrained-BRSKI resource
/requestvoucher	/rv
/voucher_status	/vs
/enrollstatus	/es

Table 1: BRSKI URI paths mapping to Constrained BRSKI URI paths

Note that /requestvoucher indicated above occurs between the Pledge and Registrar (in scope of the BRSKI-EST protocol), but it also occurs between Registrar and MASA. However, as described in Section 6, this section and above table addresses only the BRSKI-EST protocol.

Pledges that wish to discover the available BRSKI bootstrap options/formats, or reduce the size of the CoAP headers by eliminating the "/.well-known/brski" path, can do a discovery operation using [RFC6690] Section 4 by sending a discovery query to the Registrar.

For example, if the Registrar supports a short BRSKI URL (/b) and supports the voucher format "application/voucher-cose+cbor" (TBD3), and status reporting in both CBOR and JSON formats:

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

RES: 2.05 Content
Content-Format: 40
Payload:
</b>;rt=brski,
</b/rv>;rt=brski.rv;ct=TBD3,
</b/vs>;rt=brski.vs;ct="50 60",
</b/es>;rt=brski.es;ct="50 60"
```

The Registrar is under no obligation to provide shorter URLs, and MAY respond to this query with only the "/.well-known/brski/<short-name>" resources for the short names as defined in Table 1.

Registrars that have implemented shorter URLs MUST also respond in equivalent ways to the corresponding "/.well-known/brski/<short-name>" URLs, and MUST NOT distinguish between them. In particular, a Pledge MAY use the longer and shorter URLs in any combination.

When responding to a discovery request for BRSKI resources, the server MAY in addition return the full resource paths and the content types which are supported by these resources as shown in above example. This is useful when multiple content types are specified for a particular resource on the server. The server responds with only the root path for the BRSKI resources (rt=brski, resource /b in above example) and no others in case the client queries for only rt=brski type resources. (So, a query for rt=brski, without the wildcard character.)

Without discovery, a longer well-known URL can only be used, such as:

```
REQ: GET /.well-known/brski/rv
```

while with discovery of shorter URLs, a request such as:

```
REQ: GET /b/rv
```

is possible.

The return of multiple content-types in the "ct" attribute allows the Pledge to choose the most appropriate one. Note that Content-Format TBD3 ("application/voucher-cose+cbor") is defined in this document.

Content-Format TBD3 MUST be supported by the Registrar for the /rv resource. If the "ct" attribute is not indicated for the /rv resource in the link format description, this implies that at least TBD3 is supported.

Note that this specification allows for voucher-cose+cbor format requests and vouchers to be transmitted over HTTPS, as well as for voucher-cms+json and other formats yet to be defined over CoAP. The burden for this flexibility is placed upon the Registrar. A Pledge on constrained hardware is expected to support a single format only.

The Pledge and MASA need to support one or more formats (at least TBD3) for the voucher and for the voucher request. The MASA needs to support all formats that the Pledge supports.

Section 10 details how the Pledge discovers the Registrar and Join Proxy in different deployment scenarios.

6.3.1. RFC8995 Telemetry Returns

[RFC8995] defines two telemetry returns from the Pledge which are sent to the Registrar. These are the BRSKI Status Telemetry [RFC8995], Section 5.7 and the Enrollment Status Telemetry [RFC8995], Section 5.9.4. These are two POST operations made the by Pledge at two key steps in the process.

[RFC8995] defines the content of these POST operations in CDDL, which are serialized as JSON. This document extends the list of acceptable formats to CBOR as well as JSON, using the rules from [RFC8610].

The existing JSON format is described as CoAP Content-Format 50 ("application/json"), and it MAY be supported. The new CBOR format described as CoAP Content-Format 60 ("application/cbor"), MUST be supported by the Registrar for both the /vs and /es resources.

6.4. Join Proxy options

[I-D.ietf-anima-constrained-join-proxy] specifies a constrained Join Proxy that is optionally placed between Pledge and Registrar. This includes methods for discovery of the Join Proxy by the Pledge and discovery of the Registrar by the Join Proxy.

6.5. Extensions to BRSKI

6.5.1. Discovery

The Pledge discovers an IP address and port number that connects to the Registrar (possibly via a Join Proxy), and it establishes a DTLS connection.

No further discovery of hosts or port numbers is required, but a pledge that can do more than one kind of enrollment (future work offers protocols other than [I-D.ietf-ace-coap-est]), then a pledge may need to use CoAP Discovery to determine what other protocols are available.

A Pledge that only supports the EST-coaps enrollment method SHOULD NOT use discovery for BRSKI resources. It is more efficient to just try the supported enrollment method via the well-known BRSKI/EST-coaps resources. This also avoids the Pledge doing any CoRE Link Format parsing, which is specified in [I-D.ietf-ace-coap-est], Section 4.1.

The Registrar MUST support all of the EST resources at their default ".well-known" locations (on the specified port) as well as any server-specific shorter form that might also be supported.

However, when discovery is being done by the Pledge, it is possible for the Registrar to return references to resources which are on different port numbers. The Registrar SHOULD NOT use different ports numbers by default, because a Pledge that is connected via a Join Proxy can only access a single UDP port. A Registrar configured to never use Join Proxies MAY be configured to use multiple port numbers. Therefore a Registrar MUST host all discoverable BRSKI resources on the same (UDP) server port that the Pledge's DTLS connection is using. Using the same UDP server port for all resources allows the Pledge to continue via the same DTLS connection which is more efficient.

6.5.2. CoAP responses

[RFC8995], Section 5 defines a number of HTTP response codes that the Registrar is to return when certain conditions occur.

The 401, 403, 404, 406 and 415 response codes map directly to CoAP codes 4.01, 4.03, 4.04, 4.06 and 4.15.

The 202 Retry process which occurs in the voucher request, is to be handled in the same way as Section 5.7 of [I-D.ietf-ace-coap-est] process for Delayed Responses.

6.6. Extensions to EST-coaps

This document extends [I-D.ietf-ace-coap-est], and it inherits the functions described in that document: specifically, the mandatory Simple (Re-)Enrollment (/sen and /sren) and Certification Authority certificates request (/crts). Support for CSR Attributes Request (/att) and server-side key generation (/skg, /skc) remains optional for the EST server.

Collecting the resource definitions from both [RFC8995], [RFC7030], and [I-D.ietf-ace-coap-est] results in the following shorter forms of URI paths for the commonly used resources:

BRSKI + EST	Constrained-BRSKI + EST	Well-known URI namespace
/requestvoucher	/rv	brski
/voucher_status	/vs	brski
/csrattrs	/att	est
/simpleenroll	/sen	est
/cacerts	/crts	est
/enrollstatus	/es	brski

BRSKI + EST	Constrained-BRSKI + EST	Well-known URI namespace
/simplereenroll	/sren	est

Table 2: BRSKI/EST URI paths mapping to Constrained BRSKI/EST short URI paths

6.6.1. Pledge Extensions

This section defines extensions to the BRSKI Pledge, which are applicable during the BRSKI bootstrap procedure. A Pledge which only supports the EST-coaps enrollment method, SHOULD NOT use discovery for EST-coaps resources, because it is more efficient to enroll (e.g. /sen) via the well-known EST resource on the current DTLS connection. This avoids an additional round-trip of packets and avoids the Pledge having to unnecessarily implement CoRE Link Format parsing.

A constrained Pledge SHOULD NOT perform the optional EST "CSR attributes request" (/att) to minimize network traffic. The Pledge selects which attributes to include in the CSR.

One or more Subject Distinguished Name fields MUST be included. If the Pledge has no specific information on what attributes/fields are desired in the CSR, it MUST use the Subject Distinguished Name fields from its IDevID unmodified. The Pledge can receive such information via the voucher (encoded in a vendor-specific way) or via some other, out-of-band means.

A constrained Pledge MAY use the following optimized EST-coaps procedure to minimize network traffic.

1. if the voucher, that validates the current Registrar, contains a single pinned domain CA certificate, the Pledge provisionally considers this certificate as the EST trust anchor, as if it were the result of "CA certificates request" (/crts) to the Registrar.

2. Using this CA certificate as trust anchor it proceeds with EST simple enrollment (/sen) to obtain its provisionally trusted LDevID certificate.

3. If the Pledge validates that the trust anchor CA was used to sign its LDevID certificate, the Pledge accepts the pinned domain CA certificate as the legitimate trust anchor CA for the Registrar's domain and accepts the associated LDevID certificate.

4. If the trust anchor CA was NOT used to sign its LDevID certificate, the Pledge MUST perform an actual "CA certificates request" (/crts) to the EST server to obtain the EST CA trust

anchor(s) since these can differ from the (temporary) pinned domain CA.

5. When doing this /crts request, the Pledge MAY use a CoAP Accept Option with value TBD287 ("application/pkix-cert") to limit the number of returned EST CA trust anchors to only one. A constrained Pledge MAY support only this format in a /crts response, per Section 5.3 of [I-D.ietf-ace-coap-est].

6. If the Pledge cannot obtain the single CA certificate or the finally validated CA certificate cannot be chained to the LDevID certificate, then the Pledge MUST abort the enrollment process and report the error using the enrollment status telemetry (/es).

Note that even though the Pledge may avoid performing any /crts request using the above EST-coaps procedure during bootstrap, it SHOULD support retrieval of the trust anchor CA periodically as detailed in the next section.

6.6.2. EST-client Extensions

This section defines extensions to EST-coaps clients, used after the BRSKI bootstrap procedure is completed. (Note that such client is not called "Pledge" in this section, since it is already enrolled into the domain.) A constrained EST-coaps client MAY support only the Content-Format TBD287 ("application/pkix-cert") in a /crts response, per Section 5.3 of [I-D.ietf-ace-coap-est]. In this case, it can only store one trust anchor of the domain.

An EST-coaps client that has an idea of the current time (internally, or via NTP) SHOULD consider the validity time of the trust anchor CA, and MAY begin requesting a new trust anchor CA using the /crts request when the CA has 50% of it's validity time (notAfter - notBefore) left. A client without access to the current time cannot decide if the trust anchor CA has expired, and SHOULD poll periodically for a new trust anchor using the /crts request at an interval of approximately 1 month. An EST-coaps server SHOULD include the CoAP ETag Option in every response to a /crts request, to enable clients to perform low-overhead validation whether their trust anchor CA is still valid. The EST-coaps client SHOULD store the ETag resulting from a /crts response in memory and SHOULD use this value in an ETag Option in its next GET /crts request.

The above-mentioned limitation that an EST-coaps client may support only one trust anchor CA is not an issue in case the domain trust anchor remains stable. However, special consideration is needed for cases where the domain trust anchor can change over time. Such a change may happen due to relocation of the client device to a new domain, or due to key update of the trust anchor as described in [RFC4210], Section 4.4.

From the client's viewpoint, a trust anchor change typically happens during EST re-enrollment: a change of domain CA requires all devices operating under the old domain CA to acquire a new LDevID issued by the new domain CA. A client's re-enrollment may be triggered by various events, such as an instruction to re-enroll sent by a domain entity, or an imminent expiry of its LDevID certificate. How the re-enrollment is explicitly triggered on the client by a domain entity, such as a commissioner or a Registrar, is out of scope of this specification.

The mechanism described in [RFC4210], Section 4.4 for Root CA key update requires four certificates: OldWithOld, OldWithNew, NewWithOld, and NewWithNew. The OldWithOld certificate is already stored in the EST client's trust store. The NewWithNew certificate will be distributed as the single certificate in a /crts response, during EST re-enrollment. Since the EST client can only accept a single certificate in a /crts response it implies that the EST client cannot obtain the certificates OldWithNew and NewWithOld in this way, to perform the complete verification of the new domain CA. Instead, the client only verifies the EST server (Registrar) using its old domain CA certificate in its trust store as detailed below, and based on this trust in the active and valid DTLS connection it automatically trusts the new (NewWithNew) domain CA certificate that the EST server provides in the /crts response.

In this manner, even during rollover of trust anchors, it is possible to have only a single trust anchor provided in a /crts response.

During the period of the certificate renewal, it is not possible to create new communication channels between devices with NewCA certificates devices with OldCA certificates. One option is that devices should avoid restarting existing DTLS or OSCORE connections during this interval that new certificates are being deployed. The recommended period for certificate renewal is 24 hours. For re-enrollment, the constrained EST-coaps client MUST support the following EST-coaps procedure, where optional re-enrollment to a new domain is under control of the Registrar:

1. The client connects with DTLS to the Registrar, and authenticates with its present domain certificate (LDevID certificate) as usual. The Registrar authenticates itself with its domain certificate that is trusted by the client, i.e. it chains to the single trust anchor that the client has stored. This is the "old" trust anchor, the one that will be eventually

replaced in case the Registrar decides to re-enroll the client into a new domain.

2. The client performs the simple re-enrollment request (/sren) and upon success it obtains a new LDevID.

3. The client verifies the new LDevID against its (single) existing domain trust anchor. If it chains successfully, this means the trust anchor did not change and the client MAY skip retrieving the current CA certificate using the "CA certificates request" (/crts). If it does not chain successfully, this means the trust anchor was changed/updated and the client then MUST retrieve the new domain trust anchor using the "CA certificates request" (/crts).

4. If the client retrieved a new trust anchor in step 3, then it MUST verify that the new trust anchor chains with the new LDevID certificate it obtained in step 2. If it chains successfully, the client stores both, accepts the new LDevID certificate and stops using it prior LDevID certificate. If it does not chain successfully, the client MUST NOT update its LDevID certificate, it MUST NOT update its (single) domain trust anchor, and the client MUST abort the enrollment process and report the error to the Registrar using enrollment status telemetry (/es).

Note that even though the EST-coaps client may skip the /crts request in step 3, it SHOULD support retrieval of the trust anchor CA periodically as detailed earlier in this section.

6.6.3. Registrar Extensions

A Registrar SHOULD host any discoverable EST-coaps resources on the same (UDP) server port that the Pledge's DTLS initial connection is using. This avoids the overhead of the Pledge reconnecting using DTLS, when it performs EST enrollment after the BRSKI voucher request.

The Content-Format 50 (application/json) MUST be supported and 60 (application/cbor) MUST be supported by the Registrar for the /vs and /es resources.

Content-Format TBD3 MUST be supported by the Registrar for the /rv resource.

When a Registrar receives a "CA certificates request" (/crts) request with a CoAP Accept Option with value TBD287 ("application/pkix-cert") it SHOULD return only the single CA certificate that is the envisioned or actual authority for the current, authenticated Pledge making the request.

If the Pledge included in its request an Accept Option for only the TBD287 ("application/pkix-cert") Content Format, but the domain has been configured to operate with multiple CA trust anchors only, then the Registrar returns a 4.06 Not Acceptable error to signal that the Pledge needs to use the Content Format 281 ("application/pkcs7-mime; smime-type=certs-only") to retrieve all the certificates.

If the current authenticated client is an EST-coaps client that was already enrolled in the domain, and the Registrar is configured to assign this client to a new domain CA trust anchor during the next EST re-enrollment procedure, then the Registrar MUST respond with the new domain CA certificate in case the client performs the "CA Certificates request" (/crts) with an Accept Option for TBD287 only. This signals the client that a new domain is assigned to it. The client follows the procedure as defined in Section 6.6.2.

6.7. DTLS handshake fragmentation Considerations

DTLS includes a mechanism to fragment the handshake messages. This is described in Section 4.4 of [I-D.ietf-tls-dtls13]. The protocol described in this document will often be used with a Join Proxy described in [I-D.ietf-anima-constrained-join-proxy]. The Join Proxy will need some overhead, while the maximum packet sized guaranteed on 802.15.4 networks is 1280 bytes. It is RECOMMENDED that a PMTU of 1024 bytes be assumed for the DTLS handshake. It is unlikely that any Packet Too Big indications [RFC4443] will be relayed by the Join Proxy.

During the operation of the constrained BRSKI-EST protocol, the CoAP Blockwise transfer mechanism will be used when message sizes exceed the PMTU. A Pledge/EST-client on a constrained network MUST use the (D)TLS maximum fragment length extension ("max_fragment_length") defined in Section 4 of [RFC6066] with the maximum fragment length set to a value of either 2^9 or 2^10.

7. BRSKI-MASA Protocol

This section describes extensions to and clarifications of the BRSKI-MASA protocol between Registrar and MASA.

7.1. Protocol and Formats

Section 5.4 of [RFC8995] describes a connection between the Registrar and the MASA as being a normal TLS connection using HTTPS. This document does not change that. The Registrar MUST use the format "application/voucher-cose+cbor" in its voucher request to MASA, when the Pledge uses this format in its reauqtes to the Registrar [RFC8995].

The MASA only needs to support formats for which there are Pledges that use that format.

The Registrar MUST use the same format for the RVR as the Pledge used for its PVR.

The Registrar indicates the voucher format it wants to receive from MASA using the HTTP Accept header. This format MUST be the same as the format of the PVR, so that the Pledge can parse it.

At the moment of writing the creation of coaps based MASAs is deemed unrealistic. The use of CoAP for the BRSKI-MASA connection can be the subject of another document. Some consideration was made to specify CoAP support for consistency, but:

* the Registrar is not expected to be so constrained that it cannot support HTTPS client connections.

* the technology and experience to build Internet-scale HTTPS responders (which the MASA is) is common, while the experience doing the same for CoAP is much less common.

* a Registrar is likely to provide onboarding services to both constrained and non-constrained devices. Such a Registrar would need to speak HTTPS anyway.

* a manufacturer is likely to offer both constrained and non-constrained devices, so there may in practice be no situation in which the MASA could be CoAP-only. Additionally, as the MASA is intended to be a function that can easily be oursourced to a third-party service provider, reducing the complexity would also seem to reduce the cost of that function.

* security-related considerations: see Section 14.6.

7.2. Registrar Voucher Request

If the PVR contains a proximity assertion, the Registrar MUST propagate this assertion into the RVR by including the "assertion" field with the value "proximity". This conforms to the example in Section 3.3 of [RFC8995] of carrying the assertion forward.

7.3. MASA and the Server Name Indicator (SNI)

A TLS/HTTPS connection is established between the Registrar and MASA.

Section 5.4 of [RFC8995] explains this process, and there are no externally visible changes. A MASA that supports the unconstrained voucher formats should be able to support constrained voucher formats equally well.

There is no requirement that a single MASA be used for both constrained and unconstrained voucher requests: the choice of MASA is determined by the id-mod-MASAURLExtn2016 extension contained in the IDevID.

The Registrar MUST do [RFC6125] DNS-ID checks on the contents of the certificate provided by the MASA.

In constrast to the Pledge/Registrar situation, the Registrar always knows the name of the MASA, and MUST always include an [RFC6066] Server Name Indicator. The SNI is optional in TLS1.2, but common. The SNI it considered mandatory with TLS1.3.

The presence of the SNI is needed by the MASA, in order for the MASA's server to host multiple tenants (for different customers).

The Registrar SHOULD use a TLS Client Certificate to authenticate to the MASA per Section 5.4.1 of [RFC8995]. If the certificate that the Registrar uses is marked as a id-kp-cmcRA certificate, via Extended Key Usage, then it MUST also have the id-kp-clientAuth EKU attribute set.

7.3.1. Registrar Certificate Requirement

In summary for typical Registrar use, where a single Registrar certificate is used, then the certificate MUST have EKU of: id-kp-cmcRA, id-kp-serverAuth, id-kp-clientAuth.

8. Pinning in Voucher Artifacts

The voucher is a statement by the MASA for use by the Pledge that provides the identity of the Pledge's owner. This section describes how the owner's identity is determined and how it is specified within the voucher.

8.1. Registrar Identity Selection and Encoding

Section 5.5 of [RFC8995] describes BRSKI policies for selection of the owner identity. It indicates some of the flexibility that is possible for the Registrar, and recommends the Registrar to include only certificates in the voucher request (CMS) signing structure that participate in the certificate chain that is to be pinned.

The MASA is expected to evaluate the certificates included by the Registrar in its voucher request, forming them into a chain with the Registrar's (signing) identity on one end. Then, it pins a certificate selected from the chain. For instance, for a domain with a two-level certification authority (see Figure 1), where the voucher-request has been signed by "Registrar", its signing structure includes two additional CA certificates. The arrows in the figure indicate the issuing of a certificate, i.e. author of (1) issued (2) and author of (2) issued (3).

