ANIMA WG   M. Pritikin
Internet-Draft   Cisco
Intended status: Standards Track   M. Richardson
Expires: August 17, 2018   SSW
                                                         M. Behringer

                                                         S. Bjarnason
                                                      Arbor Networks
                                                            K. Watsen
                                                     Juniper Networks
                                                    February 13, 2018

Bootstrapping Remote Secure Key Infrastructures (BRSKI)
draft-ietf-anima-bootstrapping-keyinfra-10

Abstract

   This document specifies automated bootstrapping of a remote secure
   key infrastructure (BRSKI) using vendor installed X.509 certificate,
   in combination with a vendor's authorizing service, both online and
   offline. 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 lower security
   models, including devices with minimal identity, is described for
   legacy reasons but not encouraged. Bootstrapping is complete when
   the cryptographic identity of the new key infrastructure is
   successfully deployed to the device but the established secure
   connection can be used to deploy a locally issued certificate to the
   device as well.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

   BRSKI provides a foundation to securely answer the following
   questions between an element of the network domain called the
   "Registrar" and an unconfigured and untouched device called a
   "Pledge":

   o Registrar authenticating the Pledge: "Who is this device? What is
      its identity?"

   o Registrar authorization the Pledge: "Is it mine? Do I want it?
      What are the chances it has been compromised?"

   o Pledge authenticating the Registrar/Domain: "What is this domain's
      identity?"

   o Pledge authorization the Registrar: "Should I join it?"

   This document details protocols and messages to the endpoints to
   answer the above questions. The Registrar actions derive from Pledge
   identity, third party cloud service communications, and local access
   control lists. The Pledge actions derive from a cryptographically
   protected "voucher" message delivered through the Registrar but
   originating at a Manufacturer Authorized Signing Authority.

   The syntactic details of vouchers are described in detail in
   [I-D.ietf-anima-voucher]. 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).

   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 which 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. Other 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 is pre-provisioned on each new device in a costly and non-
   scalable manner. Existing mechanisms are known as non-secured 'Trust
   on First Use' (TOFU) [RFC7435], 'resurrecting duckling'
   [Stajano99theresurrecting] or 'pre-staging'.

   Another approach is to try and minimize user actions during
   bootstrapping. The enrollment protocol EST [RFC7030] details a set
   of non-autonomic bootstrapping methods in this vein:

   o using the Implicit Trust Anchor database (not an autonomic
      solution because the URL must be securely distributed),

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

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

   o 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 vendor service using a common client-
   server authentication model. After mutual authentication appropriate
   credentials to authenticate the target domain are transfered to the
   Pledge. This creates serveral problems and limitations:

   o the pledge requires realtime connectivity to the vendor service,

   o the domain identity is exposed to the vendor service (this is a
      privacy concern),

   o the vendor 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
[RFC2119].

The following terms are defined for clarity:

domainID: The domain IDentity is the 160-bit SHA-1 hash of the BIT
   STRING of the subjectPublicKey of the root certificate for the
   registrars in the domain. This is consistent with the subject key
   identifier (Section 4.2.1.2 [RFC5280]).

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.

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].

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.

Pledge: The prospective device, which has an identity installed by a
   third-party (e.g., vendor, manufacturer or integrator).

Voucher A signed statement from the MASA service 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 asserted. Multiple voucher types are defined in
   [I-D.ietf-anima-voucher]

Domain: The set of entities that trust a common key infrastructure
   trust anchor. This includes the Proxy, Registrar, Domain
   Certificate Authority, Management components and any existing
   entity that is already a member of the domain.

Domain CA: The domain Certification Authority (CA) provides
   certification functionalities to the domain. At a minimum it
   provides certification functionalities to a Registrar and stores
   the trust anchor that defines the domain. Optionally, it
   certifies all elements.

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". The term JRC is used in common with other bootstrap
   mechanisms.

Join Proxy: A domain entity that helps the pledge join the domain.
   A 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. The pledge is
   unaware that they are communicating with a proxy rather than
   directly with a Registrar.

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.

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 accurately tracking of such
   ownership. Ownership tracking information is indicated in
   vouchers as described in [I-D.ietf-anima-voucher]

IDevID: An Initial Device Identity X.509 certificate installed by
   the vendor on new equipment.

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.

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.

1.3. Scope of solution

   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+) 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 LLNs).

   In many target applications, the systems involved are large router
   platforms with multi-gigabit inter-connections, mounted in controlled
   access data centers. But this solution is not exclusive to the
   large, 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 the 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 it's
   presence through broadcasting.

   This protocol is not intended for low latency handoffs. In networks
   requiring such things, the pledge SHOULD already have been enrolled.

   Specifically, there are protocol aspects described here which 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 which could be inappropriate.

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 consistant 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.

1.4. Leveraging the new key infrastructure / next steps

As a result of the protocol described herein the bootstrapped devices
have a common trust anchor and a certificate has optionally been
issued from a local PKI. This makes it possible to automatically
deploy services across the domain in a secure manner.

Services which benefit from this:

o Device management.

o Routing authentication.

o Service discovery.

The major beneficiary 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]).

[2](#). Architectural Overview

   The logical elements of the bootstrapping framework are described in
   this section. Figure 1 provides a simplified overview of the
   components. Each component is logical and may be combined with other
   components as necessary.

```
                                         +-----------------------+
      +--------------Drop Ship--------------->| Vendor Service        |
      |                                       +-----------------------+
      |                                       | M anufacturer|        |
      |                                       | A uthorized  |Ownership|
      |                                       | S igning     |Tracker  |
      |                                       | A uthority   |        |
      |                                       +--------------+---------+
      |                                              ^
      |                                              |  BRSKI-
      V                                              |  MASA
   +-------+      .........................................|...
   |       |      .                                      | .
   |       |      . +------------+      +-----------+     | .
   |       |      . |            |      |           |     | .
   |Pledge |      . |  Circuit   |      | Domain   <-------+  .
   |       |      . |  Proxy     |      | Registrar |        .
   |         <-------->.............<-------> (PKI RA)   |        .
   |       |      .  |      BRSKI-EST   |           |        .
   |       |      . |            |      +-----+-----+        .
   |IDevID |      . +------------+            | EST RFC7030   .
   |       |      .            +----------------+----------+   .
   |       |      .            | Key Infrastructure        |   .
   |       |      .            | (e.g. PKI Certificate     |   .
   +-------+      .            |       Authority)          |   .
                  .            +---------------------------+   .
                  .                                            .
                  ..................................................
                             "Domain" components
```
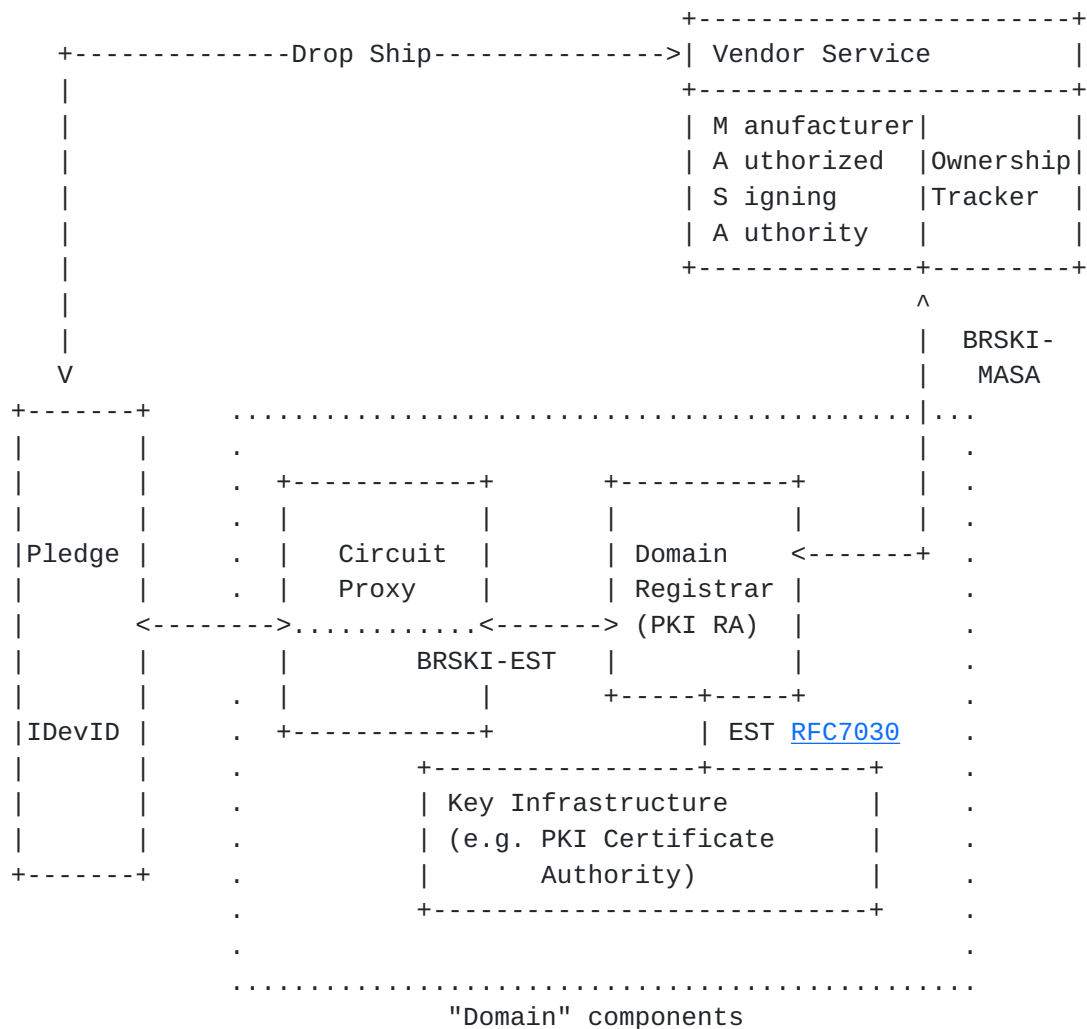
   Figure 1

   We assume a multi-vendor network. In such an environment there could
   be a Vendor Service for each vendor that supports devices following
   this document's specification, or an integrator could provide a
   generic service authorized by multiple vendors. It is unlikely that
   an integrator could provide Ownership Tracking services for multiple
   vendors 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 a domain provides initial
device connectivity sufficient for bootstrapping with a Circuit
Proxy. The Domain Registrar authenticates the Pledge, makes
authorization decisions, and distributes vouchers obtained from the
Vendor Service. Optionally the Registrar also acts as a PKI
Registration 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-------+
                 |  Discover    |
    +----------->              |
    |            +------+-------+
    |                   |
    |            +------v-------+
    |            |  Identity    |
    ^------------+              |
    | rejected   +------+-------+
    |                   |
    |            +------v-------+
    |            | Request      |
    |            | Join         |
    |            +------+-------+
    |                   |
    |            +------v-------+
    |            |  Imprint     |  Optional
    ^------------+              <--+Manual input (Appendix C)
    | Bad Vendor +------+-------+
    | response          |  send Voucher Status Telemetry
    |            +------v-------+
    |            |  Enroll      |
    ^------------+              |
    | Enroll     +------+-------+
    | Failure           |
    |            +------v-------+
    |            |  Enrolled    |
    ^------------+              |
     Factory     +--------------+
     reset
```
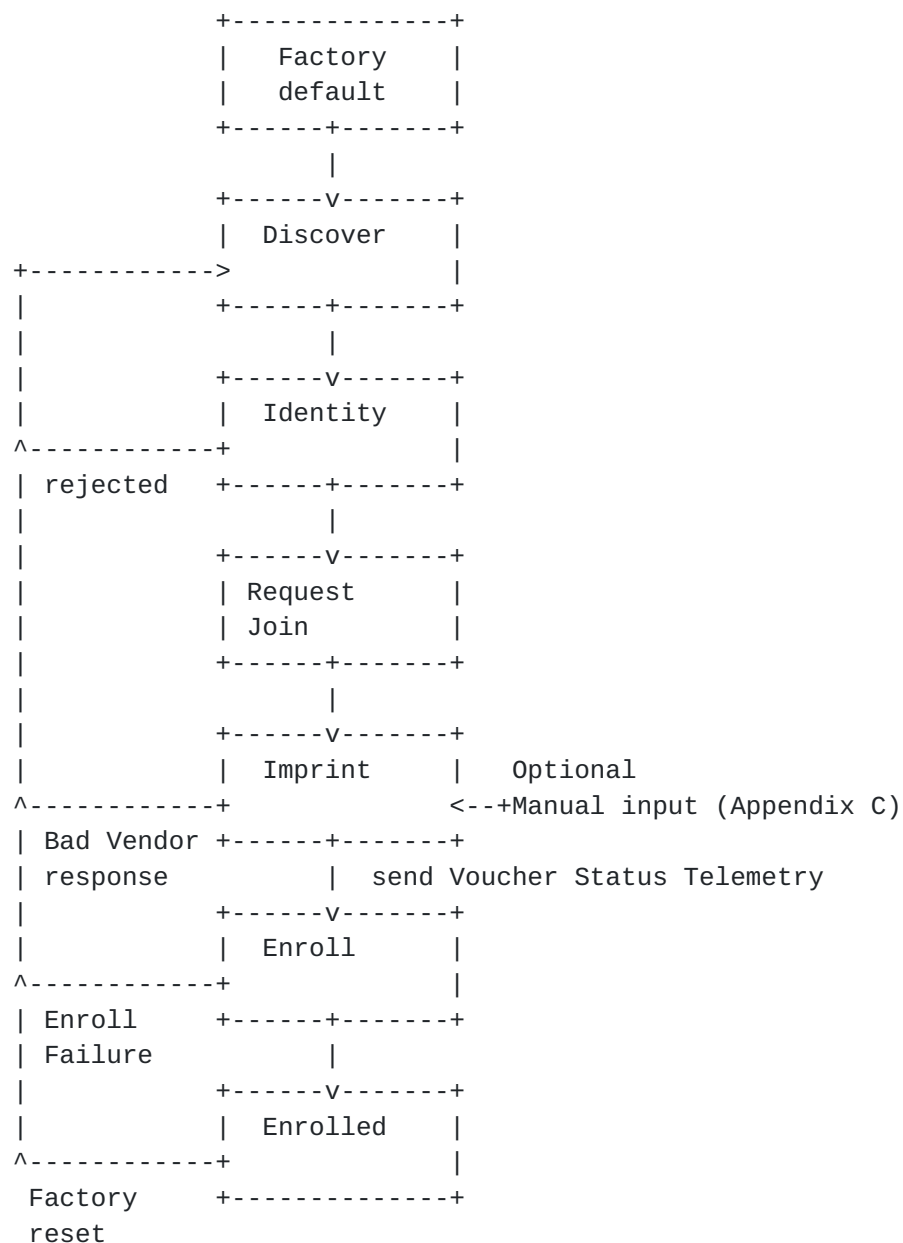
Figure 2

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. Requests to Join the discovered Registrar. A unique nonce can be
    included ensuring that any responses can be associated with this
    particular bootstrapping attempt.

4. Imprint on the Registrar. This requires verification of the
    vendor service provided voucher. A voucher contains sufficient
    information for the Pledge to complete authentication of a
    Registrar. (It enables the Pledge to finish authentication of
    the Registrar TLS server certificate).

5. Enroll. By accepting the domain specific information from a
    Registrar, and by obtaining a domain certificate from a Registrar
    using a standard enrollment protocol, e.g. Enrollment over
    Secure Transport (EST) [RFC7030].

6. 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.

2.2. Secure Imprinting using Vouchers

A voucher is a cryptographically protected statement 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 [I-D.ietf-anima-voucher].

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 server can indiscriminately issue
vouchers. At the highest security levels issuance of vouchers can be
integrated with complex sales channel integrations that are beyond
the scope of this document. 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 vendor 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
between 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 an
   X.509 certificate installed during the manufacturing process. This
   Initial Device Identifier provides a basis for authenticating the
   Pledge during subsequent protocol exchanges and informing the
   Registrar of the MASA URI. There is no requirement for a common root
   PKI hierarchy. Each device vendor can generate their own root
   certificate.

   The following previously defined fields are in the X.509 IDevID
   certificate:

   o The subject field's DN encoding MUST include the "serialNumber"
      attribute with the device's unique serial number.

   o The subject-alt field's encoding SHOULD include a non-critical
      version of the RFC4108 defined HardwareModuleName.

   In order to build the voucher "serial-number" field these IDevID
   fields need to be converted into a serial-number of "type string".
   The following methods is used depending on the first available IDevID
   certificate field (attempted in this order):

   o An RFC4514 String Representation of the Distinguished Name
      "serialNumber" attribute.

   o The HardwareModuleName hwSerialNum OCTET STRING base64 encoded.

   o The RFC4514 String Representation of the Distinguished Name
      "common name" attribute.

   The following newly defined field SHOULD be in the X.509 IDevID
   certificate: An X.509 non-critical certificate extension that
   contains a single Uniform Resource Identifier (URI) that points to an
   on-line Manufacturer Authorized Signing Authority. The URI is
   represented as described in Section 7.4 of [RFC5280].

   Any Internationalized Resource Identifiers (IRIs) MUST be mapped to
   URIs as specified in Section 3.1 of [RFC3987] before they are placed
   in the certificate extension. The URI provides the authority
   information. The BRSKI .well-known tree is described in Section 5

   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>
```

