Workgroup: ANIMA WG
Internet-Draft:
draft-ietf-anima-bootstrapping-keyinfra-37
Published: 26 February 2020
Intended Status: Standards Track
Expires: 29 August 2020
Authors: M. Pritikin, Cisco
M. Richardson, Sandelman
T.T.E. Eckert, Futurewei USA
M.H. Behringer
K.W. Watsen, Watsen Networks

Bootstrapping Remote Secure Key Infrastructures (BRSKI)

Abstract

This document specifies automated bootstrapping of an Autonomic Control Plane. To do this a Secure Key Infrastructure is bootstrapped. This is done using manufacturer-installed X.509 certificates, in combination with a manufacturer's authorizing service, both online and offline. We call this process the Bootstrapping Remote Secure Key Infrastructure (BRSKI) protocol. Bootstrapping a new device can occur using a routable address and a cloud service, or using only link-local connectivity, or on limited/disconnected networks. Support for deployment models with less stringent security requirements is included. Bootstrapping is complete when the cryptographic identity of the new key infrastructure is successfully deployed to the device. The established secure connection can be used to deploy a locally issued certificate to the device as well.

Status of This Memo

Table of Contents

1. Introduction

The Bootstrapping Remote Secure Key Infrastructure (BRSKI) protocol provides a solution for secure zero-touch (automated) bootstrap of new (unconfigured) devices that are called pledges in this document. Pledges have an IDevID installed in them at the factory.

"BRSKI" is pronounced like "brewski", a colloquial term for beer in Canada and parts of the US-midwest. [brewski]

This document primarily provides for the needs of the ISP and Enterprise focused ANIMA Autonomic Control Plane (ACP) [I-D.ietf-anima-autonomic-control-plane]. This bootstrap process satisfies the [RFC7575] requirements of section 3.3 of making all operations secure by default. Other users of the BRSKI protocol will need to provide separate applicability statements that include privacy and security considerations appropriate to that deployment. Section 9 explains the detailed applicability for this the ACP usage.

The BRSKI protocol requires a significant amount of communication between manufacturer and owner: in its default modes it provides a cryptographic transfer of control to the initial owner. In its strongest modes, it leverages sales channel information to identify the owner in advance. Resale of devices is possible, provided that the manufacturer is willing to authorize the transfer. Mechanisms to enable transfers of ownership without manufacturer authorization are not included in this version of the protocol, but could be designed into future versions.

This document describes how pledges discover (or are discovered by) an element of the network domain to which the pledge belongs that will perform the bootstrap. This element (device) is called the registrar. Before any other operation, pledge and registrar need to establish mutual trust:

1. Registrar authenticating the pledge: "Who is this device? What is its identity?"

2. Registrar authorizing the pledge: "Is it mine? Do I want it? What are the chances it has been compromised?"

3. Pledge authenticating the registrar: "What is this registrar's identity?"

4. Pledge authorizing the registrar: "Should I join this network?"

This document details protocols and messages to answer the above questions. It uses a TLS connection and an PKIX-shaped (X.509v3) certificate (an IEEE 802.1AR [IDevID] IDevID) of the pledge to answer points 1 and 2. It uses a new artifact called a "voucher" that the registrar receives from a "Manufacturer Authorized Signing Authority" (MASA) and passes to the pledge to answer points 3 and 4.

A proxy provides very limited connectivity between the pledge and the registrar.

The syntactic details of vouchers are described in detail in [RFC8366]. This document details automated protocol mechanisms to obtain vouchers, including the definition of a 'voucher-request' message that is a minor extension to the voucher format (see Section 3) defined by [RFC8366].

BRSKI results in the pledge storing an X.509 root certificate sufficient for verifying the registrar identity. In the process a TLS connection is established that can be directly used for Enrollment over Secure Transport (EST). In effect BRSKI provides an automated mechanism for the "Bootstrap Distribution of CA Certificates" described in [RFC7030] Section 4.1.1 wherein the pledge "MUST [...] engage a human user to authorize the CA certificate using out-of-band" information. With BRSKI the pledge now can automate this process using the voucher. Integration with a complete EST enrollment is optional but trivial.

BRSKI is agile enough to support bootstrapping alternative key infrastructures, such as a symmetric key solutions, but no such system is described in this document.

1.1. Prior Bootstrapping Approaches

To literally "pull yourself up by the bootstraps" is an impossible action. Similarly the secure establishment of a key infrastructure without external help is also an impossibility. Today it is commonly accepted that the initial connections between nodes are insecure, until key distribution is complete, or that domain-specific keying material (often pre-shared keys, including mechanisms like SIM cards) is pre-provisioned on each new device in a costly and non-scalable manner. Existing automated mechanisms are known as non-secured 'Trust on First Use' (TOFU) [RFC7435], 'resurrecting duckling' [Stajano99theresurrecting] or 'pre-staging'.

Another prior approach has been to try and minimize user actions during bootstrapping, but not eliminate all user-actions. The original EST protocol [RFC7030] does reduce user actions during bootstrap but does not provide solutions for how the following protocol steps can be made autonomic (not involving user actions):

* using the Implicit Trust Anchor [RFC7030] database to authenticate an owner specific service (not an autonomic solution because the URL must be securely distributed),

* engaging a human user to authorize the CA certificate using out-of-band data (not an autonomic solution because the human user is involved),

* using a configured Explicit TA database (not an autonomic solution because the distribution of an explicit TA database is not autonomic),

* and using a Certificate-Less TLS mutual authentication method (not an autonomic solution because the distribution of symmetric key material is not autonomic).

These "touch" methods do not meet the requirements for zero-touch.

There are "call home" technologies where the pledge first establishes a connection to a well known manufacturer service using a common client-server authentication model. After mutual authentication, appropriate credentials to authenticate the target domain are transferred to the pledge. This creates several problems and limitations:

* the pledge requires realtime connectivity to the manufacturer service,

* the domain identity is exposed to the manufacturer service (this is a privacy concern),

* the manufacturer is responsible for making the authorization decisions (this is a liability concern),

BRSKI addresses these issues by defining extensions to the EST protocol for the automated distribution of vouchers.

1.2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The following terms are defined for clarity:

ANI: The Autonomic Network Infrastructure as defined by [I-D.ietf-anima-reference-model]. Section 9 details specific requirements for pledges, proxies and registrars when they are part of an ANI.

Circuit Proxy: A stateful implementation of the join proxy. This is the assumed type of proxy.

drop-ship: The physical distribution of equipment containing the "factory default" configuration to a final destination. In zero-touch scenarios there is no staging or pre-configuration during drop-ship.

Domain: The set of entities that share a common local trust anchor. This includes the proxy, registrar, Domain Certificate Authority, Management components and any existing entity that is already a member of the domain.

domainID: The domain IDentity is a unique value based upon the Registrar CA's certificate. Section 5.8.2 specifies how it is calculated.

Domain CA: The domain Certification Authority (CA) provides certification functionalities to the domain. At a minimum it provides certification functionalities to a registrar and manages the private key that defines the domain. Optionally, it certifies all elements.

enrollment: The process where a device presents key material to a network and acquires a network-specific identity. For example when a certificate signing request is presented to a certification authority and a certificate is obtained in response.

imprint: The process where a device obtains the cryptographic key material to identify and trust future interactions with a network. This term is taken from Konrad Lorenz's work in biology with new ducklings: during a critical period, the duckling would assume that anything that looks like a mother duck is in fact their mother. An equivalent for a device is to obtain the fingerprint of the network's root certification authority certificate. A device that imprints on an attacker suffers a similar fate to a duckling that imprints on a hungry wolf. Securely imprinting is a primary focus of this document [imprinting]. The analogy to Lorenz's work was first noted in [Stajano99theresurrecting].

IDevID: An Initial Device Identity X.509 certificate installed by the vendor on new equipment. This is a term from 802.1AR [IDevID]

IPIP Proxy: A stateless proxy alternative.

Join Proxy: A domain entity that helps the pledge join the domain. A join proxy facilitates communication for devices that find themselves in an environment where they are not provided connectivity until after they are validated as members of the domain. For simplicity this document sometimes uses the term of 'proxy' to indicate the join proxy. The pledge is unaware that they are communicating with a proxy rather than directly with a registrar.

Join Registrar (and Coordinator): A representative of the domain that is configured, perhaps autonomically, to decide whether a new device is allowed to join the domain. The administrator of the domain interfaces with a "join registrar (and coordinator)" to control this process. Typically a join registrar is "inside" its domain. For simplicity this document often refers to this as just "registrar". Within [I-D.ietf-anima-reference-model] this is referred to as the "join registrar autonomic service agent". Other communities use the abbreviation "JRC".

LDevID: A Local Device Identity X.509 certificate installed by the owner of the equipment. This is a term from 802.1AR [IDevID]

manufacturer: the term manufacturer is used throughout this document to be the entity that created the device. This is typically the "original equipment manufacturer" or OEM, but in more complex situations it could be a "value added retailer" (VAR), or possibly even a systems integrator. In general, it a goal of BRSKI to eliminate small distinctions between different sales channels. The reason for this is that it permits a single device, with a uniform firmware load, to be shipped directly to all customers. This eliminates costs for the manufacturer. This also reduces the number of products supported in the field increasing the chance that firmware will be more up to date.

MASA Audit-Log: An anonymized list of previous owners maintained by the MASA on a per device (per pledge) basis. Described in Section 5.8.1.

MASA Service: A third-party Manufacturer Authorized Signing Authority (MASA) service on the global Internet. The MASA signs vouchers. It also provides a repository for audit-log information of privacy protected bootstrapping events. It does not track ownership.

nonced: a voucher (or request) that contains a nonce (the normal case).

nonceless: a voucher (or request) that does not contain a nonce, relying upon accurate clocks for expiration, or which does not expire.

offline: When an architectural component cannot perform realtime communications with a peer, either due to network connectivity or because the peer is turned off, the operation is said to be occurring offline.

Ownership Tracker: An Ownership Tracker service on the global Internet. The Ownership Tracker uses business processes to accurately track ownership of all devices shipped against domains that have purchased them. Although optional, this component allows vendors to provide additional value in cases where their sales and distribution channels allow for accurate tracking of such ownership. Ownership tracking information is indicated in vouchers as described in [RFC8366]

Pledge: The prospective (unconfigured) device, which has an identity installed at the factory.

(Public) Key Infrastructure: The collection of systems and processes that sustain the activities of a public key system. The registrar acts as an [RFC5280] and [RFC5272] (see section 7) "Registration Authority".

TOFU: Trust on First Use. Used similarly to [RFC7435]. This is where a pledge device makes no security decisions but rather simply trusts the first registrar it is contacted by. This is also known as the "resurrecting duckling" model.

Voucher: A signed artifact from the MASA that indicates to a pledge the cryptographic identity of the registrar it should trust.

There are different types of vouchers depending on how that trust is asserted. Multiple voucher types are defined in [RFC8366]

1.3. Scope of solution

1.3.1. Support environment

This solution (BRSKI) can support large router platforms with multi-gigabit inter-connections, mounted in controlled access data centers. But this solution is not exclusive to large equipment: it is intended to scale to thousands of devices located in hostile environments, such as ISP provided CPE devices which are drop-shipped to the end user. The situation where an order is fulfilled from distributed warehouse from a common stock and shipped directly to the target location at the request of a domain owner is explicitly supported. That stock ("SKU") could be provided to a number of potential domain owners, and the eventual domain owner will not know a-priori which device will go to which location.

The bootstrapping process can take minutes to complete depending on the network infrastructure and device processing speed. The network communication itself is not optimized for speed; for privacy reasons, the discovery process allows for the pledge to avoid announcing its presence through broadcasting.

Nomadic or mobile devices often need to acquire credentials to access the network at the new location. An example of this is mobile phone roaming among network operators, or even between cell towers. This is usually called handoff. BRSKI does not provide a low-latency handoff which is usually a requirement in such situations. For these solutions BRSKI can be used to create a relationship (an LDevID) with the "home" domain owner. The resulting credentials are then used to provide credentials more appropriate for a low-latency handoff.

1.3.2. Constrained environments

Questions have been posed as to whether this solution is suitable in general for Internet of Things (IoT) networks. This depends on the capabilities of the devices in question. The terminology of [RFC7228] is best used to describe the boundaries.

The solution described in this document is aimed in general at non-constrained (i.e., class 2+ [RFC7228]) devices operating on a non-Challenged network. The entire solution as described here is not intended to be useable as-is by constrained devices operating on challenged networks (such as 802.15.4 Low-power Lossy Networks (LLN)s).

Specifically, there are protocol aspects described here that might result in congestion collapse or energy-exhaustion of intermediate battery powered routers in an LLN. Those types of networks should not use this solution. These limitations are predominately related to the large credential and key sizes required for device authentication. Defining symmetric key techniques that meet the operational requirements is out-of-scope but the underlying protocol operations (TLS handshake and signing structures) have sufficient algorithm agility to support such techniques when defined.

The imprint protocol described here could, however, be used by non-energy constrained devices joining a non-constrained network (for instance, smart light bulbs are usually mains powered, and speak 802.11). It could also be used by non-constrained devices across a non-energy constrained, but challenged network (such as 802.15.4). The certificate contents, and the process by which the four questions above are resolved do apply to constrained devices. It is simply the actual on-the-wire imprint protocol that could be inappropriate.

1.3.3. Network Access Controls

This document presumes that network access control has either already occurred, is not required, or is integrated by the proxy and registrar in such a way that the device itself does not need to be aware of the details. Although the use of an X.509 Initial Device Identity is consistent with IEEE 802.1AR [IDevID], and allows for alignment with 802.1X network access control methods, its use here is for pledge authentication rather than network access control. Integrating this protocol with network access control, perhaps as an Extensible Authentication Protocol (EAP) method (see [RFC3748]), is out-of-scope.

1.3.4. Bootstrapping is not Booting

This document describes "bootstrapping" as the protocol used to obtain a local trust anchor. It is expected that this trust anchor, along with any additional configuration information subsequently installed, is persisted on the device across system restarts ("booting"). Bootstrapping occurs only infrequently such as when a device is transferred to a new owner or has been reset to factory default settings.

1.4. Leveraging the new key infrastructure / next steps

As a result of the protocol described herein, the bootstrapped devices have the Domain CA trust anchor in common. An end entity certificate has optionally been issued from the Domain CA. This makes it possible to securely deploy functionalities across the domain, e.g:

*Device management.

*Routing authentication.

*Service discovery.

The major intended benefit is that it possible to use the credentials deployed by this protocol to secure the Autonomic Control Plane (ACP) ([I-D.ietf-anima-autonomic-control-plane]).

1.5. Requirements for Autonomic Network Infrastructure (ANI) devices

The BRSKI protocol can be used in a number of environments. Some of the options in this document are the result of requirements that are out of the ANI scope. This section defines the base requirements for ANI devices.

For devices that intend to become part of an Autonomic Network Infrastructure (ANI) ([I-D.ietf-anima-reference-model]) that includes an Autonomic Control Plane ([I-D.ietf-anima-autonomic-control-plane]), the BRSKI protocol MUST be implemented.

The pledge must perform discovery of the proxy as described in Section 4.1 using Generic Autonomic Signaling Protocol (GRASP)'s DULL [I-D.ietf-anima-grasp] M_FLOOD announcements.

Upon successfully validating a voucher artifact, a status telemetry MUST be returned. See Section 5.7.

An ANIMA ANI pledge MUST implement the EST automation extensions described in Section 5.9. They supplement the [RFC7030] EST to better support automated devices that do not have an end user.

The ANI Join Registrar Autonomic Service Agent (ASA) MUST support all the BRSKI and above listed EST operations.

All ANI devices SHOULD support the BRSKI proxy function, using circuit proxies over the ACP. (See Section 4.3)

2. Architectural Overview

The logical elements of the bootstrapping framework are described in this section. Figure 1 provides a simplified overview of the components.

```
                                           +------------------------+
   +--------------Drop Ship----------------| Vendor Service         |
   |                                       +------------------------+
   |                                       | M anufacturer|         |
   |                                       | A uthorized  |Ownership|
   |                                       | S igning     |Tracker  |
   |                                       | A uthority   |         |
   |                                       +--------------+---------+
   |                                                      ^
   |                                                      |  BRSKI-
   V                                                      |   MASA
+-------+     ............................................|...
|       |     .                                           |  .
|       |     .  +------------+       +-----------+       |  .
|       |     .  |            |       |           |       |  .
|Pledge |     .  |   Join     |       | Domain    <-------+  .
|       |     .  |   Proxy    |       | Registrar |          .
|       <-------->............<-------> (PKI RA)  |          .
|       |        |        BRSKI-EST   |           |          .
|       |     .  |            |       +-----+-----+          .
|IDevID |     .  +------------+             | e.g. RFC7030   .
|       |     .           +-----------------+----------+     .
|       |     .           | Key Infrastructure         |     .
|       |     .           | (e.g., PKI Certificate     |     .
+-------+     .           |       Authority)           |     .
              .           +----------------------------+     .
              .                                              .
              ................................................
                            "Domain" components
```

Figure 1: Architecture Overview

We assume a multi-vendor network. In such an environment there could be a Manufacturer Service for each manufacturer that supports devices following this document's specification, or an integrator could provide a generic service authorized by multiple manufacturers. It is unlikely that an integrator could provide Ownership Tracking services for multiple manufacturers due to the required sales channel integrations necessary to track ownership.

The domain is the managed network infrastructure with a Key Infrastructure the pledge is joining. The domain provides initial device connectivity sufficient for bootstrapping through a proxy. The domain registrar authenticates the pledge, makes authorization decisions, and distributes vouchers obtained from the Manufacturer Service. Optionally the registrar also acts as a PKI Certification Authority.

2.1. Behavior of a Pledge

The pledge goes through a series of steps, which are outlined here at a high level.

```
               ------------
              /  Factory   \
              \  default   /
               -----+------
                    |
             +------v-------+
             | (1) Discover |
+------------>              |
|            +------+-------+
|                   |
|            +------v-------+
|            | (2) Identify |
^------------+              |
| rejected   +------+-------+
|                   |
|            +------v-------+
|            | (3) Request  |
|            |     Join     |
|            +------+-------+
|                   |
|            +------v-------+
|            | (4) Imprint  |
^------------+              |
| Bad MASA   +------+-------+
| response          |  send Voucher Status Telemetry
|            +------v-------+
|            | (5) Enroll   |<---+ (non-error HTTP codes  )
^------------+              |\___/ (e.g. 202 'Retry-After')
| Enroll     +------+-------+
| Failure           |
|              -----v------
|             /  Enrolled  \
^------------+             |
 Factory      \------------/
 reset
```

Figure 2: Pledge State Diagram

State descriptions for the pledge are as follows:

1. Discover a communication channel to a registrar.

2. Identify itself. This is done by presenting an X.509 IDevID credential to the discovered registrar (via the proxy) in a TLS handshake. (The registrar credentials are only provisionally accepted at this time).

3. Request to join the discovered registrar. A unique nonce is included ensuring that any responses can be associated with this particular bootstrapping attempt.

4. Imprint on the registrar. This requires verification of the manufacturer-service-provided voucher. A voucher contains sufficient information for the pledge to complete authentication of a registrar. This document details this step in depth.

5. Enroll. After imprint an authenticated TLS (HTTPS) connection exists between pledge and registrar. Enrollment over Secure Transport (EST) [RFC7030] can then be used to obtain a domain certificate from a registrar.

The pledge is now a member of, and can be managed by, the domain and will only repeat the discovery aspects of bootstrapping if it is returned to factory default settings.

This specification details integration with EST enrollment so that pledges can optionally obtain a locally issued certificate, although any Representational State Transfer (REST) (see [REST]) interface could be integrated in future work.

2.2. Secure Imprinting using Vouchers

A voucher is a cryptographically protected artifact (using a digital signature) to the pledge device authorizing a zero-touch imprint on the registrar domain.

The format and cryptographic mechanism of vouchers is described in detail in [RFC8366].

Vouchers provide a flexible mechanism to secure imprinting: the pledge device only imprints when a voucher can be validated. At the lowest security levels the MASA can indiscriminately issue vouchers and log claims of ownership by domains. At the highest security levels issuance of vouchers can be integrated with complex sales channel integrations that are beyond the scope of this document. The sales channel integration would verify actual (legal) ownership of the pledge by the domain. This provides the flexibility for a number of use cases via a single common protocol mechanism on the pledge and registrar devices that are to be widely deployed in the field. The MASA services have the flexibility to leverage either the currently defined claim mechanisms or to experiment with higher or lower security levels.

Vouchers provide a signed but non-encrypted communication channel among the pledge, the MASA, and the registrar. The registrar maintains control over the transport and policy decisions, allowing the local security policy of the domain network to be enforced.

2.3. Initial Device Identifier

Pledge authentication and pledge voucher-request signing is via a PKIX-shaped certificate installed during the manufacturing process. This is the 802.1AR Initial Device Identifier (IDevID), and it provides a basis for authenticating the pledge during the protocol exchanges described here. There is no requirement for a common root PKI hierarchy. Each device manufacturer can generate its own root certificate. Specifically, the IDevID enables:

1. Uniquely identifying the pledge by the Distinguished Name (DN) and subjectAltName (SAN) parameters in the IDevID. The unique identification of a pledge in the voucher objects are derived from those parameters as described below. Section 10.3 discusses privacy implications of the identifier.

2. Provides a cryptographic authentication of the pledge to the Registrar (see Section 5.3).

3. Secure auto-discovery of the pledge's MASA by the registrar (see Section 2.8).

4. Signing of voucher-request by the pledge's IDevID (see Section 3).

5. Provides a cryptographic authentication of the pledge to the MASA (see Section 5.5.5).

Section 7.2.13 (2009 edition) and section 8.10.3 (2018 edition) of [IDevID] discusses keyUsage and extendedKeyUsage extensions in the IDevID certificate. [IDevID] acknowledges that adding restrictions in the certificate limits applicability of these long-lived certificates. This specification emphasizes this point, and therefore RECOMMENDS that no key usage restrictions be included. This is consistent with [RFC5280] section 4.2.1.3, which does not require key usage restrictions for end entity certificates.

2.3.1. Identification of the Pledge

In the context of BRSKI, pledges have a 1:1 relationship with a "serial-number". This serial-number is used both in the "serial-number" field of voucher or voucher-requests (see Section 3) and in local policies on registrar or MASA (see Section 5).

The serialNumber field is defined in [RFC5280]. That specification allows for the field to be omitted if there is a good technical reason. IDevID certificates for use with this protocol are REQUIRED to include the "serialNumber" attribute with the device's unique serial number (from [IDevID] section 7.2.8, and [RFC5280] section 4.1.2.2's list of standard attributes).

The serialNumber field is used as follows by the pledge to build the "serial-number" that is placed in the voucher-request. In order to build it, the fields need to be converted into a serial-number of "type string".

An example of a printable form of the "serialNumber" field is provided in [RFC4519] section 2.31 ("WI-3005"). That section further provides equality and syntax attributes.

Due to the reality of existing device identity provisioning processes, some manufacturers have stored serial-numbers in other fields. Registrar's SHOULD be configurable, on a per-manufacturer basis, to look for serial-number equivalents in other fields.

As explained in Section 5.5 the Registrar MUST extract the serial-number again itself from the pledge's TLS certificate. It can consult the serial-number in the pledge-request if there are any possible confusion about the source of the serial-number.

2.3.2. MASA URI extension

This document defines a new PKIX non-critical certificate extension to carry the MASA URI. This extension is intended to be used in the IDevID certificate. The URI is represented as described in Section 7.4 of [RFC5280].

The URI provides the authority information. The BRSKI "/.well-known" tree ([RFC5785]) is described in Section 5.

A complete URI MAY be in this extension, including the 'scheme', 'authority', and 'path', The complete URI will typically be used in diagnostic or experimental situations. Typically, (and in consideration to constrained systems), this SHOULD be reduced to only the 'authority', in which case a scheme of "https://" ([RFC7230] section 2.7.3) and 'path' of "/.well-known/est" is to be assumed.

The registrar can assume that only the 'authority' is present in the extension, if there are no slash ("/") characters in the extension.

Section 7.4 of [RFC5280] calls out various schemes that MUST be supported, including LDAP, HTTP and FTP. However, the registrar MUST use HTTPS for the BRSKI-MASA connection.

The new extension is identified as follows:

```
<CODE BEGINS>

MASAURLExtnModule-2016 { iso(1) identified-organization(3) dod(6)
internet(1) security(5) mechanisms(5) pkix(7)
id-mod(0) id-mod-MASAURLExtn2016(TBD) }

DEFINITIONS IMPLICIT TAGS ::= BEGIN

-- EXPORTS ALL --

IMPORTS
EXTENSION
FROM PKIX-CommonTypes-2009
  { iso(1) identified-organization(3) dod(6) internet(1)
    security(5) mechanisms(5) pkix(7) id-mod(0)
    id-mod-pkixCommon-02(57) }

id-pe FROM PKIX1Explicit-2009
  { iso(1) identified-organization(3) dod(6) internet(1)
     security(5) mechanisms(5) pkix(7) id-mod(0)
     id-mod-pkix1-explicit-02(51) } ;

MASACertExtensions EXTENSION ::= { ext-MASAURL, ... }
ext-MASAURL EXTENSION ::= { SYNTAX MASAURLSyntax
IDENTIFIED BY id-pe-masa-url }

id-pe-masa-url OBJECT IDENTIFIER ::= { id-pe TBD }

MASAURLSyntax ::= IA5String

END


<CODE ENDS>
```

Figure 3: MASAURL ASN.1 Module

The choice of id-pe is based on guidance found in Section 4.2.2 of [RFC5280], "These extensions may be used to direct applications to on-line information about the issuer or the subject". The MASA URL is precisely that: online information about the particular subject.

2.4. Protocol Flow

A representative flow is shown in Figure 4

```
+--------+         +---------+    +------------+     +------------+
| Pledge |         | Circuit |    | Domain     |     | Vendor     |
|        |         | Join    |    | Registrar  |     | Service    |
|        |         | Proxy   |    |  (JRC)     |     | (MASA)     |
+--------+         +---------+    +------------+     +------------+
  |                     |                   |           Internet |
[discover]              |                   |                    |
  |<-RFC4862 IPv6 addr  |                   |                    |
  |<-RFC3927 IPv4 addr  | Appendix A        |  Legend            |
  |-++++++++++++++++++->|                   |  C - circuit       |
  | optional: mDNS query| Appendix B        |      join proxy    |
  | RFC6763/RFC6762 (+) |                   |  P - provisional   |
  |<-++++++++++++++++++-|                   |    TLS connection  |
  | GRASP M_FLOOD       |                   |                    |
  |   periodic broadcast|                   |                    |
[identity]              |                   |                    |
  |<------------------->C<----------------->|                    |
  |         TLS via the Join Proxy          |                    |
  |<--Registrar TLS server authentication---|                    |
[PROVISIONAL accept of server cert]         |                    |
  P---X.509 client authentication---------->|                    |
[request join]                              |                    |
  P---Voucher Request(w/nonce for voucher)->|                    |
  P                  /-------------------   |                    |
  P                  |                 [accept device?]          |
  P                  |                 [contact Vendor]          |
  P                  |                      |--Pledge ID-------->|
  P                  |                      |--Domain ID-------->|
  P                  |                      |--optional:nonce--->|
  P              optional:                  |     [extract DomainID]
  P        can occur in advance             |     [update audit log]
  P            if nonceleess                |                    |
  P                  |                      |<- voucher ---------|
  P                  \-------------------   | w/nonce if provided|
  P<------voucher---------------------------|                    |
[imprint]                                   |                    |
  |-------voucher status telemetry--------->|                    |
  |                                         |<-device audit log--|
  |                             [verify audit log and voucher]   |
  |<--------------------------------------->|                    |
[enroll]                                    |                    |
  | Continue with RFC7030 enrollment        |                    |
  | using now bidirectionally authenticated |                    |
  | TLS session.                            |                    |
[enrolled]                                  |                    |
```

Figure 4: Protocol Time Sequence Diagram

On initial bootstrap, a new device (the pledge) uses a local service autodiscovery (GRASP or mDNS) to locate a join proxy. The join proxy connects the pledge to a local registrar (the JRC).

Having found a candidate registrar, the fledgling pledge sends some information about itself to the registrar, including its serial number in the form of a voucher request and its device identity certificate (IDevID) as part of the TLS session.

The registrar can determine whether it expected such a device to appear, and locates a MASA. The location of the MASA is usually found in an extension in the IDevID. Having determined that the MASA is suitable, the entire information from the initial voucher request (including device serial number) is transmitted over the internet in a TLS protected channel to the manufacturer, along with information about the registrar/owner.

The manufacturer can then apply policy based on the provided information, as well as other sources of information (such as sales records), to decide whether to approve the claim by the registrar to own the device; if the claim is accepted, a voucher is issued that directs the device to accept its new owner.

The voucher is returned to the registrar, but not immediately to the device -- the registrar has an opportunity to examine the voucher, the MASA's audit-logs, and other sources of information to determine whether the device has been tampered with, and whether the bootstrap should be accepted.

No filtering of information is possible in the signed voucher, so this is a binary yes-or-no decision. If the registrar accepts the voucher as a proper one for its device, the voucher is returned to the pledge for imprinting.

The voucher also includes a trust anchor that the pledge uses as representing the owner. This is used to successfully bootstrap from an environment where only the manufacturer has built-in trust by the device into an environment where the owner now has a PKI footprint on the device.

When BRSKI is followed with EST this single footprint is further leveraged into the full owner's PKI and a LDevID for the device. Subsequent reporting steps provide flows of information to indicate success/failure of the process.

2.5. Architectural Components

2.5.1. Pledge

The pledge is the device that is attempting to join. The pledge is assumed to talk to the Join Proxy using link-local network connectivity. In most cases, the pledge has no other connectivity until the pledge completes the enrollment process and receives some kind of network credential.

2.5.2. Join Proxy

The join proxy provides HTTPS connectivity between the pledge and the registrar. A circuit proxy mechanism is described in Section 4. Additional mechanisms, including a CoAP mechanism and a stateless IPIP mechanism are the subject of future work.

2.5.3. Domain Registrar

The domain's registrar operates as the BRSKI-MASA client when requesting vouchers from the MASA (see Section 5.4). The registrar operates as the BRSKI-EST server when pledges request vouchers (see Section 5.1). The registrar operates as the BRSKI-EST server "Registration Authority" if the pledge requests an end entity certificate over the BRSKI-EST connection (see Section 5.9).