```
.------------------.
|  domain CA (1)   |
|  trust anchor    |
'------------------'
          |
          v
   .------------.
   | domain (2) |
   | Sub-CA     |
   '------------'
          |
          |
          v
 .----------------.
 |   domain       |
 | Registrar (3)  |
 | EE certificate |
 '----------------'
```

Figure 1: Two-Level CA PKI

When the Registrar is using a COSE-signed constrained voucher request towards MASA, instead of a regular CMS-signed voucher request, the COSE_Sign1 object contains a protected and an unprotected header. The Registrar MUST place all the certificates needed to validate the signature chain from the Registrar on the RVR in an "x5bag" attribute in the unprotected header [I-D.ietf-cose-x509].

The "x5bag" attribute is very important as it provides the required signals from the Registrar to control what identity is pinned in the resulting voucher. This is explained in the next section.

8.2. MASA Pinning Policy

The MASA, having assembled and verified the chain in the signing structure of the voucher request needs to select a certificate to pin. (For the case that only the Registrar's End-Entity certificate is included, only this certificate can be selected and this section does not apply.) The BRSKI policy for pinning by the MASA as described in Section 5.5.2 of [RFC8995] leaves much flexibility to the manufacturer.

The present document adds the following rules to the MASA pinning policy to reduce the network load:

1. for a voucher containing a nonce, it SHOULD select the most specific (lowest-level) CA certificate in the chain.

2. for a nonceless voucher, it SHOULD select the least-specific (highest-level) CA certificate in the chain that is allowed under the MASA's policy for this specific domain.

The rationale for 1. is that in case of a voucher with nonce, the voucher is valid only in scope of the present DTLS connection between Pledge and Registrar anyway, so there is no benefit to pin a higher-level CA. By pinning the most specific CA the constrained Pledge can validate its DTLS connection using less crypto operations. The rationale for pinning a CA instead of the Registrar's End-Entity certificate directly is based on the following benefit on constrained networks: the pinned certificate in the voucher can in common cases be re-used as a Domain CA trust anchor during the EST enrollment and during the operational phase that follows after EST enrollment, as explained in Section 6.6.1.

The rationale for 2. follows from the flexible BRSKI trust model for, and purpose of, nonceless vouchers (Sections 5.5.* and 7.4.1 of [RFC8995]).

Refering to Figure 1 of a domain with a two-level certification authority, the most specific CA ("Sub-CA") is the identity that is pinned by MASA in a nonced voucher. A Registrar that wished to have only the Registrar's End-Entity certificate pinned would omit the "domain CA" and "Sub-CA" certificates from the voucher-request.

In case of a nonceless voucher, depending on the trust level, the MASA pins the "Registrar" certificate (low trust in customer), or the "Sub-CA" certificate (in case of medium trust, implying that any Registrar of that sub-domain is acceptable), or even the "domain CA" certificate (in case of high trust in the customer, and possibly a pre-agreed need of the customer to obtain flexible long-lived vouchers).

8.3. Pinning of Raw Public Keys

Specifically for constrained use cases, the pinning of the raw public key (RPK) of the Registrar is also supported in the constrained voucher, instead of an X.509 certificate. If an RPK is pinned it MUST be the RPK of the Registrar.

When the Pledge is known by MASA to support RPK but not X.509 certificates, the voucher produced by the MASA pins the RPK of the Registrar in either the "pinned-domain-pubk" or "pinned-domain-pubk-sha256" field of a voucher. This is described in more detail in Section 9.2.3. A Pledge that does not support X.509 certificates cannot use EST to enroll; it has to use another method for enrollment without certificates and the Registrar has to support this method also. It is possible that the Pledge will not enroll, but instead only a network join operation will occur (See [RFC9031]). How the Pledge discovers this method and details of the enrollment method are out of scope of this document.

When the Pledge is known by MASA to support PKIX format certificates, the "pinned-domain-cert" field present in a voucher typically pins a domain certificate. That can be either the End-Entity certificate of the Registrar, or the certificate of a domain CA of the Registrar's domain as specified in Section 8.2. However, if the Pledge is known to also support RPK pinning and the MASA intends to identify the Registrar in the voucher (not the CA), then MASA MUST pin the RPK (RPK3 in Figure 2) of the Registrar instead of the Registrar's End-Entity certificate to save space in the voucher.