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 3:

```
+--------+          +---------+    +------------+     +------------+
| Pledge |          | Circuit |    | Domain     |     | Vendor     |
|        |          | Proxy   |    | Registrar  |     | Service    |
|        |          |         |    | (JRC)      |     | (MASA)     |
+--------+          +---------+    +------------+     +------------+
```
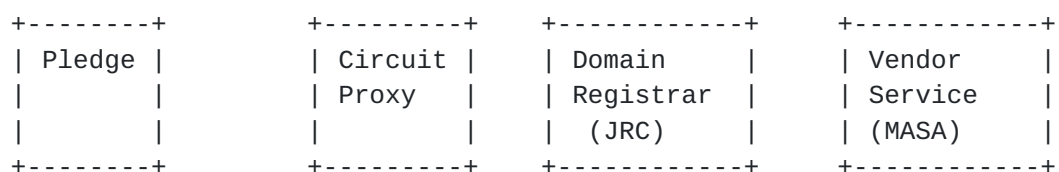
```
   |                    |                    |           Internet |
   |<-RFC4862 IPv6 addr |                    |                    |
   |<-RFC3927 IPv4 addr | Appendix A         |                    |
   |                    |                    |                    |
   |------------------->|                    |                    |
   | optional: mDNS query| Appendix B        |                    |
   | RFC6763/RFC6762    |                    |                    |
   |                    |                    |                    |
   |<-------------------|                    |                    |
   | GRASP M_FLOOD      |                    |                    |
   |    periodic broadcast|                  |                    |
   |                    |                    |                    |
   |------------------->C<----------------->|                    |
   |              TLS via the Circuit Proxy  |                    |
   |<--Registrar TLS server authentication---|                    |
 [PROVISIONAL accept of server cert]         |                    |
   P---X.509 client authentication---------->|                    |
   P                    |                    |                    |
   P---Voucher Request (include nonce)------>|                    |
   P                    |                    |                    |
   P                    |         /--->      |                    |
   P                    |         |          [accept device?]     |
   P                    |         |          [contact Vendor]     |
   P                    |         |              |--Pledge ID-------->|
   P                    |         |              |--Domain ID-------->|
   P                    |         |              |--optional:nonce--->|
   P                    |         |              |     [extract DomainID]
   P                    |         |              |                    |
   P                    |     optional:  |          [update audit log]
   P                    |         |can   |              |            |
   P                    |         |occur |              |            |
   P                    |         |in    |              |            |
   P                    |         |advance|             |            |
   P                    |         |if    |              |            |
   P                    |         |nonceless|           |            |
   P                    |         |              |<- voucher --------|
   P                    |         \---->      |                    |
   P                    |                    |                    |
   P<------voucher--------------------------|                    |
 [verify voucher ]      |                    |                    |
 [verify provisional cert|                   |                    |
   |                    |                    |                    |
   |--------------------------------------->|                    |
   |       [voucher status telemetry]       |<-device audit log--|
   |                    |          [verify audit log and voucher] |
   |                    |                    |                    |
   |<---------------------------------------|                    |
   | Continue with RFC7030 enrollment       |                    |
```

```
| using now bidirectionally authenticated |                      |
| TLS session.           |                      |                      |
|                        |                      |                      |
```

Figure 3

2.4.1. Architectural component: Pledge

The Pledge is the device which is attempting to join. Until the
pledge completes the enrollment process, it has network connectivity
only to the Proxy.

2.4.2. Architectural component: Circuit Proxy

The (Circuit) Proxy provides HTTPS connectivity between the pledge
and the registrar. The proxy mechanism is described in Section 4,
with an optional stateless mechanism described in Appendix C.

2.4.3. Architectural component: Domain Registrar

The Domain Registrar (having the formal name Join Registrar/
Coordinator (JRC)), operates as a CMC Registrar, terminating the EST
and BRSKI connections. The Registrar is manually configured or
distributed with a list of trust anchors necessary to authenticate
any Pledge device expected on the network. The Registrar
communicates with the Vendor supplied MASA to establish ownership.

2.4.4. Architectural component: Vendor Service

The Vendor Service provides two logically seperate functions: the
Manufacturer Authorized Signing Authority (MASA), and an ownership
tracking/auditing function.

2.5. Lack of realtime clock

Many devices when bootstrapping do not have knowledge of the current
time. Mechanisms like Network Time Protocols can not be secured
until bootstrapping is complete. Therefore bootstrapping is defined
in a method that does not require knowledge of the current time.

Unfortunately there are moments during bootstrapping when
certificates are verified, such as during the TLS handshake, where
validity periods are confirmed. This paradoxical "catch-22" is
resolved by the Pledge maintaining a concept of the current "window"
of presumed time validity that is continually refined throughout the
bootstrapping process as follows:

o Initially the Pledge does not know the current time.

o During Pledge authentiation by the Registrar a realtime clock can
   be used by the Registrar. This bullet expands on a closely
   related issue regarding Pledge lifetimes. RFC5280 indicates that
   long lived Pledge certifiates "SHOULD be assigned the
   GeneralizedTime value of 99991231235959Z" [RFC7030] so the
   Registrar MUST support such lifetimes and SHOULD support ignoring
   Pledge lifetimes if they did not follow the RFC5280
   recommendations.

o The Pledge authenticates the voucher presented to it. During this
   authentication the Pledge ignores certificate lifetimes (by
   necessity because it does not have a realtime clock).

o 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).

o Once the voucher is accepted the validity period of the pinned-
   domain-cert in the voucher now serves as a valid time window. Any
   subsequent certificate validity periods checked during RFC5280
   path validation MUST occur within this window.

o When accepting an enrollment certificate the validity period
   within the new certificate is assumed to be valid by the Pledge.
   The Pledge is now willing to use this credential for client
   authentication.

2.6. Cloud Registrar

The Pledge MAY contact a well known URI of a cloud Registrar if a
local Registrar can not 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
an Implicit Trust Anchor database (see [RFC7030]) MUST be used to
authenticate service as described in RFC6125. This is consistent
with the human user configuration of an EST server URI in [RFC7030]
which also depends on RFC6125.

2.7. 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 will normally contain the MASA
URL as detailed in Section 2.3. This is the RECOMMENDED mechanism.

If the Registrar is integrated with [I-D.ietf-opsawg-mud] and the
Pledge IDevID contains the id-pe-mud-url then the Registrar MAY
attempt to obtain the MASA URL from the MUD file. The MUD file
extension for the MASA URL is defined in Appendix D.

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 vendor. Note that
the Registrar can only select the configured MASA URL based on the
trust anchor -- so vendors can only leverage this approach if they
ensure a single MASA URL works for all Pledge's associated with each
trust anchor.

3. Voucher-Request artifact

The Pledge voucher-request is how a Pledge requests a voucher. The
Pledge forms a voucher-request and submits it to the Registrar. The
Registrar in turn submits a voucher-request to the MASA server. To
help differentiate this document refers to "Pledge voucher-request"
and "Registrar voucher-request" when indicating the source is
beneficial. The "proximity-registrar-cert" leaf is used in Pledge
voucher-requests. The "prior-signed-voucher-request" is used in
Registrar voucher-requests that include a Pledge voucher-request.

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

3.1. Tree Diagram

The following tree diagram illustrates a high-level view of a
voucher-request document. The notation used in this diagram is
described in [I-D.ietf-anima-voucher]. Each node in the diagram is
fully described by the YANG module in Section 3.3. 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
```

3.2. Examples

This section provides voucher examples for illustration purposes.
That 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": {
        "nonce": "62a2e7693d82fcda2624de58fb6722e5",
        "created-on": "2017-01-01T00:00:00.000Z",
        "assertion": "proximity",
        "proximity-registrar-cert": "base64encodedvalue=="
    }
}
```

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). See Section 5.4.

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

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 is offline or when
             the Registrar expects to be offline during deployment.
             See Section 5.4.

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

Example (4) The following example illustrates a Registrar voucher-
             request. The 'prior-signed-voucher-request' leaf is not
             populated with the Pledge voucher-request because the
             Pledge did not sign it's own request. This form might
             be used when more constrained Pledges are being
             deployed. The nonce is populated from the Pledge's
             request. See Section 5.4.

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

3.3. YANG Module

   Following is a YANG [RFC7950] module formally extending the
   [I-D.ietf-anima-voucher] voucher into a voucher-request.

```
<CODE BEGINS> file "yang/ietf-voucher-request@2018-02-13.yang"
module ietf-voucher-request {
  yang-version 1.1;

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

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

  import ietf-voucher {
    prefix v;
    description
      "FIXME";
    reference "RFC ????: Voucher Profile for Bootstrapping Protocols";
  }

  organization
   "IETF ANIMA Working Group";

  contact
   "WG Web:   <http://tools.ietf.org/wg/anima/>
    WG List:  <mailto:anima@ietf.org>
    Author:   Kent Watsen
              <mailto:kwatsen@juniper.net>
    Author:   Max Pritikin
              <mailto:pritikin@cisco.com>
    Author:   Michael Richardson
              <mailto:mcr+ietf@sandelman.ca>
    Author:   Toerless Eckert
              <mailto:tte+ietf@cs.fau.de>";

  description
   "This module... FIXME

    The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL NOT',
    'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'MAY', and 'OPTIONAL' in
```

```
   the module text are to be interpreted as described in RFC 2119.

   Copyright (c) 2017 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 "2018-02-13" {
   description
    "Initial version";
   reference
    "RFC XXXX: Voucher Profile for Bootstrapping Protocols";
 }

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

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

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

            The pledge MUST ignore all prior voucher information when
            accepting a voucher for imprinting. Other parties MAY
            examine the prior signed voucher information for the
            purposes of policy decisions. For example this information
            could be useful to a MASA to determine that both pledge and
            registrar agree on proximity assertions. The MASA SHOULD
            remove all prior-signed-voucher 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-T X.690.

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

}
```

<CODE ENDS>

4. Proxy details

The role of the Proxy is to facilitate communications. The Proxy
forwards packets between the Pledge and a Registrar that has been
configured on the Proxy.

The Proxy does not terminate the TLS handshake: it passes streams of bytes onward without examination.

A proxy MAY assume TLS framing for auditing purposes, but MUST NOT assume any TLS version.

A Proxy is always assumed even if it is directly integrated into a Registrar. (In a completely autonomic network, the Registrar MUST provide proxy functionality so that it can be discovered, and the network can grow concentrically around the Registrar)

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

If the Proxy joins an Autonomic Control Plane ([I-D.ietf-anima-autonomic-control-plane]) it SHOULD use Autonomic Control Plane secured GRASP ([I-D.ietf-anima-grasp]) to discovery the Registrar address and port. As part of the discovery process, the proxy mechanism (Circuit Proxy vs IPIP encapsulation) is agreed to between the Registrar and Join Proxy.

For the IPIP encapsulation methods (described in Appendix C), the port announced by the Proxy SHOULD be the same as on the registrar in order for the proxy to remain stateless.

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 but is always assumed to exist.

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. A new temporary
address SHOULD be allocated whenever the discovery process is
forced to restart due to failures. 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.

Each discovery method attempted SHOULD exponentially back-off
attempts (to a maximum of one hour) to avoid overloading the network
infrastructure with discovery. The back-off timer for each method
MUST be independent of other methods.