The registrar uses an Implicit Trust Anchor database for authenticating the BRSKI-MASA connection's MASA TLS Server Certificate. Configuration or distribution of trust anchors is out-of-scope for this specification.

The registrar uses a different Implicit Trust Anchor database for authenticating the BRSKI-EST connection's Pledge TLS Client Certificate. Configuration or distribution of the BRSKI-EST client trust anchors is out-of-scope of this specification. Note that the trust anchors in/excluded from the database will affect which manufacturers' devices are acceptable to the registrar as pledges, and can also be used to limit the set of MASAs that are trusted for enrollment.

2.5.4. Manufacturer Service

The Manufacturer Service provides two logically separate functions: the Manufacturer Authorized Signing Authority (MASA) described in Section 5.5 and Section 5.6, and an ownership tracking/auditing function described in Section 5.7 and Section 5.8.

2.5.5. Public Key Infrastructure (PKI)

The Public Key Infrastructure (PKI) administers certificates for the domain of concern, providing the trust anchor(s) for it and allowing enrollment of pledges with domain certificates.

The voucher provides a method for the distribution of a single PKI trust anchor (as the "pinned-domain-cert"). A distribution of the full set of current trust anchors is possible using the optional EST integration.

The domain's registrar acts as an [RFC5272] Registration Authority, requesting certificates for pledges from the Key Infrastructure.

The expectations of the PKI are unchanged from EST [RFC7030]. This document does not place any additional architectural requirements on the Public Key Infrastructure.

2.6. Certificate Time Validation

2.6.1. Lack of realtime clock

Many devices when bootstrapping do not have knowledge of the current time. Mechanisms such as Network Time Protocols cannot be secured until bootstrapping is complete. Therefore bootstrapping is defined with a framework that does not require knowledge of the current time. A pledge MAY ignore all time stamps in the voucher and in the certificate validity periods if it does not know the current time.

The pledge is exposed to dates in the following five places: registrar certificate notBefore, registrar certificate notAfter, voucher created-on, and voucher expires-on. Additionally, CMS signatures contain a signingTime.

A pledge with a real time clock in which it has confidence, MUST check the above time fields in all certificates and signatures that it processes.

If the voucher contains a nonce then the pledge MUST confirm the nonce matches the original pledge voucher-request. This ensures the voucher is fresh. See Section 5.2.

2.6.2. Infinite Lifetime of IDevID

[RFC5280] explains that long lived pledge certificates "SHOULD be assigned the GeneralizedTime value of 99991231235959Z" for the notAfter field.

Some deployed IDevID management systems are not compliant with the 802.1AR requirement for infinite lifetimes, and put in typical <= 3 year certificate lifetimes. Registrars SHOULD be configurable on a per-manufacturer basis to ignore pledge lifetimes when the pledge did not follow the RFC5280 recommendations.

2.7. Cloud Registrar

There exist operationally open networks wherein devices gain unauthenticated access to the Internet at large. In these use cases the management domain for the device needs to be discovered within the larger Internet. The case where a device can boot and get access to larger Internet are less likely within the ANIMA ACP scope but may be more important in the future. In the ANIMA ACP scope, new devices will be quarantined behind a Join Proxy.

There are additionally some greenfield situations involving an entirely new installation where a device may have some kind of management uplink that it can use (such as via 3G network for instance). In such a future situation, the device might use this management interface to learn that it should configure itself to become the local registrar.

In order to support these scenarios, the pledge MAY contact a well known URI of a cloud registrar if a local registrar cannot be discovered or if the pledge's target use cases do not include a local registrar.

If the pledge uses a well known URI for contacting a cloud registrar a manufacturer-assigned Implicit Trust Anchor database (see [RFC7030]) MUST be used to authenticate that service as described in [RFC6125]. The use of a DNS-ID for validation is appropriate, and it may include wildcard components on the left-mode side. This is consistent with the human user configuration of an EST server URI in [RFC7030] which also depends on RFC6125.

2.8. Determining the MASA to contact

The registrar needs to be able to contact a MASA that is trusted by the pledge in order to obtain vouchers. There are three mechanisms described:

The device's Initial Device Identifier (IDevID) will normally contain the MASA URL as detailed in Section 2.3. This is the RECOMMENDED mechanism.

It can be operationally difficult to ensure the necessary X.509 extensions are in the pledge's IDevID due to the difficulty of aligning current pledge manufacturing with software releases and development. As a final fallback the registrar MAY be manually configured or distributed with a MASA URL for each manufacturer. Note that the registrar can only select the configured MASA URL based on the trust anchor -- so manufacturers can only leverage this approach if they ensure a single MASA URL works for all pledges associated with each trust anchor.

3. Voucher-Request artifact

Voucher-requests are how vouchers are requested. The semantics of the voucher-request are described below, in the YANG model.

A pledge forms the "pledge voucher-request", signs it with it's IDevID and submits it to the registrar.

The registrar in turn forms the "registrar voucher-request", signs it with it's Registrar keypair and submits it to the MASA.

The "proximity-registrar-cert" leaf is used in the pledge voucher-requests. This provides a method for the pledge to assert the registrar's proximity.

This network proximity results from the following properties in the ACP context: the pledge is connected to the Join Proxy (Section 4) using a Link-Local IPv6 connection. While the Join Proxy does not participate in any meaningful sense in the cryptography of the TLS connection (such as via a Channel Binding), the Registrar can observe that the connection is via the private ACP (ULA) address of the join proxy, and could not come from outside the ACP. The Pledge must therefore be at most one IPv6 Link-Local hop away from an existing node on the ACP.

Other users of BRSKI will need to define other kinds of assertions if the network proximity described above does not match their needs.

The "prior-signed-voucher-request" leaf is used in registrar voucher-requests. If present, it is the signed pledge voucher-request artifact. This provides a method for the registrar to forward the pledge's signed request to the MASA. This completes transmission of the signed "proximity-registrar-cert" leaf.

Unless otherwise signaled (outside the voucher-request artifact), the signing structure is as defined for vouchers, see [RFC8366].

3.1. Nonceless Voucher Requests

A registrar MAY also retrieve nonceless vouchers by sending nonceless voucher-requests to the MASA in order to obtain vouchers for use when the registrar does not have connectivity to the MASA. No "prior-signed-voucher-request" leaf would be included. The registrar will also need to know the serial number of the pledge. This document does not provide a mechanism for the registrar to learn that in an automated fashion. Typically this will be done via scanning of bar-code or QR-code on packaging, or via some sales channel integration.

3.2. Tree Diagram

The following tree diagram illustrates a high-level view of a voucher-request document. The voucher-request builds upon the voucher artifact described in [RFC8366]. The tree diagram is described in [RFC8340]. Each node in the diagram is fully described by the YANG module in Section 3.4. Please review the YANG module for a detailed description of the voucher-request format.

```
module: ietf-voucher-request

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

Figure 5: YANG Tree diagram for Voucher-Request

3.3. Examples

This section provides voucher-request examples for illustration purposes. These examples show the JSON prior to CMS wrapping. JSON encoding rules specify that any binary content be base64 encoded ([RFC4648] section 4). The contents of the (base64) encoded certificates have been elided to save space. For detailed examples, see Appendix C.2. These examples conform to the encoding rules defined in [RFC7951].

Example (1) The following example illustrates a pledge voucher-request. The assertion leaf is indicated as 'proximity' and the registrar's TLS server certificate is included in the 'proximity-registrar-cert' leaf. See Section 5.2.

```
{
    "ietf-voucher-request:voucher": {
        "assertion": "proximity",
        "nonce": "62a2e7693d82fcda2624de58fb6722e5",
        "serial-number" : "JADA123456789",
        "created-on": "2017-01-01T00:00:00.000Z",
        "proximity-registrar-cert": "base64encodedvalue=="
    }
}
```

Figure 6: JSON representation of example Voucher-Request

Example (2) The following example illustrates a registrar voucher-request. The 'prior-signed-voucher-request' leaf is populated with the pledge's voucher-request (such as the prior example). The pledge's voucher-request is a binary CMS signed object. In the JSON encoding used here it must be base64 encoded. The nonce and assertion have been carried forward from the pledge request to the registrar request. The serial-number is extracted from the pledge's Client Certificate from the TLS connection. See Section 5.5.

```
{
    "ietf-voucher-request:voucher": {
        "assertion" : "proximity",
        "nonce": "62a2e7693d82fcda2624de58fb6722e5",
        "created-on": "2017-01-01T00:00:02.000Z",
        "idevid-issuer": "base64encodedvalue==",
        "serial-number": "JADA123456789",
        "prior-signed-voucher-request": "base64encodedvalue=="
    }
}
```

Figure 7: JSON representation of example Prior-Signed Voucher-Request

Example (3) The following example illustrates a registrar voucher-request. The 'prior-signed-voucher-request' leaf is not populated with the pledge's voucher-request nor is the nonce leaf. This form might be used by a registrar requesting a voucher when the pledge can not communicate with the registrar (such as when it is powered down, or still in packaging), and therefore could not submit a nonce. This scenario is most useful when the registrar is aware that it will not be able to reach the MASA during deployment. See Section 5.5.

```
{
    "ietf-voucher-request:voucher": {
        "created-on":    "2017-01-01T00:00:02.000Z",
        "idevid-issuer": "base64encodedvalue==",
        "serial-number": "JADA123456789"
    }
}
```


Figure 8: JSON representation of Offline Voucher-Request

3.4. YANG Module

Following is a YANG [RFC7950] module formally extending the [RFC8366] voucher into a voucher-request.

```
<CODE BEGINS> file "ietf-voucher-request@2018-02-14.yang"

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

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

  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 vch;
    description "This module defines the format for a voucher,
        which is produced by a pledge's manufacturer or
        delegate (MASA) to securely assign a pledge to
        an 'owner', so that the pledge may establish a secure
        connection to the owner's network infrastructure";

    reference "RFC 8366: Voucher Artifact for Bootstrapping Protocols";
  }

  organization
   "IETF ANIMA Working Group";

  contact
   "WG Web:   <https://datatracker.ietf.org/wg/anima/>
    WG List:  <mailto:anima@ietf.org>
    Author:   Kent Watsen
              <mailto:kent+ietf@watsen.net>
    Author:   Michael H. Behringer
              <mailto:Michael.H.Behringer@gmail.com>
    Author:   Toerless Eckert
              <mailto:tte+ietf@cs.fau.de>
    Author:   Max Pritikin
              <mailto:pritikin@cisco.com>
    Author:   Michael Richardson
              <mailto:mcr+ietf@sandelman.ca>";

  description
   "This module defines the format for a voucher request.
    It is a superset of the voucher itself.
    It provides content to the MASA for consideration
    during a voucher request.
```

```
     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.

     Copyright (c) 2019 IETF Trust and the persons identified as
     authors of the code. All rights reserved.

     Redistribution and use in source and binary forms, with or without
     modification, is permitted pursuant to, and subject to the license
     terms contained in, the Simplified BSD License set forth in Section
     4.c of the IETF Trust's Legal Provisions Relating to IETF Documents
     (http://trustee.ietf.org/license-info).

     This version of this YANG module is part of RFC XXXX; see the RFC
     itself for full legal notices.";

   revision "2018-02-14" {
     description
      "Initial version";
     reference
      "RFC XXXX: Bootstrapping Remote Secure Key Infrastructure";
   }

   // Top-level statement
   rc:yang-data voucher-request-artifact {
     uses voucher-request-grouping;
   }

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

     uses vch:voucher-artifact-grouping {
       refine "voucher/created-on" {
         mandatory false;
       }

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

       refine "voucher/domain-cert-revocation-checks" {
         description "The domain-cert-revocation-checks field
                      is not valid in a voucher request, and
                      any occurence MUST be ignored";
       }
```

```
refine "voucher/assertion" {
  mandatory false;
  description "Any assertion included in registrar voucher
        requests SHOULD be ignored by the MASA.";
}

augment "voucher"  {
  description
    "Adds leaf nodes appropriate for requesting vouchers.";

  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 voucher request
       with a proximity-registrar-cert, and the registrar
       then includes it as the prior-signed-voucher-request field.
       This is a simple mechanism for a chain of trusted
       parties to change a voucher request, while
       maintaining the prior signature information.

       The Registrar and MASA 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.";
  }

  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.X690.1994].

       The first certificate in the Registrar TLS server
       certificate_list sequence  (the end-entity TLS certificate,
       see [RFC8446]) presented by the Registrar to the Pledge.
       This MUST be populated in a Pledge's voucher request when a
       proximity assertion is requested.";
  }
}
```

```
    }
  }

}


<CODE ENDS>
```

Figure 9: YANG module for Voucher-Request

4. Proxying details (Pledge - Proxy - Registrar)

This section is normative for uses with an ANIMA ACP. The use of the GRASP mechanism is part of the ACP. Other users of BRSKI will need to define an equivalent proxy mechanism, and an equivalent mechanism to configure the proxy.

The role of the proxy is to facilitate communications. The proxy forwards packets between the pledge and a registrar that has been provisioned to the proxy via full GRASP ACP discovery.

This section defines a stateful proxy mechanism which is referred to as a "circuit" proxy. This is a form of Application Level Gateway ([RFC2663] section 2.9).

The proxy does not terminate the TLS handshake: it passes streams of bytes onward without examination. A proxy MUST NOT assume any specific TLS version. Please see [RFC8446] section 9.3 for details on TLS invariants.

A Registrar can directly provide the proxy announcements described below, in which case the announced port can point directly to the Registrar itself. In this scenario the pledge is unaware that there is no proxying occurring. This is useful for Registrars which are servicing pledges on directly connected networks.

As a result of the proxy Discovery process in Section 4.1.1, the port number exposed by the proxy does not need to be well known, or require an IANA allocation.

During the discovery of the Registrar by the Join Proxy, the Join Proxy will also learn which kinds of proxy mechanisms are available. This will allow the Join Proxy to use the lowest impact mechanism which the Join Proxy and Registrar have in common.

In order to permit the proxy functionality to be implemented on the maximum variety of devices the chosen mechanism should use the minimum amount of state on the proxy device. While many devices in the ANIMA target space will be rather large routers, the proxy function is likely to be implemented in the control plane CPU of such a device, with available capabilities for the proxy function similar to many class 2 IoT devices.

The document [I-D.richardson-anima-state-for-joinrouter] provides a more extensive analysis and background of the alternative proxy methods.

4.1. Pledge discovery of Proxy

The result of discovery is a logical communication with a registrar, through a proxy. The proxy is transparent to the pledge. The communication between the pledge and Join Proxy is over IPv6 Link-Local addresses.

To discover the proxy the pledge performs the following actions:

1. MUST: Obtains a local address using IPv6 methods as described in [RFC4862] IPv6 Stateless Address AutoConfiguration. Use of [RFC4941] temporary addresses is encouraged. To limit pervasive monitoring ([RFC7258]), a new temporary address MAY use a short lifetime (that is, set TEMP_PREFERRED_LIFETIME to be short). Pledges will generally prefer use of IPv6 Link-Local addresses, and discovery of proxy will be by Link-Local mechanisms. IPv4 methods are described in Appendix A

2. MUST: Listen for GRASP M_FLOOD ([I-D.ietf-anima-grasp]) announcements of the objective: "AN_Proxy". See section Section 4.1.1 for the details of the objective. The pledge MAY listen concurrently for other sources of information, see Appendix B.

Once a proxy is discovered the pledge communicates with a registrar through the proxy using the bootstrapping protocol defined in Section 5.

While the GRASP M_FLOOD mechanism is passive for the pledge, the non-normative other methods (mDNS, and IPv4 methods) described in Appendix B are active. The pledge SHOULD run those methods in parallel with listening to for the M_FLOOD. The active methods SHOULD back-off by doubling to a maximum of one hour to avoid overloading the network with discovery attempts. Detection of change of physical link status (Ethernet carrier for instance) SHOULD reset the back off timers.

The pledge could discover more than one proxy on a given physical interface. The pledge can have a multitude of physical interfaces as well: a layer-2/3 Ethernet switch may have hundreds of physical ports.

Each possible proxy offer SHOULD be attempted up to the point where a valid voucher is received: while there are many ways in which the attempt may fail, it does not succeed until the voucher has been validated.

The connection attempts via a single proxy SHOULD exponentially back-off to a maximum of one hour to avoid overloading the network infrastructure. The back-off timer for each MUST be independent of other connection attempts.

Connection attempts SHOULD be run in parallel to avoid head of queue problems wherein an attacker running a fake proxy or registrar could perform protocol actions intentionally slowly. Connection attempts to different proxies SHOULD be sent with an interval of 3 to 5s. The pledge SHOULD continue to listen to for additional GRASP M_FLOOD messages during the connection attempts.

Each connection attempt through a distinct Join Proxy MUST have a unique nonce in the voucher-request.

Once a connection to a registrar is established (e.g. establishment of a TLS session key) there are expectations of more timely responses, see Section 5.2.

Once all discovered services are attempted (assuming that none succeeded) the device MUST return to listening for GRASP M_FLOOD. It SHOULD periodically retry any manufacturer-specific mechanisms. The pledge MAY prioritize selection order as appropriate for the anticipated environment.

4.1.1. Proxy GRASP announcements

A proxy uses the DULL GRASP M_FLOOD mechanism to announce itself. This announcement can be within the same message as the ACP announcement detailed in [I-D.ietf-anima-autonomic-control-plane].

The formal Concise Data Definition Language (CDDL) [RFC8610] definition is:

```
flood-message = [M_FLOOD, session-id, initiator, ttl,
                 +[objective, (locator-option / [])]]

objective = ["AN_Proxy", objective-flags, loop-count,
                                       objective-value]

ttl             = 180000     ; 180,000 ms (3 minutes)
initiator = ACP address to contact Registrar
objective-flags   = sync-only  ; as in GRASP spec
sync-only         =  4         ; M_FLOOD only requires synchronization
loop-count        =  1         ; one hop only
objective-value   =  any       ; none

locator-option    = [ O_IPv6_LOCATOR, ipv6-address,
                    transport-proto, port-number ]
ipv6-address      = the v6 LL of the Proxy
$transport-proto /= IPPROTO_TCP   ; note this can be any value from the
                                 ; IANA protocol registry, as per
                                 ; [GRASP] section 2.9.5.1, note 3.
port-number      = selected by Proxy
```

Figure 10: CDDL definition of Proxy Discovery message

Here is an example M_FLOOD announcing a proxy at fe80::1, on TCP port 4443.

```
[M_FLOOD, 12340815, h'fe800000000000000000000000000001', 180000,
            ["AN_Proxy", 4, 1, ""],
            [O_IPv6_LOCATOR,
              h'fe800000000000000000000000000001', IPPROTO_TCP, 4443]]
```

Figure 11: Example of Proxy Discovery message

On a small network the Registrar MAY include the GRASP M_FLOOD announcements to locally connected networks.

The $transport-proto above indicates the method that the pledge-proxy-registrar will use. The TCP method described here is mandatory, and other proxy methods, such as CoAP methods not defined in this document are optional. Other methods MUST NOT be enabled unless the Join Registrar ASA indicates support for them in it's own announcement.

4.2. CoAP connection to Registrar

The use of CoAP to connect from pledge to registrar is out of scope for this document, and is described in future work. See [I-D.ietf-anima-constrained-voucher].

4.3. Proxy discovery and communication of Registrar

The registrar SHOULD announce itself so that proxies can find it and determine what kind of connections can be terminated.

The registrar announces itself using ACP instance of GRASP using M_FLOOD messages. A registrar may announce any convenient port number, including using a stock port 443. ANI proxies MUST support GRASP discovery of registrars.

The M_FLOOD is formatted as follows:

```
[M_FLOOD, 12340815, h'fda379a6f6ee00000200000064000001', 180000,
            ["AN_join_registrar", 4, 255, "EST-TLS"],
            [O_IPv6_LOCATOR,
              h'fda379a6f6ee00000200000064000001', IPPROTO_TCP, 8443]]
```

Figure 12: An example of a Registrar announcement message

The formal CDDL definition is:

```
flood-message = [M_FLOOD, session-id, initiator, ttl,
                 +[objective, (locator-option / [])]]

objective = ["AN_join_registrar", objective-flags, loop-count,
                                       objective-value]

initiator = ACP address to contact Registrar
objective-flags = sync-only  ; as in GRASP spec
sync-only =  4               ; M_FLOOD only requires synchronization
loop-count      = 255        ; mandatory maximum
objective-value = text       ; name of the (list of) of supported
                             ; protocols: "EST-TLS" for RFC7030.
```

Figure 13: CDDL definition for Registrar announcement message

The M_FLOOD message MUST be sent periodically. The default period SHOULD be 60 seconds, the value SHOULD be operator configurable but SHOULD NOT be smaller than 60 seconds. The frequency of sending MUST be such that the aggregate amount of periodic M_FLOODs from all flooding sources cause only negligible traffic across the ACP.

Here are some examples of locators for illustrative purposes. Only the first one ($transport-protocol = 6, TCP) is defined in this document and is mandatory to implement.