```
 .------------.
 | pub-CA (1) |
 '------------'
       |
       v
 .------------.
 | sub-CA (2) |
 '------------'
       |
       v
.--------------.
| Registrar(3) |
|    RPK3      |
'--------------'
```

Figure 2: Raw Public Key pinning

8.4. Considerations for use of IDevID-Issuer

[RFC8366] and [RFC8995] defines the idevid-issuer attribute for voucher and voucher-request (respectively), but they summarily explain when to use it.

The use of idevid-issuer is provided so that the serial-number to which the issued voucher pertains can be relative to the entity that issued the devices' IDevID. In most cases there is a one to one relationship between the trust anchor that signs vouchers (and is trusted by the pledge), and the Certification Authority that signs the IDevID. In that case, the serial-number in the voucher must refer to the same device as the serial-number that is in IDevID certificate.

However, there situations where the one to one relationship may be broken. This occurs whenever a manufacturer has a common MASA, but different products (on different assembly lines) are produced with identical serial numbers. A system of serial numbers which is just a simple counter is a good example of this. A system of serial numbers where there is some prefix relating the product type does not fit into this, even if the lower digits are a counter.

It is not possible for the Pledge or the Registrar to know which situation applies. The question to be answered is whether or not to include the idevid-issuer in the PVR and the RVR. A second question arisews as to what the format of the idevid-issuer contents are.

Analysis of the situation shows that the pledge never needs to include the idevid-issuer in it's PVR, because the pledge's IDevID certificate is available to the Registrar, and the Authority Key Identifier is contained within that. The pledge therefore has no need to repeat this.

For the RVR, the Registrar has to examine the pledge's IDevID certificate to discover the serial number for the Registrar's Voucher Request (RVR). This is clear in Section 5.5 of [RFC8995]. That section also clarifies that the idevid-issuer is to be included in the RVR.

Concerning the Authority Key Identifier, [RFC8366] specifies that the entire object i.e. the extnValue OCTET STRING is to be included: comprising the AuthorityKeyIdentifier, SEQUENCE, Choice as well as the OCTET STRING that is the keyIdentifier.

9. Artifacts

This section describes for both the voucher request and the voucher first the abstract (tree) definition as explained in [RFC8340]. 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].

9.1. Voucher Request artifact

9.1.1. Tree Diagram

The following diagram is largely a duplicate of the contents of [RFC8366], with the addition of the fields proximity-registrar-pubk, proximity-registrar-pubk-sha256, proximity-registrar-cert, and prior-signed-voucher-request.

prior-signed-voucher-request is only used between the Registrar and the MASA. proximity-registrar-pubk or proximity-registrar-pubk-sha256 optionally replaces proximity-registrar-cert for the most constrained cases where RPK is used by the Pledge.

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

  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-pubk?         binary
       +-- proximity-registrar-pubk-sha256?  binary
       +-- proximity-registrar-cert?         binary
       +-- prior-signed-voucher-request?     binary
```

9.1.2. SID values

```
      SID Assigned to
--------- --------------------------------------------------
     2501 data /ietf-voucher-request-constrained: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-voucher-request-constrained:voucher/nonce
     2509 data .../pinned-domain-cert
     2510 data .../prior-signed-voucher-request
     2511 data .../proximity-registrar-cert
     2513 data .../proximity-registrar-pubk
     2512 data .../proximity-registrar-pubk-sha256
     2514 data .../serial-number

 WARNING, obsolete definitions
```

The "assertion" attribute is an enumerated type [RFC8366], and the current PYANG tooling does not document the valid values for this attribute. In the JSON serialization, the literal strings from the enumerated types are used so there is no ambiguity. In the CBOR serialization, a small integer is used. This following values are documented here, but the YANG module should be considered authoritative. No IANA registry is provided or necessary because the YANG module provides for extensions.

Integer	Assertion Type
0	verified
1	logged
2	proximity

Table 3: CBOR integers for the "assertion" attribute enum

9.1.3. YANG Module

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

```
<CODE BEGINS> file "ietf-voucher-request-constrained@2021-04-15.yang"

module ietf-voucher-request-constrained {
  yang-version 1.1;

  namespace
    "urn:ietf:params:xml:ns:yang:ietf-voucher-request-constrained";
  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 its 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
    {{I-D.ietf-anima-bootstrap-keyinfra}}

    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 either a pinned Raw Public Key of the Registrar, or by a
      pinned X.509 certificate of the Registrar or domain CA.

      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 "2021-04-15" {
      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-pubk {
            type binary;
            description
              "The proximity-registrar-pubk replaces
               the proximity-registrar-cert in constrained uses of
               the voucher-request.
               The proximity-registrar-pubk 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, but due to
      size is discouraged.";
}

leaf proximity-registrar-pubk-sha256 {
  type binary;
  description
    "The proximity-registrar-pubk-sha256
     is an alternative to both
     proximity-registrar-pubk 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
     specification which may define a new leaf for another
     hash type.";
}

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 field or one of its
     alternatives 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>
```

9.1.4. Example voucher request artifact

Below is a CBOR serialization of an example constrained voucher request from a Pledge to a Registrar, shown in CBOR diagnostic notation. The enum value of the assertion field is calculated to be 2 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'08C2BF36....B3D2B3',  / SID=2506, idevid-issuer /
    +10: h'30820275....82c35f',  / SID=2511, proximity-registrar-cert/
    +3 : true,                   / SID=2504, domain-cert
                                                   -revocation-checks/
    +6 : "2017-10-07T19:31:42Z"  / SID=2507, last-renewal-date /
  }
}
```

9.2. 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.

9.2.1. Tree Diagram

The following diagram is largely a duplicate of the contents of [RFC8366], with only the addition of the fields pinned-domain-pubk and pinned-domain-pubk-sha256.

```
module: ietf-voucher-constrained

  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-pubk?              binary
       +-- pinned-domain-pubk-sha256?       binary
```

9.2.2. SID values

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

 WARNING, obsolete definitions
```

The "assertion" enumerated attribute is numbered as per Section 9.1.2.

9.2.3. YANG Module

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

```
<CODE BEGINS> file "ietf-voucher-constrained@2021-04-15.yang"

module ietf-voucher-constrained {
  yang-version 1.1;

  namespace
    "urn:ietf:params:xml:ns:yang:ietf-voucher-constrained";
  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 its delegate
   (MASA) to securely assign one or more pledges to an 'owner',
   so that a pledge may establish 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 either a pinned Raw Public Key of the Registrar, or by a
   pinned X.509 certificate of the Registrar or domain CA.

   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 "2021-04-15" {
      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-pubk {
            type binary;
            description
              "The pinned-domain-pubk may replace the
               pinned-domain-cert in constrained uses of
               the voucher. The pinned-domain-pubk
               is the Raw Public Key of the Registrar.
               This field is encoded as a Subject Public Key Info block
               as specified in RFC7250, in 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-domain-pubk-sha256 {
          type binary;
          description
            "The pinned-domain-pubk-sha256 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
             specification which can define a new leaf for another
             hash type.";
        }
      }
    }
  }
}
```

<CODE ENDS>

9.2.4. Example voucher artifacts

Below the CBOR serialization of an example constrained voucher is shown in CBOR diagnostic 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'E40393B4....68A3',    / SID = 2456, idevid-issuer /
    +8 : h'30820275....C35F',    / SID = 2459, pinned-domain-cert/
    +3 : true,                   / SID = 2454, domain-cert /
                                 /              -revocation-checks /
    +6 : "2017-10-07T19:31:42Z"  / SID = 2457, last-renewal-date /
  }
}
```

9.3. Signing voucher and voucher-request artifacts with COSE

The COSE_Sign1 structure is discussed in Section 4.2 of [I-D.ietf-cose-rfc8152bis-struct]. 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 SHA2-256 using curve secp256r1 (aka prime256k1) is RECOMMENDED. Most current low power hardware has support for acceleration of this algorithm. Future hardware designs could omit this in favour of a newer algorithms. This is the ES256 keytype from Table 1 of [I-D.ietf-cose-rfc8152bis-algs]. Support for curve secp256k1 is OPTIONAL.

Support for EdDSA using Curve 25519 is RECOMMENDED in new designs if hardware support is available. This is keytype EDDSA (-8) from Table 2 of [I-D.ietf-cose-rfc8152bis-algs]. A "crv" parameter is necessary to specify the Curve, which from Table 18. The 'kty' field MUST be present, and it MUST be 'OKP'. (Table 17)

A transition towards EdDSA is occuring in the industry. Some hardware can accelerate only some algorithms with specific curves, other hardware can accelerate any curve, and still other kinds of hardware provide a tool kit for acceleration of any eliptic curve algorithm.

In general, the Pledge is expected to support only a single algorithm, while the Registrar, usually not constrained, is expected to support a wide variety of algorithms: both legacy ones and up-and-coming ones via regular software updates.

An example of the supported COSE_Sign1 object structure is shown in Figure 3.