Methods SHOULD be run in parallel to avoid head of queue problems
wherein an attacker running a fake proxy or registrar can operate
protocol actions intentionally slowly.

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 the device SHOULD return
to listening for GRASP M_FLOOD. It should periodically retry the
vendor specific mechanisms. The Pledge MAY prioritize selection
order as appropriate for the anticipated environment.

4.1.1. Proxy Grasp announcements

A proxy uses the GRASP M_FLOOD mechanism to announce itself. The
pledge SHOULD listen for messages of these form. This announcement
can be within the same message as the ACP announcement detailed in
[I-D.ietf-anima-autonomic-control-plane]. The M_FLOOD is formatted
as follows:

```
[M_FLOOD, 12340815, h'fe80::1', 180000,
          ["AN_Proxy", 4, 1, ""],
          [O_IPv6_LOCATOR,
               h'fe80::1', 'TCP', 4443]]
```

Figure 6b: Proxy Discovery

The formal CDDL 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         = [ O_IPv6_LOCATOR, ipv6-address,
                      transport-proto, port-number ]
ipv6-address     = the v6 LL of the proxy
transport-proto  = IPPROTO_TCP / IPPROTO_UDP / IPPROTO_IPV6
port-number      = selected by proxy
```

Figure 6c: AN_Proxy CDDL

4.2. CoAP connection to Registrar

The use of CoAP to connect from Pledge to Registrar is out of scope
for this document, and may be described in future work.

4.3. HTTPS proxy connection to Registrar

The proxy SHOULD also provide one of: an IPIP encapsulation of HTTP
traffic to the registrar, or a TCP circuit proxy that connects the
Pledge to a Registrar.

When the Proxy provides a circuit proxy to a Registrar the Registrar
MUST accept HTTPS connections.

4.4. Proxy discovery 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 GRASP M_FLOOD messages. The
M_FLOOD is formatted as follows:

```
[M_FLOOD, 12340815, h'fda379a6f6ee0000200000064000001', 180000,
             ["AN_join_registrar", 4, 255, "EST-TLS"],
             [O_IPv6_LOCATOR,
                  h'fda379a6f6ee0000200000064000001', TCP, 80
```

Figure 6: Registrar Discovery

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 7: AN_join_registrar CDDL

The M_FLOOD message MUST be sent periodically. The period is subject
to network administrator policy (EST server configuration). It must
be so low that the aggregate amount of periodic M_FLOODs from all EST
servers causes negligible traffic across the ACP.

The locators are to be interpreted as follows:

```
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]
```

Figure 7: Registrar Response

The set of locators is to be interpreted as follows. A protocol of 6
indicates that TCP proxying on the indicated port is desired. A
protocol of 17 indicates that UDP proxying on the indicated port is

desired. In each case, the traffic SHOULD be proxied to the same
port at the ULA address provided.

A protocol of 41 indicates that packets may be IPIP proxy'ed. In the
case of that IPIP proxying is used, then the provided link-local
address MUST be advertised on the local link using proxy neighbour
discovery. The Join Proxy MAY limit forwarded traffic to the
protocol (6 and 17) and port numbers indicated by locator1 and
locator2. The address to which the IPIP traffic should be sent is
the initiator address (an ACP address of the Registrar), not the
address given in the locator.

Registrars MUST accept TCP / UDP traffic on the ports given at the
ACP address of the Registrar. If the Registrar supports IPIP
ntunnelling, it MUST also accept traffic encapsulated with IPIP.

Registrars MUST accept HTTPS/EST traffic on the TCP ports indicated.
Registrars MAY accept DTLS/CoAP/EST traffic on the UDP in addition to
TCP traffic.

5. Protocol Details

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] 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).

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

BRSKI uses EST message formats for existing operations, uses JSON
[RFC7159] for all new operations defined here, and voucher formats.

While EST section 3.2 does not insist upon use of HTTP 1.1 persistent
connections, BRSKI-EST connections SHOULD use persistent connections.
The intention of this guidance is to ensure the provisional TLS
authentication occurs only once and is properly managed.

Summarized automation extensions for the BRSKI-EST flow are:

o The Pledge provisionally accepts the Registrar certificate during
   the TLS handshake as detailed in Section 5.1.

o If the Registrar responds with a redirection to other web origins
   the Pledge MUST follow only a single redirection. (EST supports
   redirection but does not allow redirections to other web origins
   without user input).

o The Registar 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". The Pledge is RECOMMENDED to
   provide local feed (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. To avoid
   blocking on a single erroneous Registrar the Pledge MUST drop the
   connection after 5 seconds in which there has been no progress on
   the TCP connection. It should proceed to other discovered
   Registrars if there are any. If there were no other Registrars
   discovered, the pledge MAY continue to wait, as long as it is
   concurrently listening for new proxy announcements.

o Ideally the Pledge could keep track of the appropriate retry-after
   value for any number of outstanding Registrars but this would
   involve a large state table on the Pledge. Instead the pledge MAY
   ignore the exact retry-after value in favor of a single hard coded
   value that takes effect between discovery ([[ProxyDiscovery]])
   attempts. A Registrar that is unable to complete the transaction
   the first time due to timing reasons will have future chances.

o The Pledge requests and validates a voucher using the new REST
   calls described below.

o If necessary the Pledge calls the EST defined /cacerts method to
   obtain the domain owners' CA certificate. The pinned-domain-
   certificate element from the voucher should validate this
   certificate, or be identical to it.

o The Pledge completes authentication of the server certificate as
   detailed in Section 5.5.1. This moves the BRSKI-EST TLS
   connection out of the provisional state. Optionally, the BRSKI-
   EST TLS connection can now be used for EST enrollment.

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

o 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. In the language
   of RFC6125 this provides for a SERIALNUM-ID category of identifier
   that can be included in a certificate and therefore that can also
   be used for matching purposes. The SERIALNUM-ID whitelist is
   collated according to vendor trust anchor since serial numbers are
   not globally unique.

o The Registrar requests and validates the Voucher from the vendor
   authorized MASA service.

o The Registrar forwards the Voucher to the Pledge when requested.

o The Registar performs log verifications 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
Registar. 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.

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 a unverified server certificate.

The Pledge maintains a security paranoia concerning the provisional
state, and all data received, until a voucher is received and
verified as specified in Section 5.5.1

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 request media types are:

application/voucher-cms+json The request is a "YANG-defined JSON
   document that has been signed using a CMS structure" as described
   in Section 3 using the JSON encoding described in [RFC7951]. The
   Pledge SHOULD sign the request using the Section 2.3 credential.

application/json The request is the "YANG-defined JSON document" as
   described in Section 3 with exception that it is not within a
   PKCS#7 structure. It is protected only by the TLS client
   authentication. This reduces the cryptographic requirements on
   the Pledge.

For simplicity the term 'voucher-request' is used to refer to either
of these media types. Registrar impementations SHOULD anticipate
future media types but of course will simply fail the request if
those types are not yet known.

The Pledge populates the voucher-request fields as follows:

created-on: Pledges that have a realtime clock are RECOMMENDED to
   populate this field. This provides additional information to the
   MASA.

nonce: The Pledge voucher-request MUST contain a cryptographically
   strong random or pseudo-random number nonce. Doing so ensures
   Section 2.5 functionality. The nonce MUST NOT be reused for
   bootstrapping attempts.

assertion: The Pledge voucher-request MAY contain an assertion of
   "proximity".

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. This
   MUST be populated in a Pledge voucher-request if the "proximity"
   assertion is populated.

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

An example JSON payload of a Pledge voucher-request is in Section 3.2
Example 1.

The Registrar validates the client identity as described in EST
[RFC7030] section 3.3.2. If the request is signed the Registrar
confirms the 'proximity' asserion and associated 'proximity-
registrar-cert' are correct. The registrar performs authorization as
detailed in [[EDNOTE: UNRESOLVED. See Appendix D "Pledge
Authorization"]]. If these validations fail the Registrar SHOULD
respond with an appropriate HTTP error code.

If authorization is successful the Registrar obtains a voucher from
the MASA service (see Section 5.4) and returns that MASA signed
voucher to the pledge as described in Section 5.5.

5.3. 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 2.7
for RFC6125 authentication of the MASA server.

The primary method of Registrar "authentication" by the MASA is
detailed in Section 5.4. As detailed in Section 8 the MASA might
find it necessary to request additional Registrar authentication.
Registrars MUST be prepared to support TLS client certificate
authentication and HTTP Basic or Digest authentication as described
in RFC7030 for EST clients. Implementors are advised that contacting
the MASA is to establish a secured REST 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. 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. For simplicity this
is defined as an optional EST message between a Registrar and an EST
server running on the MASA service although the Registrar is not
required to make use of any other EST functionality when
communicating with the MASA service. (The MASA service MUST properly
reject any EST functionality requests it does not wish to service; a
requirement that holds for any REST interface).

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

The request media type is defined in [I-D.ietf-anima-voucher] and is
application/voucher-cms+json. 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 PKCS#7 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 populates the voucher-request fields as follows:

created-on: Registrars are RECOMMENDED to populate this field. This
   provides additional information to the MASA.

nonce: The optional nonce value from the Pledge request if desired
   (see below).

serial-number: The serial number of the Pledge the Registrar would
   like a voucher for.

idevid-issuer: The idevid-issuer value from the pledge certificate
   is included to ensure a statistically unique identity. The
   Pledge's serial number is extracted from the X.509 IDevID. See
   Section 2.3.

prior-signed-voucher: If a signed Pledge voucher-request was
   received then it SHOULD be included in the Registrar voucher-
   request. (NOTE: what is included is the complete Pledge voucher-
   request, inclusive of the 'assertion', 'proximity-registrar-cert',
   etc wrapped by the pledge's original signature).

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 is expected to be offline when the
Pledge is being deployed. These use cases require the Registrar to
learn the appropriate IDevID SerialNumber field from the physical
device labeling or from the sales channel (out-of-scope of this
document). If a nonceless voucher-reqeust is submitted the MASA
server MUST authenticate the Registrar as described in either EST
[RFC7030] section 3.2, section 3.3, or by validating the Registrar's
certificate used to sign the Registrar voucher-request. Any of these
methods reduce the risk of DDoS attacks and provide an authenticated
identity as an input to sales channel integration and authorizations
(the actual sale-channel integration is also out-of-scope of this
document).

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

Example JSON payloads of Registrar voucher-requests are in
Section 3.2 Example 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 is not know to the MASA server in
advance. The MASA validation checks before issuing a voucher are as
follows:

Renew for expired voucher: As described in [I-D.ietf-anima-voucher]
   vouchers are normally short lived to avoid revocation issues. If

the request is for a previous (expired) voucher using the same
Registrar (as determined by the Registrar pinned-domain-cert) and
the MASA has not been informed that the claim is invalid then the
request for a renewed voucher SHOULD be automatically authorized.

Voucher signature consistency: 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 this to be a
Registrar. Performing this check provides value to domain PKI by
assuring the domain administrator that the MASA service will only
respect claims from authorized Registration Authorities of the
domain. (The requirement for the Registrar to include the Domain
CA certificate in the signature structure was stated above).

Registrar revocation consistency: The MASA SHOULD check for
revocation of the Registrar certificate. The maximum lifetime of
the voucher issued SHOULD NOT exceed the lifetime of the
Registrar's revocation validation (for example if the Registrar
revocation status is indicated in a CRL that is valid for two
weeks then that is an appropriate lifetime for the voucher).
Because the Registar certificate authority is unknown to the MASA
in advance this is only an extended consistency check and is not
required. The maximum lifetime of the voucher issued SHOULD NOT
exceed the lifetime of the Registrar's revocation validation (for
example if the Registrar revocation status is indicated in a CRL
that is valid for two weeks then that is an appropriate lifetime
for the voucher).

Pledge proximity assertion: The MASA server MAY verify that the
Registrar voucher-request includes the 'prior-signed-voucher'
field populated with a Pledge voucher-request that includes a
'proximity-registrar-cert' that is consistent with the certificate
used to sign the Registrar voucher-request. The MASA server is
aware of which Pledge's support signing of their voucher requests
and can use this information to confirm proximity of the Pledge
with the Registrar.

Registar (certificate) authentication: This only occurs if the
Registrar voucher-request is nonceless. As noted above the
details concerning necessary sales-channel integration for the
MASA to authenticate a Registrar certificate is out-of-scope.

The Registrar's certificate chain is extracted from the signature
method and the root certificate is used to populate the "pinned-

domain-cert" of the Voucher being issued. The domainID (e.g. hash of
the root public key) is determined from the pinned-domain-cert and is
used to update the audit log.

5.5. Voucher Response

The voucher response to requests from the Pledge and requests from a
Registrar are in the same format. A Registrar either caches prior
MASA responses or dynamically requests a new Voucher based on local
policy.

If the join operation is successful, the server response MUST contain
an HTTP 200 response code. The server MUST answer with a suitable
4xx or 5xx HTTP [RFC2616] error code when a problem occurs. The
response data from the MASA server MUST be a plaintext human-readable
(ASCII, english) error message containing explanatory information
describing why the request was rejected.

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

A 404 (Not Found) response is appropriate when the request is for a
device which 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: headers, and algorithms used in the signature) can not be
issued, such as because the MASA knows the pledge can not process
that type.

A 415 (Unsupported Media Type) response is approriate for a request
that has a voucher encoding that is not understood.

The response media type is:

application/pkcs7-mime; smime-type=voucher The response is a "YANG-
   defined JSON document that has been signed using a PKCS#7
   structure" as described in [I-D.ietf-anima-voucher] using the JSON
   encoded described in [RFC7951]. The MASA MUST sign the request.

The syntactic details of vouchers are described in detail in
[I-D.ietf-anima-voucher]. For example, the voucher consists of:

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

   The Pledge verifies the signed voucher using the manufacturer
   installed trust anchor associated with the vendor's selected
   Manufacturer Authorized Signing Authority.

   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.

   The Pledge MUST be prepared to parse and fail gracefully from a
   Voucher response that does not contain a 'pinned-domain-cert' field.
   The Pledge MUST be prepared to ignore additional fields it does not
   recognize.

5.5.1. Completing authentication of Provisional TLS connection

   If a Registrar's credentials can not 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 attempting
   with other proxies. Attempts should be attempted in the exponential
   backoff described earlier. Attempts SHOULD be repeated as failure
   may be the result of a temporary inconsistently (an inconsistently
   rolled Registrar key, or some other mis-configuration). The
   inconsistently 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 2.5. Once the
   PKIX path validation is successful the TLS connection is no longer
   provisional.

The pinned-domain-cert is installed as an Explicit Trust Anchor for
future operations. It can therefore 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 EST 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.'

5.6. Voucher 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.

The posted data media type: application/json

The client HTTP POSTs the following to the server at the EST well
known URI /voucher_status. The Status field indicates if the Voucher
was acceptable. If it was not acceptable the Reason string indicates
why. In the failure case this message is being 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 registar expected to continue
joining the domain.