```
locator1  = [O_IPv6_LOCATOR, fd45:1345::6789, 6,  443]
locator2  = [O_IPv6_LOCATOR, fd45:1345::6789, 17, 5683]
locator3  = [O_IPv6_LOCATOR, fe80::1234, 41, nil]
```

A protocol of 6 indicates that TCP proxying on the indicated port is desired.

Registrars MUST announce the set of protocols that they support. They MUST support TCP traffic.

Registrars MUST accept HTTPS/EST traffic on the TCP ports indicated.

Registrars MUST support ANI TLS circuit proxy and therefore BRSKI across HTTPS/TLS native across the ACP.

In the ANI, the Autonomic Control Plane (ACP) secured instance of GRASP ([I-D.ietf-anima-grasp]) MUST be used for discovery of ANI registrar ACP addresses and ports by ANI proxies. The TCP leg of the proxy connection between ANI proxy and ANI registrar therefore also runs across the ACP.

5. Protocol Details (Pledge - Registrar - MASA)

The pledge MUST initiate BRSKI after boot if it is unconfigured. The pledge MUST NOT automatically initiate BRSKI if it has been configured or is in the process of being configured.

BRSKI is described as extensions to EST [RFC7030]. The goal of these extensions is to reduce the number of TLS connections and crypto operations required on the pledge. The registrar implements the BRSKI REST interface within the same "/.well-known" URI tree as the existing EST URIs as described in EST [RFC7030] section 3.2.2. The communication channel between the pledge and the registrar is referred to as "BRSKI-EST" (see Figure 1).

The communication channel between the registrar and MASA is similarly described as extensions to EST within the same "/.well-known" tree. For clarity this channel is referred to as "BRSKI-MASA". (See Figure 1).

The MASA URI is "https://" authority "/.well-known/est".

BRSKI uses existing CMS message formats for existing EST operations. BRSKI uses JSON [RFC8259] for all new operations defined here, and voucher formats. In all places where a binary value must be carried in a JSON string, the use of base64 format ([RFC4648] section 4) is to be used, as per [RFC7951] section 6.6.

While EST section 3.2 does not insist upon use of HTTP persistent connections ([RFC7230] section 6.3), BRSKI-EST connections SHOULD use persistent connections. The intention of this guidance is to ensure the provisional TLS state occurs only once, and that the subsequent resolution of the provision state is not subject to a MITM attack during a critical phase.

If non-persistent connections are used, then both the pledge and the registrar MUST remember the certificates seen, and also sent for the first connection. They MUST check each subsequent connections for the same certificates, and each end MUST use the same certificates as well. This places a difficult restriction on rolling certificates on the Registrar.

Summarized automation extensions for the BRSKI-EST flow are:

* The pledge either attempts concurrent connections via each discovered proxy, or it times out quickly and tries connections in series, as explained at the end of Section 5.1.

* The pledge provisionally accepts the registrar certificate during the TLS handshake as detailed in Section 5.1.

* The pledge requests a voucher using the new REST calls described below. This voucher is then validated.

* The pledge completes authentication of the server certificate as detailed in Section 5.6.1. This moves the BRSKI-EST TLS connection out of the provisional state.

* Mandatory bootstrap steps conclude with voucher status telemetry (see Section 5.7).

The BRSKI-EST TLS connection can now be used for EST enrollment.

The extensions for a registrar (equivalent to EST server) are:

* Client authentication is automated using Initial Device Identity (IDevID) as per the EST certificate based client authentication. The subject field's DN encoding MUST include the "serialNumber" attribute with the device's unique serial number as explained in Section 2.3.1

* The registrar requests and validates the voucher from the MASA.

* The registrar forwards the voucher to the pledge when requested.

* The registrar performs log verifications (described in Section 5.8.3) in addition to local authorization checks before accepting optional pledge device enrollment requests.

5.1. BRSKI-EST TLS establishment details

The pledge establishes the TLS connection with the registrar through the circuit proxy (see Section 4) but the TLS handshake is with the registrar. The BRSKI-EST pledge is the TLS client and the BRSKI-EST registrar is the TLS server. All security associations established are between the pledge and the registrar regardless of proxy operations.

Use of TLS 1.3 (or newer) is encouraged. TLS 1.2 or newer is REQUIRED on the Pledge side. TLS 1.3 (or newer) SHOULD be available on the Registrar server interface, and the Registrar client interface, but TLS 1.2 MAY be used. TLS 1.3 (or newer) SHOULD be available on the MASA server interface, but TLS 1.2 MAY be used.

Establishment of the BRSKI-EST TLS connection is as specified in EST [RFC7030] section 4.1.1 "Bootstrap Distribution of CA Certificates" [RFC7030] wherein the client is authenticated with the IDevID certificate, and the EST server (the registrar) is provisionally authenticated with an unverified server certificate. Configuration or distribution of the trust anchor database used for validating the IDevID certificate is out-of-scope of this specification. Note that the trust anchors in/excluded from the database will affect which manufacturers' devices are acceptable to the registrar as pledges, and can also be used to limit the set of MASAs that are trusted for enrollment.

The signatures in the certificate MUST be validated even if a signing key can not (yet) be validated. The certificate (or chain) MUST be retained for later validation.

A self-signed certificate for the Registrar is acceptable as the voucher can validate it upon successful enrollment.

The pledge performs input validation of all data received until a voucher is verified as specified in Section 5.6.1 and the TLS connection leaves the provisional state. Until these operations are complete the pledge could be communicating with an attacker.

The pledge code needs to be written with the assumption that all data is being transmitted at this point to an unauthenticated peer, and that received data, while inside a TLS connection, MUST be considered untrusted. This particularly applies to HTTP headers and CMS structures that make up the voucher.

A pledge that can connect to multiple Registrars concurrently SHOULD do so. Some devices may be unable to do so for lack of threading, or resource issues. Concurrent connections defeat attempts by a malicious proxy from causing a TCP Slowloris-like attack (see [slowloris]).

A pledge that can not maintain as many connections as there are eligible proxies will need to rotate among the various choices, terminating connections that do not appear to be making progress. If no connection is making progress after 5 seconds then the pledge SHOULD drop the oldest connection and go on to a different proxy: the proxy that has been communicated with least recently. If there were no other proxies discovered, the pledge MAY continue to wait, as long as it is concurrently listening for new proxy announcements.

5.2. Pledge Requests Voucher from the Registrar

When the pledge bootstraps it makes a request for a voucher from a registrar.

This is done with an HTTPS POST using the operation path value of "/.well-known/est/requestvoucher".

The pledge voucher-request Content-Type is:

application/voucher-cms+json [RFC8366] defines a "YANG-defined JSON document that has been signed using a CMS structure", and the voucher-request described in Section 3 is created in the same way. The media type is the same as defined in [RFC8366]. This is also used for the pledge voucher-request. The pledge MUST sign the request using the Section 2.3 credential.

Registrar implementations SHOULD anticipate future media types but of course will simply fail the request if those types are not yet known.

The pledge SHOULD include an [RFC7231] section 5.3.2 "Accept" header field indicating the acceptable media type for the voucher response. The "application/voucher-cms+json" media type is defined in [RFC8366] but constrained voucher formats are expected in the future. Registrars and MASA are expected to be flexible in what they accept.

The pledge populates the voucher-request fields as follows:

created-on: Pledges that have a realtime clock are RECOMMENDED to populate this field with the current date and time in yang:date-and-time format. This provides additional information to the MASA. Pledges that have no real-time clocks MAY omit this field.

nonce: The pledge voucher-request MUST contain a cryptographically strong random or pseudo-random number nonce (see [RFC4086] section 6.2). As the nonce is usually generated very early in the boot sequence there is a concern that the same nonce might generated across multiple boots, or after a factory reset. Different nonces MUST be generated for each bootstrapping attempt, whether in series or concurrently. The freshness of this nonce mitigates against the lack of real-time clock as explained in Section 2.6.1.

assertion: The pledge indicates support for the mechanism described in this document, by putting the value "proximity" in the voucher-request, and MUST include the "proximity-registrar-cert" field (below).

proximity-registrar-cert: In a pledge voucher-request this is the first certificate in the TLS server 'certificate_list' sequence (see [RFC5246]) presented by the registrar to the pledge. That is, it is the end-entity certificate. This MUST be populated in a pledge voucher-request.

serial-number The serial number of the pledge is included in the voucher-request from the Pledge. This value is included as a sanity check only, but it is not to be forwarded by the Registrar as described in Section 5.5.

All other fields MAY be omitted in the pledge voucher-request.

An example JSON payload of a pledge voucher-request is in Section 3.3 Example 1.

The registrar confirms that the assertion is 'proximity' and that pinned 'proximity-registrar-cert' is the Registrar's certificate. If this validation fails, then there is an On-Path Attacker (MITM), and the connection MUST be closed after the returning an HTTP 401 error code.

5.3. Registrar Authorization of Pledge

In a fully automated network all devices must be securely identified and authorized to join the domain.

A Registrar accepts or declines a request to join the domain, based on the authenticated identity presented. For different networks, examples of automated acceptance may include:

* allow any device of a specific type (as determined by the X.509 IDevID),

* allow any device from a specific vendor (as determined by the X.509 IDevID),

* allow a specific device from a vendor (as determined by the X.509 IDevID) against a domain white list. (The mechanism for checking a shared white list potentially used by multiple Registrars is out of scope).

If validation fails the registrar SHOULD respond with the HTTP 404 error code. If the voucher-request is in an unknown format, then an HTTP 406 error code is more appropriate. A situation that could be resolved with administrative action (such as adding a vendor to a whitelist) MAY be responded with an 403 HTTP error code.

If authorization is successful the registrar obtains a voucher from the MASA service (see Section 5.5) and returns that MASA signed voucher to the pledge as described in Section 5.6.

5.4. BRSKI-MASA TLS establishment details

The BRSKI-MASA TLS connection is a 'normal' TLS connection appropriate for HTTPS REST interfaces. The registrar initiates the connection and uses the MASA URL obtained as described in Section 2.8. The mechanisms in [RFC6125] SHOULD be used in authentication of the MASA using a DNS-ID that matches that which is found in the IDevID. Registrars MAY include a mechanism to override the MASA URL on a manufacturer-by-manufacturer basis, and within that override it is appropriate to provide alternate anchors. This will typically used by some vendors to establish explicit (or private) trust anchors for validating their MASA that is part of a sales channel integration.

Use of TLS 1.3 (or newer) is encouraged. TLS 1.2 or newer is REQUIRED. TLS 1.3 (or newer) SHOULD be available.

As described in [RFC7030], the MASA and the registrars SHOULD be prepared to support TLS client certificate authentication and/or HTTP Basic or Digest authentication. This connection MAY also have no client authentication at all.

Registrars SHOULD permit trust anchors to be pre-configured on a per-vendor(MASA) basis. Registrars SHOULD include the ability to configure a TLS ClientCertificate on a per-MASA basis, or to use no client certificate. Registrars SHOULD also permit HTTP Basic and Digest authentication to be configured.

The authentication of the BRSKI-MASA connection does not change the voucher-request process, as voucher-requests are already signed by the registrar. Instead, this authentication provides access control to the audit-log as described in Section 5.8.

Implementors are advised that contacting the MASA is to establish a secured API connection with a web service and that there are a number of authentication models being explored within the industry. Registrars are RECOMMENDED to fail gracefully and generate useful administrative notifications or logs in the advent of unexpected HTTP 401 (Unauthorized) responses from the MASA.

5.4.1. MASA authentication of customer Registrar

Providing per-customer options requires that the customer's registrar be uniquely identified. This can be done by any stateless method that HTTPS supports such as with HTTP Basic or Digest authentication (that is using a password), but the use of TLS Client Certificate authentication is RECOMMENDED.

Stateful methods involving API tokens, or HTTP Cookies, are not recommended.

It is expected that the setup and configuration of per-customer Client Certificates is done as part of a sales ordering process.

The use of public PKI (i.e. WebPKI) End-Entity Certificates to identify the Registrar is reasonable, and if done universally this would permit a MASA to identify a customers' Registrar simply by a FQDN.

The use of DANE records in DNSSEC signed zones would also permit use of a FQDN to identify customer Registrars.

A third (and simplest, but least flexible) mechanism would be for the MASA to simply store the Registrar's certificate pinned in a database.

A MASA without any supply chain integration can simply accept Registrars without any authentication, or can accept them on a blind Trust-on-First-Use basis as described in Section 7.4.2.

This document does not make a specific recommendation on how the MASA authenticates the Registrar as there are likely different tradeoffs in different environments and product values. Even within the ANIMA ACP applicability, there is a significant difference between supply chain logistics for $100 CPE devices and $100,000 core routers.

5.5. Registrar Requests Voucher from MASA

When a registrar receives a pledge voucher-request it in turn submits a registrar voucher-request to the MASA service via an HTTPS interface ([RFC7231]).

This is done with an HTTP POST using the operation path value of "/.well-known/est/requestvoucher".

The voucher media type "application/voucher-cms+json" is defined in [RFC8366] and is also used for the registrar voucher-request. It is a JSON document that has been signed using a CMS structure. The registrar MUST sign the registrar voucher-request. The entire registrar certificate chain, up to and including the Domain CA, MUST be included in the CMS structure.

MASA impementations SHOULD anticipate future media types but of course will simply fail the request if those types are not yet known.

The Registrar SHOULD include an [RFC7231] section 5.3.2 "Accept" header field indicating the response media types that are acceptable. This list SHOULD be the entire list presented to the Registrar in the Pledge's original request (see Section 5.2) but MAY be a subset. MASA's are expected to be flexible in what they accept.

The registrar populates the voucher-request fields as follows:

created-on: The Registrars SHOULD populate this field with the current date and time when the Registrar formed this voucher request. This field provides additional information to the MASA.

nonce: This value, if present, is copied from the pledge voucher-request. The registrar voucher-request MAY omit the nonce as per Section 3.1.

serial-number: The serial number of the pledge the registrar would like a voucher for. The registrar determines this value by parsing the authenticated pledge IDevID certificate. See Section 2.3. The registrar MUST verify that the serial number field it parsed matches the serial number field the pledge provided in its voucher-request. This provides a sanity check useful for detecting error conditions and logging. The registrar MUST NOT simply copy the serial number field from a pledge voucher request as that field is claimed but not certified.

idevid-issuer: The Issuer value from the pledge IDevID certificate is included to ensure unique interpretation of the serial-number. In the case of nonceless (offline) voucher-request, then an appropriate value needs to be configured from the same out-of-band source as the serial-number.

prior-signed-voucher-request: The signed pledge voucher-request SHOULD be included in the registrar voucher-request. The entire CMS signed structure is to be included, base64 encoded for transport in the JSON structure.

A nonceless registrar voucher-request MAY be submitted to the MASA. Doing so allows the registrar to request a voucher when the pledge is offline, or when the registrar anticipates not being able to connect to the MASA while the pledge is being deployed. Some use cases require the registrar to learn the appropriate IDevID SerialNumber field and appropriate 'Accept header field' values from the physical device labeling or from the sales channel (out-of-scope for this document).

All other fields MAY be omitted in the registrar voucher-request.

The "proximity-registrar-cert" field MUST NOT be present in the registrar voucher-request.

Example JSON payloads of registrar voucher-requests are in Section 3.3 Examples 2 through 4.

The MASA verifies that the registrar voucher-request is internally consistent but does not necessarily authenticate the registrar certificate since the registrar MAY be unknown to the MASA in advance. The MASA performs the actions and validation checks described in the following sub-sections before issuing a voucher.

5.5.1. MASA renewal of expired vouchers

As described in [RFC8366] vouchers are normally short lived to avoid revocation issues. If the request is for a previous (expired) voucher using the same registrar (that is, a Registrar with the same Domain CA) then the request for a renewed voucher SHOULD be automatically authorized. The MASA has sufficient information to determine this by examining the request, the registrar authentication, and the existing audit-log. The issuance of a renewed voucher is logged as detailed in Section 5.6.

To inform the MASA that existing vouchers are not to be renewed one can update or revoke the registrar credentials used to authorize the request (see Section 5.5.4 and Section 5.5.3). More flexible methods will likely involve sales channel integration and authorizations (details are out-of-scope of this document).

5.5.2. MASA pinning of registrar

The registrar's certificate chain is extracted from the signature method. The entire registrar certificate chain was included in the CMS structure, as specified in Section 5.5. This CA certificate will be used to populate the "pinned-domain-cert" of the voucher being issued.

If this domain CA is unknown to the MASA, then it is to be considered a temporary trust anchor for the rest of the steps in this section. The intention is not to authenticate the message as having come from a fully validated origin, but to establish the consistency of the domain PKI.

5.5.3. MASA checking of voucher request signature

As described in Section 5.5.2, the MASA has extracted Registrar's domain CA. This is used to validate the CMS signature ([RFC5652]) on the voucher-request.

Normal PKIX revocation checking is assumed during voucher-request signature validation. This CA certificate MAY have Certificate Revocation List distribution points, or Online Certificate Status Protocol (OCSP) information ([RFC6960]). If they are present, the MASA MUST be able to reach the relevant servers belonging to the Registrar's domain CA to perform the revocation checks.

The use of OCSP Stapling is preferred.

5.5.4. MASA verification of domain registrar

The MASA MUST verify that the registrar voucher-request is signed by a registrar. This is confirmed by verifying that the id-kp-cmcRA extended key usage extension field (as detailed in EST RFC7030 section 3.6.1) exists in the certificate of the entity that signed the registrar voucher-request. This verification is only a consistency check that the unauthenticated domain CA intended the voucher-request signer to be a registrar. Performing this check provides value to the domain PKI by assuring the domain administrator that the MASA service will only respect claims from authorized Registration Authorities of the domain.

Even when a domain CA is authenticated to the MASA, and there is strong sales channel integration to understand who the legitimate owner is, the above cmcRC check prevents arbitrary End-Entity certificates (such as an LDevID certificate) from having vouchers issued against them.

Other cases of inappropriate voucher issuance are detected by examination of the audit log.

If a nonceless voucher-request is submitted the MASA MUST authenticate the registrar as described in either EST [RFC7030] section 3.2.3, section 3.3.2, or by validating the registrar's certificate used to sign the registrar voucher-request using a configured trust anchor. Any of these methods reduce the risk of DDoS attacks and provide an authenticated identity as an input to sales channel integration and authorizations (details are out-of-scope of this document).

In the nonced case, validation of the Registrar's identity (via TLS Client Certificate or HTTP authentication) MAY be omitted if the device policy is to accept audit-only vouchers.

5.5.5. MASA verification of pledge prior-signed-voucher-request

The MASA MAY verify that the registrar voucher-request includes the 'prior-signed-voucher-request' field. If so the prior-signed-voucher-request MUST include a 'proximity-registrar-cert' that is consistent with the certificate used to sign the registrar voucher-request. Additionally the voucher-request serial-number leaf MUST match the pledge serial-number that the MASA extracts from the signing certificate of the prior-signed-voucher-request. The consistency check described above is checking that the 'proximity-registrar-cert' SPKI fingerprint exists within the registrar voucher-request CMS signature's certificate chain. This is substantially the same as the pin validation described in in [RFC7469] section 2.6, paragraph three.

If these checks succeed the MASA updates the voucher and audit-log assertion leafs with the "proximity" assertion, as defined by [RFC8366] section 5.3.

5.5.6. MASA nonce handling

The MASA does not verify the nonce itself. If the registrar voucher-request contains a nonce, and the prior-signed-voucher-request exists, then the MASA MUST verify that the nonce is consistent. (Recall from above that the voucher-request might not contain a nonce, see Section 5.5 and Section 5.5.4).

The MASA populates the audit-log with the nonce that was verified. If a nonceless voucher is issued, then the audit-log is to be populated with the JSON value "null".

5.6. MASA and Registrar Voucher Response

The MASA voucher response to the registrar is forwarded without changes to the pledge; therefore this section applies to both the MASA and the registrar. The HTTP signaling described applies to both the MASA and registrar responses.

When a voucher request arrives at the registrar, if it has a cached response from the MASA for the corresponding registrar voucher-request, that cached response can be used according to local policy; otherwise the registrar constructs a new registrar voucher-request and sends it to the MASA.

Registrar evaluation of the voucher itself is purely for transparency and audit purposes to further inform log verification (see Section 5.8.3) and therefore a registrar could accept future voucher formats that are opaque to the registrar.

If the voucher-request is successful, the server (MASA responding to registrar or registrar responding to pledge) response MUST contain an HTTP 200 response code. The server MUST answer with a suitable 4xx or 5xx HTTP [RFC7230] error code when a problem occurs. In this case, the response data from the MASA MUST be a plaintext human-readable (UTF-8) error message containing explanatory information describing why the request was rejected.

The registrar MAY respond with an HTTP 202 ("the request has been accepted for processing, but the processing has not been completed") as described in EST [RFC7030] section 4.2.3 wherein the client "MUST wait at least the specified 'Retry-After' time before repeating the same request". (see [RFC7231] section 6.6.4) The pledge is RECOMMENDED to provide local feedback (blinked LED etc) during this wait cycle if mechanisms for this are available. To prevent an attacker registrar from significantly delaying bootstrapping the pledge MUST limit the 'Retry-After' time to 60 seconds. Ideally the pledge would keep track of the appropriate Retry-After header field values for any number of outstanding registrars but this would involve a state table on the pledge. Instead the pledge MAY ignore the exact Retry-After value in favor of a single hard coded value (a registrar that is unable to complete the transaction after the first 60 seconds has another chance a minute later). A pledge SHOULD only maintain a 202 retry-state for up to 4 days, which is longer than a long weekend, after which time the enrollment attempt fails and the pledge returns to discovery state.

A pledge that retries a request after receiving a 202 message MUST resend the same voucher-request. It MUST NOT sign a new voucher-request each time, and in particular, it MUST NOT change the nonce value.

In order to avoid infinite redirect loops, which a malicious registrar might do in order to keep the pledge from discovering the correct registrar, the pledge MUST NOT follow more than one redirection (3xx code) to another web origins. EST supports redirection but requires user input; this change allows the pledge to follow a single redirection without a user interaction.

A 403 (Forbidden) response is appropriate if the voucher-request is not signed correctly, stale, or if the pledge has another outstanding voucher that cannot be overridden.

A 404 (Not Found) response is appropriate when the request is for a device that is not known to the MASA.

A 406 (Not Acceptable) response is appropriate if a voucher of the desired type or using the desired algorithms (as indicated by the Accept: header fields, and algorithms used in the signature) cannot be issued such as because the MASA knows the pledge cannot process that type. The registrar SHOULD use this response if it determines the pledge is unacceptable due to inventory control, MASA audit-logs, or any other reason.

A 415 (Unsupported Media Type) response is appropriate for a request that has a voucher-request or Accept: value that is not understood.

The voucher response format is as indicated in the submitted Accept header fields or based on the MASA's prior understanding of proper format for this Pledge. Only the [RFC8366] "application/voucher-cms+json" media type is defined at this time. The syntactic details of vouchers are described in detail in [RFC8366]. Figure 14 shows a sample of the contents of a voucher.


```
{
  "ietf-voucher:voucher": {
    "nonce": "62a2e7693d82fcda2624de58fb6722e5",
    "assertion": "logged",
    "pinned-domain-cert": "base64encodedvalue==",
    "serial-number": "JADA123456789"
  }
}
```


Figure 14: An example voucher

The MASA populates the voucher fields as follows:

nonce: The nonce from the pledge if available. See Section 5.5.6.

assertion: The method used to verify the relationship between pledge and registrar. See Section 5.5.5.

pinned-domain-cert: The domain CA cert. See Section 5.5.2. This figure is illustrative, for an example, see Appendix C.2

serial-number: The serial-number as provided in the voucher-request. Also see Section 5.5.5.

domain-cert-revocation-checks: Set as appropriate for the pledge's capabilities and as documented in [RFC8366]. The MASA MAY set this field to 'false' since setting it to 'true' would require that revocation information be available to the pledge and this document does not make normative requirements for [RFC6961] or equivalent integrations.

expires-on: This is set for nonceless vouchers. The MASA ensures the voucher lifetime is consistent with any revocation or pinned-domain-cert consistency checks the pledge might perform. See section Section 2.6.1. There are three times to consider: (a) a configured voucher lifetime in the MASA, (b) the expiry time for the registrar's certificate, (c) any certificate revocation information (CRL) lifetime. The expires-on field SHOULD be before the earliest of these three values. Typically (b) will be some significant time in the future, but (c) will typically be short (on the order of a week or less). The RECOMMENDED period for (a) is on the order of 20 minutes, so it will typically determine the lifespan of the resulting voucher. 20 minutes is sufficient time to reach the post-provisional state in the pledge, at which point there is an established trust relationship between pledge and registrar. The subsequent operations can take as long as required from that point onwards. The lifetime of the voucher has no impact on the lifespan of the ownership relationship.

Whenever a voucher is issued the MASA MUST update the audit-log sufficiently to generate the response as described in Section 5.8.1. The internal state requirements to maintain the audit-log are out-of-scope.

5.6.1. Pledge voucher verification

The pledge MUST verify the voucher signature using the manufacturer-installed trust anchor(s) associated with the manufacturer's MASA (this is likely included in the pledge's firmware). Management of the manufacturer-installed trust anchor(s) is out-of-scope of this document; this protocol does not update these trust anchor(s).

The pledge MUST verify the serial-number field of the signed voucher matches the pledge's own serial-number.

The pledge MUST verify the nonce information in the voucher. If present, the nonce in the voucher must match the nonce the pledge submitted to the registrar; vouchers with no nonce can also be accepted (according to local policy, see Section 7.2.

The pledge MUST be prepared to parse and fail gracefully from a voucher response that does not contain a 'pinned-domain-cert' field. Such a thing indicates a failure to enroll in this domain, and the pledge MUST attempt joining with other available Join Proxy.

The pledge MUST be prepared to ignore additional fields that it does not recognize.

5.6.2. Pledge authentication of provisional TLS connection

The 'pinned-domain-cert' element of the voucher contains the domain CA's public key. The pledge MUST use the 'pinned-domain-cert' trust anchor to immediately complete authentication of the provisional TLS connection.

If a registrar's credentials cannot be verified using the pinned-domain-cert trust anchor from the voucher then the TLS connection is immediately discarded and the pledge abandons attempts to bootstrap with this discovered registrar. The pledge SHOULD send voucher status telemetry (described below) before closing the TLS connection. The pledge MUST attempt to enroll using any other proxies it has found. It SHOULD return to the same proxy again after unsuccessful attempts with other proxies. Attempts should be made repeated at intervals according to the backoff timer described earlier. Attempts SHOULD be repeated as failure may be the result of a temporary inconsistency (an inconsistently rolled registrar key, or some other mis-configuration). The inconsistency could also be the result an active MITM attack on the EST connection.

The registrar MUST use a certificate that chains to the pinned-domain-cert as its TLS server certificate.

The pledge's PKIX path validation of a registrar certificate's validity period information is as described in Section 2.6.1. Once the PKIX path validation is successful the TLS connection is no longer provisional.

The pinned-domain-cert MAY be installed as an trust anchor for future operations such as enrollment (e.g. [RFC7030] as recommended) or trust anchor management or raw protocols that do not need full PKI based key management. It can be used to authenticate any dynamically discovered EST server that contain the id-kp-cmcRA extended key usage extension as detailed in EST RFC7030 section 3.6.1; but to reduce system complexity the pledge SHOULD avoid additional discovery operations. Instead the pledge SHOULD communicate directly with the registrar as the EST server. The 'pinned-domain-cert' is not a complete distribution of the [RFC7030] section 4.1.3 CA Certificate Response, which is an additional justification for the recommendation to proceed with EST key management operations. Once a full CA Certificate Response is obtained it is more authoritative for the domain than the limited 'pinned-domain-cert' response.

5.7. Pledge BRSKI Status Telemetry

The domain is expected to provide indications to the system administrators concerning device lifecycle status. To facilitate this it needs telemetry information concerning the device's status.

To indicate pledge status regarding the voucher, the pledge MUST post a status message to the Registrar.

The posted data media type: application/json

The client sends an HTTP POST to the server at the URI ".well-known/est/voucher_status".

The format and semantics described below are for version 1. A version field is included to permit significant changes to this feedback in the future. A Registrar that receives a status message with a version larger than it knows about SHOULD log the contents and alert a human.

The Status field indicates if the voucher was acceptable. Boolean values are acceptable, where "true" indicates the voucher was acceptable.

If the voucher was not acceptable the Reason string indicates why. In the failure case this message may be sent to an unauthenticated, potentially malicious registrar and therefore the Reason string SHOULD NOT provide information beneficial to an attacker. The operational benefit of this telemetry information is balanced against the operational costs of not recording that an voucher was ignored by a client the registrar expected to continue joining the domain.

The reason-context attribute is an arbitrary JSON object (literal value or hash of values) which provides additional information specific to this pledge. The contents of this field are not subject to standardization.

The version and status fields MUST be present. The Reason field SHOULD be present whenever the status field is false. The Reason-Context field is optional.

The keys to this JSON object are case-sensitive and MUST be lowercase. Figure 15 shows an example JSON.

```
{
    "version":"1",
    "status":false,
    "reason":"Informative human readable message",
    "reason-context": { "additional" : "JSON" }
}
```

Figure 15: Example Status Telemetry

The server SHOULD respond with an HTTP 200 but MAY simply fail with an HTTP 404 error. The client ignores any response. Within the server logs the server SHOULD capture this telemetry information.

Additional standard JSON fields in this POST MAY be added, see Section 8.5. A server that sees unknown fields should log them, but otherwise ignore them.

5.8. Registrar audit-log request

After receiving the pledge status telemetry Section 5.7, the registrar SHOULD request the MASA audit-log from the MASA service.

This is done with an HTTP POST using the operation path value of "/.well-known/est/requestauditlog".

The registrar SHOULD HTTP POST the same registrar voucher-request as it did when requesting a voucher (using the same Content-Type). It is posted to the /requestauditlog URI instead. The "idevid-issuer" and "serial-number" informs the MASA which log is requested so the appropriate log can be prepared for the response. Using the same media type and message minimizes cryptographic and message operations although it results in additional network traffic. The relying MASA implementation MAY leverage internal state to associate this request with the original, and by now already validated, voucher-request so as to avoid an extra crypto validation.

A registrar MAY request logs at future times. If the registrar generates a new request then the MASA is forced to perform the additional cryptographic operations to verify the new request.

A MASA that receives a request for a device that does not exist, or for which the requesting owner was never an owner returns an HTTP 404 ("Not found") code.

It is reasonable for a Registrar, that the MASA does not believe to be the current owner, to request the audit-log. There are probably reasons for this which are hard to predict in advance. For instance, such a registrar may not be aware that the device has been resold; it may be that the device has been resold inappropriately, and this is how the original owner will learn of the occurance. It is also possible that the device legitimately spends time in two different networks.

Rather than returning the audit-log as a response to the POST (with a return code 200), the MASA MAY instead return a 201 ("Created") response ([RFC7231] sections 6.3.2 and 7.1), with the URL to the prepared (and idempotent, therefore cachable) audit response in the Location: header field.

In order to avoid enumeration of device audit-logs, MASA that return URLs SHOULD take care to make the returned URL unguessable. [W3C.WD-capability-urls-20140218] provides very good additional guidance. For instance, rather than returning URLs containing a database number such as https://example.com/auditlog/1234 or the EUI of the device such https://example.com/auditlog/10-00-00-11-22-33, the MASA SHOULD return a randomly generated value (a "slug" in web parlance). The value is used to find the relevant database entry.

A MASA that returns a code 200 MAY also include a Location: header for future reference by the registrar.

5.8.1. MASA audit log response

A log data file is returned consisting of all log entries associated with the device selected by the IDevID presented in the request. The audit log may be abridged by removal of old or repeated values as explained below. The returned data is in JSON format ([RFC8259]), and the Content-Type SHOULD be "application/json".

The following CDDL ([RFC8610]) explains the structure of the JSON format audit-log response:

```
audit-log-response = {
  "version": uint,
  "events": [ + event ]
  "truncation": {
    ? "nonced duplicates": uint,
    ? "nonceless duplicates": uint,
    ? "arbitrary": uint,
  }
}

event = {
  "date": text,
  "domainID": text,
  "nonce": text / null,
  "assertion": "verified" / "logged" / "proximity",
  ? "truncated": uint,
}
```

Figure 16: CDDL for audit-log response

An example:

```
{
  "version":"1",
  "events":[
    {
        "date":"2019-05-15T17:25:55.644-04:00",
        "domainID":"BduJhdHPpfhQLyponf48JzXSGZ8=",
        "nonce":"VOUFT-WwrEv0NuAQEHoV7Q",
        "assertion":"proximity",
        "truncated":"0"
    },
    {
        "date":"2017-05-15T17:25:55.644-04:00",
        "domainID":"BduJhdHPpfhQLyponf48JzXSGZ8=",
        "nonce":"f4G6Vi1t8nKo/FieCVgpBg==",
        "assertion":"proximity"
    }
  ],
    "truncation": {
        "nonced duplicates": "0",
        "nonceless duplicates": "1",
        "arbitrary": "2"
     }
}
```

Figure 17: Example of audit-log response

The domainID is a binary SubjectKeyIdentifier value calculated according to Section 5.8.2. It is encoded once in base64 in order to be transported in this JSON container.

The date is in [RFC3339] format, which is consistent with typical JavaScript usage of JSON.

The truncation structure MAY be omitted if all values are zero. Any counter missing from the truncation structure is the be assumed to be zero.

The nonce is a string, as provided in the voucher-request, and used in the voucher. If no nonce was placed in the resulting voucher, then a value of null SHOULD be used in preference to omitting the entry. While the nonce is often created as a base64 encoded random series of bytes, this should not be assumed.

Distribution of a large log is less than ideal. This structure can be optimized as follows: Nonced or Nonceless entries for the same domainID MAY be abridged from the log leaving only the single most recent nonced or nonceless entry for that domainID. In the case of truncation the 'event' truncation value SHOULD contain a count of the number of events for this domainID that were omitted. The log

SHOULD NOT be further reduced but there could exist operational situation where maintaining the full log is not possible. In such situations the log MAY be arbitrarily abridged for length, with the number of removed entries indicated as 'arbitrary'.

If the truncation count exceeds 1024 then the MASA MAY use this value without further incrementing it.

A log where duplicate entries for the same domain have been omitted ("nonced duplicates" and/or "nonceless duplicates) could still be acceptable for informed decisions. A log that has had "arbitrary" truncations is less acceptable but manufacturer transparency is better than hidden truncations.

A registrar that sees a version value greater than 1 indicates an audit log format that has been enhanced with additional information. No information will be removed in future versions; should an incompatible change be desired in the future, then a new HTTP end point will be used.

This document specifies a simple log format as provided by the MASA service to the registrar. This format could be improved by distributed consensus technologies that integrate vouchers with technologies such as block-chain or hash trees or optimized logging approaches. Doing so is out of the scope of this document but is an anticipated improvement for future work. As such, the registrar SHOULD anticipate new kinds of responses, and SHOULD provide operator controls to indicate how to process unknown responses.

5.8.2. Calculation of domainID

The domainID is a binary value (a BIT STRING) that uniquely identifies a Registrar by the "pinned-domain-cert"

If the "pinned-domain-cert" certificate includes the SubjectKeyIdentifier (Section 4.2.1.2 [RFC5280]), then it is to be used as the domainID. If not, the SPKI Fingerprint as described in [RFC7469] section 2.4 is to be used. This value needs to be calculated by both MASA (to populate the audit-log), and by the Registrar (to recognize itself in the audit log).

[RFC5280] section 4.2.1.2 does not mandate that the SubjectKeyIdentifier extension be present in non-CA certificates. It is RECOMMENDED that Registrar certificates (even if self-signed), always include the SubjectKeyIdentifier to be used as a domainID.

The domainID is determined from the certificate chain associated with the pinned-domain-cert and is used to update the audit-log.

5.8.3. Registrar audit log verification

Each time the Manufacturer Authorized Signing Authority (MASA) issues a voucher, it appends details of the assignment to an internal audit log for that device. The internal audit log is processed when responding to requests for details as described in Section 5.8. The contents of the audit log can express a variety of trust levels, and this section explains what kind of trust a registrar can derive from the entries.

While the audit log provides a list of vouchers that were issued by the MASA, the vouchers are issued in response to voucher-requests, and it is the contents of the voucher-requests which determines how meaningful the audit log entries are.

A registrar SHOULD use the log information to make an informed decision regarding the continued bootstrapping of the pledge. The exact policy is out of scope of this document as it depends on the security requirements within the registrar domain. Equipment that is purchased pre-owned can be expected to have an extensive history. The following discussion is provided to help explain the value of each log element:

date: The date field provides the registrar an opportunity to divide the log around known events such as the purchase date. Depending on context known to the registrar or administrator events before/after certain dates can have different levels of importance. For example for equipment that is expected to be new, and thus have no history, it would be a surprise to find prior entries.

domainID: If the log includes an unexpected domainID then the pledge could have imprinted on an unexpected domain. The registrar can be expected to use a variety of techniques to define "unexpected" ranging from white lists of prior domains to anomaly detection (e.g. "this device was previously bound to a different domain than any other device deployed"). Log entries can also be compared against local history logs in search of discrepancies (e.g. "this device was re-deployed some number of times internally but the external audit log shows additional re-deployments our internal logs are unaware of").

nonce: Nonceless entries mean the logged domainID could theoretically trigger a reset of the pledge and then take over management by using the existing nonceless voucher.

assertion: The assertion leaf in the voucher and audit log indicates why the MASA issued the voucher. A "verified" entry means that the MASA issued the associated voucher as a result of positive verification of ownership. However, this entry does not indicate whether the pledge was actually deployed in the prior domain, or not. A "logged" assertion informs the registrar that the prior vouchers were issued with minimal verification. A "proximity" assertion assures the registrar that the pledge was truly communicating with the prior domain and thus provides assurance that the prior domain really has deployed the pledge.

A relatively simple policy is to white list known (internal or external) domainIDs, and require all vouchers to have a nonce. An alternative is to require that all nonceless vouchers be from a subset (e.g. only internal) of domainIDs. If the policy is violated a simple action is to revoke any locally issued credentials for the pledge in question or to refuse to forward the voucher. The Registrar MUST then refuse any EST actions, and SHOULD inform a human via a log. A registrar MAY be configured to ignore (i.e. override the above policy) the history of the device but it is RECOMMENDED that this only be configured if hardware assisted (i.e. TPM anchored) Network Endpoint Assessment (NEA) [RFC5209] is supported.

5.9. EST Integration for PKI bootstrapping

The pledge SHOULD follow the BRSKI operations with EST enrollment operations including "CA Certificates Request", "CSR Attributes" and "Client Certificate Request" or "Server-Side Key Generation", etc. This is a relatively seamless integration since BRSKI API calls provide an automated alternative to the manual bootstrapping method described in [RFC7030]. As noted above, use of HTTP persistent connections simplifies the pledge state machine.

Although EST allows clients to obtain multiple certificates by sending multiple Certificate Signing Requests (CSR) requests, BRSKI does not support this mechanism directly. This is because BRSKI pledges MUST use the CSR Attributes request ([RFC7030] section 4.5). The registrar MUST validate the CSR against the expected attributes. This implies that client requests will "look the same" and therefore result in a single logical certificate being issued even if the client were to make multiple requests. Registrars MAY contain more complex logic but doing so is out-of-scope of this specification. BRSKI does not signal any enhancement or restriction to this capability.

5.9.1. EST Distribution of CA Certificates

The pledge SHOULD request the full EST Distribution of CA Certificates message. See RFC7030, section 4.1.

This ensures that the pledge has the complete set of current CA certificates beyond the pinned-domain-cert (see Section 5.6.2 for a discussion of the limitations inherent in having a single certificate instead of a full CA Certificates response.) Although these limitations are acceptable during initial bootstrapping, they are not appropriate for ongoing PKIX end entity certificate validation.

5.9.2. EST CSR Attributes

Automated bootstrapping occurs without local administrative configuration of the pledge. In some deployments it is plausible that the pledge generates a certificate request containing only identity information known to the pledge (essentially the X.509 IDevID information) and ultimately receives a certificate containing domain specific identity information. Conceptually the CA has complete control over all fields issued in the end entity certificate. Realistically this is operationally difficult with the current status of PKI certificate authority deployments, where the CSR is submitted to the CA via a number of non-standard protocols. Even with all standardized protocols used, it could operationally be problematic to expect that service specific certificate fields can be created by a CA that is likely operated by a group that has no insight into different network services/protocols used. For example, the CA could even be outsourced.

To alleviate these operational difficulties, the pledge MUST request the EST "CSR Attributes" from the EST server and the EST server needs to be able to reply with the attributes necessary for use of the certificate in its intended protocols/services. This approach allows for minimal CA integrations and instead the local infrastructure (EST server) informs the pledge of the proper fields to include in the generated CSR (such as rfc822Name). This approach is beneficial to automated bootstrapping in the widest number of environments.

In networks using the BRSKI enrolled certificate to authenticate the ACP (Autonomic Control Plane), the EST CSR attributes MUST include the ACP Domain Information Fields defined in [I-D.ietf-anima-autonomic-control-plane] section 6.1.1.

The registrar MUST also confirm that the resulting CSR is formatted as indicated before forwarding the request to a CA. If the registrar is communicating with the CA using a protocol such as full CMC, which provides mechanisms to override the CSR attributes, then these mechanisms MAY be used even if the client ignores CSR Attribute guidance.

5.9.3. EST Client Certificate Request

The pledge MUST request a new client certificate. See RFC7030, section 4.2.

5.9.4. Enrollment Status Telemetry

For automated bootstrapping of devices, the administrative elements providing bootstrapping also provide indications to the system administrators concerning device lifecycle status. This might include information concerning attempted bootstrapping messages seen by the client. The MASA provides logs and status of credential enrollment. [RFC7030] assumes an end user and therefore does not include a final success indication back to the server. This is insufficient for automated use cases.

In order to communicate this indicator, the client HTTP POSTs a JSON dictionary with a number of attributes described below to the new EST endpoint at "/.well-known/est/enrollstatus".

When indicating a successful enrollment the client SHOULD first re-establish the EST TLS session using the newly obtained credentials. TLS 1.2 supports doing this in-band, but TLS 1.3 does not. The client SHOULD therefore always close the existing TLS connection, and start a new one.

In the case of a failed enrollment, the client MUST send the telemetry information over the same TLS connection that was used for the enrollment attempt, with a Reason string indicating why the most recent enrollment failed. (For failed attempts, the TLS connection is the most reliable way to correlate server-side information with what the client provides.)

The reason-context attribute is an arbitrary JSON object (literal value or hash of values) which provides additional information specific to the failure to unroll from this pledge. The contents of this field are not subject to standardization. This is represented by the group-socket "$$arbitrary-map" in the CDDL.

In the case of a SUCCESS the Reason string is omitted.