```
18( / COSE_Sign1 /
  [
    h'A101382E',        / protected header encoding: {1: -47}     /
    {                   /       which means { "alg": ES256K }     /
      4 : h'7890A03F1234'  / 4 is the "kid" binary key identifier /
    },
    h'1234....5678', / voucher-request binary content (CBOR)      /
    h'4567....1234'  / voucher-request binary public signature    /
  ]
)
```

Figure 3: COSE_Sign1 example in CBOR diagnostic notation

The [COSE-registry] specifies the integers/encoding for the "alg" and "kid" fields in Figure 3. The "alg" field restricts the key usage for verification of this COSE object to a particular cryptographic algorithm.

The "kid" field is optionally present: it is an unprotected field that identifies the public key of the key pair that was used to sign this message. The value of the key identifier "kid" parameter is an example value. Usually a hash of the public key is used to identify the public key, but a device serial number may also be used. The choice of key identifier method is vendor-specific. If "kid" is not present, then a verifying party needs to use other context information to retrieve the right public key to verify the COSE_Sign1 object against. For example, this context information may be a unique serial number encoded in the binary content (CBOR) field.

A Registrar MAY use a "kid" parameter in its RVR to identify its signing key as used to sign the RVR. The method of generating this "kid" is vendor-specific and SHOULD be configurable in the Registrar to support commonly used methods. In order to support future business cases and supply chain integrations, a Registrar MUST be configurable, on a per-manufacturer basis, to be able to configure the "kid" to a particular value. Both binary and string values are to be supported, each being inserted using a CBOR bstr or tstr. By default, a Registrar does not include a "kid" parameter in its RVR since the signing key is already identified by the included signing certificates in the COSE "x5bag" structure.

A Pledge normally SHOULD NOT use a "kid" parameter in its PVR, because its signing key is already identified by the Pledge's unique serial number that is included in the PVR. Still, where needed the Pledge MAY use a "kid" parameter in its PVR to help the MASA identify the right public key to verify against. This can occur for example if a Pledge has multiple IDevID identities. A Registrar normally SHOULD ignore a "kid" parameter used in a received PVR, as this information is intended for the MASA. In other words, there is no need for the Registrar to verify the contents of this field, but it may include it in an audit log.

In Appendix C a binary COSE_Sign1 object is shown based on the voucher-request example of Section 9.1.4.

10. Deployment-specific Discovery Considerations

This section details how discovery is done in specific deployment scenarios.

10.1. 6TSCH Deployments

In 6TISCH networks, the Constrained Join Proxy (CoJP) mechanism is described in [RFC9031]. Such networks are expected to use a [I-D.ietf-lake-edhoc] to do key management. This is the subject of future work. The Enhanced Beacon has been extended in [RFC9032] to allow for discovery of the Join Proxy.

10.2. Generic networks using GRASP

[RFC8995] defines a mechanism for the Pledge to discover a Join Proxy by listening for [RFC8990] GRASP messages. This mechanism can be used on any network which does not have another more specific mechanism. This mechanism supports mesh networks, and can also be used over unencrypted WIFI.

10.3. Generic networks using mDNS

[RFC8995] also defines a non-normative mechanism for the Pledge to discover a Join Proxy by doing mDNS queries. This mechanism can be used on any network which does not have another recommended mechanism. This mechanism does not easily support mesh networks. It can be used over unencrypted WIFI.

10.4. Thread networks using Mesh Link Establishment (MLE)

Thread [Thread] is a wireless mesh network protocol based on 6LoWPAN [RFC6282] and other IETF protocols. In Thread, a new device discovers potential Thread networks and Thread routers to join by using the Mesh Link Establishment (MLE) [I-D.ietf-6lo-mesh-link-establishment] protocol. MLE uses the UDP port number 19788. The new device sends discovery requests on different IEEE 802.15.4 radio channels, to which routers (if any present) respond with a discovery response containing information about their respective network. Once a suitable router is selected the new device initiates a DTLS transport-layer secured connection to the network's commissioning application, over a link-local single radio hop to the selected Thread router. This link is not yet secured at the radio level: link-layer security will be set up once the new device is approved by the commissioning application to join the Thread network, and it gets provisioned with network access credentials.

The Thread router acts here as a Join Proxy. The MLE discovery response message contains UDP port information to signal the new device which port to use for its DTLS connection.

10.5. Non-mesh networks using CoAP Discovery

On unencrypted constrained networks such as 802.15.4, CoAP discover may be done using the mechanism detailed in [I-D.ietf-ace-coap-est] section 5.1.

11. Design Considerations

The design considerations for the CBOR encoding of vouchers are much the same as for JSON vouchers in [RFC8366]. One key difference is that the names of the leaves in the YANG definition do not affect the size of the resulting CBOR, as the SID translation process assigns integers to the names.

Any POST request to the Registrar with resource /vs or /es returns a 2.04 Changed response with empty payload. The client should be aware that the server may use a piggybacked CoAP response (ACK, 2.04) but may also respond with a separate CoAP response, i.e. first an (ACK, 0.0) that is an acknowledgement of the request reception followed by a (CON, 2.04) response in a separate CoAP message.

12. Raw Public Key Use Considerations

This section explains techniques to reduce the number of bytes that are sent over the wire during the BRSKI bootstrap. The use of a raw public key (RPK) in the pinning process can significantly reduce the number of bytes and round trips, but it comes with a few significant operational limitations.

12.1. The Registrar Trust Anchor

When the Pledge first connects to the Registrar, the connection to the Registrar is provisional, as explained in Section 5.6.2 of [RFC8995]. The Registrar provides its public key in a TLSServerCertificate, and the Pledge uses that to validate that integrity of the (D)TLS connection, but it does not validate the identity of the provided certificate.

As the TLSServerCertificate object is never verified directly by the pledge, sending it can be considered superfluous. Instead of using a (TLSServer)Certificate of type X509 (see section 4.4.2 of [RFC8446]), a RawPublicKey object is used.

A Registrar operating in a mixed environment can determine whether to send a Certificate or a Raw Public key: this is determined by the pledge including the server_certificate_type of RawPublicKey. This is shown in section 5 of [RFC7250].

The Pledge continues to send a client_certificate_type of X509, so that the Registrar can properly identify the pledge and distill the MASA URI information from its certificate.

12.2. The Pledge Voucher Request

The Pledge puts the Registrar's public key into the proximity-registrar-pubk field of the voucher-request. (The proximity-registrar-pubk-sha256 can also be used if the 32-bytes of a SHA256 hash turns out to be smaller than a typical ECDSA key.)

As the format of the pubk field is identical to the TLS Certificate RawPublicKey, no manipulation at all is needed to insert this into a voucher-request.

12.3. The Voucher Response

A returned voucher will have a pinned-domain-subk field with the identical key as was found in the proximity-registrar-pubk field above, as well as in the TLS connection.

Validation of this key by the pledge is what takes the DTLS connection out of the provisional state see Section 5.6.2 of [RFC8995].

The voucher needs to be validated first. The Pledge needs to have a public key to validate the signature from the MASA on the voucher.

In certain cases, the MASA's public key counterpart of the (private) signing key is already installed in the Pledge at manufacturing time. In other cases, if the MASA signing key is based upon a PKI (see [I-D.richardson-anima-masa-considerations] Section 2.3), then a certificate chain may need to be included with the voucher in order for the pledge to validate the signature. In CMS signed artifacts, the CMS structure has a place for such certificates.

In the COSE-signed Constrained Vouchers described in this document, the x5bag attribute from [I-D.ietf-cose-x509] is to be used for this.

13. Use of constrained vouchers with HTTPS

This specification contains two extensions to [RFC8995]: a constrained voucher format (COSE), and a constrained transfer protocol (CoAP).

On constrained networks with constrained devices, it make senses to use both together. However, this document does not mandate that this be the only way.

A given constrained device design and software may be re-used for multiple device models, such as a model having only an IEEE 802.15.4 radio, or a model having only an IEEE 802.11 (Wi-Fi) radio, or a model having both these radios. A manufacturer of such device models may wish to have code only for the use of the constrained voucher format (COSE), and use it on all supported radios including the IEEE 802.11 radio. For this radio, the software stack to support HTTP/TLS may be already integrated into the radio module hence it is attractive for the manufacturer to reuse this. This type of approach is supported by this document. In the case that HTTPS is used, the normal [RFC8995] resource names are used, together with the media types described in this document.

Other combinations are possible, but they are not enumerated here. New work such as [I-D.ietf-anima-jws-voucher] provides new formats that may be useable over a number of different transports. In general, sending larger payloads over constrained networks makes less sense, while sending smaller payloads over unconstrained networks is perfectly acceptable.

The Pledge will in most cases support a single voucher format, which it uses without negotiation i.e. without discovery of formats supported. The Registrar, being unconstrained, is expected to support all voucher formats. There will be cases where a Registrar does not support a new format that a new Pledge uses, and this is an unfortunate situation that will result in lack of interoperation.

The responsability for supporting new formats is on the Registrar.

14. Security Considerations

14.1. Duplicate serial-numbers

In the absense of correct use of idevid-issuer by the Registrar as detailed in Section 8.4, it would be possible for a malicious Registrar to use an unauthorized voucher for a device. This would apply only to the case where a Manufacturer Authorized Signing Authority (MASA) is trusted by different products from the same manufacturer, and the manufacturer has duplicated serial numbers as a result of a merge, acquisition or mis-management.

For example, imagine the same manufacturer makes light bulbs as well as gas centrifuges, and said manufacturer does not uniquely allocate product serial numbers. This attack only works for nonceless vouchers. The attacker has obtained a light bulb which happens to have the same serial-number as a gas centrofuge which it wishes to obtain access. The attacker performs a normal BRSKI onboarding for the light bulb, but then uses the resulting voucher to onboard the gas centrofuge. The attack requires that the gas centrofuge be returned to a state where it is willing to perform a new onboarding operation.

This attack is prevented by the mechanism of having the Registrar include the idevid-issuer in the RVR, and the MASA including it in the resulting voucher. The idevid-issuer is not included by default: a MASA needs to be aware if there are parts of the organization which duplicates serial numbers, and if so, include it.

14.2. IDevID security in Pledge

The security of this protocol depends upon the Pledge identifying itself to the Registrar using it's manufacturer installed certificate: the IDevID certificate. Associated with this certificate is the IDevID private key, known only to the Pledge. Disclosure of this private key to an attacker would permit the attacker to impersonate the Pledge towards the Registrar, probably gaining access credentials to that Registrar's network.

If the IDevID private key disclosure is known to the manufacturer, there is little recourse other than recall of the relevant part numbers. The process for communicating this recall would be within the BRSKI-MASA protocol. Neither this specification nor [RFC8995] provides for consultation of a Certification Revocation List (CRL) or Open Certificate Status Protocol (OCSP) by a Registrar when evaluating an IDevID certificate. However, the BRSKI-MASA protocol submits the IDevID from the Registrar to the manufacturer's MASA and a manufacturer would have an opportunity to decline to issue a voucher for a device which they believe has become compromised.

It may be difficult for a manufacturer to determine when an IDevID private key has been disclosed. Two situations present themselves: in the first situation a compromised private key might be reused in a counterfeit device, which is sold to another customer. This would present itself as an onboarding of the same device in two different networks. The manufacturer may become suspicious seeing two voucher requests for the same device from different Registrars. Such activity could be indistinguishable from a device which has been resold from one operator to another, or re-deployed by an operator from one location to another.

In the second situation, an attacker having compromised the IDevID private key of a device might then install malware into the same device and attempt to return it to service. The device, now blank, would go through a second onboarding process with the original Registrar. Such a Registrar could notice that the device has been "factory reset" and alert the operator to this situation. One remedy against the presence of malware is through the use of Remote Attestation such as described in [I-D.ietf-rats-architecture].

Future work will need to specify a background-check Attestation flow as part of the voucher-request/voucher-response process. Attestation may still require access to a private key (e.g. IDevID private key) in order to sign Evidence, so a primary goal should be to keep any private key safe within the Pledge.

In larger, more expensive, systems there is budget (power, space, and bill of materials) to include more specific defenses for a private key. For instance, this includes putting the IDevID private key in a Trusted Platform Module (TPM), or use of Trusted Execution Environments (TEE) for access to the key. On smaller IoT devices, the cost and power budget for an extra part is often prohibitive.

It is becoming more and more common for CPUs to have an internal set of one-time fuses that can be programmed (often they are "burnt" by a laser) at the factory. This section of memory is only accessible in some priviledged CPU state. The use of this kind of CPU is appropriate as it provides significant resistance against key disclosure even when the device can be disassembled by an attacker.

In a number of industry verticals, there is increasing concern about counterfeit parts. These may be look-alike parts created in a different factory, or parts which are created in the same factory during an illegal night-shift, but which are not subject to the appropriate level of quality control. The use of a manufacturer-signed IDevID certificate provides for discovery of the pedigree of each part, and this often justifies the cost of the security measures associated with storing the private key.

14.3. Security of CoAP and UDP protocols

Section 7.1 explains that no CoAPS version of the BRSKI-MASA protocol is proposed. The connection from the Registrar to the MASA continues to be HTTPS as in [RFC8995]. This has been done to simplify the MASA deployment for the manufacturer, because no new protocol needs to be enabled on the server.

The use of UDP protocols across the open Internet is sometimes fraught with security challenges. Denial-of-service attacks against UDP based protocols are trivial as there is no three-way handshake as done for TCP. The three-way handshake of TCP guarantees that the node sending the connection request is reachable using the origin IP address. While DTLS contains an option to do a stateless challenge -- a process actually stronger than that done by TCP -- it is not yet common for this mechanism to be available in hardware at multigigabit speeds. It is for this reason that this document defines using HTTPS for the Registrar to MASA connection.

14.4. Registrar Certificate may be self-signed

The provisional (D)TLS connection formed by the Pledge with the Registrar does not authenticate the Registrar's identity. This Registrar's identity is validated by the [RFC8366] voucher that is issued by the MASA, signed with an anchor that was built-in to the Pledge.

The Registrar may therefore use any certificate, including a self-signed one. The only restrictions on the certificate is that it MUST have EKU bits set as detailed in Section 7.3.1.

14.5. Use of RPK alternatives to proximity-registrar-cert

In Section 9.1 two compact alternative fields for proximity-registrar-cert are defined that include an RPK: proximity-registrar-pubk and proximity-registrar-pubk-sha256. The Pledge can use these fields in its PVR to identify the Registrar based on its public key only. Since the full certificate of the proximate Registrar is not included, use of these fields by a Pledge implies that a Registrar could insert another certificate with the same public key identity into the RVR. For example, an older or a newer version of its certificate. The MASA will not be able to detect such act by the Registrar. But since any 'other' certificate the Registrar could insert in this way still encodes its identity the additional risk of using the RPK alternatives is neglible.

When a Registrar sees a PVR that uses one of proximity-registrar-pubk or proximity-registrar-pubk-sha256 fields, this implies the Registrar must include the certificate identified by these fields into its RVR. Otherwise, the MASA is unable to verify proximity. This requirement is already implied by the "MUST" requirement in Section 8.1.

14.6. MASA support of CoAPS

The use of CoAP for the BRSKI-MASA connection is not in scope of the current document. The following security considerations have led to this choice of scope:

* the technology and experience to build secure Internet-scale HTTPS responders (which the MASA is) is common, while the experience in doing the same for CoAP is much less common.

* in many enterprise networks, outgoing UDP connections are often treated as suspicious, which could effectively block CoAP connections for some firewall configurations.

* reducing the complexity of MASA (i.e. less protocols supported) would also reduce its potential attack surface, which is relevant

since the MASA is 24/7 exposed on the Internet and accepting (untrusted) incoming connections.

15. IANA Considerations

15.1. Resource Type Registry

Additions to the sub-registry "Resource Type Link Target Attribute Values", within the "CoRE Parameters" IANA registry are specified below.

Reference: [This RFC]

Attribute	Description
brski	Root path of Bootstrapping Remote Secure Key Infrastructure (BRSKI) resources
brski.rv	BRSKI request voucher resource
brski.vs	BRSKI voucher status telemetry resource
brski.es	BRSKI enrollment status telemetry resource

Table 4: Resource Type (rt) link target attribute values for IANA registration

15.2. 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-voucher-constrained
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-voucher-request-constrained
Registrant Contact: The ANIMA WG of the IETF.
XML: N/A, the requested URI is an XML namespace.
```

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

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

15.4. The RFC SID range assignment sub-registry

```
------------ ------ --------------------------- ------------
Entry-point | Size | Module name               | RFC Number
------------ ------ --------------------------- ------------
2450         50     ietf-voucher-constrained    [This RFC]
2500         50     ietf-voucher-request        [This RFC]
                                -constrained
----------- ------  --------------------------- ------------
```

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

IANA: please update the names in the Registry to match these revised names, if they have not already been revised.