```
{
  "version":"1",
  "Status":FALSE /* TRUE=Success, FALSE=Fail"
  "Reason":"Informative human readable message"
  "reason-context": { additional JSON }
}
```

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.

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."

Additional standard responses MAY be added via Specification
Required.

5.7. MASA authorization log Request

After receiving the voucher status telemetry Section 5.6, the
Registrar SHOULD request the MASA authorization log from the MASA
service using this EST extension. If a device had previously
registered with another domain, a Registrar of that domain would show
in the log.

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

The Registrar MUST HTTP POSTs the same Registrar voucher-request as
it did when requesting a Voucher. It is posted to the
/requestauditlog URI instead. The "idevid-issuer" and "serial-
number" informs the MASA server 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
server implementation MAY leverage internal state to associate this
request with the original, and by now already validated, Registrar
voucher-request so as to avoid an extra crypto validation.

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

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")
RESTful response ([RFC7231] section 7.1) containing a URL to the
prepared (and easily cachable) audit response.

MASA servers that return URLs SHOULD take care to make the returned
URL unguessable. 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, would be easily
enumerable by an attacker. It is recommended put to put some
meaningless randomly generated slug that indexes a database instead.

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

The request media type is:

application/voucher-cms+json The request is a "YANG-defined JSON
   document that has been signed using a CMS structure" as described
   in Section 3 using the JSON encoded described in [RFC7951]. The
   Registrar MUST sign the request. The entire Registrar certificate
   chain, up to and including the Domain CA, MUST be included in the
   CMS structure.

5.7.1. MASA authorization log Response

A log data file is returned consisting of all log entries. For
example:

```
{
  "version":"1",
  "events":[
    {
     "date":"<date/time of the entry>",
     "domainID":"<domainID extracted from voucher-request>",
     "nonce":"<any nonce if supplied (or the exact string 'NULL')>"
    },
    {
     "date":"<date/time of the entry>",
     "domainID":"<domainID extracted from voucher-request>",
     "nonce":"<any nonce if supplied (or the exact string 'NULL')>"
    }
  ],
  "truncation": {
    "nonced duplicates": <number of entries truncated>,
    "nonceless duplicates": <number of entries truncated>,
    "arbitrary": <number of entries trucated>
    }
}
```

A Registrar SHOULD use this log information to make an informed
decision regarding the continued bootstrapping of the Pledge. For
example if the log includes an unexpected domainID then the Pledge
could have imprinted on an unexpected domain. If the log includes
nonceless entries then any registrar in the same domain could
theoretically trigger a reset of the device and take over management
of the Pledge. Equipment that is purchased pre-owned can be expected
to have an extensive history. A Registrar MAY request logs at future
times. A Registrar MAY be configured to ignore the history of the
device but it is RECOMMENDED that this only be configured if hardware
assisted NEA [RFC5209] is supported.

Log entries can be compared against local history logs in search of discrepancies.

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 truncated from the log leaving only the single most recent nonced or nonceless entry. 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 truncated for length. The trunctation method(s) used MUST be indicated in the JSON truncation dictionary using "nonced duplicates", "nonceless duplicates", and "arbitrary" where the number of entries that have been truncation is indicated. 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 truncated ("nonced duplicates" and/or "nonceless duplicates) could still be acceptable for informed decisions. A log that has had "arbitrary" truncations is less acceptable but vendor transparency is better than hidden truncations.

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

5.8. 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 REST calls provide an automated alternative to the manual bootstrapping method described in [RFC7030]. As noted above, use of HTTP 1.1 persistent connections simplifies the Pledge state machine.

The Pledge is also RECOMMENDED to implement the following EST automation extensions. They supplement the RFC7030 EST to better support automated devices that do not have an end user.

Although EST allows clients to obtain multiple certificates by sending multiple CSR requests BRSKI mandates use of the CSR Attributes request and mandates that the Registrar 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. Pledges that require
multiple certificates could establish direct EST connections to the
Registrar.

5.8.1. EST Distribution of CA Certificates

The Pledge MUST 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 5.5.1 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.

5.8.2. EST CSR Attributes

Automated bootstrapping occurs without local administrative
configuration of the Pledge. In some deployments its 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. This approach is beneficial to automated boostrapping
in the widest number of environments.

If the hardwareModuleName in the X.509 IDevID is populated then it
SHOULD by default be propagated to the LDevID along with the
hwSerialNum. The EST server SHOULD support local policy concerning
this functionality.

In networks using the BRSKI enrolled certificate to authenticate the
ACP (Autonomic Control Plane), the EST attributes MUST include the
"ACP information" field. See
[I-D.ietf-anima-autonomic-control-plane] for more details.

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

5.8.3. EST Client Certificate Request

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

5.8.4. Enrollment Status Telemetry

For automated bootstrapping of devices the adminstrative 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, MASA provides logs and status of credential
enrollment. The EST protocol assumes an end user and therefore does
not include a final success indication back to the server. This is
insufficient for automated use cases.

To indicate successful enrollment the client SHOULD re-negotiate the
EST TLS session using the newly obtained credentials. This occurs by
the client initiating a new TLS ClientHello message on the existing
TLS connection. The client MAY simply close the old TLS session and
start a new one. The server MUST support either model.

In the case of a FAIL the Reason string indicates why the most recent
enrollment failed. The SubjectKeyIdentifier field MUST be included
if the enrollment attempt was for a keypair that is locally known to
the client. If EST /serverkeygen was used and failed then the field
is omitted from the status telemetry.

In the case of a SUCCESS the Reason string is omitted. The
SubjectKeyIdentifier is included so that the server can record the
successful certificate distribution.

Status media type: application/json

The client HTTP POSTs the following to the server at the new EST well
known URI /enrollstatus.

```
{
  "version":"1",
  "Status":TRUE /* TRUE=Success, FALSE=Fail"
  "Reason":"Informative human readable message"
  "reason-context": "Additional information"
}
```

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.
This allows for clients that wish to minimize their crypto operations
to simply POST this response without renegotiating the TLS session -
at the cost of the server not being able to accurately verify that
enrollment was truly successful.

5.8.5. 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 defintion 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.vanderstok-ace-coap-est] and that
CoAP mappings for BRSKI will be discussed either there or in future
work.

6. Reduced security operational modes

A common requirement of bootstrapping is to support less secure
operational modes for support specific use cases. The following
sections detail specific ways that the Pledge, Registrar and MASA can
be configured to run in a less secure mode for the indicated reasons.

6.1. Trust Model