```
enrollstatus-post = {
    "version": uint,
    "status": bool,
    "reason": text,
    ? "reason-context" : { $$arbitrary-map }
  }
}
```

Figure 18: CDDL for enrollment status POST

An example status report can be seen below. It is sent with with the media type: application/json


```
{
    "version":"1",
    "status":true,
    "reason":"Informative human readable message",
    "reason-context": { "additional" : "JSON" }
}
```


Figure 19: Example of enrollment status POST

The server SHOULD respond with an HTTP 200 but MAY simply fail with an HTTP 404 error.

Within the server logs the server MUST capture if this message was received over an TLS session with a matching client certificate.

5.9.5. Multiple certificates

Pledges that require multiple certificates could establish direct EST connections to the registrar.

5.9.6. EST over CoAP

This document describes extensions to EST for the purposes of bootstrapping of remote key infrastructures. Bootstrapping is relevant for CoAP enrollment discussions as well. The definition of EST and BRSKI over CoAP is not discussed within this document beyond ensuring proxy support for CoAP operations. Instead it is anticipated that a definition of CoAP mappings will occur in subsequent documents such as [I-D.ietf-ace-coap-est] and that CoAP mappings for BRSKI will be discussed either there or in future work.

6. Clarification of transfer-encoding

[RFC7030] defines its endpoints to include a "Content-Transfer-Encoding" heading, and the payloads to be [RFC4648] Base64 encoded DER.

When used within BRSKI, the original RFC7030 EST endpoints remain Base64 encoded, but the new BRSKI end points which send and receive binary artifacts (specifically, "/.well-known/est/requestvoucher") are binary. That is, no encoding is used.

In the BRSKI context, the EST "Content-Transfer-Encoding" header field if present, SHOULD be ignored. This header field does not need to be included.

7. Reduced security operational modes

A common requirement of bootstrapping is to support less secure operational modes for support specific use cases. This section suggests a range of mechanisms that would alter the security assurance of BRSKI to accommodate alternative deployment architectures and mitigate lifecycle management issues identified in Section 10. They are presented here as informative (non-normative) design guidance for future standardization activities. Section 9 provides standardization applicability statements for the ANIMA ACP. Other users would be expected that subsets of these mechanisms could be profiled with an accompanying applicability statements similar to the one described in Section 9.

This section is considered non-normative in the generality of the protocol. Use of the suggested mechanisms here MUST be detailed in specific profiles of BRSKI, such as in Section 9.

7.1. Trust Model

This section explains the trust relationships detailed in Section 2.4:

```
+--------+         +---------+    +------------+     +------------+
| Pledge |         | Join    |    | Domain     |     |Manufacturer|
|        |         | Proxy   |    | Registrar  |     | Service    |
|        |         |         |    |            |     | (Internet) |
+--------+         +---------+    +------------+     +------------+
```

Figure 10

Pledge: The pledge could be compromised and providing an attack vector for malware. The entity is trusted to only imprint using secure methods described in this document. Additional endpoint assessment techniques are RECOMMENDED but are out-of-scope of this document.

Join Proxy: Provides proxy functionalities but is not involved in security considerations.

Registrar: When interacting with a MASA a registrar makes all decisions. For Ownership Audit Vouchers (see [RFC8366]) the registrar is provided an opportunity to accept MASA decisions.

Vendor Service, MASA: This form of manufacturer service is trusted to accurately log all claim attempts and to provide authoritative log information to registrars. The MASA does not know which devices are associated with which domains. These claims could be strengthened by using cryptographic log techniques to provide append only, cryptographic assured, publicly auditable logs.

Vendor Service, Ownership Validation: This form of manufacturer service is trusted to accurately know which device is owned by which domain.

7.2. Pledge security reductions

The following is a list of alternative behaviours that the pledge can be programmed to implement. These behaviours are not mutually exclusive, nor are they dependent upon each other. Some of these methods enable offline and emergency (touch based) deployment use cases. Normative language is used as these behaviours are referenced in later sections in a normative fashion.

1. The pledge MUST accept nonceless vouchers. This allows for a use case where the registrar can not connect to the MASA at the deployment time. Logging and validity periods address the security considerations of supporting these use cases.

2. Many devices already support "trust on first use" for physical interfaces such as console ports. This document does not change that reality. Devices supporting this protocol MUST NOT support "trust on first use" on network interfaces. This is because "trust on first use" over network interfaces would undermine the logging based security protections provided by this specification.

3. The pledge MAY have an operational mode where it skips voucher validation one time. For example if a physical button is depressed during the bootstrapping operation. This can be useful if the manufacturer service is unavailable. This behavior SHOULD be available via local configuration or physical presence methods (such as use of a serial/craft console) to ensure new entities can always be deployed even when autonomic methods fail. This allows for unsecured imprint.

4. A craft/serial console could include a command such as "est-enroll [2001:db8:0:1]:443" that begins the EST process from the point after the voucher is validated. This process SHOULD include server certificate verification using an on-screen fingerprint.

It is RECOMMENDED that "trust on first use" or any method of skipping voucher validation (including use of craft serial console) only be available if hardware assisted Network Endpoint Assessment (NEA: [RFC5209]) is supported. This recommendation ensures that domain network monitoring can detect inappropriate use of offline or emergency deployment procedures when voucher-based bootstrapping is not used.

7.3. Registrar security reductions

A registrar can choose to accept devices using less secure methods. They MUST NOT be the default behavior. These methods may be acceptable in situations where threat models indicate that low security is adequate. This includes situations where security decisions are being made by the local administrator:

1. A registrar MAY choose to accept all devices, or all devices of a particular type, at the administrator's discretion. This could occur when informing all registrars of unique identifiers of new entities might be operationally difficult.

2. A registrar MAY choose to accept devices that claim a unique identity without the benefit of authenticating that claimed identity. This could occur when the pledge does not include an X.509 IDevID factory installed credential. New Entities without an X.509 IDevID credential MAY form the Section 5.2 request using the Section 5.5 format to ensure the pledge's serial number information is provided to the registrar (this includes the IDevID AuthorityKeyIdentifier value, which would be statically configured on the pledge.) The pledge MAY refuse to provide a TLS client certificate (as one is not available.) The pledge SHOULD support HTTP-based or certificate-less TLS authentication as described in EST RFC7030 section 3.3.2. A registrar MUST NOT accept unauthenticated New Entities unless it has been configured to do so by an administrator that has verified that only expected new entities can communicate with a registrar (presumably via a physically secured perimeter.)

3. A registrar MAY submit a nonceless voucher-requests to the MASA service (by not including a nonce in the voucher-request.) The resulting vouchers can then be stored by the registrar until they are needed during bootstrapping operations. This is for use cases where the target network is protected by an air gap and therefore cannot contact the MASA service during pledge deployment.

4. A registrar MAY ignore unrecognized nonceless log entries. This could occur when used equipment is purchased with a valid history being deployed in air gap networks that required offline vouchers.

5. A registrar MAY accept voucher formats of future types that can not be parsed by the Registrar. This reduces the Registrar's

visibility into the exact voucher contents but does not change the protocol operations.

7.4. MASA security reductions

Lower security modes chosen by the MASA service affect all device deployments unless the lower-security behavior is tied to specific device identities. The modes described below can be applied to specific devices via knowledge of what devices were sold. They can also be bound to specific customers (independent of the device identity) by authenticating the customer's Registrar.

7.4.1. Issuing Nonceless vouchers

A MASA has the option of not including a nonce in the voucher, and/or not requiring one to be present in the voucher-request. This results in distribution of a voucher that may never expire and in effect makes the specified Domain an always trusted entity to the pledge during any subsequent bootstrapping attempts. That a nonceless voucher was issued is captured in the log information so that the registrar can make appropriate security decisions when a pledge joins the Domain. Nonceless vouchers are useful to support use cases where registrars might not be online during actual device deployment.

While a nonceless voucher may include an expiry date, a typical use for a nonceless voucher is for it to be long-lived. If the device can be trusted to have an accurate clock (the MASA will know), then a nonceless voucher CAN be issued with a limited lifetime.

A more typical case for a nonceless voucher is for use with offline onboarding scenarios where it is not possible to pass a fresh voucher-request to the MASA. The use of a long-lived voucher also eliminates concern about the availability of the MASA many years in the future. Thus many nonceless vouchers will have no expiry dates.

Thus, the long lived nonceless voucher does not require the proof that the device is online. Issuing such a thing is only accepted when the registrar is authenticated by the MASA and the MASA is authorized to provide this functionality to this customer. The MASA is RECOMMENDED to use this functionality only in concert with an enhanced level of ownership tracking, the details of which are out of scope for this document.

If the pledge device is known to have a real-time-clock that is set from the factory, use of a voucher validity period is RECOMMENDED.

7.4.2. Trusting Owners on First Use

A MASA has the option of not verifying ownership before responding with a voucher. This is expected to be a common operational model because doing so relieves the manufacturer providing MASA services from having to track ownership during shipping and supply chain and allows for a very low overhead MASA service. A registrar uses the audit log information as a defense in depth strategy to ensure that this does not occur unexpectedly (for example when purchasing new equipment the registrar would throw an error if any audit log information is reported.) The MASA SHOULD verify the 'prior-signed-voucher-request' information for pledges that support that functionality. This provides a proof-of-proximity check that reduces the need for ownership verification. The proof-of-proximity comes from the assumption that the pledge and Join Proxy are on the same link-local connection.

A MASA that practices Trust-on-First-Use (TOFU) for Registrar identity may wish to annotate the origin of the connection by IP address or netblock, and restrict future use of that identity from other locations. A MASA that does this SHOULD take care to not create nuisance situations for itself when a customer has multiple registrars, or uses outgoing IPv4 NAT44 connections that change frequently.

7.4.3. Updating or extending voucher trust anchors

This section deals with the problem of a MASA that is no longer available due to a failed business, or the situation where a MASA is uncooperative to a secondary sale.

A manufacturer could offer a management mechanism that allows the list of voucher verification trust anchors to be extended. [I-D.ietf-netconf-keystore] is one such interface that could be implemented using YANG. Pretty much any configuration mechanism used today could be extended to provide the needed additional update. A manufacturer could even decide to install the domain CA trust anchors received during the EST "cacerts" step as voucher verification anchors. Some additional signals will be needed to clearly identify which keys have voucher validation authority from among those signed by the domain CA. This is future work.

With the above change to the list of anchors, vouchers can be issued by an alternate MASA. This could be the previous owner (the seller), or some other trusted third party who is mediating the sale. If it was a third party, then the seller would need to have taken steps to introduce the third party configuration to the device prior disconnection. The third party (e.g. a wholesaler of used equipment) could however use a mechanism described in Section 7.2 to take control of the device after receiving it physically. This would permit the third party to act as the MASA for future onboarding actions. As the IDevID certificate probably can not be replaced, the new owner's Registrar would have to support an override of the MASA URL.

To be useful for resale or other transfers of ownership one of two situations will need to occur. The simplest is that the device is not put through any kind of factory default/reset before going through onboarding again. Some other secure, physical signal would be needed to initiate it. This is most suitable for redeploying a device within the same Enterprise. This would entail having previous configuration in the system until entirely replaced by the new owner, and represents some level of risk.

The second mechanism is that there would need to be two levels of factory reset. One would take the system back entirely to manufacturer state, including removing any added trust anchors, and the second (more commonly used) one would just restore the configuration back to a known default without erasing trust anchors. This weaker factory reset might leave valuable credentials on the device and this may be unacceptable to some owners.

As a third option, the manufacturer's trust anchors could be entirely overwritten with local trust anchors. A factory default would never restore those anchors. This option comes with a lot of power, but also a lot of responsibility: if access to the private part of the new anchors are lost the manufacturer may be unable to help.

8. IANA Considerations

This document requires the following IANA actions:

8.1. The IETF XML Registry

This document registers a URI in the "IETF XML Registry" [RFC3688]. IANA is asked to register the following:

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

8.2. YANG Module Names Registry

This document registers a YANG module in the "YANG Module Names" registry [RFC6020]. IANA is asked to register the following:

```
name:         ietf-voucher-request
namespace:    urn:ietf:params:xml:ns:yang:ietf-voucher-req
prefix:       vch
reference:    THIS DOCUMENT
```

8.3. Well-known EST registration

This document extends the definitions of "est" (so far defined via RFC7030) in the "https://www.iana.org/assignments/well-known-uris/well-known-uris.xhtml" registry. IANA is asked to change the registration of "est" to include RFC7030 and this document.

8.4. PKIX Registry

IANA is requested to register the following:

This document requests a number for id-mod-MASAURLExtn2016(TBD) from the pkix(7) id-mod(0) Registry.

This document has received an early allocation from the id-pe registry (SMI Security for PKIX Certificate Extension) for id-pe-masa-url with the value 32, resulting in an OID of 1.3.6.1.5.5.7.1.32.

8.5. Pledge BRSKI Status Telemetry

IANA is requested to create a new Registry entitled: "BRSKI Parameters", and within that Registry to create a table called: "Pledge BRSKI Status Telemetry Attributes". New items can be added using the Specification Required. The following items are to be in the initial registration, with this document (Section 5.7) as the reference:

* version

* Status

* Reason

* reason-context

8.6. DNS Service Names

IANA is requested to register the following Service Names:

Service Name: brski-proxy
Transport Protocol(s): tcp
Assignee: IESG <iesg@ietf.org>.
Contact: IESG <iesg@ietf.org>
Description: The Bootstrapping Remote Secure Key
             Infrastructures Proxy
Reference: [This document]

Service Name: brski-registrar
Transport Protocol(s): tcp
Assignee: IESG <iesg@ietf.org>.
Contact: IESG <iesg@ietf.org>
Description: The Bootstrapping Remote Secure Key
             Infrastructures Registrar
Reference: [This document]

9. Applicability to the Autonomic Control Plane (ACP)

This document provides a solution to the requirements for secure bootstrap set out in Using an Autonomic Control Plane for Stable Connectivity of Network Operations, Administration, and Maintenance [RFC8368], A Reference Model for Autonomic Networking [I-D.ietf-anima-reference-model] and specifically the An Autonomic Control Plane (ACP) [I-D.ietf-anima-autonomic-control-plane], section 3.2 (Secure Bootstrap), and section 6.1 (ACP Domain, Certificate and Network).

The protocol described in this document has appeal in a number of other non-ANIMA use cases. Such uses of the protocol will be deploying into other environments with different tradeoffs of privacy, security, reliability and autonomy from manufacturers. As such those use cases will need to provide their own applicability statements, and will need to address unique privacy and security considerations for the environments in which they are used.

The autonomic control plane (ACP) that is bootstrapped by the BRSKI protocol is typically used in medium to large Internet Service Provider organizations. Equivalent enterprises that have significant layer-3 router connectivity also will find significant benefit, particularly if the Enterprise has many sites. (A network consisting of primarily layer-2 is not excluded, but the adjacencies that the ACP will create and maintain will not reflect the topology until all devices participate in the ACP).

In the ACP, the Join Proxy is found to be proximal because communication between the pledge and the join proxy is exclusively on IPv6 Link-Local addresses. The proximity of the Join Proxy to the Registrar is validated by the Registrar using ANI ACP IPv6 Unique Local Addresses (ULA). ULAs are not routable over the Internet, so as long as the Join Proxy is operating correctly the proximity asssertion is satisfied. Other uses of BRSKI will need make similar analysis if they use proximity assertions.

As specified in the ANIMA charter, this work "..focuses on professionally-managed networks." Such a network has an operator and can do things like install, configure and operate the Registrar function. The operator makes purchasing decisions and is aware of what manufacturers it expects to see on its network.

Such an operator is also capable of performing bootstrapping of a device using a serial-console (craft console). The zero-touch mechanism presented in this and the ACP document [I-D.ietf-anima-autonomic-control-plane] represents a significiant efficiency: in particular it reduces the need to put senior experts on airplanes to configure devices in person.

There is a recognition as the technology evolves that not every situation may work out, and occasionally a human may still have to visit. In recognition of this, some mechanisms are presented in Section 7.2. The manufacturer MUST provide at least one of the one-touch mechanisms described that permit enrollment to be proceed without availability of any manufacturer server (such as the MASA).

The BRSKI protocol is going into environments where there have already been quite a number of vendor proprietary management systems. Those are not expected to go away quickly, but rather to leverage the secure credentials that are provisioned by BRSKI. The connectivity requirements of said management systems are provided by the ACP.

9.1. Operational Requirements

This section collects operational requirements based upon the three roles involved in BRSKI: The Manufacturer Authorized Signing Authority (MASA), the (Domain) Owner and the Device. It should be recognized that the manufacturer may be involved in two roles, as it creates the software/firmware for the device, and also may be the operator of the MASA.

The requirements in this section are presented using BCP14 ([RFC2119], [RFC8174]) language. These do not represent new normative statements, just a review of a few such things in one place by role. They also apply specifically to the ANIMA ACP use case. Other use cases likely have similar, but MAY have different requirements.

9.1.1. MASA Operational Requirements

The manufacturer MUST arrange for an online service to be available called the MASA. It MUST be available at the URL which is encoded in the IDevID certificate extensions described in Section 2.3.2.

The online service MUST have access to a private key with which to sign [RFC8366] format voucher artifacts. The public key, certificate, or certificate chain MUST be built in to the device as part of the firmware.

It is RECOMMENDED that the manufacturer arrange for this signing key (or keys) to be escrowed according to typical software source code escrow practices [softwareescrow].

The MASA accepts voucher requests from Domain Owners according to an operational practice appropriate for the device. This can range from any domain owner (first-come first-served, on a TOFU-like basis), to full sales channel integration where Domain Owners need to be positively identified by TLS Client Certicate pinned, or HTTP Authentication process. The MASA creates signed voucher artifacts according to its internally defined policies.

The MASA MUST operate an audit log for devices that is accessible. The audit log is designed to be easily cacheable and the MASA MAY find it useful to put this content on a CDN.

9.1.2. Domain Owner Operational Requirements

The domain owner MUST operate an EST ([RFC7030]) server with the extensions described in this document. This is the JRC or Registrar. This JRC/EST server MUST announce itself using GRASP within the ACP. This EST server will typically reside with the Network Operations Center for the organization.

The domain owner MAY operate an internal certificate authority (CA) that is seperate from the EST server, or it MAY combine all activities into a single device. The determination of the architecture depends upon the scale and resiliency requirements of the organization. Multiple JRC instances MAY be announced into the ACP from multiple locations to achieve an appropriate level of redundancy.

In order to recognize which devices and which manufacturers are welcome on the domain owner's network, the domain owner SHOULD maintain a white list of manufacturers. This MAY extend to integration with purchasing departments to know the serial numbers of devices.

The domain owner SHOULD use the resulting overlay ACP network to manage devices, replacing legacy out-of-band mechanisms.

The domain owner SHOULD operate one or more EST servers which can be used to renew the domain certificates (LDevIDs) which are deployed to devices. These servers MAY be the same as the JRC, or MAY be a distinct set of devices, as approriate for resiliency.

The organization MUST take appropriate precautions against loss of access to the certificate authority private key. Hardware security modules and/or secret splitting are appropriate.

9.1.3. Device Operational Requirements

Devices MUST come with built-in trust anchors that permit the device to validate vouchers from the MASA.

Device MUST come with (unique, per-device) IDevID certificates that include their serial numbers, and the MASA URL extension.

Devices are expected to find Join Proxies using GRASP, and then connect to the JRC using the protocol described in this document.

Once a domain owner has been validated with the voucher, devices are expected to enroll into the domain using EST. Devices are then expected to form ACPs using IPsec over IPv6 Link-Local addresses as described in [I-D.ietf-anima-autonomic-control-plane]

Once a device has been enrolled it SHOULD listen for the address of the JRC using GRASP, and it SHOULD enable itself as a Join Proxy, and announce itself on all links/interfaces using GRASP DULL.

Devices are expected to renew their certificates before they expire.

10. Privacy Considerations

10.1. MASA audit log

The MASA audit log includes the domainID for each domain a voucher has been issued to. This information is closely related to the actual domain identity. A MASA may need additional defenses against Denial of Service attacks (Section 11.1), and this may involve collecting additional (unspecified here) information. This could provide sufficient information for the MASA service to build a detailed understanding the devices that have been provisioned within a domain.

There are a number of design choices that mitigate this risk. The domain can maintain some privacy since it has not necessarily been authenticated and is not authoritatively bound to the supply chain.

Additionally the domainID captures only the unauthenticated subject key identifier of the domain. A privacy sensitive domain could theoretically generate a new domainID for each device being deployed. Similarly a privacy sensitive domain would likely purchase devices that support proximity assertions from a manufacturer that does not require sales channel integrations. This would result in a significant level of privacy while maintaining the security characteristics provided by Registrar based audit log inspection.

10.2. What BRSKI-EST reveals

During the provisional phase of the BRSKI-EST connection between the Pledge and the Registrar, each party reveals its certificates to each other. For the Pledge, this includes the serialNumber attribute, the MASA URL, and the identity that signed the IDevID certificate.

TLS 1.2 reveals the certificate identities to on-path observers, including the Join Proxy.

TLS 1.3 reveals the certificate identities only to the end parties, but as the connection is provisional, an on-path attacker (MTIM) can see the certificates. This includes not just malicious attackers, but also Registrars that are visible to the Pledge, but which are not part of the intended domain.

The certificate of the Registrar is rather arbitrary from the point of view of the BRSKI protocol. As no [RFC6125] validations are expected to be done, the contents could be easily pseudonymized. Any device that can see a join proxy would be able to connect to the Registrar and learn the identity of the network in question. Even if the contents of the certificate are pseudonymized, it would be possible to correlate different connections in different locations belong to the same entity. This is unlikely to present a significant privacy concern to ANIMA ACP uses of BRSKI, but may be a concern to other users of BRSKI.

The certificate of the Pledge could be revealed by a malicious Join Proxy that performed a MITM attack on the provisional TLS connection. Such an attacker would be able to reveal the identity of the Pledge to third parties if it chose to so.

Research into a mechanism to do multi-step, multi-party authenticated key agreement, incorporating some kind of zero-knowledge proof would be valuable. Such a mechanism would ideally avoid disclosing identities until pledge, registrar and MASA agree to the transaction. Such a mechanism would need to discover the location of the MASA without knowing the identity of the pledge, or the identity of the MASA. This part of the problem may be unsolveable.

10.3. What BRSKI-MASA reveals to the manufacturer

With consumer-oriented devices, the "call-home" mechanism in IoT devices raises significant privacy concerns. See [livingwithIoT] and [IoTstrangeThings] for exemplars. The Autonomic Control Plane (ACP) usage of BRSKI is not targeted at individual usage of IoT devices, but rather at the Enterprise and ISP creation of networks in a zero-touch fashion where the "call-home" represents a different class of privacy and lifecycle management concerns.

As the Autonomic Control Plane (ACP) usage of BRSKI is not targeted at individual usage of IoT devices, but rather at the Enterprise and ISP creation of networks in a zero-touch fashion, the "call-home" represents a different kind of concern.

It needs to be re-iterated that the BRSKI-MASA mechanism only occurs once during the commissioning of the device. It is well defined, and although encrypted with TLS, it could in theory be made auditable as the contents are well defined. This connection does not occur when the device powers on or is restarted for normal routines. (It is conceivable, but remarkably unusual, that a device could be forced to go through a full factory reset during an exceptional firmware update situation, after which enrollment would have be repeated, and a new connection would occur)

The BRSKI call-home mechanism is mediated via the owner's Registrar, and the information that is transmitted is directly auditable by the device owner. This is in stark contrast to many "call-home" protocols where the device autonomously calls home and uses an undocumented protocol.

While the contents of the signed part of the pledge voucher request can not be changed, they are not encrypted at the registrar. The ability to audit the messages by the owner of the network is a mechanism to defend against exfiltration of data by a nefarious pledge. Both are, to re-iterate, encrypted by TLS while in transit.

The BRSKI-MASA exchange reveals the following information to the manufacturer:

* the identity of the device being enrolled. This is revealed by transmission of a signed voucher-request containing the serial-number. The manufacturer can usually link the serial number to a device model.

* an identity of the domain owner in the form of the domain trust anchor. However, this is not a global PKI anchored name within

the WebPKI, so this identity could be pseudonymous. If there is sales channel integration, then the MASA will have authenticated the domain owner, either via pinned certificate, or perhaps another HTTP authentication method, as per Section 5.5.4.

* the time the device is activated,

* the IP address of the domain Owner's Registrar. For ISPs and Enterprises, the IP address provides very clear geolocation of the owner. No amount of IP address privacy extensions ([RFC4941]) can do anything about this, as a simple whois lookup likely identifies the ISP or Enterprise from the upper bits anyway. A passive attacker who observes the connection definitely may conclude that the given enterprise/ISP is a customer of the particular equipment vendor. The precise model that is being enrolled will remain private.

Based upon the above information, the manufacturer is able to track a specific device from pseudonymous domain identity to the next pseudonymous domain identity. If there is sales-channel integration, then the identities are not pseudonymous.

The manufacturer knows the IP address of the Registrar, but it can not see the IP address of the device itself. The manufacturer can not track the device to a detailed physical or network location, only to the location of the Registrar. That is likely to be at the Enterprise or ISPs headquarters.

The above situation is to be distinguished from a residential/ individual person who registers a device from a manufacturer. Individuals do not tend to have multiple offices, and their registrar is likely on the same network as the device. A manufacturer that sells switching/routing products to enterprises should hardly be surprised if additional purchases switching/routing products are made. Deviations from a historical trend or an establish baseline would, however, be notable.

The situation is not improved by the enterprise/ISP using anonymization services such as ToR [Dingledine2004], as a TLS 1.2 connection will reveal the ClientCertificate used, clearly identifying the enterprise/ISP involved. TLS 1.3 is better in this regard, but an active attacker can still discover the parties involved by performing a Man-In-The-Middle-Attack on the first attempt (breaking/killing it with a TCP RST), and then letting subsequent connection pass through.

A manufacturer could attempt to mix the BRSKI-MASA traffic in with general traffic their site by hosting the MASA behind the same (set) of load balancers that the companies normal marketing site is hosted behind. This makes lots of sense from a straight capacity planning point of view as the same set of services (and the same set of Distributed Denial of Service mitigations) may be used. Unfortunately, as the BRSKI-MASA connections include TLS ClientCertificate exchanges, this may easily be observed in TLS 1.2, and a traffic analysis may reveal it even in TLS 1.3. This does not make such a plan irrelevant. There may be other organizational reasons to keep the marketing site (which is often subject to frequent re-designs, outsourcing, etc.) separate from the MASA, which may need to operate reliably for decades.

10.4. Manufacturers and Used or Stolen Equipment

As explained above, the manufacturer receives information each time that a device which is in factory-default mode does a zero-touch bootstrap, and attempts to enroll into a domain owner's registrar.

The manufacturer is therefore in a position to decline to issue a voucher if it detects that the new owner is not the same as the previous owner.

1. This can be seen as a feature if the equipment is believed to have been stolen. If the legitimate owner notifies the manufacturer of the theft, then when the new owner brings the device up, if they use the zero-touch mechanism, the new (illegitimate) owner reveals their location and identity.

2. In the case of Used equipment, the initial owner could inform the manufacturer of the sale, or the manufacturer may just permit resales unless told otherwise. In which case, the transfer of ownership simply occurs.

3. A manufacturer could however decide not to issue a new voucher in response to a transfer of ownership. This is essentially the same as the stolen case, with the manufacturer having decided that the sale was not legitimate.

4. There is a fourth case, if the manufacturer is providing protection against stolen devices. The manufacturer then has a responsibility to protect the legitimate owner against fraudulent claims that the equipment was stolen. In the absence of such manufacturer protection, such a claim would cause the manufacturer to refuse to issue a new voucher. Should the device go through a deep factory reset (for instance, replacement of a damaged main board component, the device would not bootstrap.

5. Finally, there is a fifth case: the manufacturer has decided to end-of-line the device, or the owner has not paid a yearly support amount, and the manufacturer refuses to issue new

vouchers at that point. This last case is not new to the industry: many license systems are already deployed that have significantly worse effect.

This section has outlined five situations in which a manufacturer could use the voucher system to enforce what are clearly license terms. A manufacturer that attempted to enforce license terms via vouchers would find it rather ineffective as the terms would only be enforced when the device is enrolled, and this is not (to repeat), a daily or even monthly occurrence.

10.5. Manufacturers and Grey market equipment

Manufacturers of devices often sell different products into different regional markets. Which product is available in which market can be driven by price differentials, support issues (some markets may require manuals and tech-support to be done in the local language), government export regulation (such as whether strong crypto is permitted to be exported, or permitted to be used in a particular market). When an domain owner obtains a device from a different market (they can be new) and transfers it to a different location, this is called a Grey Market.

A manufacturer could decide not to issue a voucher to an enterprise/ ISP based upon their location. There are a number of ways which this could be determined: from the geolocation of the registrar, from sales channel knowledge about the customer, and what products are (un-)available in that market. If the device has a GPS the coordinates of the device could even be placed into an extension of the voucher.

The above actions are not illegal, and not new. Many manufacturers have shipped crypto-weak (exportable) versions of firmware as the default on equipment for decades. The first task of an enterprise/ ISP has always been to login to a manufacturer system, show one's "entitlement" (country information, proof that support payments have been made), and receive either a new updated firmware, or a license key that will activate the correct firmware.

BRSKI permits the above process to automated (in an autonomic fashion), and therefore perhaps encourages this kind of differentiation by reducing the cost of doing it.

An issue that manufacturers will need to deal with in the above automated process is when a device is shipped to one country with one set of rules (or laws or entitlements), but the domain registry is in another one. Which rules apply is something will have to be worked out: the manufacturer could come to believe they are dealing with Grey market equipment, when it is simply dealing with a global enterprise.

10.6. Some mitigations for meddling by manufacturers

The most obvious mitigation is not to buy the product. Pick manufacturers that are up-front about their policies, who do not change them gratuitously.

Section 7.4.3 describes some ways in which a manufacturer could provide a mechanism to manage the trust anchors and built-in certificates (IDevID) as an extension. There are a variety of mechanism, and some may take a substantial amount of work to get exactly correct. These mechanisms do not change the flow of the protocol described here, but rather allow the starting trust assumptions to be changed. This is an area for future standardization work.

Replacement of the voucher validation anchors (usually pointing to the original manufacturer's MASA) with those of the new owner permits the new owner to issue vouchers to subsequent owners. This would be done by having the selling (old) owner to run a MASA.

The BRSKI protocol depends upon a trust anchor on the device and an identity on the device. Management of these entities facilitates a few new operational modes without making any changes to the BRSKI protocol. Those modes include: offline modes where the domain owner operates an internal MASA for all devices, resell modes where the first domain owner becomes the MASA for the next (resold-to) domain owner, and services where an aggregator acquires a large variety of devices, and then acts as a pseudonymized MASA for a variety of devices from a variety of manufacturers.

Although replacement of the IDevID is not required for all modes described above, a manufacturers could support such a thing. Some may wish to consider replacement of the IDevID as an indication that the device's warrantee is terminated. For others, the privacy requirements of some deployments might consider this a standard operating practice.

As discussed at the end of Section 5.8.1, new work could be done to use a distributed consensus technology for the audit log. This would permit the audit log to continue to be useful, even when there is a chain of MASA due to changes of ownership.

10.7. Death of a manufacturer

A common concern has been that a manufacturer could go out of business, leaving owners of devices unable to get new vouchers for existing products. Said products might have been previously deployed, but need to be re-initialized, they might have been purchased used, or they might have kept in a warehouse as long-term spares.

The MASA was named the Manufacturer *Authorized* Signing Authority to emphasize that it need not be the manufacturer itself that performs this. It is anticipated that specialist service providers will come to exist that deal with the creation of vouchers in much the same way that many companies have outsourced email, advertising and janitorial services.

Further, it is expected that as part of any service agreement that the manufacturer would arrange to escrow appropriate private keys such that a MASA service could be provided by a third party. This has routinely been done for source code for decades.

11. Security Considerations

This document details a protocol for bootstrapping that balances operational concerns against security concerns. As detailed in the introduction, and touched on again in Section 7, the protocol allows for reduced security modes. These attempt to deliver additional control to the local administrator and owner in cases where less security provides operational benefits. This section goes into more detail about a variety of specific considerations.

To facilitate logging and administrative oversight, in addition to triggering Registrar verification of MASA logs, the pledge reports on voucher parsing status to the registrar. In the case of a failure, this information is informative to a potentially malicious registrar. This is mandated anyway because of the operational benefits of an informed administrator in cases where the failure is indicative of a problem. The registrar is RECOMMENDED to verify MASA logs if voucher status telemetry is not received.

To facilitate truly limited clients EST RFC7030 section 3.3.2 requirements that the client MUST support a client authentication model have been reduced in Section 7 to a statement that the registrar "MAY" choose to accept devices that fail cryptographic authentication. This reflects current (poor) practices in shipping devices without a cryptographic identity that are NOT RECOMMENDED.

During the provisional period of the connection the pledge MUST treat all HTTP header and content data as untrusted data. HTTP libraries are regularly exposed to non-secured HTTP traffic: mature libraries should not have any problems.

Pledges might chose to engage in protocol operations with multiple discovered registrars in parallel. As noted above they will only do so with distinct nonce values, but the end result could be multiple vouchers issued from the MASA if all registrars attempt to claim the device. This is not a failure and the pledge choses whichever voucher to accept based on internal logic. The registrars verifying log information will see multiple entries and take this into account for their analytics purposes.

11.1. Denial of Service (DoS) against MASA

There are uses cases where the MASA could be unavailable or uncooperative to the Registrar. They include active DoS attacks, planned and unplanned network partitions, changes to MASA policy, or other instances where MASA policy rejects a claim. These introduce an operational risk to the Registrar owner in that MASA behavior might limit the ability to bootstrap a pledge device. For example this might be an issue during disaster recovery. This risk can be mitigated by Registrars that request and maintain long term copies of "nonceless" vouchers. In that way they are guaranteed to be able to bootstrap their devices.

The issuance of nonceless vouchers themselves creates a security concern. If the Registrar of a previous domain can intercept protocol communications then it can use a previously issued nonceless voucher to establish management control of a pledge device even after having sold it. This risk is mitigated by recording the issuance of such vouchers in the MASA audit log that is verified by the subsequent Registrar and by Pledges only bootstrapping when in a factory default state. This reflects a balance between enabling MASA independence during future bootstrapping and the security of bootstrapping itself. Registrar control over requesting and auditing nonceless vouchers allows device owners to choose an appropriate balance.

The MASA is exposed to DoS attacks wherein attackers claim an unbounded number of devices. Ensuring a registrar is representative of a valid manufacturer customer, even without validating ownership of specific pledge devices, helps to mitigate this. Pledge signatures on the pledge voucher-request, as forwarded by the registrar in the prior-signed-voucher-request field of the registrar voucher-request, significantly reduce this risk by ensuring the MASA can confirm proximity between the pledge and the registrar making the request. Supply chain integration ("know your customer") is an additional step that MASA providers and device vendors can explore.

11.2. DomainID must be resistant to second-preimage attacks

The domainID is used as the reference in the audit log to the domain. The domainID is expected to be calculated by a hash that is resistant to a second-preimage attack. Such an attack would allow a second registrar to create audit log entries that are fake.

11.3. Availability of good random numbers

The nonce used by the Pledge in the voucher-request SHOULD be generated by a Strong Cryptographic Sequence ([RFC4086] section 6.2). TLS has a similar requirement.

In particular implementations should pay attention to the advance in [RFC4086] section 3, particularly section 3.4. The random seed used by a device at boot MUST be unique across all devices and all bootstraps. Resetting a device to factory default state does not obviate this requirement.

11.4. Freshness in Voucher-Requests

A concern has been raised that the pledge voucher-request should contain some content (a nonce) provided by the registrar and/or MASA in order for those actors to verify that the pledge voucher-request is fresh.

There are a number of operational problems with getting a nonce from the MASA to the pledge. It is somewhat easier to collect a random value from the registrar, but as the registrar is not yet vouched for, such a registrar nonce has little value. There are privacy and logistical challenges to addressing these operational issues, so if such a thing were to be considered, it would have to provide some clear value. This section examines the impacts of not having a fresh pledge voucher-request.

Because the registrar authenticates the pledge, a full Man-in-the-Middle attack is not possible, despite the provisional TLS authentication by the pledge (see Section 5.) Instead we examine the case of a fake registrar (Rm) that communicates with the pledge in parallel or in close time proximity with the intended registrar. (This scenario is intentionally supported as described in Section 4.1.)

The fake registrar (Rm) can obtain a voucher signed by the MASA either directly or through arbitrary intermediaries. Assuming that the MASA accepts the registrar voucher-request (either because Rm is collaborating with a legitimate registrar according to supply chain information, or because the MASA is in audit-log only mode), then a voucher linking the pledge to the registrar Rm is issued.

Such a voucher, when passed back to the pledge, would link the pledge to registrar Rm, and would permit the pledge to end the provisional state. It now trusts Rm and, if it has any security vulnerabilities leveragable by an Rm with full administrative control, can be assumed to be a threat against the intended registrar.

This flow is mitigated by the intended registrar verifying the audit logs available from the MASA as described in Section 5.8. Rm might chose to collect a voucher-request but wait until after the intended registrar completes the authorization process before submitting it. This pledge voucher-request would be 'stale' in that it has a nonce that no longer matches the internal state of the pledge. In order to successfully use any resulting voucher the Rm would need to remove the stale nonce or anticipate the pledge's future nonce state. Reducing the possibility of this is why the pledge is mandated to generate a strong random or pseudo-random number nonce.

Additionally, in order to successfully use the resulting voucher the Rm would have to attack the pledge and return it to a bootstrapping enabled state. This would require wiping the pledge of current configuration and triggering a re-bootstrapping of the pledge. This is no more likely than simply taking control of the pledge directly but if this is a consideration the target network is RECOMMENDED to take the following steps:

* Ongoing network monitoring for unexpected bootstrapping attempts by pledges.

* Retrieval and examination of MASA log information upon the occurrence of any such unexpected events. Rm will be listed in the logs along with nonce information for analysis.

11.5. Trusting manufacturers

The BRSKI extensions to EST permit a new pledge to be completely configured with domain specific trust anchors. The link from built-in manufacturer-provided trust anchors to domain-specific trust anchors is mediated by the signed voucher artifact.

If the manufacturer's IDevID signing key is not properly validated, then there is a risk that the network will accept a pledge that should not be a member of the network. As the address of the manufacturer's MASA is provided in the IDevID using the extension from Section 2.3, the malicious pledge will have no problem collaborating with it's MASA to produce a completely valid voucher.

BRSKI does not, however, fundamentally change the trust model from domain owner to manufacturer. Assuming that the pledge used its IDevID with RFC7030 EST and BRSKI, the domain (registrar) still needs to trust the manufacturer.

Establishing this trust between domain and manufacturer is outside the scope of BRSKI. There are a number of mechanisms that can adopted including:

* Manually configuring each manufacturer's trust anchor.

* A Trust-On-First-Use (TOFU) mechanism. A human would be queried upon seeing a manufacturer's trust anchor for the first time, and then the trust anchor would be installed to the trusted store. There are risks with this; even if the key to name mapping is validated using something like the WebPKI, there remains the possibility that the name is a look alike: e.g, dem0.example. vs demO.example.

* scanning the trust anchor from a QR code that came with the packaging (this is really a manual TOFU mechanism)

* some sales integration process where trust anchors are provided as part of the sales process, probably included in a digital packing "slip", or a sales invoice.

* consortium membership, where all manufacturers of a particular device category (e.g, a light bulb, or a cable-modem) are signed by an certificate authority specifically for this. This is done by CableLabs today. It is used for authentication and authorization as part of TR-79: [docsisroot] and [TR069].

The existing WebPKI provides a reasonable anchor between manufacturer name and public key. It authenticates the key. It does not provide a reasonable authorization for the manufacturer, so it is not directly useable on it's own.

11.6. Manufacturer Maintenance of trust anchors

BRSKI depends upon the manufacturer building in trust anchors to the pledge device. The voucher artifact which is signed by the MASA will be validated by the pledge using that anchor. This implies that the manufacturer needs to maintain access to a signing key that the pledge can validate.

The manufacturer will need to maintain the ability to make signatures that can be validated for the lifetime that the device could be onboarded. Whether this onboarding lifetime is less than the device lifetime depends upon how the device is used. An inventory of devices kept in a warehouse as spares might not be onboarded for many decades.

There are good cryptographic hygiene reasons why a manufacturer would not want to maintain access to a private key for many decades. A manufacturer in that situation can leverage a long-term certificate authority anchor, built-in to the pledge, and then a certificate chain may be incorporated using the normal CMS certificate set. This may increase the size of the voucher artifacts, but that is not a significant issues in non-constrained environments.

There are a few other operational variations that manufacturers could consider. For instance, there is no reason that every device need have the same set of trust anchors pre-installed. Devices built in different factories, or on different days, or any other consideration could have different trust anchors built in, and the record of which batch the device is in would be recorded in the asset database. The manufacturer would then know which anchor to sign an artifact against.

Aside from the concern about long-term access to private keys, a major limiting factor for the shelf-life of many devices will be the age of the cryptographic algorithms included. A device produced in 2019 will have hardware and software capable of validating algorithms common in 2019, and will have no defense against attacks (both quantum and von-neuman brute force attacks) which have not yet been invented. This concern is orthogonal to the concern about access to private keys, but this concern likely dominates and limits the lifespan of a device in a warehouse. If any update to firmware to support new cryptographic mechanism were possible (while the device was in a warehouse), updates to trust anchors would also be done at the same time.

The set of standard operating procedures for maintaining high value private keys is well documented. For instance, the WebPKI provides a number of options for audits at [cabforumaudit], and the DNSSEC root operations are well documented at [dnssecroot].

It is not clear if Manufacturers will take this level of precaution, or how strong the economic incentives are to maintain an appropriate level of security.

This next section examines the risk due to a compromised manufacturer IDevID signing key. This is followed by examination of the risk due to a compromised MASA key. The third section sections below examines the situation where MASA web server itself is under attacker control, but that the MASA signing key itself is safe in a not-directly connected hardware module.

11.6.1. Compromise of Manufacturer IDevID signing keys

An attacker that has access to the key that the manufacturer uses to sign IDevID certificates can create counterfeit devices. Such devices can claim to be from a particular manufacturer, but be entirely different devices: Trojan horses in effect.

As the attacker controls the MASA URL in the certificate, the registrar can be convinced to talk to the attackers' MASA. The Registrar does not need to be in any kind of promiscuous mode to be vulnerable.

In addition to creating fake devices, the attacker may also be able to issue revocations for existing certificates if the IDevID certificate process relies upon CRL lists that are distributed.

There does not otherwise seem to be any risk from this compromise to devices which are already deployed, or which are sitting locally in boxes waiting for deployment (local spares). The issue is that operators will be unable to trust devices which have been in an uncontrolled warehouse as they do not know if those are real devices.

11.6.2. Compromise of MASA signing keys

There are two periods of time in which to consider: when the MASA key has fallen into the hands of an attacker, and after the MASA recognizes that the key has been compromised.

11.6.2.1. Attacker opportunties with compromised MASA key

An attacker that has access to the MASA signing key could create vouchers. These vouchers could be for existing deployed devices, or for devices which are still in a warehouse. In order to exploit these vouchers two things need to occur: the device has to go through a factory default boot cycle, and the registrar has to be convinced to contact the attacker's MASA.

If the attacker controls a Registrar which is visible to the device, then there is no difficulty in delivery of the false voucher. A possible practical example of an attack like this would be in a data center, at an ISP peering point (whether a public IX, or a private peering point). In such a situation, there are already cables attached to the equipment that lead to other devices (the peers at the IX), and through those links, the false voucher could be delivered. The difficult part would be get the device put through a factory reset. This might be accomplished through social engineering of data center staff. Most locked cages have ventilation holes, and possibly a long "paperclip" could reach through to depress a factory reset button. Once such a piece of ISP equipment has been compromised, it could be used to compromise equipment that was connected to (through long haul links even), assuming that those pieces of equipment could also be forced through a factory reset.

The above scenario seems rather unlikely as it requires some element of physical access; but were there a remote exploit that did not cause a direct breach, but rather a fault that resulted in a factory reset, this could provide a reasonable path.

The above deals with ANI uses of BRSKI. For cases where 802.11 or 802.15.4 is involved, the need to connect directly to the device is eliminated, but the need to do a factory reset is not. Physical possession of the device is not required as above, provided that there is some way to force a factory reset. With some consumers devices with low overall implementation quality, the end users might be familiar with needing to reset the device regularly.

The authors are unable to come up with an attack scenario where a compromised voucher signature enables an attacker to introduce a compromised pledge into an existing operator's network. This is the case because the operator controls the communication between Registrar and MASA, and there is no opportunity to introduce the fake voucher through that conduit.

11.6.2.2. Risks after key compromise is known

Once the operator of the MASA realizes that the voucher signing key has been compromised it has to do a few things.

First, it MUST issue a firmware update to all devices that had that key as a trust anchor, such that they will no longer trust vouchers from that key. This will affect devices in the field which are operating, but those devices, being in operation, are not performing onboarding operations, so this is not a critical patch.

Devices in boxes (in warehouses) are vulnerable, and remain vulnerable until patched. An operator would be prudent to unbox the devices, onboard them in a safe environment, and then perform firmware updates. This does not have to be done by the end-operator; it could be done by a distributor that stores the spares. A recommended practice for high value devices (which typically have a <4hr service window) may be to validate the device operation on a regular basis anyway.

If the onboarding process includes attestations about firmware versions, then through that process the operator would be advised to upgrade the firmware before going into production. Unfortunately, this does not help against situations where the attacker operates their own Registrar (as listed above).

[RFC8366] section 6.1 explains the need for short-lived vouchers. The nonce guarantees freshness, and the short-lived nature of the voucher means that the window to deliver a fake voucher is very short. A nonceless, long-lived voucher would be the only option for the attacker, and devices in the warehouse would be vulnerable to such a thing.

A key operational recommendation is for manufacturers to sign nonceless, long-lived vouchers with a different key that they sign short-lived vouchers. That key needs significantly better protection. If both keys come from a common trust-anchor (the manufacturer's CA), then a compromise of the manufacturer's CA would compromise both keys. Such a compromise of the manufacturer's CA likely compromises all keys outlined in this section.

11.6.3. Compromise of MASA web service

An attacker that takes over the MASA web service has a number of attacks. The most obvious one is simply to take the database listing customers and devices and to sell this data to other attackers who will now know where to find potentially vulnerable devices.

The second most obvious thing that the attacker can do is to kill the service, or make it operate unreliably, making customers frustrated. This could have a serious affect on ability to deploy new services by customers, and would be a significant issue during disaster recovery.

While the compromise of the MASA web service may lead to the compromise of the MASA voucher signing key, if the signing occurs offboard (such as in a hardware signing module, HSM), then the key may well be safe, but control over it resides with the attacker.

Such an attacker can issue vouchers for any device presently in service. Said device still needs to be convinced to do through a factory reset process before an attack.

If the attacker has access to a key that is trusted for long-lived nonceless vouchers, then they could issue vouchers for devices which are not yet in service. This attack may be very hard to verify and as it would involve doing firmware updates on every device in warehouses (a potentially ruinously expensive process), a manufacturer might be reluctant to admit this possibility.

11.7. YANG Module Security Considerations

As described in the Security Considerations section of [RFC8366] (section 7.4), the YANG module specified in this document defines the schema for data that is subsequently encapsulated by a CMS signed-data content type, as described in Section 5 of [RFC5652]. As such, all of the YANG modeled data is protected from modification.

The use of YANG to define data structures, via the 'yang-data' statement, is relatively new and distinct from the traditional use of YANG to define an API accessed by network management protocols such as NETCONF [RFC6241] and RESTCONF [RFC8040]. For this reason, these guidelines do not follow template described by Section 3.7 of [RFC8407].

12. Acknowledgements

We would like to thank the various reviewers for their input, in particular William Atwood, Brian Carpenter, Fuyu Eleven, Eliot Lear, Sergey Kasatkin, Anoop Kumar, Tom Petch, Markus Stenberg, Peter van der Stok, and Thomas Werner

Significant reviews were done by Jari Arko, Christian Huitema and Russ Housley.

Henk Birkholz contributed the CDDL for the audit log response.

This document started it's life as a two-page idea from Steinthor Bjarnason.

In addition, significant review comments were received by many IESG members, including Adam Roach, Alexey Melnikov, Alissa Cooper, Benjamin Kaduk, Eric Vyncke, Roman Danyliw, and Magnus Westerlund.

13. References

13.1. Normative References

[I-D.ietf-anima-autonomic-control-plane]
Eckert, T., Behringer, M., and S. Bjarnason, "An Autonomic Control Plane (ACP)", Work in Progress, Internet-Draft, draft-ietf-anima-autonomic-control-plane-22, 3 February 2020, <http://www.ietf.org/internet-drafts/draft-ietf-anima-autonomic-control-plane-22.txt>.

[I-D.ietf-anima-grasp] Bormann, C., Carpenter, B., and B. Liu, "A Generic Autonomic Signaling Protocol (GRASP)", Work in Progress, Internet-Draft, draft-ietf-anima-grasp-15, 13 July 2017, <http://www.ietf.org/internet-drafts/draft-ietf-anima-grasp-15.txt>.

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

[ITU.X690.1994] International Telecommunications Union, "Information Technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ITU-T Recommendation X.690, 1994.

[REST] Fielding, R.F., "Architectural Styles and the Design of Network-based Software Architectures", 2000, <http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>.

[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>.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <https://www.rfc-editor.org/info/rfc3339>.

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

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, Ed., "Extensible Authentication Protocol (EAP)", RFC 3748, DOI 10.17487/RFC3748, June 2004, <https://www.rfc-editor.org/info/rfc3748>.

[RFC3927] Cheshire, S., Aboba, B., and E. Guttman, "Dynamic Configuration of IPv4 Link-Local Addresses", RFC 3927, DOI 10.17487/RFC3927, May 2005, <https://www.rfc-editor.org/info/rfc3927>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <https://www.rfc-editor.org/info/rfc4086>.

[RFC4519] Sciberras, A., Ed., "Lightweight Directory Access Protocol (LDAP): Schema for User Applications", RFC 4519, DOI 10.17487/RFC4519, June 2006, <https://www.rfc-editor.org/info/rfc4519>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/info/rfc4648>.

[RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <https://www.rfc-editor.org/info/rfc4862>.

[RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in

IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007, <https://www.rfc-editor.org/info/rfc4941>.

[RFC5272] Schaad, J. and M. Myers, "Certificate Management over CMS (CMC)", RFC 5272, DOI 10.17487/RFC5272, June 2008, <https://www.rfc-editor.org/info/rfc5272>.

[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>.

[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>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <https://www.rfc-editor.org/info/rfc6762>.

[RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013, <https://www.rfc-editor.org/info/rfc6763>.

[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>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and

Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <https://www.rfc-editor.org/info/rfc7230>.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <https://www.rfc-editor.org/info/rfc7231>.

[RFC7469] Evans, C., Palmer, C., and R. Sleevi, "Public Key Pinning Extension for HTTP", RFC 7469, DOI 10.17487/RFC7469, April 2015, <https://www.rfc-editor.org/info/rfc7469>.

[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>.

[RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG", RFC 7951, DOI 10.17487/RFC7951, August 2016, <https://www.rfc-editor.org/info/rfc7951>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[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>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/info/rfc8259>.

[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>.

[RFC8368] Eckert, T., Ed. and M. Behringer, "Using an Autonomic Control Plane for Stable Connectivity of Network Operations, Administration, and Maintenance (OAM)", RFC 8368, DOI 10.17487/RFC8368, May 2018, <https://www.rfc-editor.org/info/rfc8368>.

[RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC

8407, DOI 10.17487/RFC8407, October 2018, <https://www.rfc-editor.org/info/rfc8407>.

[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>.

13.2. Informative References

[brewski] "Urban Dictionary: Brewski", October 2019, <https://www.urbandictionary.com/define.php?term=brewski>.

[cabforumaudit] "Information for Auditors and Assessors", August 2019, <https://cabforum.org/information-for-auditors-and-assessors/>.

[Dingledine2004] Dingledine, R., Mathewson, N., and P. Syverson, "Tor: the second-generation onion router", 2004, <https://spec.torproject.org/tor-spec>.

[dnssecroot] "DNSSEC Practice Statement for the Root Zone ZSK Operator", December 2017, <https://www.iana.org/dnssec/dps/zsk-operator/dps-zsk-operator-v2.0.pdf>.

[docsisroot] "CableLabs Digital Certificate Issuance Service", February 2018, <https://www.cablelabs.com/resources/digital-certificate-issuance-service/>.

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

[I-D.ietf-anima-constrained-voucher]
Richardson, M., Stok, P., and P. Kampanakis, "Constrained Voucher Artifacts for Bootstrapping Protocols", Work in Progress, Internet-Draft, draft-ietf-anima-constrained-voucher-07, 15 January 2020, <http://www.ietf.org/

internet-drafts/draft-ietf-anima-constrained-voucher-07.txt>.

[I-D.ietf-anima-reference-model] Behringer, M., Carpenter, B., Eckert, T., Ciavaglia, L., and J. Nobre, "A Reference Model for Autonomic Networking", Work in Progress, Internet-Draft, draft-ietf-anima-reference-model-10, 22 November 2018, <http://www.ietf.org/internet-drafts/draft-ietf-anima-reference-model-10.txt>.

[I-D.ietf-netconf-keystore] Watsen, K., "A YANG Data Model for a Keystore", Work in Progress, Internet-Draft, draft-ietf-netconf-keystore-15, 20 November 2019, <http://www.ietf.org/internet-drafts/draft-ietf-netconf-keystore-15.txt>.

[I-D.richardson-anima-state-for-joinrouter] Richardson, M., "Considerations for stateful vs stateless join router in ANIMA bootstrap", Work in Progress, Internet-Draft, draft-richardson-anima-state-for-joinrouter-02, 25 January 2018, <http://www.ietf.org/internet-drafts/draft-richardson-anima-state-for-joinrouter-02.txt>.

[imprinting] "Wikipedia article: Imprinting", July 2015, <https://en.wikipedia.org/wiki/Imprinting_(psychology)>.

[IoTstrangeThings] "IoT of toys stranger than fiction: Cybersecurity and data privacy update (accessed 2018-12-02)", March 2017, <https://www.welivesecurity.com/2017/03/03/internet-of-things-security-privacy-iot-update/>.

[livingwithIoT] "What is it actually like to live in a house filled with IoT devices? (accessed 2018-12-02)", February 2018, <https://www.siliconrepublic.com/machines/iot-smart-devices-reality>.

[minerva] Richardsdon, M., "Minerva reference implementation for BRSKI", 2020, <https://minerva.sandelman.ca/>.

[minervagithub] Richardsdon, M., "GITHUB hosting of Minerva reference code", 2020, <https://github.com/ANIMAgus-minerva>.

[openssl] "OpenSSL X509 utility", September 2019, <https://www.openssl.org/docs/man1.1.1/man1/openssl-x509.html/>.

[RFC2663] Srisuresh, P. and M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations", RFC

2663, DOI 10.17487/RFC2663, August 1999, <https://www.rfc-editor.org/info/rfc2663>.

[RFC5209] Sangster, P., Khosravi, H., Mani, M., Narayan, K., and J. Tardo, "Network Endpoint Assessment (NEA): Overview and Requirements", RFC 5209, DOI 10.17487/RFC5209, June 2008, <https://www.rfc-editor.org/info/rfc5209>.

[RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)", RFC 5785, DOI 10.17487/RFC5785, April 2010, <https://www.rfc-editor.org/info/rfc5785>.

[RFC6960] Santesson, S., Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 6960, DOI 10.17487/RFC6960, June 2013, <https://www.rfc-editor.org/info/rfc6960>.

[RFC6961] Pettersen, Y., "The Transport Layer Security (TLS) Multiple Certificate Status Request Extension", RFC 6961, DOI 10.17487/RFC6961, June 2013, <https://www.rfc-editor.org/info/rfc6961>.

[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>.

[RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, DOI 10.17487/RFC7258, May 2014, <https://www.rfc-editor.org/info/rfc7258>.

[RFC7435] Dukhovni, V., "Opportunistic Security: Some Protection Most of the Time", RFC 7435, DOI 10.17487/RFC7435, December 2014, <https://www.rfc-editor.org/info/rfc7435>.

[RFC7575] Behringer, M., Pritikin, M., Bjarnason, S., Clemm, A., Carpenter, B., Jiang, S., and L. Ciavaglia, "Autonomic Networking: Definitions and Design Goals", RFC 7575, DOI 10.17487/RFC7575, June 2015, <https://www.rfc-editor.org/info/rfc7575>.

[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>.

[slowloris] "Slowloris (computer security)", February 2019, <https://en.wikipedia.org/wiki/Slowloris_(computer_security)/>.

[softwareescrow]
"Wikipedia article: Software Escrow", October 2019, <https://en.wikipedia.org/wiki/Source_code_escrow>.

[Stajano99theresurrecting] Stajano, F. and R. Anderson, "The resurrecting duckling: security issues for ad-hoc wireless networks", 1999, <https://www.cl.cam.ac.uk/~fms27/papers/1999-StajanoAnd-duckling.pdf>.

[TR069] "TR-69: CPE WAN Management Protocol", February 2018, <https://www.broadband-forum.org/standards-and-software/technical-specifications/tr-069-files-tools>.

[W3C.WD-capability-urls-20140218]
Tennison, J., "Good Practices for Capability URLs", World Wide Web Consortium WD WD-capability-urls-20140218, 18 February 2014, <http://www.w3.org/TR/2014/WD-capability-urls-20140218>.

Appendix A. IPv4 and non-ANI operations

The specification of BRSKI in Section 4 intentionally only covers the mechanisms for an IPv6 pledge using Link-Local addresses. This section describes non-normative extensions that can be used in other environments.

A.1. IPv4 Link Local addresses

Instead of an IPv6 link-local address, an IPv4 address may be generated using [RFC3927] Dynamic Configuration of IPv4 Link-Local Addresses.

In the case that an IPv4 Link-Local address is formed, then the bootstrap process would continue as in the IPv6 case by looking for a (circuit) proxy.

A.2. Use of DHCPv4

The Pledge MAY obtain an IP address via DHCP [RFC2131]. The DHCP provided parameters for the Domain Name System can be used to perform DNS operations if all local discovery attempts fail.

Appendix B. mDNS / DNSSD proxy discovery options

Pledge discovery of the proxy (Section 4.1) MAY be performed with DNS-based Service Discovery [RFC6763] over Multicast DNS [RFC6762] to discover the proxy at "_brski-proxy._tcp.local.".

Proxy discovery of the registrar (Section 4.3) MAY be performed with DNS-based Service Discovery over Multicast DNS to discover registrars by searching for the service "_brski-registrar._tcp.local.".

To prevent unaccceptable levels of network traffic, when using mDNS, the congestion avoidance mechanisms specified in [RFC6762] section 7 MUST be followed. The pledge SHOULD listen for an unsolicited broadcast response as described in [RFC6762]. This allows devices to avoid announcing their presence via mDNS broadcasts and instead silently join a network by watching for periodic unsolicited broadcast responses.

Discovery of registrar MAY also be performed with DNS-based service discovery by searching for the service "_brski-registrar._tcp.<domain>". In this case the domain "example.com" is discovered as described in [RFC6763] section 11 (Appendix A.2 suggests the use of DHCP parameters).

If no local proxy or registrar service is located using the GRASP mechanisms or the above mentioned DNS-based Service Discovery methods, the pledge MAY contact a well known manufacturer provided bootstrapping server by performing a DNS lookup using a well known URI such as "brski-registrar.manufacturer.example.com". The details of the URI are manufacturer specific. Manufacturers that leverage this method on the pledge are responsible for providing the registrar service. Also see Section 2.7.

The current DNS services returned during each query are maintained until bootstrapping is completed. If bootstrapping fails and the pledge returns to the Discovery state, it picks up where it left off and continues attempting bootstrapping. For example, if the first Multicast DNS _bootstrapks._tcp.local response doesn't work then the second and third responses are tried. If these fail the pledge moves on to normal DNS-based Service Discovery.

Appendix C. Example Vouchers

Three entities are involved in a voucher: the MASA issues (signs) it, the registrar's public key is mentioned in the voucher, and the pledge validates it. In order to provide reproduceable examples the public and private keys for an example MASA and registrar are first listed.

The keys come from an open source reference implementation of BRSKI, called "Minerva" [minerva]. It is available on github [minervagithub]. The keys presented here are used in the unit and integration tests. The MASA code is called "highway", the Registrar code is called "fountain", and the example client is called "reach".

The public key components of each are presented as both base64 certificates, as well as being decoded by openssl's x509 utility so that the extensions can be seen. This was version 1.1.1c of the [openssl] library and utility.

C.1. Keys involved

The Manufacturer has a Certificate Authority that signs the pledge's IDevID. In addition the Manufacturer's signing authority (the MASA) signs the vouchers, and that certificate must distributed to the devices at manufacturing time so that vouchers can be validated.

C.1.1. Manufacturer Certificate Authority for IDevID signatures

This private key is Certificate Authority that signs IDevID certificates:

```
<CODE BEGINS> file "vendor.key"

-----BEGIN EC PRIVATE KEY-----
MIGkAgEBBDCAYkoLW1IEA5SKKhMMdkTK7sJxk5ybKqYq9Yr5aR7tNwqXyLGS7z8G
8S4w/UJ58BqgBwYFK4EEACKhZANiAAQu5/yktJbFLjMC87h7b+yTreFuF8GwewKH
L4mS0r0dVAQubqDUQcTrjvpXrXCpTojiLCzgp8fzkcUDkZ9LD/M90LDipiLNIOkP
juF8QkoAbT8pMrY83MS8y76wZ7AalNQ=
-----END EC PRIVATE KEY-----

<CODE ENDS>
```

This public key validates IDevID certificates:

file: examples/vendor.key

```
<CODE BEGINS> file "vendor.cert"

Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 519772114 (0x1efb17d2)
        Signature Algorithm: ecdsa-with-SHA256
        Issuer: C = Canada, ST = Ontario, OU = Sandelman, CN = highway-test.example.com CA
        Validity
            Not Before: Feb 12 22:22:21 2019 GMT
            Not After : Feb 11 22:22:21 2021 GMT
        Subject: C = Canada, ST = Ontario, OU = Sandelman, CN = highway-test.example.com CA
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (384 bit)
                pub:
                    04:2e:e7:fc:a4:b4:96:c5:2e:33:02:f3:b8:7b:6f:
                    ec:93:ad:e1:6e:17:c1:b0:7b:02:87:2f:89:92:d2:
                    bd:1d:54:04:2e:6e:a0:d4:41:c4:eb:8e:fa:57:ad:
                    70:a9:4e:88:e2:2c:2c:e0:a7:c7:f3:91:c5:03:91:
                    9f:4b:0f:f3:3d:d0:b0:e2:a6:22:cd:20:e9:0f:8e:
                    e1:7c:42:4a:00:6d:3f:29:32:b6:3c:dc:c4:bc:cb:
                    be:b0:67:b0:1a:94:d4
                ASN1 OID: secp384r1
                NIST CURVE: P-384
        X509v3 extensions:
            X509v3 Basic Constraints: critical
                CA:TRUE
            X509v3 Key Usage: critical
                Certificate Sign, CRL Sign
            X509v3 Subject Key Identifier:
                5E:0C:A9:52:5A:8C:DF:A9:0F:03:14:E9:96:F1:80:76:8C:53:8A:08
            X509v3 Authority Key Identifier:
                keyid:5E:0C:A9:52:5A:8C:DF:A9:0F:03:14:E9:96:F1:80:76:8C:53:8A:08

    Signature Algorithm: ecdsa-with-SHA256
         30:65:02:30:5f:21:fd:c6:ab:d6:94:a6:cd:ca:37:2c:81:33:
         87:fe:7b:e1:b5:1a:e8:6c:05:43:a6:8b:4e:22:b5:55:e9:48:
         0c:b5:97:f3:c9:1a:65:d9:97:4b:f0:21:86:0d:cb:26:02:31:
         00:e3:2d:0d:08:49:4d:a3:f5:dc:57:1f:a7:13:26:a4:e0:d6:
         3a:c2:d5:4a:50:83:62:26:2e:79:2b:d0:a5:ee:66:d5:bf:16:
         9a:33:75:b4:d1:8d:ba:d3:50:77:6b:92:df
-----BEGIN CERTIFICATE-----
MIICTDCCAdKgAwIBAgIEHvsX0jAKBggqhkjOPQQDAjBdMQ8wDQYDVQQGEwZDYW5h
ZGExEDAOBgNVBAgMB09udGFyaW8xEjAQBgNVBAsMCVNhbmRlbG1hbjEkMCIGA1UE
AwwbaGlnaHdheS10ZXN0LmV4YW1wbGUuY29tIENBMB4XDTE5MDIxMjIyMjIyMVoX
DTIxMDIxMTIyMjIyMVowXTEPMA0GA1UEBhMGQ2FuYWRhMRAwDgYDVQQIDAdPbnRh
cmlvMRIwEAYDVQQLDAlTYW5kZWxtYW4xJDAiBgNVBAMMG2hpZ2h3YXktdGVzdC5l
eGFtcGxlLmNvbSBDQTB2MBAGByqGSM49AgEGBSuBBAAiA2IABC7n/KS0lsUuMwLz
```

```
uHtv7JOt4W4XwbB7AocviZLSvR1UBC5uoNRBxOuO+letcKlOiOIsLOCnx/ORxQOR
n0sP8z3QsOKmIs0g6Q+O4XxCSgBtPykytjzcxLzLvrBnsBqU1KNjMGEwDwYDVR0T
AQH/BAUwAwEB/zAOBgNVHQ8BAf8EBAMCAQYwHQYDVR0OBBYEFF4MqVJajN+pDwMU
6ZbxgHaMU4oIMB8GA1UdIwQYMBaAFF4MqVJajN+pDwMU6ZbxgHaMU4oIMAoGCCqG
SM49BAMCA2gAMGUCMF8h/car1pSmzco3LIEzh/574bUa6GwFQ6aLTiK1VelIDLWX
88kaZdmXS/Ahhg3LJgIxAOMtDQhJTaP13FcfpxMmpODWOsLVSlCDYiYueSvQpe5m
1b8WmjN1tNGNutNQd2uS3w==
-----END CERTIFICATE-----

<CODE ENDS>
```

C.1.2. MASA key pair for voucher signatures

This private key signs vouchers:

```
<CODE BEGINS> file "masa.key"

-----BEGIN EC PRIVATE KEY-----
MHcCAQEEIFhdd0eDdzip67kXx72K+KHGJQYJHNy8pkiLJ6CcvxMGoAoGCCqGSM49
AwEHoUQDQgAEqgQVo0S54kT4yfkbBxumdHOcHrpsqbOpMKmiMln3oB1HAW25MJV+
gqi4tMFfSJ0iEwt8kszfWXK4rLgJS2mnpQ==
-----END EC PRIVATE KEY-----

<CODE ENDS>
```

This public key validates vouchers, and it has been signed by the CA above:

file: examples/masa.key

```
<CODE BEGINS> file "masa.cert"

Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 463036244 (0x1b995f54)
        Signature Algorithm: ecdsa-with-SHA256
        Issuer: C = Canada, ST = Ontario, OU = Sandelman, CN = highway-test.example.com CA
        Validity
            Not Before: Feb 12 22:22:41 2019 GMT
            Not After : Feb 11 22:22:41 2021 GMT
        Subject: C = Canada, ST = Ontario, OU = Sandelman, CN = highway-test.example.com MASA
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    04:aa:04:15:a3:44:b9:e2:44:f8:c9:f9:1b:07:1b:
                    a6:74:73:9c:1e:ba:6c:a9:b3:a9:30:a9:a2:32:59:
                    f7:a0:1d:47:01:6d:b9:30:95:7e:82:a8:b8:b4:c1:
                    5f:48:9d:22:13:0b:7c:92:cc:df:59:72:b8:ac:b8:
                    09:4b:69:a7:a5
                ASN1 OID: prime256v1
                NIST CURVE: P-256
        X509v3 extensions:
            X509v3 Basic Constraints: critical
                CA:FALSE
    Signature Algorithm: ecdsa-with-SHA256
         30:66:02:31:00:bd:55:e5:9b:0e:fb:fc:5e:95:29:e3:81:b3:
         15:35:aa:93:18:a2:04:be:44:72:b2:51:7d:4d:6d:eb:d1:d5:
         c1:10:3a:b2:39:7b:57:3f:c5:cc:b0:a3:0e:e7:99:46:ba:02:
         31:00:f6:7f:44:7d:b7:14:fa:d1:67:6a:d4:11:c3:4b:ae:e6:
         fb:9a:98:56:fa:85:21:2e:5c:48:4c:f0:3f:f2:9b:3f:ae:88:
         20:a7:ae:f9:72:ff:5b:f9:78:68:cf:0f:48:c9
-----BEGIN CERTIFICATE-----
MIIB3zCCAWSgAwIBAgIEG5lfVDAKBggqhkjOPQQDAjBdMQ8wDQYDVQQGEwZDYW5h
ZGExEDAOBgNVBAgMB09udGFyaW8xEjAQBgNVBAsMCVNhbmRlbG1hbjEkMCIGA1UE
AwwbaGlnaHdheS10ZXN0LmV4YW1wbGUuY29tIENBMB4XDTE5MDIxMjIyMjI0MVoX
DTIxMDIxMTIyMjI0MVowXzEPMA0GA1UEBhMGQ2FuYWRhMRAwDgYDVQQIDAdPbnRh
cmlvMRIwEAYDVQQLDAlTYW5kZWxtYW4xJjAkBgNVBAMMHWhpZ2h3YXktdGVzdC5l
eGFtcGxlLmNvbSBNQVNBMFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEqgQVo0S5
4kT4yfkbBxumdHOcHrpsqbOpMKmiMln3oB1HAW25MJV+gqi4tMFfSJ0iEwt8kszf
WXK4rLgJS2mnpaMQMA4wDAYDVR0TAQH/BAIwADAKBggqhkjOPQQDAgNpADBmAjEA
vVXlmw77/F6VKeOBsxU1qpMYogS+RHKyUX1NbevR1cEQOrI5e1c/xcywow7nmUa6
AjEA9n9EfbcU+tFnatQRw0uu5vuamFb6hSEuXEhM8D/ymz+uiCCnrvly/1v5eGjP
D0jJ
-----END CERTIFICATE-----

<CODE ENDS>
```

C.1.3. Registrar Certificate Authority

This Certificate Authority enrolls the pledge once it is authorized, and it also signs the Registrar's certificate.

```
<CODE BEGINS> file "ownerca_secp384r1.key"

-----BEGIN EC PRIVATE KEY-----
MIGkAgEBBDCHnLI0MSOLf8XndiZqoZdqblcPR5YSoPGhPOuFxWy1gFi9HbWv8b/R
EGdRgGEVSjKgBwYFK4EEACKhZANiAAQbf1m6F8MavGaNjGzgw/oxcQ9l9iKRvbdW
gAfb37h6pUVNeYpGlxlZljGxj2l9Mr48yD5bY7VG9qjVb5v5wPPTuRQ/ckdRpHbd
0vC/9cqPMAF/+MJf0/UgA0SLi/IHbLQ=
-----END EC PRIVATE KEY-----

<CODE ENDS>
```

The public key is indicated in a pledge voucher-request to show proximity.

file: examples/ownerca_secp384r1.key

```
<CODE BEGINS> file "ownerca_secp384r1.cert"

Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 694879833 (0x296b0659)
        Signature Algorithm: ecdsa-with-SHA256
        Issuer: DC = ca, DC = sandelman, CN = fountain-test.example.com Unstrung Fountain Roo
        Validity
            Not Before: Feb 25 21:31:45 2020 GMT
            Not After : Feb 24 21:31:45 2022 GMT
        Subject: DC = ca, DC = sandelman, CN = fountain-test.example.com Unstrung Fountain Ro
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (384 bit)
                pub:
                    04:1b:7f:59:ba:17:c3:1a:bc:66:8d:8c:6c:e0:c3:
                    fa:31:71:0f:65:f6:22:91:bd:b7:56:80:07:db:df:
                    b8:7a:a5:45:4d:79:8a:46:97:19:59:96:31:b1:8f:
                    69:7d:32:be:3c:c8:3e:5b:63:b5:46:f6:a8:d5:6f:
                    9b:f9:c0:f3:d3:b9:14:3f:72:47:51:a4:76:dd:d2:
                    f0:bf:f5:ca:8f:30:01:7f:f8:c2:5f:d3:f5:20:03:
                    44:8b:8b:f2:07:6c:b4
                ASN1 OID: secp384r1
                NIST CURVE: P-384
        X509v3 extensions:
            X509v3 Basic Constraints: critical
                CA:TRUE
            X509v3 Key Usage: critical
                Certificate Sign, CRL Sign
            X509v3 Subject Key Identifier:
                B9:A5:F6:CB:11:E1:07:A4:49:2C:A7:08:C6:7C:10:BC:87:B3:74:26
            X509v3 Authority Key Identifier:
                keyid:B9:A5:F6:CB:11:E1:07:A4:49:2C:A7:08:C6:7C:10:BC:87:B3:74:26

    Signature Algorithm: ecdsa-with-SHA256
         30:64:02:30:20:83:06:ce:8d:98:a4:54:7a:66:4c:4a:3a:70:
         c2:52:36:5a:52:8d:59:7d:20:9b:2a:69:14:58:87:38:d8:55:
         79:dd:fd:29:38:95:1e:91:93:76:b4:f5:66:29:44:b4:02:30:
         6f:38:f9:af:12:ed:30:d5:85:29:7c:b1:16:58:bd:67:91:43:
         c4:0d:30:f9:d8:1c:ac:2f:06:dd:bc:d5:06:42:2c:84:a2:04:
         ea:02:a4:5f:17:51:26:fb:d9:2f:d2:5c
```

```
-----BEGIN CERTIFICATE-----
MIICazCCAfKgAwIBAgIEKWsGWTAKBggqhkjOPQQDAjBtMRIwEAYKCZImiZPyLGQB
GRYCY2ExGTAXBgoJkiaJk/IsZAEZFglzYW5kZWxtYW4xPDA6BgNVBAMMM2ZvdW50
YWluLXRlc3QuZXhhbXBsZS5jb20gVW5zdHJ1bmcgRm91bnRhaW4gUm9vdCBDQTAe
Fw0yMDAyMjUyMTMxNDVaFw0yMjAyMjQyMTMxNDVaMG0xEjAQBgoJkiaJk/IsZAEZ
FgJjYTEZMBcGCgmSJomT8ixkARkWCXNhbmRlbG1hbjE8MDoGA1UEAwwzZm91bnRh
aW4tdGVzdC5leGFtcGxlLmNvbSBVbnN0cnVuZyBGb3VudGFpbiBSb290IENBMHYw
```

```
EAYHKoZIzj0CAQYFK4EEACIDYgAEG39ZuhfDGrxmjYxs4MP6MXEPZfYikb23VoAH
29+4eqVFTXmKRpcZWZYxsY9pfTK+PMg+W2O1Rvao1W+b+cDz07kUP3JHUaR23dLw
v/XKjzABf/jCX9P1IANEi4vyB2y0o2MwYTAPBgNVHRMBAf8EBTADAQH/MA4GA1Ud
DwEB/wQEAwIBBjAdBgNVHQ4EFgQUuaX2yxHhB6RJLKcIxnwQvIezdCYwHwYDVR0j
BBgwFoAUuaX2yxHhB6RJLKcIxnwQvIezdCYwCgYIKoZIzj0EAwIDZwAwZAIwIIMG
zo2YpFR6ZkxKOnDCUjZaUo1ZfSCbKmkUWIc42FV53f0pOJUekZN2tPVmKUS0AjBv
OPmvEu0w1YUpfLEWWL1nkUPEDTD52BysLwbdvNUGQiyEogTqAqRfF1Em+9kv0lw=
-----END CERTIFICATE-----

<CODE ENDS>
```

C.1.4. Registrar key pair

The Registrar is the representative of the domain owner. This key signs registrar voucher-requests, and terminates the TLS connection from the pledge.

```
<CODE BEGINS> file "jrc_prime256v1.key"

-----BEGIN EC PRIVATE KEY-----
MHcCAQEEIFZodk+PC5Mu24+ra0sbOjKzan+dW5rvDAR7YuJUOC1YoAoGCCqGSM49
AwEHoUQDQgAElmVQcjS6n+Xd5l/28IFv6UiegQwSBztGj5dkK2MAjQIPV8l8lH+E
jLIOYdbJiI0VtEIf1/Jqt+TOBfinTNOLOg==
-----END EC PRIVATE KEY-----

<CODE ENDS>
```

The public key is indicated in a pledge voucher-request to show proximity.

```
<CODE BEGINS> file "jrc_prime256v1.cert"

Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 1066965842 (0x3f989b52)
        Signature Algorithm: ecdsa-with-SHA256
        Issuer: DC = ca, DC = sandelman, CN = fountain-test.example.com Unstrung Fountain Roo
        Validity
            Not Before: Feb 25 21:31:54 2020 GMT
            Not After : Feb 24 21:31:54 2022 GMT
        Subject: DC = ca, DC = sandelman, CN = fountain-test.example.com
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    04:96:65:50:72:34:ba:9f:e5:dd:e6:5f:f6:f0:81:
                    6f:e9:48:9e:81:0c:12:07:3b:46:8f:97:64:2b:63:
                    00:8d:02:0f:57:c9:7c:94:7f:84:8c:b2:0e:61:d6:
                    c9:88:8d:15:b4:42:1f:d7:f2:6a:b7:e4:ce:05:f8:
                    a7:4c:d3:8b:3a
                ASN1 OID: prime256v1
                NIST CURVE: P-256
        X509v3 extensions:
            X509v3 Extended Key Usage: critical
                CMC Registration Authority
            X509v3 Key Usage: critical
                Digital Signature
    Signature Algorithm: ecdsa-with-SHA256
         30:65:02:30:66:4f:60:4c:55:48:1e:96:07:f8:dd:1f:b9:c8:
         12:8d:45:36:87:9b:23:c0:bc:bb:f1:cb:3d:26:15:56:6f:5f:
         1f:bf:d5:1c:0e:6a:09:af:1b:76:97:99:19:23:fd:7e:02:31:
         00:bc:ac:c3:41:b0:ba:0d:af:52:f9:9c:6e:7a:7f:00:1d:23:
         c8:62:01:61:bc:4b:c5:c0:47:99:35:0a:0c:77:61:44:01:4a:
         07:52:70:57:00:75:ff:be:07:0e:98:cb:e5
-----BEGIN CERTIFICATE-----
MIIB/DCCAYKgAwIBAgIEP5ibUjAKBggqhkjOPQQDAjBtMRIwEAYKCZImiZPyLGQB
GRYCY2ExGTAXBgoJkiaJk/IsZAEZFglzYW5kZWxtYW4xPDA6BgNVBAMMM2ZvdW50
YWluLXRlc3QuZXhhbXBsZS5jb20gVW5zdHJ1bmcgRm91bnRhaW4gUm9vdCBDQTAe
Fw0yMDAyMjUyMTMxNTRaFw0yMjAyMjQyMTMxNTRaMFMxEjAQBgoJkiaJk/IsZAEZ
FgJjYTEZMBcGCgmSJomT8ixkARkWCXNhbmRlbG1hbjEiMCAGA1UEAwwZZm91bnRh
aW4tdGVzdC5leGFtcGxlLmNvbTBZMBMGByqGSM49AgEGCCqGSM49AwEHA0IABJZl
UHI0up/l3eZf9vCBb+lInoEMEgc7Ro+XZCtjAI0CD1fJfJR/hIyyDmHWyYiNFbRC
H9fyarfkzgX4p0zTizqjKjAoMBYGA1UdJQEB/wQMMAoGCCsGAQUFBwMcMA4GA1Ud
DwEB/wQEAwIHgDAKBggqhkjOPQQDAgNoADBlAjBmT2BMVUgelgf43R+5yBKNRTaH
myPAvLvxyz0mFVZvXx+/1RwOagmvG3aXmRkj/X4CMQC8rMNBsLoNr1L5nG56fwAd
I8hiAWG8S8XAR5k1Cgx3YUQBSgdScFcAdf++Bw6Yy+U=
-----END CERTIFICATE-----
```

<CODE ENDS>

C.1.5. Pledge key pair

The pledge has an IDevID key pair built in at manufacturing time:

<CODE BEGINS> file "idevid_00-D0-E5-F2-00-02.key"

```
-----BEGIN EC PRIVATE KEY-----
MHcCAQEEIBHNh6r8QRevRuo+tEmBJeFjQKf6bpFA/9NGoltv+9sNoAoGCCqGSM49
AwEHoUQDQgAEA6N1Q4ezfMAKmoecrfb0OBMc1AyEH+BATkF58FsTSyBxs0SbSWLx
FjDOuwB9gLGn2TsTUJumJ6VPw5Z/TP4hJw==
-----END EC PRIVATE KEY-----
```

<CODE ENDS>

The public key is used by the registrar to find the MASA. There is a
second

```
<CODE BEGINS> file "idevid_00-D0-E5-F2-00-02.cert"

Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 226876461 (0xd85dc2d)
        Signature Algorithm: ecdsa-with-SHA256
        Issuer: C = Canada, ST = Ontario, OU = Sandelman, CN = highway-test.example.com CA
        Validity
            Not Before: Feb  3 06:47:20 2020 GMT
            Not After : Dec 31 00:00:00 2999 GMT
        Subject: serialNumber = 00-D0-E5-F2-00-02
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    04:03:a3:75:43:87:b3:7c:c0:0a:9a:87:9c:ad:f6:
                    f4:38:13:1c:d4:0c:84:1f:e0:40:4e:41:79:f0:5b:
                    13:4b:20:71:b3:44:9b:49:62:f1:16:30:ce:bb:00:
                    7d:80:b1:a7:d9:3b:13:50:9b:a6:27:a5:4f:c3:96:
                    7f:4c:fe:21:27
                ASN1 OID: prime256v1
                NIST CURVE: P-256
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                45:88:CC:96:96:00:64:37:B0:BA:23:65:64:64:54:08:06:6C:56:AD
            X509v3 Basic Constraints:
                CA:FALSE
            1.3.6.1.5.5.7.1.32:
                ..highway-test.example.com:9443
    Signature Algorithm: ecdsa-with-SHA256
         30:65:02:30:23:e1:a9:2e:ef:22:12:34:5a:a5:c2:15:d6:28:
         bd:ed:3d:96:d6:ce:04:95:ef:a7:c8:dc:18:a8:31:c7:b8:04:
         34:f2:b7:4d:79:8a:67:22:24:03:4f:c5:cd:d6:06:ba:02:31:
         00:b3:8d:5c:0a:d0:fe:04:83:90:d3:4f:6d:72:97:b3:3e:02:
         ea:f1:c8:5a:32:72:58:b7:45:02:50:78:bc:04:1d:23:5e:22:
         6f:c3:7f:8c:7c:d7:9b:70:20:91:b4:e1:7f
-----BEGIN CERTIFICATE-----
MIIB5jCCAWygAwIBAgIEDYXcLTAKBggqhkjOPQQDAjBdMQ8wDQYDVQQGEwZDYW5h
ZGExEDAOBgNVBAgMB09udGFyaW8xEjAQBgNVBAsMCVNhbmRlbG1hbjEkMCIGA1UE
AwwbaGlnaHdheS10ZXN0LmV4YW1wbGUuY29tIENBMCAXDTIwMDIwMzA2NDcyMFoY
DzI5OTkxMjMxMDAwMDAwWjAcMRowGAYDVQQFDBEwMC1EMC1FNS1GMi0wMC0wMjBZ
MBMGByqGSM49AgEGCCqGSM49AwEHA0IABAOjdUOHs3zACpqHnK329DgTHNQMhB/g
QE5BefBbE0sgcbNEm0li8RYwzrsAfYCxp9k7E1CbpielT8OWf0z+ISejWTBXMB0G
A1UdDgQWBBRFiMyWlgBkN7C6I2VkZFQIBmxWrTAJBgNVHRMEAjAAMCsGCCsGAQUF
BwEgBB8MHWhpZ2h3YXktdGVzdC5leGFtcGxlLmNvbTo5NDQzMAoGCCqGSM49BAMC
A2gAMGUCMCPhqS7vIhI0WqXCFdYove09ltbOBJXvp8jcGKgxx7gENPK3TXmKZyIk
A0/FzdYGugIxALONXArQ/gSDkNNPbXKXsz4C6vHIWjJyWLdFAlB4vAQdI14ib8N/
jHzXm3AgkbThfw==
```

```
-----END CERTIFICATE-----
```

<CODE ENDS>

C.2. Example process

The JSON examples below are wrapped at 60 columns. This results in strings that have newlines in them, which makes them invalid JSON as is. The strings would otherwise be too long, so they need to be unwrapped before processing.

C.2.1. Pledge to Registrar

As described in Section 5.2, the pledge will sign a pledge voucher-request containing the registrar's public key in the proximity-registrar-cert field. The base64 has been wrapped at 60 characters for presentation reasons.

<CODE BEGINS> file "vr_00-D0-E5-F2-00-02.b64"