15.5. Media Types Registry

This section registers the 'application/voucher-cose+cbor' in the IANA "Media Types" registry. This media type is used to indicate that the content is a CBOR voucher or voucher request signed with a COSE_Sign1 structure [I-D.ietf-cose-rfc8152bis-struct].

15.5.1. application/voucher-cose+cbor

Type name: application
Subtype name: voucher-cose+cbor
Required parameters: N/A
Optional parameters: N/A
Encoding considerations: binary (CBOR)
Security considerations: Security Considerations of [This RFC].
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 onboarding systems
Fragment identifier considerations: The syntax and semantics of
  fragment identifiers specified for application/voucher-cose+cbor
  are as specified for application/cbor. (At publication of this
  document, there is no fragment identification syntax defined for
  application/cbor.)
Additional information:
  Deprecated alias names for this type: N/A
  Magic number(s): N/A
  File extension(s): .vch
  Macintosh file type code(s): N/A
Person & email address to contact for further information: IETF
  ANIMA Working Group (anima@ietf.org) or IETF Operations and
  Management Area Working Group (opsawg@ietf.org)
Intended usage: COMMON
Restrictions on usage: N/A
Author: ANIMA WG
Change controller: IETF
Provisional registration? (standards tree only): NO

15.6. CoAP Content-Format Registry

One addition to the sub-registry "CoAP Content-Formats", within the "CoRE Parameters" registry is needed for a new content-format. It can be registered in the Expert Review range (0-255) or the IETF Review range (256-9999).

Media type	Encoding	ID	Reference
application/voucher-cose+cbor	-	TBD3	[This RFC]

15.7. Update to BRSKI Parameters Registry

This section updates the BRSKI Well-Known URIs sub-registry of the IANA Bootstrapping Remote Secure Key Infrastructures (BRSKI) Parameters Registry by adding a new column "Short URI". The contents of this field MUST be specified for any newly registered URI as follows:

Short URI: A short name for the "URI" resource that can be used by a Constrained BRSKI Pledge in a CoAP request to the Registrar. In case the "URI" resource is only used between Registrar and MASA, the value "--" is registered denoting that a short name is not applicable.

The initial contents of the sub-registry including the new column are as follows:

URI	Short URI	Description	Reference
requestvoucher	rv	Request voucher: Pledge to Registrar, and Registrar to MASA	[RFC8995], [This RFC]
voucher_status	vs	Voucher status telemetry: Pledge to Registrar	[RFC8995], [This RFC]
requestauditlog	--	Request audit log: Registrar to MASA	[RFC8995]
enrollstatus	es	Enrollment status telemetry: Pledge to Registrar	[RFC8995], [This RFC]

Table 5: Update of the BRSKI Well-Known URI Sub-Registry

16. Acknowledgements

We are very grateful to Jim Schaad for explaining COSE and CMS choices. Also thanks to Jim Schaad for correcting earlier versions 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 its first users.

Daniel Franke and Henk Birkholtz provided review feedback.

The BRSKI design team has met on many Thursdays for document review. It includes: Aurelio Schellenbaum, David von Oheimb Steffen Fries, Thomas Werner, Toerless Eckert,

17. Changelog

-11 to -16 (For change details see GitHub issues https://github.com/anima-wg/constrained-voucher/issues)

-10 Design considerations extended Examples made consistent

-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

18. References

18.1. Normative References

[I-D.ietf-ace-coap-est] Stok, P. V. D., Kampanakis, P., Richardson, M. C., and S. Raza, "EST over secure CoAP (EST-coaps)", Work in Progress, Internet-Draft, draft-ietf-ace-coap-est-18, 6 January 2020, <https://www.ietf.org/archive/id/draft-ietf-ace-coap-est-18.txt>.

[I-D.ietf-core-sid] Veillette, M., Pelov, A., Petrov, I., Bormann, C., and M. Richardson, "YANG Schema Item iDentifier (YANG SID)", Work in Progress, Internet-Draft, draft-ietf-core-sid-18, 18 November 2021, <https://www.ietf.org/archive/id/draft-ietf-core-sid-18.txt>.

[I-D.ietf-core-yang-cbor] Veillette, M., Petrov, I., Pelov, A., Bormann, C., and M. Richardson, "CBOR Encoding of Data Modeled with YANG", Work in Progress, Internet-Draft, draft-ietf-core-yang-cbor-19, 20 March 2022, <https://www.ietf.org/archive/id/draft-ietf-core-yang-cbor-19.txt>.

[I-D.ietf-cose-rfc8152bis-algs]
Schaad, J., "CBOR Object Signing and Encryption (COSE): Initial Algorithms", Work in Progress, Internet-Draft, draft-ietf-cose-rfc8152bis-algs-12, 24 September 2020,

<https://www.ietf.org/archive/id/draft-ietf-cose-rfc8152bis-algs-12.txt>.

[I-D.ietf-cose-rfc8152bis-struct]
Schaad, J., "CBOR Object Signing and Encryption (COSE): Structures and Process", Work in Progress, Internet-Draft, draft-ietf-cose-rfc8152bis-struct-15, 1 February 2021, <https://www.ietf.org/archive/id/draft-ietf-cose-rfc8152bis-struct-15.txt>.

[I-D.ietf-cose-x509] Schaad, J., "CBOR Object Signing and Encryption (COSE): Header parameters for carrying and referencing X.509 certificates", Work in Progress, Internet-Draft, draft-ietf-cose-x509-08, 14 December 2020, <https://www.ietf.org/archive/id/draft-ietf-cose-x509-08.txt>.

[I-D.ietf-tls-dtls13] Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", Work in Progress, Internet-Draft, draft-ietf-tls-dtls13-43, 30 April 2021, <https://www.ietf.org/archive/id/draft-ietf-tls-dtls13-43.txt>.

[ieee802-1AR] IEEE Standard, "IEEE 802.1AR Secure Device Identifier", 2009, <http://standards.ieee.org/findstds/standard/802.1AR-2009.html>.

[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>.

[RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, DOI 10.17487/RFC4193, October 2005, <https://www.rfc-editor.org/info/rfc4193>.

[RFC4210] Adams, C., Farrell, S., Kause, T., and T. Mononen, "Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)", RFC 4210, DOI 10.17487/RFC4210, September 2005, <https://www.rfc-editor.org/info/rfc4210>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[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>.

[RFC6066] Eastlake 3rd, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, DOI 10.17487/RFC6066, January 2011, <https://www.rfc-editor.org/info/rfc6066>.

[RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <https://www.rfc-editor.org/info/rfc6125>.

[RFC7250] Wouters, P., Ed., Tschofenig, H., Ed., Gilmore, J., Weiler, S., and T. Kivinen, "Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", RFC 7250, DOI 10.17487/RFC7250, June 2014, <https://www.rfc-editor.org/info/rfc7250>.

[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>.

[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>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <https://www.rfc-editor.org/info/rfc8610>.

[RFC8949]
Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/info/rfc8949>.

[RFC8995] Pritikin, M., Richardson, M., Eckert, T., Behringer, M., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructure (BRSKI)", RFC 8995, DOI 10.17487/RFC8995, May 2021, <https://www.rfc-editor.org/info/rfc8995>.

[RFC9031] Vučinić, M., Ed., Simon, J., Pister, K., and M. Richardson, "Constrained Join Protocol (CoJP) for 6TiSCH", RFC 9031, DOI 10.17487/RFC9031, May 2021, <https://www.rfc-editor.org/info/rfc9031>.

[RFC9032] Dujovne, D., Ed. and M. Richardson, "Encapsulation of 6TiSCH Join and Enrollment Information Elements", RFC 9032, DOI 10.17487/RFC9032, May 2021, <https://www.rfc-editor.org/info/rfc9032>.

18.2. Informative References

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

[I-D.ietf-6lo-mesh-link-establishment]
Kelsey, R., "Mesh Link Establishment", Work in Progress, Internet-Draft, draft-ietf-6lo-mesh-link-establishment-00, 1 December 2015, <https://www.ietf.org/archive/id/draft-ietf-6lo-mesh-link-establishment-00.txt>.

[I-D.ietf-anima-constrained-join-proxy] Richardson, M., Stok, P. V. D., and P. Kampanakis, "Constrained Join Proxy for Bootstrapping Protocols", Work in Progress, Internet-Draft, draft-ietf-anima-constrained-join-proxy-09, 25 March 2022, <https://www.ietf.org/archive/id/draft-ietf-anima-constrained-join-proxy-09.txt>.

[I-D.ietf-anima-jws-voucher] Richardson, M. and T. Werner, "JWS signed Voucher Artifacts for Bootstrapping Protocols", Work in Progress, Internet-Draft, draft-ietf-anima-jws-voucher-03, 7 March 2022, <https://www.ietf.org/archive/id/draft-ietf-anima-jws-voucher-03.txt>.

[I-D.ietf-lake-edhoc] Selander, G., Mattsson, J. P., and F. Palombini, "Ephemeral Diffie-Hellman Over COSE (EDHOC)", Work in Progress, Internet-Draft, draft-ietf-lake-edhoc-12, 20 October 2021, <https://www.ietf.org/archive/id/draft-ietf-lake-edhoc-12.txt>.

[I-D.ietf-rats-architecture] Birkholz, H., Thaler, D., Richardson, M., Smith, N., and W. Pan, "Remote Attestation Procedures Architecture", Work in Progress, Internet-Draft, draft-ietf-rats-architecture-15, 8 February 2022, <https://www.ietf.org/archive/id/draft-ietf-rats-architecture-15.txt>.

[I-D.kuehlewind-update-tag] Kuehlewind, M. and S. Krishnan, "Definition of new tags for relations between RFCs", Work in Progress, Internet-Draft, draft-kuehlewind-update-tag-04, 12 July 2021, <https://www.ietf.org/archive/id/draft-kuehlewind-update-tag-04.txt>.

[I-D.richardson-anima-masa-considerations] Richardson, M. and W. Pan, "Operatonal Considerations for Voucher infrastructure for BRSKI MASA", Work in Progress, Internet-Draft, draft-richardson-anima-masa-considerations-06, 13 November 2021, <https://www.ietf.org/archive/id/draft-richardson-anima-masa-considerations-06.txt>.

[RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <https://www.rfc-editor.org/info/rfc4443>.

[RFC6282] Hui, J., Ed. and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks", RFC 6282, DOI 10.17487/RFC6282, September 2011, <https://www.rfc-editor.org/info/rfc6282>.

[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>.

[RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, DOI 10.17487/

RFC7228, May 2014, <https://www.rfc-editor.org/info/rfc7228>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8990] Bormann, C., Carpenter, B., Ed., and B. Liu, Ed., "GeneRic Autonomic Signaling Protocol (GRASP)", RFC 8990, DOI 10.17487/RFC8990, May 2021, <https://www.rfc-editor.org/info/rfc8990>.

[Thread] Thread Group, Inc, "Thread support page, White Papers", November 2021, <https://www.threadgroup.org/support#Whitepapers>.

Appendix A. Library Support for BRSKI

For the implementation of BRSKI, the use of a software library to manipulate certificates and use crypto algorithms is often beneficial. Two C-based examples are OpenSSL and mbedtls. Others more targeted to specific platforms or languages exist. It is important to realize that the library interfaces differ significantly between libraries.

Libraries do not support all known crypto algorithms. Before deciding on a library, it is important to look at their supported crypto algorithms and the roadmap for future support. Apart from availability, the library footprint, and the required execution cycles should be investigated beforehand.