```
+--------+        +---------+     +------------+     +------------+
| Pledge |        | Circuit |     | Domain     |     | Vendor     |
|        |        | Proxy   |     | Registrar  |     | Service    |
|        |        |         |     |            |     | (Internet  |
+--------+        +---------+     +------------+     +------------+
```
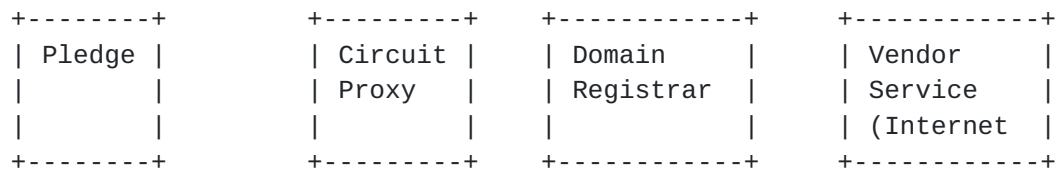
Figure 10

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.

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

Registrar: When interacting with a MASA server a Registrar makes all
   decisions. When Ownership Vouchers are involved a Registrar is
   only a conduit and all security decisions are made on the vendor
   service.

Vendor Service, MASA: This form of vendor 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.
   Current text provides only for a trusted vendor.

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

6.2. Pledge security reductions

The Pledge can choose to accept vouchers using less secure methods.
These methods enable offline and emergency (touch based) deployment
use cases:

1. The Pledge MUST accept nonceless vouchers. This allows for
    offline use cases. Logging and validity periods address the
    inherent security considerations of supporting these use cases.

2. The Pledge MAY support "trust on first use" for physical
    interfaces such as a local console port or physical user
    interface but MUST NOT support "trust on first use" on network
    interfaces. This is because "trust on first use" permanently
    degrades the security for all use cases.

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 vendor service is unavailable. This behavior SHOULD be
    available via local configuration or physical presence methods to
    ensure new entities can always be deployed even when autonomic
    methods fail. This allows for unsecured imprint.

It is RECOMMENDED that "trust on first use" or skipping voucher
validation only be available if hardware assisted Network Endpoint
Assessment [RFC5209] is supported. This recommendation ensures that
domain network monitoring can detect innappropriate use of offline or
emergency deployment procedures.

6.3. Registrar security reductions

A Registrar can choose to accept devices using less secure methods.
These methods are acceptable when low security models are needed, as
the security decisions are being made by the local administrator, but
they MUST NOT be the default behavior:

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 5.4 format to ensure the Pledge's serial number
    information is provided to the Registar (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 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 target network is protected by an air gap and

therefore can not 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 permanent
    Vouchers.

6.4. MASA security reductions

Lower security modes chosen by the MASA service effect all device
deployments unless bound to the specific device identities. In which
case these modes can be provided as additional features for specific
customers. The MASA service can choose to run in less secure modes
by:

1. Not enforcing that a nonce is in the Voucher. This results in
    distribution of Voucher that never expires and in effect makes
    the Domain an always trusted entity to the Pledge during any
    subsequent bootstrapping attempts. That this occurred is
    captured in the log information so that the Registrar can make
    appropriate security decisions when a Pledge joins the Domain.
    This is useful to support use cases where Registrars might not be
    online during actual device deployment. Because this results in
    long lived Voucher and does not require the proof that the device
    is online this is only accepted when the Registrar is
    authenticated by the MASA server and authorized to provide this
    functionality. The MASA server is RECOMMENDED to use this
    functionality only in concert with an enhanced level of ownership
    tracking (out-of-scope). 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.

2. Not verifying ownership before responding with an Voucher. This
    is expected to be a common operational model because doing so
    relieves the vendor 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' information for Pledge's that support that
    functionality. This provides a proof-of-proximity check that
    reduces the need for ownership verification.

7. IANA Considerations

   This document requires the following IANA actions:

7.1. 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. [[EDNOTE: fix names]]

   This document requests a number from the id-pe registry for id-pe-
   masa-url. XXX

7.2. Voucher Status Telemetry

   IANA is requested to create a registry entitled: _Voucher 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 as the reference:

   o version

   o Status

   o Reason

   o reason-context

8. Security Considerations

   There are uses cases where the MASA could be unavailable or
   uncooperative to the Registrar. They include 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 that MASA/vendor behavior might limit the ability
   to re-boostrap 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 repeat bootstrapping
   for their devices.

   The issuance of nonceless vouchers themselves create 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. This reduces the resale value of the equipment
because future owners will detect the lowered security inherent in
the existence of a nonceless voucher that would be trusted by their
Pledge. This reflects a balance between partition resistant recovery
and security of future bootstrapping. Registrars take the Pledge's
audit history into account when applying policy to new devices.

The MASA server is exposed to DoS attacks wherein attackers claim an
unbounded number of devices. Ensuring a Registrar is representative
of a valid vendor 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 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.
This mechanism is optional to allow for constrained devices.

To facilitate logging and administrative oversight in addition to
triggering Registration 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 Registar
but 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.

The MASA authorization log includes a hash of the domainID for each
registrar a voucher has been issued to. This information is closely
related to the actual domain identity, especially when paired with
the anti-DDoS authentication information the MASA might collect.
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
vendor 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.

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 6 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.

Pledge's 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
voucher's 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 Registrar's verifying
log information will see multiple entries and take this into account
for their analytics purposes.

8.1. 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 Registar 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 Registar verifying the audit
logs available from the MASA as described in Section 5.7. Rm might
chose to wait until after the intended Registrar completes the
authorization process before submitting the now-stale Pledge voucher-
request. The Rm would need to remove the Pledge's nonce.

In order to successfully use the resulting "stale voucher" 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:

o Ongoing network monitoring for unexpected bootstrapping attempts
   by Pledges.

o Retreival and examination of MASA log information upon the
   occurance of any such unexpected events. Rm will be listed in the
   logs.

9. Acknowledgements

We would like to thank the various reviewers for their input, in
particular Brian Carpenter, Toerless Eckert, Fuyu Eleven, Eliot Lear,
Sergey Kasatkin, Markus Stenberg, and Peter van der Stok

10. References

10.1. Normative References

[I-D.ietf-anima-autonomic-control-plane]
          Eckert, T., Behringer, M., and S. Bjarnason, "An Autonomic
          Control Plane (ACP)", draft-ietf-anima-autonomic-control-
          plane-13 (work in progress), December 2017.

[I-D.ietf-anima-grasp]
          Bormann, C., Carpenter, B., and B. Liu, "A Generic
          Autonomic Signaling Protocol (GRASP)", draft-ietf-anima-
          grasp-15 (work in progress), July 2017.

[I-D.ietf-anima-voucher]
          Watsen, K., Richardson, M., Pritikin, M., and T. Eckert,
          "Voucher Profile for Bootstrapping Protocols", draft-ietf-
          anima-voucher-07 (work in progress), January 2018.

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

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

[RFC3542] Stevens, W., Thomas, M., Nordmark, E., and T. Jinmei,
          "Advanced Sockets Application Program Interface (API) for
          IPv6", RFC 3542, DOI 10.17487/RFC3542, May 2003,
          <https://www.rfc-editor.org/info/rfc3542>.

[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>.

[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>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security
          (TLS) Protocol Version 1.2", RFC 5246,
          DOI 10.17487/RFC5246, August 2008,
          <https://www.rfc-editor.org/info/rfc5246>.

   [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>.

   [RFC5386] Williams, N. and M. Richardson, "Better-Than-Nothing
              Security: An Unauthenticated Mode of IPsec", RFC 5386,
              DOI 10.17487/RFC5386, November 2008,
              <https://www.rfc-editor.org/info/rfc5386>.

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

   [RFC5660] Williams, N., "IPsec Channels: Connection Latching",
              RFC 5660, DOI 10.17487/RFC5660, October 2009,
              <https://www.rfc-editor.org/info/rfc5660>.

   [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>.

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

   [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>.

   [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>.

10.2. Informative References

   [I-D.behringer-homenet-trust-bootstrap]
              Behringer, M., Pritikin, M., and S. Bjarnason,
              "Bootstrapping Trust on a Homenet", draft-behringer-
              homenet-trust-bootstrap-02 (work in progress), February
              2014.

   [I-D.ietf-netconf-zerotouch]
              Watsen, K., Abrahamsson, M., and I. Farrer, "Zero Touch
              Provisioning for NETCONF or RESTCONF based Management",
              draft-ietf-netconf-zerotouch-19 (work in progress),
              October 2017.

   [I-D.ietf-opsawg-mud]
              Lear, E., Droms, R., and D. Romascanu, "Manufacturer Usage
              Description Specification", draft-ietf-opsawg-mud-15 (work
              in progress), January 2018.

   [I-D.richardson-anima-state-for-joinrouter]
              Richardson, M., "Considerations for stateful vs stateless
              join router in ANIMA bootstrap", draft-richardson-anima-
              state-for-joinrouter-02 (work in progress), January 2018.

   [I-D.vanderstok-ace-coap-est]
              Stok, P., Kampanakis, P., Kumar, S., Richardson, M.,
              Furuhed, M., and S. Raza, "EST over secure CoAP (EST-
              coaps)", draft-vanderstok-ace-coap-est-04 (work in
              progress), January 2018.

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

   [RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in
              IPv6 Specification", RFC 2473, DOI 10.17487/RFC2473,
              December 1998, <https://www.rfc-editor.org/info/rfc2473>.

   [RFC7217] Gont, F., "A Method for Generating Semantically Opaque
              Interface Identifiers with IPv6 Stateless Address
              Autoconfiguration (SLAAC)", RFC 7217,
              DOI 10.17487/RFC7217, April 2014,
              <https://www.rfc-editor.org/info/rfc7217>.

   [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>.

   [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>.

   [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>.

Appendix A. IPv4 operations

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 Local-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 Plege 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

   The Pledge MAY perform DNS-based Service Discovery [RFC6763] over
   Multicast DNS [RFC6762] searching for the service
   "_bootstrapks._tcp.local.".

   To prevent unaccceptable levels of network traffic 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.

Performs DNS-based Service Discovery [RFC6763] over normal DNS
operations. The service searched for is
"_bootstrapks._tcp.example.com". In this case the domain
"example.com" is discovered as described in [RFC6763] section 11.
This method is only available if the host has received a useable IPv4
address via DHCPv4 as suggested in Appendix A.

If no local bootstrapks service is located using the GRASP
mechanisms, or the above mentioned DNS-based Service Discovery
methods the Pledge MAY contact a well known vendor provided
bootstrapping server by performing a DNS lookup using a well known
URI such as "bootstrapks.vendor-example.com". The details of the URI
are vendor specific. Vendors that leverage this method on the Pledge
are responsible for providing the bootstrapks service.

The current DNS services returned during each query is 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. IPIP Join Proxy mechanism

The Circuit Proxy mechanism suffers from requiring a state on the
Join Proxy for each connection that is relayed. The Circuit Proxy
can be considered a kind of Algorithm Gateway [FIND-good-REF].

An alternative to proxying at the TCP layer is to selectively forward
at the IP layer. This moves all per-connection to the Join
Registrar. The IPIP tunnel statelessly forwards packets. This
section provides some explanation of some of the details of the
Registrar discovery procotol which are not important to Circuit
Proxy, and some implementation advice.

The IPIP tunnel is described in [RFC2473]. Each such tunnel is
considered a unidirectional construct, but two tunnels may be
associated to form a bidirectional mechanism. An IPIP tunnel is
setup as follows. The outer addresses are an ACP address of the Join
Proxy, and the ACP address of the Join Registrar. The inner
addresses seen in the tunnel are the link-local addresses of the
network on which the join activity is occuring.

One way to look at this construct is to consider that the Registrar
is extending attaching an interface to the network on which the Join
Proxy is physically present. The Registrar then interacts as if it
were present on that network using link-local (fe80::) addresses.

The Join node is unaware that the traffic is being proxied through a
tunnel, and does not need any special routing.

There are a number of considerations with this mechanism which
require cause some minor amounts of complexity. Note that due to the
tunnels, the Registrar sees multiple connections to a fe80::/10
network on not just physical interfaces, but on each of the virtual
interfaces represending the tunnels.

C.1. Multiple Join networks on the Join Proxy side

The Join Proxy will in the general case be a routing device with
multiple interfaces. Even a device as simple as a wifi access point
may have wired, and multiple frequencies of wireless interfaces,
potentially with multiple ESSIDs.

Each of these interfaces on the Join Proxy may be seperate L3 routing
domains, and therefore will have a unique set of link-local
addresses. An IPIP packet being returned by the Registrar needs to
be forwarded to the correct interface, so the Join Proxy needs an
additional key to distinguish which network the packet should be
returned to.

The simplest way to get this additional key is to allocate an
additional ACP address; one address for each network on which join
traffic is occuring. The Join Proxy SHOULD do a GRASP M_NEG_SYN for
each interface which they wish to relay traffic, as this allows the
Registrar to do any static tunnel configuration that may be required.

C.2. Automatic configuration of tunnels on Registrar

The Join Proxy is expected to do a GRASP negotiation with the proxy
for each Join Interface that it needs to relay traffic from. This is
to permit Registrars to configure the appropriate virtual interfaces
before join traffic arrives.

A Registrar serving a large number of interfaces may not wish to
allocate resources to every interface at all times, but can instead
dynamically allocate interfaces. It can do this by monitoring IPIP
traffic that arrives on it's ACP interface, and when packets arrive
from new Join Proxys, it can dynamically configure virtual
interfaces.

A more sophisticated Registrar willing to modify the behaviour of
it's TCP and UDP stack could note the IPIP traffic origination in the
socket control block and make information available to the TCP layer
(for HTTPS connections), or to the application (for CoAP connections)
via a proprietary extension to the socket API.

C.3. **Proxy Neighbor Discovery by Join Proxy**

   The Join Proxy MUST answer neighbor discovery messages for the
   address given by the Registrar as being it's link-local address. The
   Join Proxy must also advertise this address as the address to which
   to connect to when advertising it's existence.

   This proxy neighbor discovery means that the pledge will create TCP
   and UDP connections to the correct Registrar address. This matters
   as the TCP and UDP pseudo-header checksum includes the destination
   address, and for the proxy to remain completely stateless, it must
   not be necessary for the checksum to be updated.

C.4. **Use of connected sockets; or IP_PKTINFO for CoAP on Registrar**

   TCP connections on the registrar SHOULD properly capture the ifindex
   of the incoming connection into the socket structure. This is normal
   IPv6 socket API processing. The outgoing responses will go out on
   the same (virtual) interface by ifindex.

   When using UDP sockets with CoAP, the application will have to pay
   attention to the incoming ifindex on the socket. Access to this
   information is available using the IP_PKTINFO auxiliary extension
   which is a standard part of the IPv6 sockets API.

   A registrar application could, after receipt of an initial CoAP
   message from the Pledge, create a connected UDP socket (including the
   ifindex information). The kernel would then take care of accurate
   demultiplexing upon receive, and subsequent transmission to the
   correct interface.

C.5. **Use of socket extension rather than virtual interface**

   Some operating systems on which a Registrar need be implemented may
   find need for a virtual interface per Join Proxy to be problematic.
   There are other mechanism which can make be done.

   If the IPIP decapsulator can mark the (SYN) packet inside the kernel
   with the address of the Join Proxy sending the traffic, then an
   interface per Join Proxy may not be needed. The outgoing path need
   just pay attention to this extra information and add an appropriate
   IPIP header on outgoing. A CoAP over UDP mechanism may need to
   expose this extra information to the application as the UDP sockets
   are often not connected, and the application will need to specify the
   outgoing path on each packet send.

   Such an additional socket mechanism has not been standardized.
   Terminating L2TP connections over IPsec transport mode suffers from
   the same challenges.

Appendix D. MUD Extension

   The following extension augments the MUD model to include a single
   node, as described in [I-D.ietf-opsawg-mud] section 3.6, using the
   following sample module that has the following tree structure:

   module: ietf-mud-brski-masa
   augment /ietf-mud:mud:
   +--rw masa-server?   inet:uri

   The model is defined as follows:

```
<CODE BEGINS>
module ietf-mud-brski-masa {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-mud-brski-masa";
  prefix ietf-mud-brski-masa;
  import ietf-mud {
    prefix ietf-mud;
  }
  import ietf-inet-types {
    prefix inet;
  }

  organization
    "IETF ANIMA (Autonomic Networking Integrated Model and
    Approach) Working Group";
    contact
    "WG Web: http://tools.ietf.org/wg/anima/
    WG List: anima@ietf.org
    ";
  description
    "BRSKI extension to a MUD file to indicate the
    MASA URL.";

  revision 2017-10-09 {
    description
    "Initial revision.";
    reference
    "RFC XXXX: Manufacturer Usage Description
    Specification";
  }

  augment "/ietf-mud:mud" {
    description
    "BRSKI extension to a MUD file to indicate the
    MASA URL.";
    leaf masa-server {
      type inet:uri;
      description
      "This value is the URI of the MASA server";
    }
  }
}
<CODE ENDS>
```

[Appendix E](#). **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.

[E.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.

[E.1.1](#). **MASA key pair for voucher signatures**

This private key signs vouchers:

```
-----BEGIN EC PRIVATE KEY-----
MIGkAgEBBDAgiRoYqKoEcfOfvRvmZ5P5Azn58tuI7nSnIy7OgFnCeiNo+BmbgMho
r6lcU60gwVagBwYFK4EEACKhZANiAATZAH3Rb2FvIJOnts+vXuWW35ofyNbCHzjA
zOi2kWZFE1ByurKImNcNMFGirGnRXIXGqWCfw5ICgJ8CuM3vV5ty9bf7KUlOkejz
Tvv+5PV++elkP9HQ83vqTAws2WwWTxI=
-----END EC PRIVATE KEY-----
```

This public key validates vouchers:

```
-----BEGIN CERTIFICATE-----
MIIBzzCCAVagAwIBAgIBATAKBggqhkjOPQQDAjBNMRIwEAYKCZImiZPyLGQBGRYC
Y2ExGTAXBgoJkiaJk/IsZAEZFglzYW5kZWxtYW4xHDAaBgNVBAMME1Vuc3RydW5n
IEhpZ2h3YXkgQ0EwHhcNMTcwMzI2MTYxOTQwWhcNMTkwMzI2MTYxOTQwWjBHMRIw
EAYKCZImiZPyLGQBGRYCY2ExGTAXBgoJkiaJk/IsZAEZFglzYW5kZWxtYW4xFjAU
BgNVBAMMDVVuc3RydW5nIE1BU0EwdjAQBgcqhkjOPQIBBgUrgQQAIgNiAATZAH3R
b2FvIJOnts+vXuWW35ofyNbCHzjAzOi2kWZFE1ByurKImNcNMFGirGnRXIXGqWCf
w5ICgJ8CuM3vV5ty9bf7KUlOkejzTvv+5PV++elkP9HQ83vqTAws2WwWTxKjEDAO
MAwGA1UdEwEB/wQCMAAwCgYIKoZIzj0EAwIDZwAwZAIwGb0oyM0doP6t3/LSPL5O
DuatEwMYh7WGO+IYTHC8K7EyHBOmCYReKT2+GhV/CLWzAjBNy6UMJTt1tsxJsJqd
MPUIFj+4wZg1AOIb/JoA6M7r33pwLQTrHRxEzVMGfWOkYUw=
-----END CERTIFICATE-----
```

[E.1.2](#). **Manufacturer key pair for IDevID signatures**

This private key signs IDevID certificates:

     -----BEGIN EC PRIVATE KEY-----
     MIGkAgEBBDAgiRoYqKoEcfOfvRvmZ5P5Azn58tuI7nSnIy7OgFnCeiNo+BmbgMho
     r6lcU60gwVagBwYFK4EEACKhZANiAATZAH3Rb2FvIJOnts+vXuWW35ofyNbCHzjA
     zOi2kWZFE1ByurKImNcNMFGirGnRXIXGqWCfw5ICgJ8CuM3vV5ty9bf7KUlOkejz
     Tvv+5PV++elkP9HQ83vqTAws2WwWTxI=
     -----END EC PRIVATE KEY-----

     This public key validates IDevID certificates:

     -----BEGIN CERTIFICATE-----
     MIIBzzCCAVagAwIBAgIBATAKBggqhkjOPQQDAjBNMRIwEAYKCZImiZPyLGQBGRYC
     Y2ExGTAXBgoJkiaJk/IsZAEZFglzYW5kZWxtYW4xHDAaBgNVBAMME1Vuc3RydW5n
     IEhpZ2h3YXkgQ0EwHhcNMTcwMzI2MTYxOTQwWhcNMTkwMzI2MTYxOTQwWjBHMRIw
     EAYKCZImiZPyLGQBGRYCY2ExGTAXBgoJkiaJk/IsZAEZFglzYW5kZWxtYW4xFjAU
     BgNVBAMMDVVuc3RydW5nIE1BU0EwdjAQBgcqhkjOPQIBBgUrgQQAIgNiAATZAH3R
     b2FvIJOnts+vXuWW35ofyNbCHzjAzOi2kWZFE1ByurKImNcNMFGirGnRXIXGqWCf
     w5ICgJ8CuM3vV5ty9bf7KUlOkejzTvv+5PV++elkP9HQ83vqTAws2WwWTxKjEDAO
     MAwGA1UdEwEB/wQCMAAwCgYIKoZIzj0EAwIDZwAwZAIwGb0oyM0doP6t3/LSPL5O
     DuatEwMYh7WGO+IYTHC8K7EyHBOmCYReKT2+GhV/CLWzAjBNy6UMJTt1tsxJsJqd
     MPUIFj+4wZg1AOIb/JoA6M7r33pwLQTrHRxEzVMGfWOkYUw=
     -----END CERTIFICATE-----

[E.1.3](#). Registrar key pair

     The registrar key (or chain) is the representative of the domain
     owner. This key signs Registrar voucher-requests:

     -----BEGIN EC PRIVATE KEY-----
     MHcCAQEEIF+obiToYYYeMifPsZvrjWJ0yFsCJwIFhpokmT/TULmXoAoGCCqGSM49
     AwEHoUQDQgAENWQOzcNMUjP0NrtfeBc0DJLWfeMGgCFdIv6FUz4DifM1ujMBec/g
     6W/P6boTmyTGdFOh/8HwKUerL5bpneK8sg==
     -----END EC PRIVATE KEY-----

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

     -----BEGIN CERTIFICATE-----
     MIIBrjCCATOgAwIBAgIBAzAKBggqhkjOPQQDAzBOMRIwEAYKCZImiZPyLGQBGRYC
     Y2ExGTAXBgoJkiaJk/IsZAEZFglzYW5kZWxtYW4xHTAbBgNVBAMMFFVuc3RydW5n
     IEZvdW50YWluIENBMB4XDTE3MDkwNTAxMTI0NVoXDTE5MDkwNTAxMTI0NVowQzES
     MBAGCgmSJomT8ixkARkWAmNhMRkwFwYKCZImiZPyLGQBGRYJc2FuZGVsbWFuMRIw
     EAYDVQQDDAlsb2NhbGhvc3QwWTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAAQ1ZA7N
     w0xSM/Q2u194FzQMktZ94waAIV0i/oVTPgOJ8zW6MwF5z+Dpb8/puhObJMZ0U6H/
     wfApR6svlumd4ryyow0wCzAJBgNVHRMEAjAAMAoGCCqGSM49BAMDA2kAMGYCMQC3
     /iTQJ3evYYcgbXhbmzrp64t3QC6qjIeY2jkDx062nuNifVKtyaara3F30AIkKSEC
     MQDi29efbTLbdtDk3tecY/rD7V77XaJ6nYCmdDCR54TrSFNLgxvt1lyFM+0fYpYR
     c3o=
     -----END CERTIFICATE-----

   The registrar public certificate as decoded by openssl's x509
   utility. Note that the registrar certificate is marked with the
   cmcRA extension.

   Certificate:
       Data:
           Version: 3 (0x2)
           Serial Number: 3 (0x3)
       Signature Algorithm: ecdsa-with-SHA384
           Issuer: DC = ca, DC = sandelman, CN = Unstrung Fount
   ain CA
           Validity
               Not Before: Sep 5 01:12:45 2017 GMT
               Not After : Sep 5 01:12:45 2019 GMT
           Subject: DC = ca, DC = sandelman, CN = localhost
           Subject Public Key Info:
               Public Key Algorithm: id-ecPublicKey
                   Public-Key: (256 bit)
                   pub:
                       04:35:64:0e:cd:c3:4c:52:33:f4:36:bb:5f:7
   8:17:
                       34:0c:92:d6:7d:e3:06:80:21:5d:22:fe:85:5
   3:3e:
                       03:89:f3:35:ba:33:01:79:cf:e0:e9:6f:cf:e
   9:ba:
                       13:9b:24:c6:74:53:a1:ff:c1:f0:29:47:ab:2
   f:96:
                       e9:9d:e2:bc:b2
                   ASN1 OID: prime256v1
                   NIST CURVE: P-256
           X509v3 extensions:
               X509v3 Basic Constraints:
                   CA:FALSE
       Signature Algorithm: ecdsa-with-SHA384
            30:66:02:31:00:b7:fe:24:d0:27:77:af:61:87:20:6d:78:
   5b:
            9b:3a:e9:eb:8b:77:40:2e:aa:8c:87:98:da:39:03:c7:4e:
   b6:
            9e:e3:62:7d:52:ad:c9:a6:ab:6b:71:77:d0:02:24:29:21:
   02:
            31:00:e2:db:d7:9f:6d:32:db:76:d0:e4:de:d7:9c:63:fa:
   c3:
            ed:5e:fb:5d:a2:7a:9d:80:a6:74:30:91:e7:84:eb:48:53:
   4b:
            83:1b:ed:d6:5c:85:33:ed:1f:62:96:11:73:7a

E.1.4. **Pledge key pair**

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

   -----BEGIN EC PRIVATE KEY-----
   MHcCAQEEIL+ue8PQcN+M7LFBGPsompYwobI/rsoHnTb2a+0hO+8joAoGCCqGSM49
   AwEHoUQDQgAEumBVaDlX87WyME8CJToyt9NWy6sYw0DTbjjJIn79pgr7ALa//Y8p
   r70WpK1SIaiUeeFw7e+lCzTp1Z+wJu14Bg==
   -----END EC PRIVATE KEY-----

   The public key is used by the registrar to find the MASA. The MASA
   URL is in an extension described in Section 2.3. RFC-EDITOR: Note
   that these certificates are using a Private Enterprise Number for the
   not-yet-assigned by IANA MASA URL, and need to be replaced before
   AUTH48.

   -----BEGIN CERTIFICATE-----
   MIICMjCCAbegAwIBAgIBDDAKBggqhkjOPQQDAjBNMRIwEAYKCZImiZPyLGQBGRYC
   Y2ExGTAXBgoJkiaJk/IsZAEZFglzYW5kZWxtYW4xHDAaBgNVBAMME1Vuc3RydW5n
   IEhpZ2h3YXkgQ0EwIBcNMTcxMDEyMTM1MjUyWhgPMjk5OTEyMzEwMDAwMDBaMEsx
   EjAQBgoJkiaJk/IsZAEZFgJjYTEZMBcGCgmSJomT8ixkARkWCXNhbmRlbG1hbjEa
   MBgGA1UEAwwRMDAtRDAtRTUtRjItMDAtMDIwWTATBgcqhkjOPQIBBggqhkjOPQMB
   BwNCAARJp5i0dU1aUnR2u8wMRwgkNupNbNM7m1n0mj+0KJZjcPIqID+trPjTSobt
   uIdpRPfGZ8hU/nIUveqwyoYI8BPbo4GHMIGEMB0GA1UdDgQWBBQdMRZhthFQmzz6
   E7YVXzkL7XZDKjAJBgNVHRMEAjAAMCsGA1UdEQQkMCKgIAYJKwYBBAGC7lIBoBMM
   ETAwLUQwLUU1LUYyLTAwLTAyMCsGCSsGAQQBgu5SAgQeDBxodHRwczovL2hpZ2h3
   YXkuc2FuZGVsbWFuLmNhMAoGCCqGSM49BAMCA2kAMGYCMQDhJ1N+eanW1U/e5qoM
   SGvUvWHR7uic8cJbh7vXy580nBs8bpNn60k/+IzvEUetMzICMQCr1uxvdYeKq7mb
   RXCR4ZCJsw67fJ7jyXZbCUSir+3wBT2+lWggzPDRgYB5ABb7sAw=
   -----END CERTIFICATE-----

   The pledge public certificate as decoded by openssl's x509 utility so
   that the extensions can be seen. A second custom Extension is
   included to provided to contain the EUI48/EUI64 that the pledge will
   configure.

```
   Certificate:
       Data:
           Version: 3 (0x2)
           Serial Number: 12 (0xc)
       Signature Algorithm: ecdsa-with-SHA256
           Issuer: DC = ca, DC = sandelman, CN = Unstrung Highw
   ay CA
           Validity
               Not Before: Oct 12 13:52:52 2017 GMT
               Not After : Dec 31 00:00:00 2999 GMT
           Subject: DC = ca, DC = sandelman, CN = 00-D0-E5-F2-0
   0-02
```

```
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    04:49:a7:98:b4:75:4d:5a:52:74:76:bb:cc:0
  c:47:
                    08:24:36:ea:4d:6c:d3:3b:9b:59:f4:9a:3f:b
  4:28:
                    96:63:70:f2:2a:20:3f:ad:ac:f8:d3:4a:86:e
  d:b8:
                    87:69:44:f7:c6:67:c8:54:fe:72:14:bd:ea:b
  0:ca:
                    86:08:f0:13:db
                ASN1 OID: prime256v1
                NIST CURVE: P-256
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                1D:31:16:61:B6:11:50:9B:3C:FA:13:B6:15:5F:39
  :0B:ED:76:43:2A
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Subject Alternative Name:
                othername:<unsupported>
            1.3.6.1.4.1.46930.2:
                ..https://highway.sandelman.ca
    Signature Algorithm: ecdsa-with-SHA256
         30:66:02:31:00:e1:27:53:7e:79:a9:d6:d5:4f:de:e6:aa:
  0c:
         48:6b:d4:bd:61:d1:ee:e8:9c:f1:c2:5b:87:bb:d7:cb:9f:
  34:
         9c:1b:3c:6e:93:67:eb:49:3f:f8:8c:ef:11:47:ad:33:32:
  02:
         31:00:ab:d6:ec:6f:75:87:8a:ab:b9:9b:45:70:91:e1:90:
  89:
         b3:0e:bb:7c:9e:e3:c9:76:5b:09:44:a2:af:ed:f0:05:3d:
  be:
         95:68:20:cc:f0:d1:81:80:79:00:16:fb:b0:0c
```

E.2. Example process

   RFC-EDITOR: these examples will need to be replaced with CMS versions
   once IANA has assigned the eContentType in [I-D.ietf-anima-voucher].

E.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.

MIIHHAYJKoZIhvcNAQcCoIIHDTCCBwkCAQExDzANBglghkgBZQMEAgEFADCC
Aw4GCSqGSIb3DQEHAaCCAv8EggL7eyJpZXRmLXZvdWNoZXItcmVxdWVzdDp2
b3VjaGVyIjp7ImFzc2VydGlvbiI6InByb3hpbWl0eSIsImNyZWF0ZWQtb24i
OiIyMDE3LTA5LTAxIiwic2VyaWFsLW51bWJlciI6IjAwLUQwLUU1LUYyLTAw
LTAyIiwibm9uY2UiOiJEc3M5OXNCcjNwTk1PQUNlLUxZWTd3IiwicHJveGlt
aXR5LXJlZ2lzdHJhci1jZXJ0IjoiTUlJQnJqQ0NBVE9nQXdJQkFnSUJekFL
QmdncWhrak9QUVFEQXpCT01SSXdFQVlLQ1pJbWlaUHlMR1FCR1JZQ1kyRXhH
VEFYQmdvSmtppYUprL0lzWkFFWkZnbHpaVzVVcWld4dFlXNHhIVEFiQmdOVkJB
TU1GRlZ1YzNSeWRXXW5JRVp2ZFc1MFhbHVJRU9CTUI0WERURTNNNGt3TlRRB
eE1UB0Vm9YRFRFNU1Ea3dOVEF4TVRJME5Wb3dRekVTVTUJBR0NnbVNKb21U
OGl4a0FSa1dBU1oyTVJrd0Z3WUtDWkltaVpQeUxHUUJHUllKY2JGdVpHHVnNi
V0Z1TVJJJd0VBWURWUVFEREFsc2IyTmhiR2h2YzNRd1dUQVRCZ2NxaGtqT1BR
SUJCZ2dxaGtqT1BRTUJCd05EQUFRMVpBBN053MHhTTS9RMnUxOTRGGelFNa3Ra
OTR3YUFJVjBpL29WVFBnT0o4elc2TXdGNXorRHBiOC9wdWhPYkppWjBVNkgv
d2ZBcFI2c3ZsdW1kNHJ5eW93MHddDekFKQmdOVkhTVUVBakFBUFvR0NDcUdT
TTQ5QkFNREEya0FNR1lDTVFDMy9pVFFFFK2V2WVljZ2JYaGJtenJwNjR0M1FD
NnFqSWVZZMmprRHgwNjJudU5ppZlZLdHlhYXJhM0YzMEFJa0tTRUNNUURpMjll
ZmJJUTGJkdERrM3RlY1kvckQ3Vjc3WGFKNm5ZQ21kRENSNTRUclNGTkxxneHZ0
MWx5Rk0rMGZZcFlSYzNvPSJ9faCCAjYwggIyMIIBt6ADAgECAgEMMAoGCCqG
SM49BAMCME0xEjAQBgoJkiaJk/IsZAEZFgJjYTEZMBcGCgmSJomT8ixkARkW
CXNhbmRlbG1hbjEcMBoGA1UEAwwTVW5zdHJ1bmcgSGlnaHdheSBDQTAgFw0x
NzEwMTIxMzUyNTJaGA8yOTk5MTIzMTAwMDAwMFowSzESMBAGCgmSJomT8ixk
ARkWAmNhMRkwFwYKCZImiZPyLGQBGRYJc2FuZGVsbWFuMRowGAYDVQQDDBEw
MC1EMC1FNS1GMi0wMC0wMjBZMBMGByqGSM49AgEGCCqGSM49AwEHA0IABEmn
mLR1TVpSdHa7zAxHCCQ26k1s0zubWfSaP7QolmNw8iogP62s+NNKhu24h2lE
98ZnyFT+chS96rDKhgjwE9ujgYcwgYQwHQYDVR0OBBYEFB0xFmG2EVCbPPoT
thVfOQvtdkMqMAkGA1UdEwQCMAAwKwYDVR0RBCQwIqAgBgkrBgEEAYLuUgGg
EwwRMDAtRDAtRTUtRjItMDAtMDIwKwYJKwYBBAGC7lICBB4MHGh0dHBzOi8v
aGlnaHdheS5zYW5kZWxtYW4uY2EwCgYIKoZIzj0EAwIDaQAwZgIxAOEnU355
qdbVT97mqgxIa9S9YdHu6JzxwluHu9fLnzScGzxuk2frST/4jO8RR60zMgIx
AKvW7G91h4qruZtFcJHhkImzDrt8nuPJdlsJRKKv7fAFPb6VaCDM8NGBgHkA
FvuwDDGCAaUwggGhAgEBMFIwTTESMBAGCgmSJomT8ixkARkWAmNhMRkwFwYK
CZImiZPyLGQBGRYJc2FuZGVsbWFuMRwwGgYDVQQDDBNVbnN0cnVuZyBIaWdo
d2F5IENBAgEMMA0GCWCGSAFlAwQCAQUAoIHkMBgGCSqGSIb3DQEJAzELBgkq
hkiG9w0BBwEwHAYJKoZIhvcNAQkFMQ8XDTE3MTAxMjE3NTQzMFowLwYJKoZI
hvcNAQkEMSIEIP59cuKVAPkKOOlQIaIV/W1AsWKbmVmBd9wFSuD5yLafMHkG
CSqGSIb3DQEJDzFsMGowCwYJYIZIAWUDBAEqMAsGCWCGSAFlAwQBFjALBglg
hkgBZQMEAQIwCgYIKoZIhvcNAwcwDgYIKoZIhvcNAwICAgCAMA0GCCqGSIb3
DQMCAgFAMAcGBSsOAwIHMA0GCCqGSIb3DQMCAgEoMAoGCCqGSM49BAMCBEYw
RAIgYUy0NTdP+xTkm/Et69eI++S/2z3dQwPKOwdL0cDCSvACIAh3jJbybMnK
cf7DKKnsn2G/O06HeB/8imMI+hnA7CfN

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

The ASN1 decoding of the artifact:

```
    0:d=0  hl=4 l=1820 cons: SEQUENCE
    4:d=1  hl=2 l=   9 prim: OBJECT              :pkcs7-signed
Data
   15:d=1  hl=4 l=1805 cons: cont [ 0 ]
   19:d=2  hl=4 l=1801 cons: SEQUENCE
   23:d=3  hl=2 l=   1 prim: INTEGER            :01
   26:d=3  hl=2 l=  15 cons: SET
   28:d=4  hl=2 l=  13 cons: SEQUENCE
   30:d=5  hl=2 l=   9 prim: OBJECT             :sha256
   41:d=5  hl=2 l=   0 prim: NULL
   43:d=3  hl=4 l= 782 cons: SEQUENCE
   47:d=4  hl=2 l=   9 prim: OBJECT             :pkcs7-data
   58:d=4  hl=4 l= 767 cons: cont [ 0 ]
   62:d=5  hl=4 l= 763 prim: OCTET STRING       :{"ietf-vouch
er-request:voucher":{"assertion":"proximity","created-on":"2
017-09-01","serial-number":"00-D0-E5-F2-00-02","nonce":"Dss9
9sBr3pNMOACe-LYY7w","proximity-registrar-cert":"MIIBrjCCATOg
AwIBAgIBAzAKBggqhkjOPQQDAzBOMRIwEAYKCZImiZPyLGQBGRYCY2ExGTAX
BgoJkiaJk/IsZAEZFglzYW5kZWxtYW4xHTAbBgNVBAMMFFVuc3RydW5nIEZv
dW50YWluIENBMB4XDTE3MDkwNTAxMTI0NVoXDTE5MDkwNTAxMTI0NVowQzES
MBAGCgmSJomT8ixkARkWAmNhMRkwFwYKCZImiZPyLGQBGRYJc2FuZGVsbWFu
MRIwEAYDVQQDDAlsb2NhbGhvc3QwWTATBgcqhkjOPQIBBggqhkjOPQMBBwNC
AAQ1ZA7Nw0xSM/Q2u194FzQMktZ94waAIV0i/oVTPgOJ8zW6MwF5z+Dpb8/p
uhObJMZ0U6H/wfApR6svlumd4ryyow0wCzAJBgNVHRMEAjAAMAoGCCqGSM49
BAMDA2kAMGYCMQC3/iTQJ3evYYcgbXhbmzrp64t3QC6qjIeY2jkDx062nuNi
fVKtyaara3F30AIkKSECMQDi29efbTLbdtDk3tecY/rD7V77XaJ6nYCmdDCR
54TrSFNLgxvt1lyFM+0fYpYRc3o="}}
  829:d=3  hl=4 l= 566 cons: cont [ 0 ]
  833:d=4  hl=4 l= 562 cons: SEQUENCE
  837:d=5  hl=4 l= 439 cons: SEQUENCE
  841:d=6  hl=2 l=   3 cons: cont [ 0 ]
  843:d=7  hl=2 l=   1 prim: INTEGER            :02
  846:d=6  hl=2 l=   1 prim: INTEGER            :0C
  849:d=6  hl=2 l=  10 cons: SEQUENCE
  851:d=7  hl=2 l=   8 prim: OBJECT             :ecdsa-with-S
HA256
  861:d=6  hl=2 l=  77 cons: SEQUENCE
  863:d=7  hl=2 l=  18 cons: SET
  865:d=8  hl=2 l=  16 cons: SEQUENCE
  867:d=9  hl=2 l=  10 prim: OBJECT             :domainCompon
ent
  879:d=9  hl=2 l=   2 prim: IA5STRING          :ca
  883:d=7  hl=2 l=  25 cons: SET
  885:d=8  hl=2 l=  23 cons: SEQUENCE
  887:d=9  hl=2 l=  10 prim: OBJECT             :domainCompon
ent
  899:d=9  hl=2 l=   9 prim: IA5STRING          :sandelman
  910:d=7  hl=2 l=  28 cons: SET
```

```
   912:d=8  hl=2 l=   26 cons: SEQUENCE
   914:d=9  hl=2 l=    3 prim: OBJECT            :commonName
   919:d=9  hl=2 l=   19 prim: UTF8STRING        :Unstrung Hig
 hway CA
   940:d=6  hl=2 l=   32 cons: SEQUENCE
   942:d=7  hl=2 l=   13 prim: UTCTIME           :171012135252
 Z
   957:d=7  hl=2 l=   15 prim: GENERALIZEDTIME   :299912310000
 00Z
   974:d=6  hl=2 l=   75 cons: SEQUENCE
   976:d=7  hl=2 l=   18 cons: SET
   978:d=8  hl=2 l=   16 cons: SEQUENCE
   980:d=9  hl=2 l=   10 prim: OBJECT            :domainCompon
 ent
   992:d=9  hl=2 l=    2 prim: IA5STRING         :ca
   996:d=7  hl=2 l=   25 cons: SET
   998:d=8  hl=2 l=   23 cons: SEQUENCE
  1000:d=9  hl=2 l=   10 prim: OBJECT            :domainCompon
 ent
  1012:d=9  hl=2 l=    9 prim: IA5STRING         :sandelman
  1023:d=7  hl=2 l=   26 cons: SET
  1025:d=8  hl=2 l=   24 cons: SEQUENCE
  1027:d=9  hl=2 l=    3 prim: OBJECT            :commonName
  1032:d=9  hl=2 l=   17 prim: UTF8STRING        :00-D0-E5-F2-
 00-02
  1051:d=6  hl=2 l=   89 cons: SEQUENCE
  1053:d=7  hl=2 l=   19 cons: SEQUENCE
  1055:d=8  hl=2 l=    7 prim: OBJECT            :id-ecPublicK
 ey
  1064:d=8  hl=2 l=    8 prim: OBJECT            :prime256v1
  1074:d=7  hl=2 l=   66 prim: BIT STRING
  1142:d=6  hl=3 l=  135 cons: cont [ 3 ]
  1145:d=7  hl=3 l=  132 cons: SEQUENCE
  1148:d=8  hl=2 l=   29 cons: SEQUENCE
  1150:d=9  hl=2 l=    3 prim: OBJECT            :X509v3 Subje
 ct Key Identifier
  1155:d=9  hl=2 l=   22 prim: OCTET STRING      [HEX DUMP]:04
 141D311661B611509B3CFA13B6155F390BED76432A
  1179:d=8  hl=2 l=    9 cons: SEQUENCE
  1181:d=9  hl=2 l=    3 prim: OBJECT            :X509v3 Basic
  Constraints
  1186:d=9  hl=2 l=    2 prim: OCTET STRING      [HEX DUMP]:30
 00
  1190:d=8  hl=2 l=   43 cons: SEQUENCE
  1192:d=9  hl=2 l=    3 prim: OBJECT            :X509v3 Subje
 ct Alternative Name
  1197:d=9  hl=2 l=   36 prim: OCTET STRING      [HEX DUMP]:30
 22A02006092B0601040182EE5201A0130C1130302D44302D45352D46322D
```

```
  30302D3032
   1235:d=8  hl=2 l=  43 cons: SEQUENCE
   1237:d=9  hl=2 l=   9 prim: OBJECT              :1.3.6.1.4.1.
  46930.2
   1248:d=9  hl=2 l=  30 prim: OCTET STRING       [HEX DUMP]:0C
  1C68747470733A2F2F686967687761792E73616E64656C6D616E2E6361
   1280:d=5  hl=2 l=  10 cons: SEQUENCE
   1282:d=6  hl=2 l=   8 prim: OBJECT              :ecdsa-with-S
  HA256
   1292:d=5  hl=2 l= 105 prim: BIT STRING
   1399:d=3  hl=4 l= 421 cons: SET
   1403:d=4  hl=4 l= 417 cons: SEQUENCE
   1407:d=5  hl=2 l=   1 prim: INTEGER             :01
   1410:d=5  hl=2 l=  82 cons: SEQUENCE
   1412:d=6  hl=2 l=  77 cons: SEQUENCE
   1414:d=7  hl=2 l=  18 cons: SET
   1416:d=8  hl=2 l=  16 cons: SEQUENCE
   1418:d=9  hl=2 l=  10 prim: OBJECT              :domainCompon
  ent
   1430:d=9  hl=2 l=   2 prim: IA5STRING           :ca
   1434:d=7  hl=2 l=  25 cons: SET
   1436:d=8  hl=2 l=  23 cons: SEQUENCE
   1438:d=9  hl=2 l=  10 prim: OBJECT              :domainCompon
  ent
   1450:d=9  hl=2 l=   9 prim: IA5STRING           :sandelman
   1461:d=7  hl=2 l=  28 cons: SET
   1463:d=8  hl=2 l=  26 cons: SEQUENCE
   1465:d=9  hl=2 l=   3 prim: OBJECT              :commonName
   1470:d=9  hl=2 l=  19 prim: UTF8STRING          :Unstrung Hig
  hway CA
   1491:d=6  hl=2 l=   1 prim: INTEGER             :0C
   1494:d=5  hl=2 l=  13 cons: SEQUENCE
   1496:d=6  hl=2 l=   9 prim: OBJECT              :sha256
   1507:d=6  hl=2 l=   0 prim: NULL
   1509:d=5  hl=3 l= 228 cons: cont [ 0 ]
   1512:d=6  hl=2 l=  24 cons: SEQUENCE
   1514:d=7  hl=2 l=   9 prim: OBJECT              :contentType
   1525:d=7  hl=2 l=  11 cons: SET
   1527:d=8  hl=2 l=   9 prim: OBJECT              :pkcs7-data
   1538:d=6  hl=2 l=  28 cons: SEQUENCE
   1540:d=7  hl=2 l=   9 prim: OBJECT              :signingTime
   1551:d=7  hl=2 l=  15 cons: SET
   1553:d=8  hl=2 l=  13 prim: UTCTIME             :171012175430
  Z
   1568:d=6  hl=2 l=  47 cons: SEQUENCE
   1570:d=7  hl=2 l=   9 prim: OBJECT              :messageDiges
  t
   1581:d=7  hl=2 l=  34 cons: SET
```