```
MIIG3gYJKoZIhvcNAQcCoIIGzzCCBssCAQExDTALBglghkgBZQMEAgEwggOJBgkqhkiG9w0BBwGg
ggN6BIIDdnsiaWV0Zi12b3VjaGVyLXJlcXVlc3Q6dm91Y2hlciI6eyJhc3NlcnRpb24iOiJwcm94
aW1pdHkiLCJjcmVhdGVkLW9uIjoiMjAyMC0wMi0yNVQxNjozMzoxMS45ODQtMDU6MDAiLCJzZXJp
YWwtbnVtYmVyIjoiMDAtRDAtRTUtRjItMDAtMDIiLCJub25jZSI6InkyQmZOYUlTMEtKU3loS2Ft
VEdYYVEiLCJwcm94aW1pdHktcmVnaXN0cmFyLWNlcnQiOiJNSUlCL0RDQ0FZS2dBd0lCQWdJRVA1
aWJVakFLQmdncWhrak9QUVFEQWpCdE1SSXdFQVlLQ1pJbWlaUHlMR1FCR1JZQ1kyRXhHVEFYQmdv
SmtpYUprL0lzWkFFWkZnbHpZVzVrWld4dFlXNHhQREE2QmdOVkJBTU1NMlp2ZFc1MFlXbHVMWFJs
YzNRdVpYaGhiWEJzWlM1amIyMGdWVzV6ZEhKMWJtY2dSbTkxYm5SaGFXNGdVbTl2ZENCRFFUQWVG
dzB5TURBeU1qVXlNVE14TlRSYUZ3MHlNakF5TWpReU1UTXhOVFJhTUZNeEVqQVFCZ29Ka2lhSmsv
SXNaQUVaRmdKallURVpNQmNHQ2dtU0pvbVQ4aXhrQVJrV0NYTmhibVJsYkcxaGJqRWlNQ0FHQTFV
RUF3d1pabTkxYm5SaGFXNHRkR1Z6ZEM1bGVHRnRjR3hsTG1OdmJUQlpNQk1HQnlxR1NNNDlBZ0VH
Q0NxR1NNNDlBd0VIQTBJQUJKWmxVSEkwdXAvbDNlWmY5dkNCYitsSW5vRU1FZ2M3Um8rWFpDdGpB
STBDRDFmSmZKUi9oSXl5RG1IV3lZaU5GYlJDSDlmeWFyZmt6Z1g0cDB6VGl6cWpLakFvTUJZR0Ex
VWRKUUVCL3dRTU1Bb0dDQ3NHQVFVRkJ3TWNNQTRHQTFVZER3RUIvd1FFQXdJSGdEQUtCZ2dxaGtq
T1BRUURBZ05vQURCbEFqQm1UMkJNVlVnZWxnZjQzUis1eUJLTlJUYUhteVBBdkx2eHl6MG1GVlp2
WHgrLzFSd09hZ212RzNhWG1Sa2ovWDRDTVFDOHJNTkJzTG9OcjFMNW5HNTZmd0FkSThoaUFXRzhT
OFhBUjVrMUNneDNZVVFCU2dkU2NGY0FkZisrQnc2WXkrVT0ifX2gggHqMIIB5jCCAWygAwIBAgIE
DYXcLTAKBggqhkjOPQQDAjBdMQ8wDQYDVQQGEwZDYW5hZGExEDAOBgNVBAgMB09udGFyaW8xEjAQ
BgNVBAsMCVNhbmRlbG1hbjEkMCIGA1UEAwwbaGlnaHdheS10ZXN0LmV4YW1wbGUuY29tIENBMCAX
DTIwMDIwMzA2NDcyMFoYDzI5OTkxMjMxMDAwMDAwWjAcMRowGAYDVQQFDBEwMC1EMC1FNS1GMi0w
MC0wMjBZMBMGByqGSM49AgEGCCqGSM49AwEHA0IABAOjdUOHs3zACpqHnK329DgTHNQMhB/gQE5B
efBbE0sgcbNEm0li8RYwzrsAfYCxp9k7E1CbpielT8OWf0z+ISejWTBXMB0GA1UdDgQWBBRFiMyW
lgBkN7C6I2VkZFQIBmxWrTAJBgNVHRMEAjAAMCsGCCsGAQUFBwEgBB8MHWhpZ2h3YXktdGVzdC5l
eGFtcGxlLmNvbTo5NDQzMAoGCCqGSM49BAMCA2gAMGUCMCPhqS7vIhI0WqXCFdYove09ltbOBJXv
p8jcGKgxx7gENPK3TXmKZyIkA0/FzdYGugIxALONXArQ/gSDkNNPbXKXsz4C6vHIWjJyWLdFAlB4
vAQdI14ib8N/jHzXm3AgkbThfzGCATowggE2AgEBMGUwXTEPMA0GA1UEBhMGQ2FuYWRhMRAwDgYD
VQQIDAdPbnRhcmlvMRIwEAYDVQQLDAlTYW5kZWxtYW4xJDAiBgNVBAMMG2hpZ2h3YXktdGVzdC5l
eGFtcGxlLmNvbSBDQQIEDYXcLTALBglghkgBZQMEAgGgaTAYBgkqhkiG9w0BCQMxCwYJKoZIhvcN
AQcBMBwGCSqGSIb3DQEJBTEPFw0yMDAyMjUyMTMzMTFaMC8GCSqGSIb3DQEJBDEiBCB0yBo/clJ6
GlTKYRILxPrHxYFL9+4MAL0/DavOh/IkETAKBggqhkjOPQQDAgRGMEQCIFpErwe+ypjpXtYpnsIZ
FsfoLFIOfH1p2p+Cr3eo4F1tAiAhvsI/GRsBd2LP7ZA+W0b+sBXwc2heR19a+LV5hwLIOg==
```


<CODE ENDS>

The ASN1 decoding of the artifact:

file: examples/vr_00-D0-E5-F2-00-02.b64

```
    0:d=0  hl=4 l=1758 cons: SEQUENCE
    4:d=1  hl=2 l=   9 prim: OBJECT            :pkcs7-signedData
   15:d=1  hl=4 l=1743 cons: cont [ 0 ]
   19:d=2  hl=4 l=1739 cons: SEQUENCE
   23:d=3  hl=2 l=   1 prim: INTEGER           :01
   26:d=3  hl=2 l=  13 cons: SET
   28:d=4  hl=2 l=  11 cons: SEQUENCE
   30:d=5  hl=2 l=   9 prim: OBJECT            :sha256
   41:d=3  hl=4 l= 905 cons: SEQUENCE
   45:d=4  hl=2 l=   9 prim: OBJECT            :pkcs7-data
   56:d=4  hl=4 l= 890 cons: cont [ 0 ]
   60:d=5  hl=4 l= 886 prim: OCTET STRING      :{"ietf-voucher-request:v
  950:d=3  hl=4 l= 490 cons: cont [ 0 ]
  954:d=4  hl=4 l= 486 cons: SEQUENCE
  958:d=5  hl=4 l= 364 cons: SEQUENCE
  962:d=6  hl=2 l=   3 cons: cont [ 0 ]
  964:d=7  hl=2 l=   1 prim: INTEGER           :02
  967:d=6  hl=2 l=   4 prim: INTEGER           :0D85DC2D
  973:d=6  hl=2 l=  10 cons: SEQUENCE
  975:d=7  hl=2 l=   8 prim: OBJECT            :ecdsa-with-SHA256
  985:d=6  hl=2 l=  93 cons: SEQUENCE
  987:d=7  hl=2 l=  15 cons: SET
  989:d=8  hl=2 l=  13 cons: SEQUENCE
  991:d=9  hl=2 l=   3 prim: OBJECT            :countryName
  996:d=9  hl=2 l=   6 prim: PRINTABLESTRING   :Canada
 1004:d=7  hl=2 l=  16 cons: SET
 1006:d=8  hl=2 l=  14 cons: SEQUENCE
 1008:d=9  hl=2 l=   3 prim: OBJECT            :stateOrProvinceName
 1013:d=9  hl=2 l=   7 prim: UTF8STRING        :Ontario
 1022:d=7  hl=2 l=  18 cons: SET
 1024:d=8  hl=2 l=  16 cons: SEQUENCE
 1026:d=9  hl=2 l=   3 prim: OBJECT            :organizationalUnitName
 1031:d=9  hl=2 l=   9 prim: UTF8STRING        :Sandelman
 1042:d=7  hl=2 l=  36 cons: SET
 1044:d=8  hl=2 l=  34 cons: SEQUENCE
 1046:d=9  hl=2 l=   3 prim: OBJECT            :commonName
 1051:d=9  hl=2 l=  27 prim: UTF8STRING        :highway-test.example.com
 1080:d=6  hl=2 l=  32 cons: SEQUENCE
 1082:d=7  hl=2 l=  13 prim: UTCTIME           :200203064720Z
 1097:d=7  hl=2 l=  15 prim: GENERALIZEDTIME   :29991231000000Z
 1114:d=6  hl=2 l=  28 cons: SEQUENCE
 1116:d=7  hl=2 l=  26 cons: SET
 1118:d=8  hl=2 l=  24 cons: SEQUENCE
 1120:d=9  hl=2 l=   3 prim: OBJECT            :serialNumber
 1125:d=9  hl=2 l=  17 prim: UTF8STRING        :00-D0-E5-F2-00-02
 1144:d=6  hl=2 l=  89 cons: SEQUENCE
 1146:d=7  hl=2 l=  19 cons: SEQUENCE
 1148:d=8  hl=2 l=   7 prim: OBJECT            :id-ecPublicKey
 1157:d=8  hl=2 l=   8 prim: OBJECT            :prime256v1
```

```
1167:d=7  hl=2 l=  66 prim: BIT STRING
1235:d=6  hl=2 l=  89 cons: cont [ 3 ]
1237:d=7  hl=2 l=  87 cons: SEQUENCE
1239:d=8  hl=2 l=  29 cons: SEQUENCE
1241:d=9  hl=2 l=   3 prim: OBJECT            :X509v3 Subject Key Ident
1246:d=9  hl=2 l=  22 prim: OCTET STRING      [HEX DUMP]:04144588CC9696
1270:d=8  hl=2 l=   9 cons: SEQUENCE
1272:d=9  hl=2 l=   3 prim: OBJECT            :X509v3 Basic Constraints
1277:d=9  hl=2 l=   2 prim: OCTET STRING      [HEX DUMP]:3000
1281:d=8  hl=2 l=  43 cons: SEQUENCE
1283:d=9  hl=2 l=   8 prim: OBJECT            :1.3.6.1.5.5.7.1.32
1293:d=9  hl=2 l=  31 prim: OCTET STRING      [HEX DUMP]:0C1D6869676877
1326:d=5  hl=2 l=  10 cons: SEQUENCE
1328:d=6  hl=2 l=   8 prim: OBJECT            :ecdsa-with-SHA256
1338:d=5  hl=2 l= 104 prim: BIT STRING
1444:d=3  hl=4 l= 314 cons: SET
1448:d=4  hl=4 l= 310 cons: SEQUENCE
1452:d=5  hl=2 l=   1 prim: INTEGER           :01
1455:d=5  hl=2 l= 101 cons: SEQUENCE
1457:d=6  hl=2 l=  93 cons: SEQUENCE
1459:d=7  hl=2 l=  15 cons: SET
1461:d=8  hl=2 l=  13 cons: SEQUENCE
1463:d=9  hl=2 l=   3 prim: OBJECT            :countryName
1468:d=9  hl=2 l=   6 prim: PRINTABLESTRING   :Canada
1476:d=7  hl=2 l=  16 cons: SET
1478:d=8  hl=2 l=  14 cons: SEQUENCE
1480:d=9  hl=2 l=   3 prim: OBJECT            :stateOrProvinceName
1485:d=9  hl=2 l=   7 prim: UTF8STRING        :Ontario
1494:d=7  hl=2 l=  18 cons: SET
1496:d=8  hl=2 l=  16 cons: SEQUENCE
1498:d=9  hl=2 l=   3 prim: OBJECT            :organizationalUnitName
1503:d=9  hl=2 l=   9 prim: UTF8STRING        :Sandelman
1514:d=7  hl=2 l=  36 cons: SET
1516:d=8  hl=2 l=  34 cons: SEQUENCE
1518:d=9  hl=2 l=   3 prim: OBJECT            :commonName
1523:d=9  hl=2 l=  27 prim: UTF8STRING        :highway-test.example.com
1552:d=6  hl=2 l=   4 prim: INTEGER           :0D85DC2D
1558:d=5  hl=2 l=  11 cons: SEQUENCE
1560:d=6  hl=2 l=   9 prim: OBJECT            :sha256
1571:d=5  hl=2 l= 105 cons: cont [ 0 ]
1573:d=6  hl=2 l=  24 cons: SEQUENCE
1575:d=7  hl=2 l=   9 prim: OBJECT            :contentType
1586:d=7  hl=2 l=  11 cons: SET
1588:d=8  hl=2 l=   9 prim: OBJECT            :pkcs7-data
1599:d=6  hl=2 l=  28 cons: SEQUENCE
1601:d=7  hl=2 l=   9 prim: OBJECT            :signingTime
1612:d=7  hl=2 l=  15 cons: SET
1614:d=8  hl=2 l=  13 prim: UTCTIME           :200225213311Z
1629:d=6  hl=2 l=  47 cons: SEQUENCE
```

```
1631:d=7  hl=2 l=   9 prim: OBJECT            :messageDigest
1642:d=7  hl=2 l=  34 cons: SET
1644:d=8  hl=2 l=  32 prim: OCTET STRING      [HEX DUMP]:74C81A3F72527A
1678:d=5  hl=2 l=  10 cons: SEQUENCE
1680:d=6  hl=2 l=   8 prim: OBJECT            :ecdsa-with-SHA256
1690:d=5  hl=2 l=  70 prim: OCTET STRING      [HEX DUMP]:304402205A44AF
```

The JSON contained in the voucher request:

{"ietf-voucher-request:voucher":{"assertion":"proximity","cr
eated-on":"2020-02-25T16:33:11.984-05:00","serial-number":"0
0-D0-E5-F2-00-02","nonce":"y2BfNaIS0KJSyhKamTGXaQ","proximit
y-registrar-cert":"MIIB/DCCAYKgAwIBAgIEP5ibUjAKBggqhkjOPQQDA
jBtMRIwEAYKCZImiZPyLGQBGRYCY2ExGTAXBgoJkiaJk/IsZAEZFglzYW5kZ
WxtYW4xPDA6BgNVBAMMM2ZvdW50YWluLXRlc3QuZXhhbXBsZS5jb20gVW5zd
HJ1bmcgRm91bnRhaW4gUm9vdCBDQTAeFw0yMDAyMjUyMTMxNTRaFw0yMjAyM
jQyMTMxNTRaMFMxEjAQBgoJkiaJk/IsZAEZFgJjYTEZMBcGCgmSJomT8ixkA
RkWCXNhbmRlbG1hbjEiMCAGA1UEAwwZZm91bnRhaW4tdGVzdC5leGFtcGxlL
mNvbTBZMBMGByqGSM49AgEGCCqGSM49AwEHA0IABJZlUHI0up/l3eZf9vCBb
+lInoEMEgc7Ro+XZCtjAI0CD1fJfJR/hIyyDmHWyYiNFbRCH9fyarfkzgX4p
0zTizqjKjAoMBYGA1UdJQEB/wQMMAoGCCsGAQUFBwMcMA4GA1UdDwEB/wQEA
wIHgDAKBggqhkjOPQQDAgNoADBlAjBmT2BMVUgelgf43R+5yBKNRTaHmyPAv
Lvxyz0mFVZvXx+/1RwOagmvG3aXmRkj/X4CMQC8rMNBsLoNr1L5nG56fwAdI
8hiAWG8S8XAR5k1Cgx3YUQBSgdScFcAdf++Bw6Yy+U="}}

C.2.2. Registrar to MASA

As described in Section 5.5 the registrar will sign a registrar voucher-request, and will include pledge's voucher request in the prior-signed-voucher-request.

<CODE BEGINS> file "parboiled_vr_00-D0-E5-F2-00-02.b64"

MIIP9wYJKoZIhvcNAQcCoIIP6DCCD+QCAQExDTALBglghkgBZQMEAgEwggoMBgkqhkiG9w0BBwGg
ggn9BIIJ+XsiaWV0Zi12b3VjaGVyLXJlcXVlc3Q6dm91Y2hlciI6eyJhc3NlcnRpb24iOiJwcm94
aW1pdHkiLCJjcmVhdGVkLW9uIjoiMjAyMC0wMi0yNVQyMzowNDo0OS4wNTRaIiwic2VyaWFsLW51
bWJlciI6IjAwLUQwLUU1LUYyLTAwLTAyIiwibm9uY2UiOiJhTWpndWVLVVQtMjJ3VmltajZ6MjdR
IiwicHJpb3Itc2lnbmVkLXZvdWNoZXItcmVxdWVzdCI6Ik1JSUczd1lKS29aSWh2Y05BUWNDb0lJ
RzBEQ0NCc3dDQVFFeERUQUxCZ2xnaGtnQlpRTUVBZ0V3Z2dPSkJna3Foa2lHOXcwQkJ3R2dnZ042
QklJRGRuc2lhV1YwWmkxMmIzVmphR1Z5TFhKbGNYVmxjM1E2ZG05MVkyaGxjaUk2ZXlKaGMzTmxj
blJwYjI0aU9pSndjbTk0YVcxcGRIa2lMQ0pqY21WaGRHVmtMVzl1SWpvaU1qQXlNQzB3TWkweU5W
UXhPRG93TkRvME9DNDJOVEl0TURVNk1EQWlMQ0p6WlhKcFlXd3RiblZ0WW1WeUlqb2lNREF0UkRB
dFJUVXRSakl0TURBdE1ESWlMQ0p1YjI1alpTSTZJbUZOYW1kMVpVdFZWQzB5TW5kV2FXMXFObm95
TjFFaUxDSndjbTk0YVcxcGRIa3RjbVZuYVhOMGNtRnlMV05sY25RaU9pSk5TVWxDTDBSRFEwRlpT
MmRCZDBsQ1FXZEpSVkExYVdKVmFrRkxRbWRuY1docmFrOVBVFVkZFUVdwQ2RFMVNNTWGRGUVZsTFEx
cEpiV2xhVUhsTVIxRkNSMUpaUTFreVJYaEhWRUZZUW1kdlNtdHBZVXBXTDBseldrRkZXa1puYkhw
WlZ6VnJXbGQ0ZEZsWE5IaFFSRUUyUW1kT1ZrSkJJUVTFOTWxwMlpGYzFNRmxYYkhWTVdGSnNZek5S
ZFZwWWFHaGlXRUp6V2xNMWFtSXlNR2RXVnpWNlpYaEtNV0p0U2JKUa3hZbTVTYUdWE5HZFZi
VGwyWkVOQ1JGRlVRV1ZHZHpCNVRVUkJlVTFxVlhsTlZFMTRUbFJTTWVVaM01IbE5ha0Y1VFdwUmVV
MVVUWGhPVkZKaFRVWk5lRVZxUVZGQ1oyOUthMmxoSmtTZ0pBbFVSVnBOUW1O
SFEyZHRVMHB2YlZRNGFYaHJRVkpyVjBOWVRtaGliVkpzWWtjeGFHSnFSV2xOUTBGSFFURlZSVUYz
ZDFwYWJJUa3hZbTVTYUdWE5IUmtSMVo2WkVNMWJHVkhhSblJqUUjNoc1RHMU9kbUpUVUWxwTlFrMUhR
bmx4UjFOTk5EbEJaUXhZ0xCZDBsQkZVVVQkRVUpLV214VlNFa3dkWEF2YkROFdt
WTVka05NWl0c1NXNXZTVTFGWjJNM1VtOHJXnBEZEdwQlNUQkRSEZtU21aS1VpOW9TWGw1Ukcx
SVYzbFphVTVVHWWxKRFNEbG1lV0Z5Wm10NloxZzBjREI2VkdsNmNXcExxha0Z2VFVKWlIwRXhWV1JL
VVVWQ0wzZFJUVFCYjBkRFEzTkhRVkZWWUmtKM1RXTk5RVFJIUVRGRGVlpFUjNSVUl2ZDFGRlYZEpT
R2RFUVV0Q1oyZHhhR3RxVDFCQUlVVUkJaMDV2UVVSQ2JFRnFRbTFVTWtKTlZsVm5aV3huWmpRelVp
czFlVUpMVGxxKVVlVaHRlVkJCZGt4MmVIbDZNRzFHVmxwMldIZ3JMekZTZDA5aFoyMTJSek5oV0cx
U2Eyb3ZXRFJFVFZGGRE9ISk5Ua0p6VEc5T2NqRk1OVzVVVTlRabWQRmtTVGhvYVVVVGVGJ6aFRPRmhC
VWpWck1VTm5lRE5aVlZGGQ1UyZGtVMk5HWTBGa1ppc3JSbmMyV1hrclZUMGlmWDJnZ2dIcU1JSUI1
akNNQVd5Z0F3SUJBZ0lFTFFlYY0xUQXpCQ2dxaGtqT1BRUURBakJtTVE4d0RRWURWUVFHRXdKRFFlX
NWhaR0V4RURBT0JnTlZCQWdNQjA5dWRHRnlhVzh4RWpBUUJnTlZCQXNNQ1ZOaGJtUmxiRzFoYmpF
a01DSUdBMVVFQXd3YmFHbG5hSGRoZVMxMFpYTjBMbVY0WVcxd2JHVXVZMjl0SUVOQk1DQVhEVEl3
TURJd016QTJORGN5TUZvWUR6STVPVGt4TWpNeE1EQXdNREF3V2pBY01Sb3dHQVlEVlFRRkRBOFV0
QzFFTUxNRk5TMUdNVGMwTURAdE1EQXRNREl3V1RBVEJnY3Foa2pPUFFJQkJnZ3Foa2pPUFFNQkJ3
T2pkVU9IczN6QUNwcUhuSzMyOURnVEhOUU1oQi9nUUU1QmVmQmJFMHNnY2JORW0wbGk4Ull3enJz
QWZZQ3hwOWs3RTFDYnBZWxUOE9XZjB6K0lTZWpXVEJJYTU1WEVyNEExWEVZREZxQkJSRmlNeVdsZ0Jr
TjdDNkkyVmtaRlFJQm14V3JUQUpCZ05WSFJNRUFqQU5NQ3NHQ0NzR0FRVUZCd0VnQkI4TUhXaHBa
MmgzWVhrdGRHVnpkRzVsZUdGdFlsUmVM5UY29QNW9NQW9HQ0NxR1NNNDlCQU1DQU1DQTJnQU1HVUNN
VUNNQ1BocVM3dkloSTBXcVhhRmRZb3lMMDlsdGJPQkpGQmpKYkNvOWdLYk5zMkRHU1NUTmlUWnlJ
a0EwL0Z6ZFlHWdJeEFMT05YQXJRL2dTRGtOTlBiWEtYc3o0QzZ2SElXa3p5ZE9OMThsM1QySVVFRk
STE0aWI4Ti9qSHpYbTNBZ2tiVGhmekdDQVRzd2dnRTNBZ0VCTUdVd1hURVBNQTBHQTFVRUJoTUdR
MkZ1WVdSaE1SQXdEZ1lEVlFRSURBZFBibnJhY21sdk1SSXdFQVlEVlFRTERBbFRRZVZrVrWld4dFlX
NHhKREFpQmdOVkJBTU1HMmhwWjJoM1lYa3RkR1Z6ZEM1bGVHRnRjR3hsTG1OdmJTQkRRUUlFRFFlY
Y0xUQUxCZ2xnaGtnQlpRTUVBZ0dnYVRBWUJna3Foa2lHOXcwQkNRTXhDd1lKS29aSWh2Y05BUWNC
TUJ3R0NTcUdTSWIzRFFFSkJURVBGdzB5TURBeU1qVXhOekEwTkRoYU1DOEdDU3FHU0liM0RRRUpC
REVpQkNDeDZJcndzdEhGNjA5WTBFcURLNjJRS2J5NGR1eXlJV3VkdnMxNU0xNkJCVEFLQmdncWhr
ak9QUVFEQWdSSE1FVUNJQnh3QTFFVbGtJa3VRRGYvajdrWi9NVmVmZ3IxNDEraEtCRmdybk5uZ2p3
cEFpRUF5OGFYdDBHU0I5bTFFbWlFVXBlZkNFaHTdjJ4TFl1ckdsdWd2MGRmci9FPSJ9faCCBG8w
ggH8MIIBgqADAgECAgQ/mJtSMAoGCCqGSM49BAMCMG0xEjAQBgoJkiaJk/IsZAEZFgJjYTEZMBcG

CgmSJomT8ixkARkWCXNhbmRlbG1hbjE8MDoGA1UEAwwzZm91bnRhaW4tdGVzdC5leGFtcGxlLmNv
bSBVbnN0cnVuZyBGb3VudGFpbiBSb290IENBMB4XDTIwMDIyNTIxMzE1NFoXDTIyMDIyNDIxMzE1
NFowUzESMBAGCgmSJomT8ixkARkWAmNhMRkwFwYKCZImiZPyLGQBGRYJc2FuZGVsbWFuMSIwIAYD
VQQDDBlmb3VudGFpbi10ZXN0LmV4YW1wbGUuY29tMFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAE
lmVQcjS6n+Xd5l/28IFv6UiegQwSBztGj5dkK2MAjQIPV8l8lH+EjLIOYdbJiI0VtEIf1/Jqt+TO
BfinTNOLOqMqMCgwFgYDVR0lAQH/BAwwCgYIKwYBBQUHAxwwDgYDVR0PAQH/BAQDAgeAMAoGCCqG
SM49BAMCA2gAMGUCMGZPYExVSB6WB/jdH7nIEo1FNoebI8C8u/HLPSYVVm9fH7/VHA5qCa8bdpeZ
GSP9fgIxALysw0Gwug2vUvmcbnp/AB0jyGIBYbxLxcBHmTUKDHdhRAFKB1JwVwB1/74HDpjL5TCC
AmswggHyoAMCAQICBClrBlkwCgYIKoZIzj0EAwIwbTESMBAGCgmSJomT8ixkARkWAmNhMRkwFwYK
CZImiZPyLGQBGRYJc2FuZGVsbWFuMTwwOgYDVQQDDDNmb3VudGFpbi10ZXN0LmV4YW1wbGUuY29t
IFVuc3RydW5nIEZvdW50YWluIFJvb3QgQ0EwHhcNMjAwMjI1MjEzMTQ1WhcNMjIwMjI0MjEzMTQ1
WjBtMRIwEAYKCZImiZPyLGQBGRYCY2ExGTAXBgoJkiaJk/IsZAEZFglzYW5kZWxtYW4xPDA6BgNV
BAMMM2ZvdW50YWluLXRlc3QuZXhhbXBsZS5jb20gVW5zdHJ1bmcgRm91bnRhaW4gUm9vdCBDQTB2
MBAGByqGSM49AgEGBSuBBAAiA2IABBt/WboXwxq8Zo2MbODD+jFxD2X2IpG9t1aAB9vfuHqlRU15
ikaXGVmWMbGPaX0yvjzIPltjtUb2qNVvm/nA89O5FD9yR1Gkdt3S8L/1yo8wAX/4wl/T9SADRIuL
8gdstKNjMGEwDwYDVR0TAQH/BAUwAwEB/zAOBgNVHQ8BAf8EBAMCAQYwHQYDVR0OBBYEFLml9ssR
4QekSSynCMZ8ELyHs3QmMB8GA1UdIwQYMBaAFLml9ssR4QekSSynCMZ8ELyHs3QmMAoGCCqGSM49
BAMCA2cAMGQCMCCDBs6NmKRUemZMSjpwwlI2WlKNWX0gmyppFFiHONhVed39KTiVHpGTdrT1ZilE
tAIwbzj5rxLtMNWFKXyxFli9Z5FDxA0w+dgcrC8G3bzVBkIshKIE6gKkXxdRJvvZL9JcMYIBSzCC
AUcCAQEwdTBtMRIwEAYKCZImiZPyLGQBGRYCY2ExGTAXBgoJkiaJk/IsZAEZFglzYW5kZWxtYW4x
PDA6BgNVBAMMM2ZvdW50YWluLXRlc3QuZXhhbXBsZS5jb20gVW5zdHJ1bmcgRm91bnRhaW4gUm9v
dCBDQQIEP5ibUjALBglghkgBZQMEAgGgaTAYBgkqhkiG9w0BCQMxCwYJKoZIhvcNAQcBMBwGCSqG
SIb3DQEJBTEPFw0yMDAyMjUyMzA0NDlaMC8GCSqGSIb3DQEJBDEiBCA9gYxR1sS0giII3PwvOK/N
5RUBwjSL/cDcrH/Bd+E1ajAKBggqhkjOPQQDAgRHMEUCIFieXZaO7P9eZMpCVn2laB4czw7I0s0P
s9+frcJtEBTTAiEAhCcB//qmgqcEA+90mquvVNENmFH9dxCH8Ihhz6SCVDI=

<CODE ENDS>

The ASN1 decoding of the artifact:

file: examples/parboiled_vr_00_D0-E5-02-00-2D.b64

```
   0:d=0  hl=4 l=4087 cons: SEQUENCE
   4:d=1  hl=2 l=   9 prim: OBJECT            :pkcs7-signedData
  15:d=1  hl=4 l=4072 cons: cont [ 0 ]
  19:d=2  hl=4 l=4068 cons: SEQUENCE
  23:d=3  hl=2 l=   1 prim: INTEGER           :01
  26:d=3  hl=2 l=  13 cons: SET
  28:d=4  hl=2 l=  11 cons: SEQUENCE
  30:d=5  hl=2 l=   9 prim: OBJECT            :sha256
  41:d=3  hl=4 l=2572 cons: SEQUENCE
  45:d=4  hl=2 l=   9 prim: OBJECT            :pkcs7-data
  56:d=4  hl=4 l=2557 cons: cont [ 0 ]
  60:d=5  hl=4 l=2553 prim: OCTET STRING      :{"ietf-voucher-request:v
2617:d=3  hl=4 l=1135 cons: cont [ 0 ]
2621:d=4  hl=4 l= 508 cons: SEQUENCE
2625:d=5  hl=4 l= 386 cons: SEQUENCE
2629:d=6  hl=2 l=   3 cons: cont [ 0 ]
2631:d=7  hl=2 l=   1 prim: INTEGER           :02
2634:d=6  hl=2 l=   4 prim: INTEGER           :3F989B52
2640:d=6  hl=2 l=  10 cons: SEQUENCE
2642:d=7  hl=2 l=   8 prim: OBJECT            :ecdsa-with-SHA256
2652:d=6  hl=2 l= 109 cons: SEQUENCE
2654:d=7  hl=2 l=  18 cons: SET
2656:d=8  hl=2 l=  16 cons: SEQUENCE
2658:d=9  hl=2 l=  10 prim: OBJECT            :domainComponent
2670:d=9  hl=2 l=   2 prim: IA5STRING         :ca
2674:d=7  hl=2 l=  25 cons: SET
2676:d=8  hl=2 l=  23 cons: SEQUENCE
2678:d=9  hl=2 l=  10 prim: OBJECT            :domainComponent
2690:d=9  hl=2 l=   9 prim: IA5STRING         :sandelman
2701:d=7  hl=2 l=  60 cons: SET
2703:d=8  hl=2 l=  58 cons: SEQUENCE
2705:d=9  hl=2 l=   3 prim: OBJECT            :commonName
2710:d=9  hl=2 l=  51 prim: UTF8STRING        :fountain-test.example.co
2763:d=6  hl=2 l=  30 cons: SEQUENCE
2765:d=7  hl=2 l=  13 prim: UTCTIME           :200225213154Z
2780:d=7  hl=2 l=  13 prim: UTCTIME           :220224213154Z
2795:d=6  hl=2 l=  83 cons: SEQUENCE
2797:d=7  hl=2 l=  18 cons: SET
2799:d=8  hl=2 l=  16 cons: SEQUENCE
2801:d=9  hl=2 l=  10 prim: OBJECT            :domainComponent
2813:d=9  hl=2 l=   2 prim: IA5STRING         :ca
2817:d=7  hl=2 l=  25 cons: SET
2819:d=8  hl=2 l=  23 cons: SEQUENCE
2821:d=9  hl=2 l=  10 prim: OBJECT            :domainComponent
2833:d=9  hl=2 l=   9 prim: IA5STRING         :sandelman
2844:d=7  hl=2 l=  34 cons: SET
2846:d=8  hl=2 l=  32 cons: SEQUENCE
2848:d=9  hl=2 l=   3 prim: OBJECT            :commonName
2853:d=9  hl=2 l=  25 prim: UTF8STRING        :fountain-test.example.co
```