The handling of certificates usually includes the checking of a certificate chain. In some libraries, chains are constructed and verified on the basis of a set of certificates, the trust anchor (usually self signed root CA), and the target certificate. In other libraries, the chain must be constructed beforehand and obey order criteria. Verification always includes the checking of the signatures. Less frequent is the checking the validity of the dates or checking the existence of a revoked certificate in the chain against a set of revoked certificates. Checking the chain on the consistency of the certificate extensions which specify the use of the certificate usually needs to be programmed explicitly.

A libary can be used to construct a (D)TLS connection. It is useful to realize that differences beetween (D)TLS implementations will occur due to the differences in the certicate checks supported by the library. On top of that, checks between client and server certificates enforced by (D)TLS are not always helpful for a BRSKI implementation. For example, the certificates of Pledge and Registrar are usually not related when the BRSKI protocol is started. It must be verified that checks on the relation between client and server certificates do not hamper a succeful DTLS connection establishment.

A.1. OpensSSL

From openssl's apps/verify.c :

<CODE BEGINS>

```
X509 *x = NULL;
int i = 0, ret = 0;
X509_STORE_CTX *csc;
STACK_OF(X509) *chain = NULL;
int num_untrusted;

x = load_cert(file, "certificate file");
if (x == NULL)
    goto end;

csc = X509_STORE_CTX_new();
if (csc == NULL) {
    BIO_printf(bio_err, "error %s: X.509 store context"
               "allocation failed\n",
               (file == NULL) ? "stdin" : file);
    goto end;
}

X509_STORE_set_flags(ctx, vflags);
if (!X509_STORE_CTX_init(csc, ctx, x, uchain)) {
    X509_STORE_CTX_free(csc);
    BIO_printf(bio_err,
               "error %s: X.509 store context"
               "initialization failed\n",
               (file == NULL) ? "stdin" : file);
    goto end;
}
if (tchain != NULL)
    X509_STORE_CTX_set0_trusted_stack(csc, tchain);
if (crls != NULL)
    X509_STORE_CTX_set0_crls(csc, crls);

i = X509_verify_cert(csc);
X509_STORE_CTX_free(csc);
```

<CODE ENDS>

A.2. mbedTLS

```
<CODE BEGINS>
mbedtls_x509_crt cert;
mbedtls_x509_crt caCert;
uint32_t         certVerifyResultFlags;
...
int result = mbedtls_x509_crt_verify(&cert, &caCert, NULL, NULL,
                             &certVerifyResultFlags, NULL, NULL);

<CODE ENDS>
```

Appendix B. Constrained BRSKI-EST Message Examples

This appendix extends the message examples from Appendix A of [I-D.ietf-ace-coap-est] with constrained BRSKI messages. The CoAP headers are only fully worked out for the first example, enrollstatus.

B.1. enrollstatus

A coaps enrollstatus message from Pledge to Registrar can be as follows:

```
 POST coaps://192.0.2.1:8085/b/es
   Content-Format: 60
   Payload: <binary CBOR enrollstatus document>
```

The corresponding CoAP header fields are shown below.

```
Ver = 1
T = 0 (CON)
TKL = 1
Code = 0x02 (0.02 is POST method)
Message ID = 0xab0f
Token = 0x4d
Options
 Option  (Uri-Path)
   Option Delta = 0xb   (option nr = 11)
   Option Length = 0x1
   Option Value = "b"
 Option  (Uri-Path)
   Option Delta = 0x0   (option nr = 11)
   Option Length = 0x2
   Option Value = "es"
 Option  (Content-Format)
   Option Delta = 0x1   (option nr = 12)
   Option Length = 0x1
   Option Value = 60    (application/cbor)
Payload Marker = 0xFF
Payload = A26776657273696F6E0166737461747573F5 (18 bytes binary)
```

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

The above binary CBOR enrollstatus payload looks as follows in CBOR diagnostic notation, for the case of enrollment success:

```
{
  "version": 1,
  "status": true
 }
```

Alternatively the payload could look as follows in case of enrollment failure, using the reason field to describe the failure:

```
Payload = A36776657273696F6E0166737461747573F466726561736F6E782A3C
          496E666F726D61746976652068756D616E207265616461626C652065
          72726F72206D6573736167653E

{
  "version": 1,
  "status": false,
  "reason": "<Informative human readable error message>"
}
```

To indicate successful reception of the enrollmentstatus telemetry report, a response from the Registrar may then be:

2.04 Changed

With CoAP fields:

```
Ver=1
T=2 (ACK)
TKL=1
Code = 0x44 (2.04 Changed)
Message ID = 0xab0f
Token = 0x4d
```

B.2. voucher_status

A coaps voucher_status message from Pledge to Registrar can be as follows:

```
 POST coaps://[2001:db8::2:1]/.well-known/brski/vs
   Content-Format: 60 (application/cbor)
   Payload:
```


```
a46776657273696f6e0166737461747573f466726561736f6e7828496e66
6f726d61746976652068756d616e2d7265616461626c65206572726f7220
6d6573736167656e726561736f6e2d636f6e74657874a100764164646974
696f6e616c20696e666f726d6174696f6e
```


The request payload above is binary CBOR but represented here in hexadecimal for readability. Below is the equivalent CBOR diagnostic format.

```
{
  "version": 1,
  "status": false,
  "reason": "Informative human-readable error message",
  "reason-context": { 0: "Additional information" }
}
```

A success response without payload will then be sent by the Registrar back to the Pledge to indicate reception of the telemetry report:

2.04 Changed

Appendix C. COSE-signed Voucher (Request) Examples

This appendix provides examples of COSE-signed voucher requests and vouchers. First, the used test keys and certificates are described, following by examples of a constrained PVR, RVR and voucher.

C.1. Pledge, Registrar and MASA Keys

This section documents the public and private keys used for all examples in this appendix. These keys are not used in any production system, and must only be used for testing purposes.

C.1.1. Pledge IDevID private key

```
<CODE BEGINS>
Private-Key: (256 bit)
priv:
    9b:4d:43:b6:a9:e1:7c:04:93:45:c3:13:d9:b5:f0:
    41:a9:6a:9c:45:79:73:b8:62:f1:77:03:3a:fc:c2:
    9c:9a
pub:
    04:d6:b7:6f:74:88:bd:80:ae:5f:28:41:2c:72:02:
    ef:5f:98:b4:81:e1:d9:10:4c:f8:1b:66:d4:3e:5c:
    ea:da:73:e6:a8:38:a9:f1:35:11:85:b6:cd:e2:04:
    10:be:fe:d5:0b:3b:14:69:2e:e1:b0:6a:bc:55:40:
    60:eb:95:5c:54
ASN1 OID: prime256v1
NIST CURVE: P-256
<CODE ENDS>
```

C.1.2. Registrar private key

```
<CODE BEGINS>
Private-Key: (256 bit)
priv:
    81:df:bb:50:a3:45:58:06:b5:56:3b:46:de:f3:e9:
    e9:00:ae:98:13:9e:2f:36:68:81:fc:d9:65:24:fb:
    21:7e
pub:
    04:50:7a:c8:49:1a:8c:69:c7:b5:c3:1d:03:09:ed:
    35:ba:13:f5:88:4c:e6:2b:88:cf:30:18:15:4f:a0:
    59:b0:20:ec:6b:eb:b9:4e:02:b8:93:40:21:89:8d:
    a7:89:c7:11:ce:a7:13:39:f5:0e:34:8e:df:0d:92:
    3e:d0:2d:c7:b7
ASN1 OID: prime256v1
NIST CURVE: P-256
<CODE ENDS>
```

C.1.3. MASA private key

```
<CODE BEGINS>
Private-Key: (256 bit)
priv:
    c6:bb:a5:8f:b6:d3:c4:75:28:d8:d3:d9:46:c3:31:
    83:6d:00:0a:9a:38:ce:22:5c:e9:d9:ea:3b:98:32:
    ec:31
pub:
    04:59:80:94:66:14:94:20:30:3c:66:08:85:55:86:
    db:e7:d4:d1:d7:7a:d2:a3:1a:0c:73:6b:01:0d:02:
    12:15:d6:1f:f3:6e:c8:d4:84:60:43:3b:21:c5:83:
    80:1e:fc:e2:37:85:77:97:94:d4:aa:34:b5:b6:c6:
    ed:f3:17:5c:f1
ASN1 OID: prime256v1
NIST CURVE: P-256
<CODE ENDS>
```

C.2. Pledge, Registrar and MASA Certificates

All keys and certificates used for the examples have been generated with OpenSSL - see Appendix D for more details on certificate generation. Below the certificates are listed that accompany the keys shown above. Each certificate description is followed by the hexadecimal representation of the X.509 ASN.1 DER encoded certificate. This representation can be for example decoded using an online ASN.1 decoder.

C.2.1. Pledge IDevID Certificate

```
<CODE BEGINS>
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: Dec  9 10:02:36 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:d6:b7:6f:74:88:bd:80:ae:5f:28:41:2c:72:02:
                  ef:5f:98:b4:81:e1:d9:10:4c:f8:1b:66:d4:3e:5c:
                  ea:da:73:e6:a8:38:a9:f1:35:11:85:b6:cd:e2:04:
                  10:be:fe:d5:0b:3b:14:69:2e:e1:b0:6a:bc:55:40:
                  60:eb:95:5c:54
              ASN1 OID: prime256v1
              NIST CURVE: P-256
      X509v3 extensions:
          X509v3 Basic Constraints:
              CA:FALSE
          X509v3 Authority Key Identifier:
              keyid:
      E4:03:93:B4:C3:D3:F4:2A:80:A4:77:18:F6:96:49:03:01:17:68:A3

  Signature Algorithm: ecdsa-with-SHA256
         30:46:02:21:00:d2:e6:45:3b:b0:c3:00:b3:25:8d:f1:83:fe:
         d9:37:c1:a2:49:65:69:7f:6b:b9:ef:2c:05:07:06:31:ac:17:
         bd:02:21:00:e2:ce:9e:7b:7f:74:50:33:ad:9e:ff:12:4e:e9:
         a6:f3:b8:36:65:ab:7d:80:bb:56:88:bc:03:1d:e5:1e:31:6f
<CODE ENDS>
```

Below is the hexadecimal representation:

<CODE BEGINS>

30820226308201cba003020102020711223344556600300a06082a8648ce3d04
0302306f310b3009060355040613024e4c310b300906035504080c024e423110
300e06035504070c0748656c6d6f6e6431133011060355040a0c0a76616e6465
7273746f6b311530130603550400b0c0c6d616e75666163747572657231153013
06035504030c0c6d6173612e73746f6b2e6e6c3020170d323031323039313030
3233365a180f39393939313233313233353935395a308190310b300906035504
0613024e4c310b300906035504080c024e423110300e06035504070c0748656c
6d6f6e6431133011060355040a0c0a76616e64657273746f6b31163014060355
040b0c0d6d616e75666163747572696e67311c301a06035504030c1375756964
3a706c656467652e312e322e332e3431173015060355040513 0e706c65646765
2e312e322e332e343059301306072a8648ce3d020106082a8648ce3d03010703
420004d6b76f7488bd80ae5f28412c7202ef5f98b481e1d9104cf81b66d43e5c
eada73e6a838a9f1351185b6cde20410befed50b3b14692ee1b06abc554060eb
955c54a32e302c30090603551d1304023000301f0603551d23041830168014e4
0393b4c3d3f42a80a47718f6964903011768a3300a06082a8648ce3d04030203
49003046022100d2e6453bb0c300b3258df183fed937c1a24965697f6bb9ef2c
05070631ac17bd022100e2ce9e7b7f745033ad9eff124ee9a6f3b83665ab7d80
bb5688bc031de51e316f

<CODE ENDS>

C.2.2. Registrar Certificate

```
<CODE BEGINS>
Certificate:
  Data:
      Version: 3 (0x2)
      Serial Number:
        70:56:ea:aa:30:66:d8:82:6a:55:5b:90:88:d4:62:bf:9c:f2:8c:fd
      Signature Algorithm: ecdsa-with-SHA256
      Issuer: C=NL, ST=NB, L=Helmond, O=vanderstok, OU=consultancy,
                                              CN=registrar.stok.nl
      Validity
          Not Before: Dec  9 10:02:36 2020 GMT
          Not After : Dec  9 10:02:36 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:
                  04:50:7a:c8:49:1a:8c:69:c7:b5:c3:1d:03:09:ed:
                  35:ba:13:f5:88:4c:e6:2b:88:cf:30:18:15:4f:a0:
                  59:b0:20:ec:6b:eb:b9:4e:02:b8:93:40:21:89:8d:
                  a7:89:c7:11:ce:a7:13:39:f5:0e:34:8e:df:0d:92:
                  3e:d0:2d:c7:b7
              ASN1 OID: prime256v1
              NIST CURVE: P-256
      X509v3 extensions:
          X509v3 Subject Key Identifier:
    08:C2:BF:36:88:7F:79:41:21:85:87:2F:16:A7:AC:A6:EF:B3:D2:B3
          X509v3 Authority Key Identifier:
              keyid:
    08:C2:BF:36:88:7F:79:41:21:85:87:2F:16:A7:AC:A6:EF:B3:D2:B3

          X509v3 Basic Constraints: critical
              CA:TRUE
          X509v3 Extended Key Usage:
              CMC Registration Authority, TLS Web Server
              Authentication, TLS Web Client Authentication
          X509v3 Key Usage: critical
              Digital Signature, Non Repudiation, Key Encipherment,
              Data Encipherment, Certificate Sign, CRL Sign
  Signature Algorithm: ecdsa-with-SHA256
       30:44:02:20:74:4c:99:00:85:13:b2:f1:bc:fd:f9:02:1a:46:
       fb:17:4c:f8:83:a2:7c:a1:d9:3f:ae:ac:f3:1e:4e:dd:12:c6:
       02:20:11:47:14:db:f5:1a:5e:78:f5:81:b9:42:1c:6e:47:02:
       ab:53:72:70:c5:ba:fb:2d:16:c3:de:9a:a1:82:c3:5f
<CODE ENDS>
```

Below is the hexadecimal representation which is in (request-)voucher examples referred to as regis-cert-hex:


```
<CODE BEGINS>
308202753082021ca00302010202147056eaaa3066d8826a555b9088d462bf9c
f28cfd300a06082a8648ce3d0403023073310b3009060355040613024e4c310b
300906035504080c024e423110300e06035504070c0748656c6d6f6e64311330
11060355040a0c0a76616e64657273746f6b311430120603550400b0c0b636f6e
73756c74616e6379311a301806035504030c117265676973747261722e73746f
6b2e6e6c301e170d3230313230393130303233365a170d323131323039313030
3233365a3073310b3009060355040613024e4c310b300906035504080c024e42
3110300e06035504070c0748656c6d6f6e643113301106035504 0a0c0a76616e
64657273746f6b311430120603550400b0c0b636f6e73756c74616e6379311a30
1806035504030c117265676973747261722e73746f6b2e6e6c3059301306072a
8648ce3d020106082a8648ce3d03010703420004507ac8491a8c69c7b5c31d03
09ed35ba13f5884ce62b88cf3018154fa059b020ec6bebb94e02b8934021898d
a789c711cea71339f50e348edf0d923ed02dc7b7a3818d30818a301d0603551d
0e0416041408c2bf36887f79412185872f16a7aca6efb3d2b3301f0603551d23
04183016801408c2bf36887f79412185872f16a7aca6efb3d2b3300f0603551d
130101ff040530030101ff30270603551d250420301e06082b0601050507031c
06082b0601050507030106082b06010505070302300e0603551d0f0101ff0404
030201f6300a06082a8648ce3d0403020347003044022074 4c99008513b2f1bc
fdf9021a46fb174cf883a27ca1d93faeacf31e4edd12c60220114714dbf51a5e
78f581b9421c6e4702ab537270c5bafb2d16c3de9aa182c35f
<CODE ENDS>
```

C.2.3. MASA Certificate

```
<CODE BEGINS>
Certificate:
  Data:
      Version: 3 (0x2)
      Serial Number:
        14:26:b8:1c:ce:d8:c3:e8:14:05:cb:87:67:0d:be:ef:d5:81:25:b4
      Signature Algorithm: ecdsa-with-SHA256
      Issuer: C=NL, ST=NB, L=Helmond, O=vanderstok,
          OU=manufacturer, CN=masa.stok.nl

      Validity
          Not Before: Dec  9 10:02:36 2020 GMT
          Not After : Sep  5 10:02:36 2023 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:
                  04:59:80:94:66:14:94:20:30:3c:66:08:85:55:86:
                  db:e7:d4:d1:d7:7a:d2:a3:1a:0c:73:6b:01:0d:02:
                  12:15:d6:1f:f3:6e:c8:d4:84:60:43:3b:21:c5:83:
                  80:1e:fc:e2:37:85:77:97:94:d4:aa:34:b5:b6:c6:
                  ed:f3:17:5c:f1
              ASN1 OID: prime256v1
              NIST CURVE: P-256
      X509v3 extensions:
          X509v3 Subject Key Identifier:
    E4:03:93:B4:C3:D3:F4:2A:80:A4:77:18:F6:96:49:03:01:17:68:A3
          X509v3 Authority Key Identifier:
              keyid:
     E4:03:93:B4:C3:D3:F4:2A:80:A4:77:18:F6:96:49:03:01:17:68:A3

          X509v3 Basic Constraints: critical
              CA:TRUE
          X509v3 Extended Key Usage:
              CMC Registration Authority,
              TLS Web Server Authentication,
              TLS Web Client Authentication
          X509v3 Key Usage: critical
              Digital Signature, Non Repudiation, Key Encipherment,
                    Data Encipherment, Certificate Sign, CRL Sign
  Signature Algorithm: ecdsa-with-SHA256
       30:44:02:20:2e:c5:f2:24:72:70:20:ea:6e:74:8b:13:93:67:
       8a:e6:fe:fb:8d:56:7f:f5:34:18:a9:ef:a5:0f:c3:99:ca:53:
       02:20:3d:dc:91:d0:e9:6a:69:20:01:fb:e4:20:40:de:7c:7d:
       98:ed:d8:84:53:61:84:a7:f9:13:06:4c:a9:b2:8f:5c
<CODE ENDS>
```

Below is the hexadecimal representation:

```
<CODE BEGINS>
3082026d30820214a00302010202141426b81cced8c3e81405cb87670dbeefd5
8125b4300a06082a8648ce3d040302306f310b3009060355040613024e4c310b
300906035504080c024e423110300e06035504070c0748656c6d6f6e64311330
11060355040a0c0a76616e64657273746f6b311530130603550403b0c0c6d616e
75666163747572657231153013060355040030c0c6d6173612e73746f6b2e6e6c
301e170d3230313230393130303233365a170d3233303930353130303233365a
306f310b3009060355040613024e4c310b300906035504080c024e423110300e
06035504070c0748656c6d6f6e643113301106035504a0c0a76616e64657273
746f6b311530130603550400b0c0c6d616e75666163747572657231153013060603
5504030c0c6d6173612e73746f6b2e6e6c3059301306072a8648ce3d02010608
2a8648ce3d03010703420004598094661494203 03c6608855586dbe7d4d1d77a
d2a31a0c736b010d021215d61ff36ec8d48460433b21c583801efce237857797
94d4aa34b5b6c6edf3175cf1a3818d30818a301d0603551d0e04160414e40393
b4c3d3f42a80a47718f6964903011768a3301f0603551d23041830168014e403
93b4c3d3f42a80a47718f6964903011768a3300f0603551d130101ff04053003
0101ff30270603551d250420301e06082b0601050507031c06082b0601050507
030106082b06010505070302300e0603551d0f0101ff0404030201f6300a0608
2a8648ce3d040302034700304402202ec5f224727020ea6e748b1393678ae6fe
fb8d567ff53418a9efa50fc399ca5302203ddc91d0e96a692001fbe42040de7c
7d98edd884536184a7f913064ca9b28f5c
<CODE ENDS>
```

C.3. COSE-signed Pledge Voucher Request (PVR)

In this COSE example the voucher request has been signed by the Pledge using the private key of Appendix C.1.1, and has been sent to the link-local JRC (Registrar) over CoAPS.

```
 POST coaps://[JRC-link-local-address]/b/rv
 Content-Format: TBD3
 Payload: signed_request_voucher
```

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

```
<CODE BEGINS>
d28444a101382ea104582097113db094eee8eae48683e7337875c0372164
be89d023a5f3df52699c0fbfb55902d2a11909c5a60274323032302d3132
2d32335431323a30353a32325a0474323032322d31322d32335431323a30
353a32325a01020750684ca83e27230aff97630cf2c1ec409a0d6e706c65
6467652e312e322e332e340a5902793082027530820210ca0030201020214
7056eaaa3066d8826a555b9088d462bf9cf28cfd300a06082a8648ce3d04
03023073310b300906035504061302 4e4c310b300906035504080c024e42
3110300e06035504070c0748656c6d6f6e6431133011060355040a0c0a76
616e64657273746f6b31143012060355040b0c0b636f6e73756c74616e63
79311a301806035504030c117265676973747261722e73746f6b2e6e6c30
1e170d3230313230393130303233365a170d32313132303931303032333
5a3073310b3009060355040613024e4c310b300906035504080c024e4231
10300e06035504070c0748656c6d6f6e6431133011060355040a0c0a7661
6e64657273746f6b31143012060355040b0c0b636f6e73756c74616e6379
311a301806035504030c117265676973747261722e73746f6b2e6e6c3059
301306072a8648ce3d020106082a8648ce3d03010703420004507ac8491a
8c69c7b5c31d0309ed35ba13f5884ce62b88cf3018154fa059b020ec6beb
b94e02b8934021898da789c711cea71339f50e348edf0d923ed02dc7b7a3
818d30818a301d0603551d0e0416041408c2bf36887f79412185872f16a7
aca6efb3d2b3301f0603551d2304183016801408c2bf36887f7941218587
2f16a7aca6efb3d2b3300f0603551d130101ff040530030101ff30270603
551d250420301e06082b0601050507031c06082b0601050507030106082b
0601050507030230 0e0603551d0f0101ff0404030201f6300a06082a8648
ce3d0403020347003044022074 4c99008513b2f1bcfdf9021a46fb174cf8
83a27ca1d93faeacf31e4edd12c60220114714dbf51a5e78f581b9421c6e
4702ab537270c5bafb2d16c3de9aa182c35f58473045022063766c7bbd1b
339dbc398e764af3563e93b25a69104befe9aac2b3336b8f56e1022100cd
0419559ad960ccaed4dee3f436eca40b7570b25a52eb60332bc1f2991484
e9
<CODE ENDS>
```

The Pledge uses the "proximity" (SID 2502, enum 2) assertion together with an included proximity-registrar-cert field (SID 2511) to inform MASA about its proximity to the specific Registrar. The representiation of signed_voucher_request in CBOR diagnostic format is:

```
<CODE BEGINS>
Diagnose(signed_request_voucher) =
18([
h'A101382E',     / {"alg": -47} /
{4: h'97113DB094EEE8EAE48683E7337875C0372164B
      E89D023A5F3DF52699C0FBFB5'},
h'<request_voucher>',  / byte string as detailed below /
h'3045022063766C7BBD1B339DBC398E764AF3563E93B
25A69104BEFE9AAC2B3336B8F56E1022100CD0419559A
D960CCAED4DEE3F436ECA40B7570B25A52EB60332BC1F
2991484E9'
])

Diagnose(request_voucher) =
{2501: {2: "2020-12-23T12:05:22Z",
        4: "2022-12-23T12:05:22Z",
        1: 2,
        7: h'684CA83E27230AFF97630CF2C1EC409A',
        13: "pledge.1.2.3.4",
        10: h'<regis-cert-hex>' / byte string as defined in C.2.2 /
}}
<CODE ENDS>
```

C.4. COSE-signed Registrar Voucher Request (RVR)

In this example the Registrar's voucher request has been signed by the JRC (Registrar) using the private key from Appendix C.1.2. Contained within this voucher request is the voucher request PVR that was made by the Pledge to JRC. Note that the RVR uses the HTTPS protocol (not CoAP) and corresponding long URI path names as defined in [RFC8995]. The Content-Type and Accept headers indicate the constrained voucher format that is defined in the present document. Because the Pledge used this format in the PVR, the JRC must also use this format in the RVR.

```
POST https://masa.example.com/.well-known/brski/requestvoucher
Content-Type: application/voucher-cose+cbor
Accept: application/voucher-cose+cbor
Body: signed_masa_request_voucher
```