```
2880:d=6  hl=2 l=  89 cons: SEQUENCE
2882:d=7  hl=2 l=  19 cons: SEQUENCE
2884:d=8  hl=2 l=   7 prim: OBJECT            :id-ecPublicKey
2893:d=8  hl=2 l=   8 prim: OBJECT            :prime256v1
2903:d=7  hl=2 l=  66 prim: BIT STRING
2971:d=6  hl=2 l=  42 cons: cont [ 3 ]
2973:d=7  hl=2 l=  40 cons: SEQUENCE
2975:d=8  hl=2 l=  22 cons: SEQUENCE
2977:d=9  hl=2 l=   3 prim: OBJECT            :X509v3 Extended Key Usag
2982:d=9  hl=2 l=   1 prim: BOOLEAN           :255
2985:d=9  hl=2 l=  12 prim: OCTET STRING      [HEX DUMP]:300A06082B0601
2999:d=8  hl=2 l=  14 cons: SEQUENCE
3001:d=9  hl=2 l=   3 prim: OBJECT            :X509v3 Key Usage
3006:d=9  hl=2 l=   1 prim: BOOLEAN           :255
3009:d=9  hl=2 l=   4 prim: OCTET STRING      [HEX DUMP]:03020780
3015:d=5  hl=2 l=  10 cons: SEQUENCE
3017:d=6  hl=2 l=   8 prim: OBJECT            :ecdsa-with-SHA256
3027:d=5  hl=2 l= 104 prim: BIT STRING
3133:d=4  hl=4 l= 619 cons: SEQUENCE
3137:d=5  hl=4 l= 498 cons: SEQUENCE
3141:d=6  hl=2 l=   3 cons: cont [ 0 ]
3143:d=7  hl=2 l=   1 prim: INTEGER           :02
3146:d=6  hl=2 l=   4 prim: INTEGER           :296B0659
3152:d=6  hl=2 l=  10 cons: SEQUENCE
3154:d=7  hl=2 l=   8 prim: OBJECT            :ecdsa-with-SHA256
3164:d=6  hl=2 l= 109 cons: SEQUENCE
3166:d=7  hl=2 l=  18 cons: SET
3168:d=8  hl=2 l=  16 cons: SEQUENCE
3170:d=9  hl=2 l=  10 prim: OBJECT            :domainComponent
3182:d=9  hl=2 l=   2 prim: IA5STRING         :ca
3186:d=7  hl=2 l=  25 cons: SET
3188:d=8  hl=2 l=  23 cons: SEQUENCE
3190:d=9  hl=2 l=  10 prim: OBJECT            :domainComponent
3202:d=9  hl=2 l=   9 prim: IA5STRING         :sandelman
3213:d=7  hl=2 l=  60 cons: SET
3215:d=8  hl=2 l=  58 cons: SEQUENCE
3217:d=9  hl=2 l=   3 prim: OBJECT            :commonName
3222:d=9  hl=2 l=  51 prim: UTF8STRING        :fountain-test.example.co
3275:d=6  hl=2 l=  30 cons: SEQUENCE
3277:d=7  hl=2 l=  13 prim: UTCTIME           :200225213145Z
3292:d=7  hl=2 l=  13 prim: UTCTIME           :220224213145Z
3307:d=6  hl=2 l= 109 cons: SEQUENCE
3309:d=7  hl=2 l=  18 cons: SET
3311:d=8  hl=2 l=  16 cons: SEQUENCE
3313:d=9  hl=2 l=  10 prim: OBJECT            :domainComponent
3325:d=9  hl=2 l=   2 prim: IA5STRING         :ca
3329:d=7  hl=2 l=  25 cons: SET
3331:d=8  hl=2 l=  23 cons: SEQUENCE
3333:d=9  hl=2 l=  10 prim: OBJECT            :domainComponent
```

```
3345:d=9  hl=2 l=   9 prim: IA5STRING         :sandelman
3356:d=7  hl=2 l=  60 cons: SET
3358:d=8  hl=2 l=  58 cons: SEQUENCE
3360:d=9  hl=2 l=   3 prim: OBJECT            :commonName
3365:d=9  hl=2 l=  51 prim: UTF8STRING        :fountain-test.example.co
3418:d=6  hl=2 l= 118 cons: SEQUENCE
3420:d=7  hl=2 l=  16 cons: SEQUENCE
3422:d=8  hl=2 l=   7 prim: OBJECT            :id-ecPublicKey
3431:d=8  hl=2 l=   5 prim: OBJECT            :secp384r1
3438:d=7  hl=2 l=  98 prim: BIT STRING
3538:d=6  hl=2 l=  99 cons: cont [ 3 ]
3540:d=7  hl=2 l=  97 cons: SEQUENCE
3542:d=8  hl=2 l=  15 cons: SEQUENCE
3544:d=9  hl=2 l=   3 prim: OBJECT            :X509v3 Basic Constraints
3549:d=9  hl=2 l=   1 prim: BOOLEAN           :255
3552:d=9  hl=2 l=   5 prim: OCTET STRING      [HEX DUMP]:30030101FF
3559:d=8  hl=2 l=  14 cons: SEQUENCE
3561:d=9  hl=2 l=   3 prim: OBJECT            :X509v3 Key Usage
3566:d=9  hl=2 l=   1 prim: BOOLEAN           :255
3569:d=9  hl=2 l=   4 prim: OCTET STRING      [HEX DUMP]:03020106
3575:d=8  hl=2 l=  29 cons: SEQUENCE
3577:d=9  hl=2 l=   3 prim: OBJECT            :X509v3 Subject Key Ident
3582:d=9  hl=2 l=  22 prim: OCTET STRING      [HEX DUMP]:0414B9A5F6CB11
3606:d=8  hl=2 l=  31 cons: SEQUENCE
3608:d=9  hl=2 l=   3 prim: OBJECT            :X509v3 Authority Key Ide
3613:d=9  hl=2 l=  24 prim: OCTET STRING      [HEX DUMP]:30168014B9A5F6
3639:d=5  hl=2 l=  10 cons: SEQUENCE
3641:d=6  hl=2 l=   8 prim: OBJECT            :ecdsa-with-SHA256
3651:d=5  hl=2 l= 103 prim: BIT STRING
3756:d=3  hl=4 l= 331 cons: SET
3760:d=4  hl=4 l= 327 cons: SEQUENCE
3764:d=5  hl=2 l=   1 prim: INTEGER           :01
3767:d=5  hl=2 l= 117 cons: SEQUENCE
3769:d=6  hl=2 l= 109 cons: SEQUENCE
3771:d=7  hl=2 l=  18 cons: SET
3773:d=8  hl=2 l=  16 cons: SEQUENCE
3775:d=9  hl=2 l=  10 prim: OBJECT            :domainComponent
3787:d=9  hl=2 l=   2 prim: IA5STRING         :ca
3791:d=7  hl=2 l=  25 cons: SET
3793:d=8  hl=2 l=  23 cons: SEQUENCE
3795:d=9  hl=2 l=  10 prim: OBJECT            :domainComponent
3807:d=9  hl=2 l=   9 prim: IA5STRING         :sandelman
3818:d=7  hl=2 l=  60 cons: SET
3820:d=8  hl=2 l=  58 cons: SEQUENCE
3822:d=9  hl=2 l=   3 prim: OBJECT            :commonName
3827:d=9  hl=2 l=  51 prim: UTF8STRING        :fountain-test.example.co
3880:d=6  hl=2 l=   4 prim: INTEGER           :3F989B52
3886:d=5  hl=2 l=  11 cons: SEQUENCE
3888:d=6  hl=2 l=   9 prim: OBJECT            :sha256
```

```
3899:d=5  hl=2 l= 105 cons: cont [ 0 ]
3901:d=6  hl=2 l=  24 cons: SEQUENCE
3903:d=7  hl=2 l=   9 prim: OBJECT            :contentType
3914:d=7  hl=2 l=  11 cons: SET
3916:d=8  hl=2 l=   9 prim: OBJECT            :pkcs7-data
3927:d=6  hl=2 l=  28 cons: SEQUENCE
3929:d=7  hl=2 l=   9 prim: OBJECT            :signingTime
3940:d=7  hl=2 l=  15 cons: SET
3942:d=8  hl=2 l=  13 prim: UTCTIME           :200225230449Z
3957:d=6  hl=2 l=  47 cons: SEQUENCE
3959:d=7  hl=2 l=   9 prim: OBJECT            :messageDigest
3970:d=7  hl=2 l=  34 cons: SET
3972:d=8  hl=2 l=  32 prim: OCTET STRING      [HEX DUMP]:3D818C51D6C4B4
4006:d=5  hl=2 l=  10 cons: SEQUENCE
4008:d=6  hl=2 l=   8 prim: OBJECT            :ecdsa-with-SHA256
4018:d=5  hl=2 l=  71 prim: OCTET STRING      [HEX DUMP]:30450220589E5D
```

The JSON contained in the voucher request. Note that the previous voucher request is in the prior-signed-voucher-request attribute.


```
{"ietf-voucher-request:voucher":{"assertion":"proximity","cr
eated-on":"2020-02-25T23:04:49.054Z","serial-number":"00-D0-
E5-F2-00-02","nonce":"aMjgueKUT-22wVimj6z27Q","prior-signed-
voucher-request":"MIIG3wYJKoZIhvcNAQcCoIIG0DCCBswCAQExDTALBg
lghkgBZQMEAgEwggOJBgkqhkiG9w0BBwGgggN6BIIDdnsiaWV0Zi12b3VjaG
VyLXJlcXVlc3Q6dm91Y2hlciI6eyJhc3NlcnRpb24iOiJwcm94aW1pdHkiLC
JjcmVhdGVkLW9uIjoiMjAyMC0wMi0yNVQxODowNDo0OC42NTItMDU6MDAiLC
JzZXJpYWwtbnVtYmVyIjoiMDAtRDAtRTUtRjItMDAtMDIiLCJub25jZSI6Im
FNamd1ZUtVVC0yMndWaW1qNnoyN1EiLCJwcm94aW1pdHktcmVnaXN0cmFyLW
NlcnQiOiJNSUlCL0RDQ0FZS2dBd0lCQWdJRVA1aWJVakFLQmdncWhrak9QUV
FEQWpCdE1SSXdFQVlLQ1pJbWlaUHlMR1FCR1JZQ1kyRXhHVEFYQmdvSmtpYU
prL0lzWkFFWkZnbHpZVzVrVld4dFlXNHhQREE2QmdOVkJBTU1NMlp2ZFc1MF
lXbHVMWFJsYzNRdVpYaGhiWEJzWlM1amIyMGdWVzV6ZEhKMWJtY2dSbTkxYm
5SaGFXNGdVbTl2ZENCRFFUQWVGdzB5TURBeU1qVXlNVE14TlRSYUZ3MHlNak
F5TWpReU1UTXhOVFJhTUZNeEVqQVFCZ29Ka2lhSmsvSXNaQUVaRmdKallURV
pNQmNHQ2dtU0pvbVQ4aXhrQVJrV0NYTmhibVJsYkcxaGJqRWlNQ0FHQTFVRU
F3d1pabTkxYm5SaGFXNHRkR1Z6ZEM1bGVHRnRjR3hsTG1OdmJUQlpNQk1HQn
lxR1NNNDlBZ0VHQ0NxR1NNNDlBd0VIQTBJQUJKWmxVSEkwdXAvbDNlWmY5dk
NCYitsSW5vRU1FZ2M3Um8rWFpDdGpBSTBDRDFmSmZKUi9oSXl5RG1IV3lZaU
5GYlJDSDlmeWFyZmt6Z1g0cDB6VGl6cWpLakFvTUJZR0ExVWRKUUVCL3dRTU
1Bb0dDQ3NHQVFVRkJ3TWNNQTRHQTFVZER3RUIvd1FFQXdJSGdEQUtCZ2dxaG
tqT1BRUURBZ05vQURCbEFqQm1UMkJNVlVnZWxnZjQzUis1eUJLTlJUYUhteV
BBdkx2eHl6MG1GVlp2WHgrLzFSd09hZ212RzNhWG1Sa2ovWDRDTVFDOHJNTk
JzTG9OcjFMNW5HNTZmd0FkSThoaUFXRzhTOFhBUjVrMUNneDNZVVFCU2dkU2
NGY0FkZisrQnc2WXkrVT0ifX2gggHqMIIB5jCCAWygAwIBAgIEDYXcLTAKBg
gqhkjOPQQDAjBdMQ8wDQYDVQQGEwZDYW5hZGExEDAOBgNVBAgMB09udGFyaW
8xEjAQBgNVBAsMCVNhbmRlbG1hbjEkMCIGA1UEAwwbaGlnaHdheS10ZXN0Lm
V4YW1wbGUuY29tIENBMCAXDTIwMDIwMzA2NDcyMFoYDzI5OTkxMjMxMDAwMD
AwWjAcMRowGAYDVQQFDBEwMC1EMC1FNS1GMi0wMC0wMjBZMBMGByqGSM49Ag
EGCCqGSM49AwEHA0IABAOjdUOHs3zACpqHnK329DgTHNQMhB/gQE5BefBbE0
sgcbNEm0li8RYwzrsAfYCxp9k7E1CbpielT8OWf0z+ISejWTBXMB0GA1UdDg
QWBBRFiMyWlgBkN7C6I2VkZFQIBmxWrTAJBgNVHRMEAjAAMCsGCCsGAQUFBw
EgBB8MHWhpZ2h3YXktdGVzdC5leGFtcGxlLmNvbTo5NDQzMAoGCCqGSM49BA
MCA2gAMGUCMCPhqS7vIhI0WqXCFdYove09ltbOBJXvp8jcGKgxx7gENPK3TX
mKZyIkA0/FzdYGugIxALONXArQ/gSDkNNPbXKXsz4C6vHIWjJyWLdFAlB4vA
QdI14ib8N/jHzXm3AgkbThfzGCATswggE3AgEBMGUwXTEPMA0GA1UEBhMGQ2
FuYWRhMRAwDgYDVQQIDAdPbnRhcmlvMRIwEAYDVQQLDAlTYW5kZWxtYW4xJD
AiBgNVBAMMG2hpZ2h3YXktdGVzdC5leGFtcGxlLmNvbSBDQQIEDYXcLTALBg
lghkgBZQMEAgGgaTAYBgkqhkiG9w0BCQMxCwYJKoZIhvcNAQcBMBwGCSqGSI
b3DQEJBTEPFw0yMDAyMjUyMzA0NDhaMC8GCSqGSIb3DQEJBDEiBCCx6Irwst
HF609Y0EqDK62QKby4duyyIWudvs15M16BBTAKBggqhkjOPQQDAgRHMEUCIB
xwA1UlkIkuQDf/j7kZ/MVefgr141+hKBFgrnNngjwpAiEAy8aXt0GSB9m1bm
iEUpefCEhxSv2xLYurGlugv0dfr/E="}}
```

C.2.3. MASA to Registrar

The MASA will return a voucher to the registrar, to be relayed to the pledge.

```
<CODE BEGINS> file "voucher_00-D0-E5-F2-00-02.b64"
```


```
MIIGyAYJKoZIhvcNAQcCoIIGuTCCBrUCAQExDTALBglghkgBZQMEAgEwggN4BgkqhkiG9w0BBwGg
ggNpBIIDZXsiaWV0Zi12b3VjaGVyOnZvdWNoZXIiOnsiYXNzZXJ0aW9uIjoibG9nZ2VkIiwiY3Jl
YXRlZC1vbiI6IjIwMjAtMDItMjVUMTY6MzM6MTIuODQ5LTA1OjAwIiwic2VyaWFsLW51bWJlciI6
IjAwLUQwLUU1LUYyLTAwLTAyIiwibm9uY2UiOiJ5MkJmTmFJUzBLSlN5aEthbVRHWGFRIiwicGlu
bmVkLWRvbWFpbi1jZXJ0IjoiTUlJQi9EQ0NBWUtnQXdJQkFnSUVQNWliVWpBS0JnZ3Foa2pPUFFR
REFqQnRNUkl3RUFZS0NaSW1pWlB5TEdRQkdSWUNZMkV4R1RBWEJnb0praWFKay9Jc1pBRVpGZ2x6
WVc1a1pXeHRZVzR4UERBNkJnTlZCQU1NTTJadmRXNTBZV2x1TFhSbGMzUXVaWGhoYlhCc1pTNWpi
MjBnVlc1emRISjFibWNnUm05MWJuUmhhVzRnVW05dmRDQkRRVEFlRncweU1EQXlNalV5TVRNeE5U
UmFGdzB5TWpBeU1qUXlNVE14TlRSYU1GTXhFakFRQmdvSmtpYUprL0lzWkFFWkZnSmpZVEVaTUJj
R0NnbVNKb21UOGl4a0FSa1dDWE5oYm1SbGJHMWhiakVpTUNBR0ExVUVBd3daWm05MWJuUmhhVzR0
ZEdWemRDNWxlR0Z0Y0d4bExtTnZiVEJaTUJNR0J5cUdTTTQ5QWdFR0NDcUdTTTQ5QXdFSEEwSUFC
SlpsVUhJMHVwL2wzZVpmOXZDQmIrbElub0VNRWdjN1JvK1haQ3RqQUkwQ0QxZkpmSlIvaEl5eURt
SFd5WWlORmJSQ0g5ZnlhcmZremdYNHAwelRpenFqS2pBb01CWUdBMVVkSlFFQi93UU1NQW9HQ0Nz
R0FRVUZCd01jTUE0R0ExVWREd0VCL3dRRUF3SUhnREFLQmdncWhrak9QUVFEQWdOb0FEQmxBakJt
VDJCTVZVZ2VsZ2Y0M1IrNXlCS05SVGFIbXlQQXZMdnh5ejBtRlZadlh4Ky8xUndPYWdtdkczYVht
UmtqL1g0Q01RQzhyTU5Cc0xvTnIxTDVuRzU2ZndBZEk4aGlBV0c4UzhYQVI1azFDZ3gzWVVRQlNn
ZFNjRmNBZGYrK0J3Nll5K1U9In19oIIB4zCCAd8wggFkoAMCAQICBBuZX1QwCgYIKoZIzj0EAwIw
XTEPMA0GA1UEBhMGQ2FuYWRhMRAwDgYDVQQIDAdPbnRhcmlvMRIwEAYDVQQLDAlTYW5kZWxtYW4x
JDAiBgNVBAMMG2hpZ2h3YXktdGVzdC5leGFtcGxlLmNvbSBDQTAeFw0xOTAyMTIyMjIyNDFaFw0y
MTAyMTEyMjIyNDFaMF8xDzANBgNVBAYTBkNhbmFkYTEQMA4GA1UECAwHT250YXJpbzESMBAGA1UE
CwwJU2FuZGVsbWFuMSYwJAYDVQQDDB1oaWdod2F5LXRlc3QuZXhhbXBsZS5jb20gTUFTQTBZMBMG
ByqGSM49AgEGCCqGSM49AwEHA0IABKoEFaNEueJE+Mn5GwcbpnRznB66bKmzqTCpojJZ96AdRwFt
uTCVfoKouLTBX0idIhMLfJLM31lyuKy4CUtpp6WjEDAOMAwGA1UdEwEB/wQCMAAwCgYIKoZIzj0E
AwIDaQAwZgIxAL1V5ZsO+/xelSnjgbMVNaqTGKIEvkRyslF9TW3r0dXBEDqyOXtXP8XMsKMO55lG
ugIxAPZ/RH23FPrRZ2rUEcNLrub7mphW+oUhLlxITPA/8ps/roggp675cv9b+Xhozw9IyTGCATww
ggE4AgEBMGUwXTEPMA0GA1UEBhMGQ2FuYWRhMRAwDgYDVQQIDAdPbnRhcmlvMRIwEAYDVQQLDAlT
YW5kZWxtYW4xJDAiBgNVBAMMG2hpZ2h3YXktdGVzdC5leGFtcGxlLmNvbSBDQQIEG5lfVDALBglg
hkgBZQMEAgGgaTAYBgkqhkiG9w0BCQMxCwYJKoZIhvcNAQcBMBwGCSqGSIb3DQEJBTEPFw0yMDAy
MjUyMTMzMTJaMC8GCSqGSIb3DQEJBDEiBCAUaEb583jZIHCPYTPwZSRKJ7XONLRtjvf071ZCHb6i
6jAKBggqhkjOPQQDAgRIMEYCIQCOzdl41Cdb0OycWymjmoT6alVT+mPvGfVPl2xyMxqwHwIhAPri
bTuYNFARgkPLS5MX6iZx5IKhiTCjbJKY8hwfAqCE
```


```
<CODE ENDS>
```

The ASN1 decoding of the artifact:

file: examples/voucher_00-D0-E5-F2-00-02.b64

```
   0:d=0  hl=4 l=1736 cons: SEQUENCE
   4:d=1  hl=2 l=   9 prim: OBJECT            :pkcs7-signedData
  15:d=1  hl=4 l=1721 cons: cont [ 0 ]
  19:d=2  hl=4 l=1717 cons: SEQUENCE
  23:d=3  hl=2 l=   1 prim: INTEGER           :01
  26:d=3  hl=2 l=  13 cons: SET
  28:d=4  hl=2 l=  11 cons: SEQUENCE
  30:d=5  hl=2 l=   9 prim: OBJECT            :sha256
  41:d=3  hl=4 l= 888 cons: SEQUENCE
  45:d=4  hl=2 l=   9 prim: OBJECT            :pkcs7-data
  56:d=4  hl=4 l= 873 cons: cont [ 0 ]
  60:d=5  hl=4 l= 869 prim: OCTET STRING      :{"ietf-voucher:voucher":
 933:d=3  hl=4 l= 483 cons: cont [ 0 ]
 937:d=4  hl=4 l= 479 cons: SEQUENCE
 941:d=5  hl=4 l= 356 cons: SEQUENCE
 945:d=6  hl=2 l=   3 cons: cont [ 0 ]
 947:d=7  hl=2 l=   1 prim: INTEGER           :02
 950:d=6  hl=2 l=   4 prim: INTEGER           :1B995F54
 956:d=6  hl=2 l=  10 cons: SEQUENCE
 958:d=7  hl=2 l=   8 prim: OBJECT            :ecdsa-with-SHA256
 968:d=6  hl=2 l=  93 cons: SEQUENCE
 970:d=7  hl=2 l=  15 cons: SET
 972:d=8  hl=2 l=  13 cons: SEQUENCE
 974:d=9  hl=2 l=   3 prim: OBJECT            :countryName
 979:d=9  hl=2 l=   6 prim: PRINTABLESTRING   :Canada
 987:d=7  hl=2 l=  16 cons: SET
 989:d=8  hl=2 l=  14 cons: SEQUENCE
 991:d=9  hl=2 l=   3 prim: OBJECT            :stateOrProvinceName
 996:d=9  hl=2 l=   7 prim: UTF8STRING        :Ontario
1005:d=7  hl=2 l=  18 cons: SET
1007:d=8  hl=2 l=  16 cons: SEQUENCE
1009:d=9  hl=2 l=   3 prim: OBJECT            :organizationalUnitName
1014:d=9  hl=2 l=   9 prim: UTF8STRING        :Sandelman
1025:d=7  hl=2 l=  36 cons: SET
1027:d=8  hl=2 l=  34 cons: SEQUENCE
1029:d=9  hl=2 l=   3 prim: OBJECT            :commonName
1034:d=9  hl=2 l=  27 prim: UTF8STRING        :highway-test.example.com
1063:d=6  hl=2 l=  30 cons: SEQUENCE
1065:d=7  hl=2 l=  13 prim: UTCTIME           :190212222241Z
1080:d=7  hl=2 l=  13 prim: UTCTIME           :210211222241Z
1095:d=6  hl=2 l=  95 cons: SEQUENCE
1097:d=7  hl=2 l=  15 cons: SET
1099:d=8  hl=2 l=  13 cons: SEQUENCE
1101:d=9  hl=2 l=   3 prim: OBJECT            :countryName
1106:d=9  hl=2 l=   6 prim: PRINTABLESTRING   :Canada
1114:d=7  hl=2 l=  16 cons: SET
1116:d=8  hl=2 l=  14 cons: SEQUENCE
1118:d=9  hl=2 l=   3 prim: OBJECT            :stateOrProvinceName
1123:d=9  hl=2 l=   7 prim: UTF8STRING        :Ontario
```

```
1132:d=7  hl=2 l=  18 cons: SET
1134:d=8  hl=2 l=  16 cons: SEQUENCE
1136:d=9  hl=2 l=   3 prim: OBJECT            :organizationalUnitName
1141:d=9  hl=2 l=   9 prim: UTF8STRING        :Sandelman
1152:d=7  hl=2 l=  38 cons: SET
1154:d=8  hl=2 l=  36 cons: SEQUENCE
1156:d=9  hl=2 l=   3 prim: OBJECT            :commonName
1161:d=9  hl=2 l=  29 prim: UTF8STRING        :highway-test.example.com
1192:d=6  hl=2 l=  89 cons: SEQUENCE
1194:d=7  hl=2 l=  19 cons: SEQUENCE
1196:d=8  hl=2 l=   7 prim: OBJECT            :id-ecPublicKey
1205:d=8  hl=2 l=   8 prim: OBJECT            :prime256v1
1215:d=7  hl=2 l=  66 prim: BIT STRING
1283:d=6  hl=2 l=  16 cons: cont [ 3 ]
1285:d=7  hl=2 l=  14 cons: SEQUENCE
1287:d=8  hl=2 l=  12 cons: SEQUENCE
1289:d=9  hl=2 l=   3 prim: OBJECT            :X509v3 Basic Constraints
1294:d=9  hl=2 l=   1 prim: BOOLEAN           :255
1297:d=9  hl=2 l=   2 prim: OCTET STRING      [HEX DUMP]:3000
1301:d=5  hl=2 l=  10 cons: SEQUENCE
1303:d=6  hl=2 l=   8 prim: OBJECT            :ecdsa-with-SHA256
1313:d=5  hl=2 l= 105 prim: BIT STRING
1420:d=3  hl=4 l= 316 cons: SET
1424:d=4  hl=4 l= 312 cons: SEQUENCE
1428:d=5  hl=2 l=   1 prim: INTEGER           :01
1431:d=5  hl=2 l= 101 cons: SEQUENCE
1433:d=6  hl=2 l=  93 cons: SEQUENCE
1435:d=7  hl=2 l=  15 cons: SET
1437:d=8  hl=2 l=  13 cons: SEQUENCE
1439:d=9  hl=2 l=   3 prim: OBJECT            :countryName
1444:d=9  hl=2 l=   6 prim: PRINTABLESTRING   :Canada
1452:d=7  hl=2 l=  16 cons: SET
1454:d=8  hl=2 l=  14 cons: SEQUENCE
1456:d=9  hl=2 l=   3 prim: OBJECT            :stateOrProvinceName
1461:d=9  hl=2 l=   7 prim: UTF8STRING        :Ontario
1470:d=7  hl=2 l=  18 cons: SET
1472:d=8  hl=2 l=  16 cons: SEQUENCE
1474:d=9  hl=2 l=   3 prim: OBJECT            :organizationalUnitName
1479:d=9  hl=2 l=   9 prim: UTF8STRING        :Sandelman
1490:d=7  hl=2 l=  36 cons: SET
1492:d=8  hl=2 l=  34 cons: SEQUENCE
1494:d=9  hl=2 l=   3 prim: OBJECT            :commonName
1499:d=9  hl=2 l=  27 prim: UTF8STRING        :highway-test.example.com
1528:d=6  hl=2 l=   4 prim: INTEGER           :1B995F54
1534:d=5  hl=2 l=  11 cons: SEQUENCE
1536:d=6  hl=2 l=   9 prim: OBJECT            :sha256
1547:d=5  hl=2 l= 105 cons: cont [ 0 ]
1549:d=6  hl=2 l=  24 cons: SEQUENCE
1551:d=7  hl=2 l=   9 prim: OBJECT            :contentType
```

```
1562:d=7  hl=2 l=  11 cons: SET
1564:d=8  hl=2 l=   9 prim: OBJECT            :pkcs7-data
1575:d=6  hl=2 l=  28 cons: SEQUENCE
1577:d=7  hl=2 l=   9 prim: OBJECT            :signingTime
1588:d=7  hl=2 l=  15 cons: SET
1590:d=8  hl=2 l=  13 prim: UTCTIME           :200225213312Z
1605:d=6  hl=2 l=  47 cons: SEQUENCE
1607:d=7  hl=2 l=   9 prim: OBJECT            :messageDigest
1618:d=7  hl=2 l=  34 cons: SET
1620:d=8  hl=2 l=  32 prim: OCTET STRING      [HEX DUMP]:146846F9F378D9
1654:d=5  hl=2 l=  10 cons: SEQUENCE
1656:d=6  hl=2 l=   8 prim: OBJECT            :ecdsa-with-SHA256
1666:d=5  hl=2 l=  72 prim: OCTET STRING      [HEX DUMP]:30460221008ECD
```

Appendix D. Additional References

RFC EDITOR Please remove this section before publication. It exists just to include references to the things in the YANG descriptions which are not otherwise referenced in the text so that xml2rfc will not complain.

[ITU.X690.1994]

Authors' Addresses

Max Pritikin
Cisco

Email: pritikin@cisco.com

Michael C. Richardson
Sandelman Software Works

Email: mcr+ietf@sandelman.ca
URI: http://www.sandelman.ca/

Toerless Eckert
Futurewei Technologies Inc. USA
2330 Central Expy
Santa Clara, 95050
United States of America

Email: tte+ietf@cs.fau.de

Michael H. Behringer

Email: Michael.H.Behringer@gmail.com

Kent Watsen
Watsen Networks

Email: kent+ietf@watsen.net