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

```
<CODE BEGINS>
d28444a101382ea1045820e8735bc4b470c3aa6a7aa9aa8ee584c09c1113
1b205efec5d0313bad84c5cd05590414a11909c5a60274323032302d3132
2d32385431303a30333a33355a0474323032322d31322d32385431303a30
333a33355a07501551631f6e0416bd162ba53ea00c2a050d6e706c656467
652e312e322e332e3405587131322d32385431303a30333a33355a075015
51631f6e0416bd162ba53ea00c2a050d6e706c656467652e312e322e332e
3405587131322d32385431303a3000000000000000000000000000416bd16
2ba53ea00c2a050d6e706c656467652e312e322e332e3405587131322d32
385431303a09590349d28444a101382ea104582097113db094eee8eae486
83e7337875c0372164be89d023a5f3df52699c0fbfb55902d2a11909c5a6
0274323032302d31322d32385431303a30333a33355a0474323032322d31
322d32385431303a30333a33355a010207501551631f6e0416bd162ba53e
a00c2a050d6e706c656467652e312e322e332e340a590279308202753082
021ca00302010202147056eaaa3066d8826a555b9088d462bf9cf28cfd30
0a06082a8648ce3d0403023073310b3009060355040613024e4c310b3009
06035504080c024e423110300e06035504070c0748656c6d6f6e64311330
11060355040a0c0a76616e64657273746f6b31143012060355040b0c0b63
6f6e73756c74616e6379311a301806035504030c11726567697374726172
2e73746f6b2e6e6c301e170d3230313230393130303233365a170d323131
3230393130303233365a3073310b3009060355040613024e4c310b300906
035504080c024e423110300e06035504070c0748656c6d6f6e6431133011
060355040a0c0a76616e64657273746f6b31143012060355040b0c0b636f
6e73756c74616e6379311a301806035504030c117265676973747261722e
73746f6b2e6e6c3059301306072a8648ce3d020106082a8648ce3d030107
03420004507ac8491a8c69c7b5c31d0309ed35ba13f5884ce62b88cf3018
154fa059b020ec6bebb94e02b8934021898da789c711cea71339f50e348e
df0d923ed02dc7b7a3818d30818a301d0603551d0e0416041408c2bf3688
7f79412185872f16a7aca6efb3d2b3301f0603551d2304183016801408c2
bf36887f79412185872f16a7aca6efb3d2b3300f0603551d130101ff0405
30030101ff30270603551d250420301e06082b0601050507031c06082b06
010505070301060822b0601050507030230e0603551d0f0101ff04040302
01f6300a06082a8648ce3d0403020347003044022074c99008513b2f1bc
fdf9021a46fb174cf883a27ca1d93faeacf31e4edd12c60220114714dbf5
1a5e78f581b9421c6e4702ab537270c5bafb2d16c3de9aa182c35f584730
45022063766c7bbd1b339dbc398e764af3563e93b25a69104befe9aac2b3
336b8f56e1022100cd0419559ad960ccaed4dee3f436eca40b7570b25a52
eb60332bc1f2991484e958473045022100e6b45558c1b806bba23f4ac626
c9bdb6fd354ef4330d8dfb7c529f29cca934c802203c1f2ccbbac89733d1
7ee7775bc2654c5f1cc96afba2741cc31532444aa8fed8
<CODE ENDS>
```

The representiation of signed_masa_voucher_request in CBOR diagnostic format is:

```
<CODE BEGINS>
Diagnose(signed_registrar_request-voucher)
18([
h'A101382E',     / {"alg": -47} /
h'E8735BC4B470C3AA6A7AA9AA8EE584C09C11131B205EFEC5D0313BAD84
C5CD05'},
h'<registrar_request_voucher>', / byte string as detailed below /
h'3045022100E6B45558C1B806BBA23F4AC626C9BDB6FD354EF4330D8DFB
7C529F29CCA934C802203C1F2CCBBAC89733D17EE7775BC2654C5F1CC96A
FBA2741CC31532444AA8FED8'
])

Diagnose(registrar_request_voucher)
{2501:
    {2: "2020-12-28T10:03:35Z",
     4: "2022-12-28T10:03:35Z",
     7: h'1551631F6E0416BD162BA53EA00C2A05',
    13: "pledge.1.2.3.4",
     5: h'31322D32385431303A30333A33355A07501551631F6E0416BD
          162BA53EA00C2A050D6E706C656467652E312E322E332E3405
          587131322D32385431303A30000000000000000000000000004
          16BD162BA53EA00C2A050D6E706C656467652E312E322E332E
          3405587131322D32385431303A', / idevid-issuer /
     9: h'<prior-pvr>'  / prior-signed-voucher-request = PVR /
     }
}
<CODE ENDS>
```

C.5. COSE-signed Voucher from MASA

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 (see Appendix C.1.3) and can be verified by the Pledge using the MASA's public key that it stores.

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

```
<CODE BEGINS>
d28444a101382ea104582039920a34ee92d3148ab3a729f58611193270c9
029f7784daf112614b19445d5158cfa1190993a70274323032302d31322d
32335431353a30333a31325a0474323032302d31322d32335431353a3233
3a31325a010007506508e06b2959d5089d7a3169ea889a490b6e706c6564
67652e312e322e332e340858753073310b3009060355040613024e4c310b
300906035504080c024e423110300e06035504070c0748656c6d6f6e6431
133011060355040a0c0a76616e64657273746f6b31143012060355040b0c
0b636f6e73756c74616e6379311a301806035504030c1172656769737472
61722e73746f6b2e6e6c03f45847304502202515d96cd12224ee5d3ac67
3237163bba24ad84815699285d9618f463ee73fa022100a6bff9d8585c1c
9256371ece94da3d26264a5dfec0a354fe7b3aef58344c512f
<CODE ENDS>
```

The representiation of signed_voucher in CBOR diagnostic format is:

```
<CODE BEGINS>
Diagnose(signed_voucher) =
18([
h'A101382E',     / {"alg": -47} /
{4: h'39920A34EE92D3148AB3A729F58611193270C9029F7784DAF112614B194
45D51'},
h'<voucher>',    / byte string as detailed below /
h'3045022022515D96CD12224EE5D3AC673237163BBA24AD84815699285D9618F
463EE73FA022100A6BFF9D8585C1C9256371ECE94DA3D26264A5DFEC0A354FE7B
3AEF58344C512F'
])

Diagnose(voucher) =
{2451:
   {2: "2020-12-23T15:03:12Z",
    4: "2020-12-23T15:23:12Z",
    1: 0,
    7: h'6508E06B2959D5089D7A3169EA889A49',
   11: "pledge.1.2.3.4",
    8: h'<regis-cert-hex>', / as detailed in C.2.2 /
    3: false}
}
<CODE ENDS>
```

In above, regis-cert-hex represents the hexadecimal encoding of the Registrar certificate of Appendix C.2.2.

Appendix D. Generating Certificates with OpenSSL

This informative appendix shows an example of a Bash shell script to generate test certificates for the Pledge IDevID, the Registrar and the MASA. This shell script cannot be run stand-alone because it depends on particular input files which are not included in this appendix. Nevertheless, this example script may provide guidance on how OpenSSL can be configured for generating Constrained BRSKI certificates.

Note: the *-comb.crt certificate files combine the certificate with the private key. These are generated to be used by libcoap for DTLS connection establishment.

```
<CODE BEGINS>
#!/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 # Some hex
export subjectAltName="otherName:1.3.6.1.5.5.7.8.4;SEQ:hmodname"
echo  $hwType - $hwSerialNum
echo $serialNumber
OPENSSL_BIN="openssl"

# 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 sha 256 key"
$OPENSSL_BIN ecparam -name prime256v1 -genkey \
   -noout -out $cadir/private/ca-regis.key

$OPENSSL_BIN req -new -x509 \
 -config $cnfdir/openssl-regis.cnf \
 -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_BIN pkcs12 -passin pass:watnietWT -passout pass:watnietWT\
   -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_BIN pkcs12 -passin pass:watnietWT -passout pass:watnietWT\
   -in $cadir/certs/ca-regis-comb.pfx \
   -out $cadir/certs/ca-regis-comb.crt -nodes

#show certificate in registrar combined certificate
$OPENSSL_BIN  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_BIN ecparam -name prime256v1 -genkey -noout \
   -out $cadir/private/ca-masa.key

$OPENSSL_BIN req -new -x509 \
 -config $cnfdir/openssl-masa.cnf \
 -days 1000 -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_BIN pkcs12 -passin pass:watnietWT -passout pass:watnietWT\
   -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_BIN pkcs12 -passin pass:watnietWT -passout pass:watnietWT\
   -in $cadir/certs/ca-masa-comb.pfx \
   -out $cadir/certs/ca-masa-comb.crt -nodes

#show certificate in pledge combined certificate
$OPENSSL_BIN  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 sha 256 key"
$OPENSSL_BIN ecparam -name prime256v1 -genkey -noout \
   -out $dir/private/pledge.key

echo "create pledge certificate request"
$OPENSSL_BIN 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_BIN 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 derived pledge key and derived pledge \
 certificate to pkcs12 file"
$OPENSSL_BIN pkcs12  -passin pass:watnietWT -passout pass:watnietWT\
   -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_BIN pkcs12 -passin pass:watnietWT -passout pass:watnietWT\
   -in $dir/certs/pledge-comb.pfx \
   -out $dir/certs/pledge-comb.crt -nodes

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

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

<CODE ENDS>
```

Appendix E. Pledge Device Class Profiles

This specification allows implementers to select between various functional options for the Pledge, yielding different code size footprints and different requirements on Pledge hardware. Thus for each product an optimal trade-off between functionality, development/maintenance cost and hardware cost can be made.

This appendix illustrates different selection outcomes by means of defining different example "profiles" of constrained Pledges. In the following subsections, these profiles are defined and a comparison is provided.

E.1. Minimal Pledge

The Minimal Pledge profile (Min) aims to reduce code size and hardware cost to a minimum. This comes with some severe functional restrictions, in particular:

* No support for EST re-enrollment: whenever this would be needed, a factory reset followed by a new bootstrap process is required.

* No support for change of Registrar: for this case, a factory reset followed by a new bootstrap process is required.

This profile would be appropriate for single-use devices which must be replaced rather than re-deployed. That might include medical devices, but also sensors used during construction, such as concrete temperature sensors.

E.2. Typical Pledge

The Typical Pledge profile (Typ) aims to support a typical Constrained BRSKI feature set including EST re-enrollment support and Registrar changes.

E.3. Full-featured Pledge

The Full-featured Pledge profile (Full) illustrates a Pledge category that supports multiple bootstrap methods, hardware real-time clock, BRSKI/EST resource discovery, and CSR Attributes request/response. It also supports most of the optional features defined in this specification.

E.4. Comparison Chart of Pledge Classes

The below table specifies the functions implemented in the three example Pledge classes Min, Typ and Full.

Function |====================| Profiles ->	**Min**	**Typ**	**Full**
General	===	===	====
Support Constrained BRSKI bootstrap	Y	Y	Y
Support other bootstrap method(s)	-	-	Y
Real-time clock and cert time checks	-	-	Y
Constrained BRSKI	===	===	====
Discovery for rt=brski*	-	-	Y
Support pinned Registrar public key (RPK)	Y	-	Y
Support pinned Registrar certificate	-	Y	Y
Support pinned Domain CA	-	Y	Y
Constrained EST	===	===	====
Discovery for rt=ace.est*	-	-	Y
GET /att and response parsing	-	-	Y
GET /crts format 281 (multiple CA certs)	-	-	Y
GET /crts only format TBD287 (one CA cert only)	Y	Y	-
ETag handling support for GET /crts	-	Y	Y
Re-enrollment supported	- (1)	Y	Y
6.6.1 optimized procedure	Y	Y	-
Pro-active cert re-enrollment at own initiative	N/A	-	Y
Periodic trust anchor retrieval GET /crts	- (1)	Y	Y
Supports change of Registrar identity	- (1)	Y	Y

Table 6

Notes: (1) is possible only by doing a factory-reset followed by a new bootstrap procedure.

Contributors

Russ Housley

Email: housley@vigilsec.com

Authors' Addresses

Michael Richardson
Sandelman Software Works

Email: mcr+ietf@sandelman.ca

Peter van der Stok
vanderstok consultancy

Email: stokcons@bbhmail.nl

Panos Kampanakis
Cisco Systems

Email: pkampana@cisco.com

Esko Dijk
IoTconsultancy.nl

Email: esko.dijk@iotconsultancy.nl