```
 1583:d=8  hl=2 l=  32 prim: OCTET STRING      [HEX DUMP]:FE
7D72E29500F90A38E95021A215FD6D40B1629B99598177DC054AE0F9C8B6
9F
 1617:d=6  hl=2 l= 121 cons: SEQUENCE
 1619:d=7  hl=2 l=   9 prim: OBJECT            :S/MIME Capab
ilities
 1630:d=7  hl=2 l= 108 cons: SET
 1632:d=8  hl=2 l= 106 cons: SEQUENCE
 1634:d=9  hl=2 l=  11 cons: SEQUENCE
 1636:d=10 hl=2 l=   9 prim: OBJECT            :aes-256-cbc
 1647:d=9  hl=2 l=  11 cons: SEQUENCE
 1649:d=10 hl=2 l=   9 prim: OBJECT            :aes-192-cbc
 1660:d=9  hl=2 l=  11 cons: SEQUENCE
 1662:d=10 hl=2 l=   9 prim: OBJECT            :aes-128-cbc
 1673:d=9  hl=2 l=  10 cons: SEQUENCE
 1675:d=10 hl=2 l=   8 prim: OBJECT            :des-ede3-cbc
 1685:d=9  hl=2 l=  14 cons: SEQUENCE
 1687:d=10 hl=2 l=   8 prim: OBJECT            :rc2-cbc
 1697:d=10 hl=2 l=   2 prim: INTEGER           :80
 1701:d=9  hl=2 l=  13 cons: SEQUENCE
 1703:d=10 hl=2 l=   8 prim: OBJECT            :rc2-cbc
 1713:d=10 hl=2 l=   1 prim: INTEGER           :40
 1716:d=9  hl=2 l=   7 cons: SEQUENCE
 1718:d=10 hl=2 l=   5 prim: OBJECT            :des-cbc
 1725:d=9  hl=2 l=  13 cons: SEQUENCE
 1727:d=10 hl=2 l=   8 prim: OBJECT            :rc2-cbc
 1737:d=10 hl=2 l=   1 prim: INTEGER           :28
 1740:d=5  hl=2 l=  10 cons: SEQUENCE
 1742:d=6  hl=2 l=   8 prim: OBJECT            :ecdsa-with-S
HA256
 1752:d=5  hl=2 l=  70 prim: OCTET STRING      [HEX DUMP]:30
440220614CB435374FFB14E49BF12DEBD788FBE4BFDB3DDD4303CA3B074B
D1C0C24AF0022008778C96F26CC9CA71FEC328A9EC9F61BF3B4E87781FFC
8A6308FA19C0EC27CD
```

The JSON contained in the voucher request:

    {"ietf-voucher-request:voucher":{"assertion":"proximity","cr
    eated-on":"2017-09-01","serial-number":"00-D0-E5-F2-00-02","
    nonce":"Dss99sBr3pNMOACe-LYY7w","proximity-registrar-cert":"
    MIIBrjCCATOgAwIBAgIBAzAKBggqhkjOPQQDAzBOMRIwEAYKCZImiZPyLGQB
    GRYCY2ExGTAXBgoJkiaJk/IsZAEZFglzYW5kZWxtYW4xHTAbBgNVBAMMFFVu
    c3RydW5nIEZvdW50YWluIENBMB4XDTE3MDkwNTAxMTI0NVoXDTE5MDkwNTAx
    MTI0NVowQzESMBAGCgmSJomT8ixkARkWAmNhMRkwFwYKCZImiZPyLGQBGRYJ
    c2FuZGVsbWFuMRIwEAYDVQQDDAlsb2NhbGhvc3QwWTATBgcqhkjOPQIBBggq
    hkjOPQMBBwNCAAQ1ZA7Nw0xSM/Q2u194FzQMktZ94waAIV0i/oVTPgOJ8zW6
    MwF5z+Dpb8/puhObJMZ0U6H/wfApR6svlumd4ryyow0wCzAJBgNVHRMEAjAA
    MAoGCCqGSM49BAMDA2kAMGYCMQC3/iTQJ3evYYcgbXhbmzrp64t3QC6qjIeY
    2jkDx062nuNifVKtyaara3F30AIkKSECMQDi29efbTLbdtDk3tecY/rD7V77
    XaJ6nYCmdDCR54TrSFNLgxvt1lyFM+0fYpYRc3o="}}

E.2.2. Registrar to MASA

As described in Section 5.4 the Registrar will sign a registrar
voucher-request, and will include pledge's voucher request in the
prior-signed-voucher-request.

    MIIN2gYJKoZIhvcNAQcCoIINyzCCDccCAQExDzANBglghkgBZQMEAgEFADCC
    Ck4GCSqGSIb3DQEHAaCCCj8Eggo7eyJpZXRmLXZvdWNoZXItcmVxdWVzdDp2
    b3VjaGVyIjp7ImFzc2VydGlvbiI6InByb3hpbWl0eSIsImNyZWF0ZWQtb24i
    OiIyMDE3LTA5LTE1VDAwOjAwOjAwLjAwMFoiLCJzZXJpYWwtbnVtYmVyIjoi
    SkFEQTEyMzQ1Njc4OSIsIm5vbmNlIjoiYWJjZDEyMzQiLCJwcmlvci1zaWdu
    ZWQtdm91Y2hlci1yZXF1ZXN0IjoiTUlJSEhRWUpLb1pJaHZjTkFRY0NvSUlI
    RGpDQ0J3b0NBUUV4RHpBTkJnbGdoa2dCWlFNRUFnRUZBRENDc0R0TTcUdT
    SWIzRFFFSEFhQ0NBdjhFZ2dMM2V5SnBaWFJtTFhadmRXTm9aWEl0Y21WeGdR
    VnpkRHAyYjNWamFISjQySW1GemMyVmlkR2x2YmlJNmluQnliM2hwbWYlds
    MGVTSXNJbU55WldGcFUXRiMjRpT2lJeU1ERTNMVEE1TFRBeElpd2ljMlZ5
    YVdGc0xXNTFiV0psY2lJNklqQXdMVVF3TFVVMUxVWlMlMVEF3TFRBeUlpd2li
    bTl1WTJV9pSkVjM001T1hOQ2NqTndUazFPUVVVObExeVlpXG9zSWl3aWNI
    SnZlR2x0YVhSNUxYSmlhMmx6ZEhKaGKaGNpMWXpiWEowSWpvaVVRVbEpRbkpxTBO
    QlZFT09W5RWGRKUWtHblNVSkJaM0ZMWU1kkbmNNXaHJhczlRVVZGRVYcENUMDFT
    U1hkRlFWbExRMXBKYldsYVVIbE1TMUZDUjFKWlExa3lSWGhJVkRXVFtZHZT
    bXRwWVVwckwwWnRabmJJcFpwWelZyV2xkNGRGbFhPSGhJVkdaVFt
    ZE9Wa0pCVFuxR1JsWjZek5TZVdSWE5UUpSVnAyYkxjjMU1GbFhSFZKZKUlU1
    Q1RVSTBXRVJVUlROTlJHNUbFJCZUUxVVNUQk9WbTlZUkVSRk5VTVVVhM2RP
    VkdGNFRWUkpNRTVWYjNkUmExVlVRUpCObmJWXtiMjFFVT0dsNGEwRlNh
    MWRCYlU1b1RWSnJkMFozV1V0RFdrBHRhVnBBZVV4SFVVSkhVbGxXXpKR2RW
    cEhWbk5pVjBaMVRVWSkpkMFZCV1VSVldRkVZYzJJeVVRtaGlSSMmgyWXpO
    UmQxZFFRVlJlJDWjJOeGFHdHFFUMUJSU1VKQ1oyHhhhR3RxFFCUlRVSkNkMDVE
    UVVGUk1WcEJPMDUzTUhoVFROVlJNbVV4T1RSR2VsRk5hM1ZaS1FrdkUyhQWWtzTld
    UVVGUk1WcEJPMDUzTUhoVFRPVJNblV4T1RSR2VsRk5hM1ZaS1FrdkUyhQWWtzTldq
    akJwTDI5V1ZGQm5UMG80ZWxjMlYZEdOWG9yRWdEdkV2hQWWtwTlddq
    QlZOa2d2ZDJaQmNGT2STJjM1pzZFcxa05ISjVlVzkzTUhkRGVrRktbBWRVmto
    U1RVVkJha0hla0ZFVGdlIwTkRjVWRUVFRRNVFRk5SRk5YTBGTlIxbIIxbERVkZE
    TXk5cFFZGRktNMlYyV1ZZaaloySllobR0p0ZW5kd05qUjBNUZETm5GcVNVZVlpN
    bXByUkhnd05qSnVkVTVpZlZWWmxaTGRibGhZWEpoTTBSBZek1FRkphMHRUVOTlVV

UnBNamxsWm1KVVRHSmtkRVJyTTNSbFkxa3Zja1EzVmpjM1dHRktObTVaUTIx
a1JFTlNOVFJVY2xOR1RreG5lSFowTVd4NVJrMHJNR1paY0ZsU1l6TnZQU0o5
ZmFDQ0FqWXdnZ0l5TUlJQnlQUURBZ0VDQWdFQTU1Bb0dDQ3FHU000OUJBTUNN
RTB4RWpBUUJnb0praWFKay9Jc1pBRVpHZ0pqWVRFWk1CY0dDZ21TSm9tVDhp
eGtBUmtXQ1hOaGGJtUmxiRzFoYmpFY01Cb0dBMVVFQXd3VFZXNXpkSEoxWm1j
Z1NHbG5hSGRoZVZCRFFUQWdGdGJ44TnpGd01USXhOelV5TlRKRYUdBOHlPVGs1
TVRJek1UUQXdNREE3TUZvd01N6RVNNQkFHQ1FHQ2dtU0pvbVQ0aXhrQVJrV0FtTmhN
Umt3RndnS1JndZS0NaSW1pWlRpTQBGQDEZSY0j2Fu4ZGVsbFFuUjR0wGwYDVQQD
DDBRVbnN0cnVuZyBGb3VudGFpbiBEQ1RBZWxOMNZA5MDUwMTEyNDVaFwx0TA5
MDUwMTEyNDVaMEMxEjAQBgoJkiaJk/IsZAEZFgJjYTEZMBcGCgmSJomT8ixk
ARkWCXNhbmRsbG1hbjESMBAGA1UEAwwJbG9jYWxob3N0MFkwEwYHKoZIzj0C
AQYIKoZIzj0DAQcDQgAENWQOzcNMUjP0NrtfeBc0DJLWfeMGgCFdIv6FUz4DifM
1ujMBec/g6W/P6boTmyTGdFOh/8HwKUerL5bpneK8sqMNMAswCQYDVR0TBAIw
ADAKBggqhkjOPQQDAwNpADBmAjEAt/4k0Cd3r2GHIG14W5s66euLd0AuqoyHm
No5A8dOtp7jYn1Srcmmq2txd9ACJCkhAjEA4tvXn20y23bQ5N7XnGP6w+1e+1
2iep2ApnQwkeeE60hTS4Mb7dZchTPtH2KWEXN6MYIBpzCCAaMCAQEwUzBOMR
IwEAYKCZImiZPyLGQBGRYCY2ExGTAXBgoJkiaJk/IsZAEZFglzYW5kZWxtYW
4xHTAbBgNVBAMMFFVuc3RydW5nIEZvdW50YWluIENBAgEDMA0GCWCGSAFlAw
QCAQUAoIHkMBgGCSqGSIb3DQEJAzELBgkqhkiG9w0BBwEwHAYJKoZIhvcNAQk
FMQ8XDTE3MTAyNjAxMzYxOFowLwYJKoZIhvcNAQkEMSIEIEQBM73PZzPo7tE9
Mj8gQvaaYeMQOsxlACaW/HenAqNwMHkGCSqGSIb3DQEJDzFsMGowCwYJYIZIA
WUDBAEqMAsGCWCGSAFlAwQBFjALBglghkgBZQMEAQIwCgYIKoZIhvcNAwcwDgYIKoZIhvcN

         AwICAgCAMA0GCCqGSIb3DQMCAgFAMAcGBSsOAwIHMA0GCCqGSIb3DQMCAgEo
         MAoGCCqGSM49BAMCBEcwRQIgDdp5uPUlMKp7GFQAD7ypAgqFv8q+KkJt6c3O
         7iVpVI8CIQCD1u8BkxipvigwvIDmWfjlYdJxcvozNjffq5j3UHg7Rg==

         file: examples/parboiled_vr_00-D0-E5-F2-00-02.pkcs

         The ASN1 decoding of the artifact:

            0:d=0 hl=4 l=3546 cons: SEQUENCE
            4:d=1 hl=2 l=   9 prim: OBJECT   :pkcs7-signed
         Data
           15:d=1 hl=4 l=3531 cons: cont [0]
           19:d=2 hl=4 l=3527 cons: SEQUENCE
           23:d=3 hl=2 l=   1 prim: INTEGER   :01
           26:d=3 hl=2 l= 15 cons: SET
           28:d=4 hl=2 l= 13 cons: SEQUENCE
           30:d=5 hl=2 l=   9 prim: OBJECT   :sha256
           41:d=5 hl=2 l=   0 prim: NULL
           43:d=3 hl=4 l=2638 cons: SEQUENCE
           47:d=4 hl=2 l=   9 prim: OBJECT   :pkcs7-data
           58:d=4 hl=4 l=2623 cons: cont [0]
           62:d=5 hl=4 l=2619 prim: OCTET STRING   :{"ietf-vouch
         er-request:voucher":{"assertion":"proximity","created-on":"2
         017-09-15T00:00:00.000Z","serial-number":"JADA123456789","no
         nce":"abcd1234","prior-signed-voucher-request":"MIIHHQYJKoZI
         hvcNAQcCoIIHDjCCBwoCAQExDzANBglghkgBZQMEAgEFADCCAw4GCSqGSIb3
         DQEHAaCCAv8EggL7eyJpZXRmLXZvdWNoZXItcmVxdWVzdDp2b3VjaGVyIjp7
         ImFzc2VydGlvbiI6InByb3hpbWl0eSIsImNyZWF0ZWQtb24iOiIyMDE3LTA5
         LTAxIiwic2VyaWFsLW51bWJlciI6IjAwLUQwLUU1LUYyLTAwLTAyIiwibm9u
         Y2UiOiJEc3M5OXNCcjNwTk1PQUNlLUxZWTd3IiwicHJveGltaXR5LXJlZ2lz
         dHJhci1jZXJ0IjoiTUlJQnJqQ0NBVE9nQXdJQkFnSUJBekFLQmdncWhrak9Q
         UVFEQXpCCT01SSXdFQVlLQ1pJbWlaUHlMR1FCR1JZQ1kyRXhHVEFYQmdvSmtp
         YUprL0lzWkFFWkZnbHpZVzVrYld4dFlXNHhIVEFiQmdOVkJBTU1GR1YzNS
         eWRXNW5JRVVp2ZFc1MFIXbHVJRU5CTUI0WERURTNNRGt3TlRBeE1USTBOVm9Y
         RFRFNU1Ea3dOVEF4TVRJME5Wb3dRRekVTTUJBR0NnbVNKb21UOGl4a0FSa1dB
         bU5oTVJrd0Z3WUtDWktaVnZoVUJHUUllYZJGdVpHVnNiVFZJJd0VB
         WURWUVFEREFsc2IyYnRhiR2h2YzNRd1dUQVRCZ2NxaGtqT1BRSUJCZ2dxaGtq
         T1BRTUJCd05DQUFRMVpBN053SmhTTS9RRMnUxOTRGZelFNna0OTR3YUFJVjjBp
         L29WWFBnT0o4elc2TXdlNXorRHBiOC9wdWhPYYkpNWjBVNkgvd2ZBcFI2c3Zs
         dW1kNHJ5eW93MHdHDekFKQmdOVkhTWUVBakFBTUFvR0NDcUdTTTQ5QkFNREEy
         a0FNR1lDTVVFDTVFM3y9pVFFKM2V2WVljZ2JYaGGJtenJwNjR0M1FDNnFpSWVZMmpr
         RHgwNjJudU5pZlZLdDlheXHhM0YzMEFJa0tTRUNNUURpMjllZmJUTGJkERr
         M3RlY1kvckQ3Vjc3WGFKNm5ZQ21kRENSNTRUUclNGTkxneHZ0MWx5Rk00rMGZZ
         cFlSYzNvPSJ9faCCAjYwggIyMIIBt6ADAgECAgEMMAoGCCqGSM49BAMCME0x
         EjAQBgoJkiaJk/IsZAEZFgJjYTEZMBcGCgmSJomT8ixkARkWCXNhbmRlbG1h
         bjEcMBoGA1UEAwwTVW5zdHJ1bmcgSGlnaHdheSBDQTAgFw0xNzEwMTIxMzUy
         NTJaGA8yOTk5MTIzMTAwMDAwMFowSzESMBAGCgmSJomT8ixkARkWAmNhMRkw
         FwYKCZImiZPyLGQBGRYJc2FuZGVsbWFuMRowGAYDVQQDDBEwMC1EMC1FNS1G

        Mi0wMC0wMjBZMBMGByqGSM49AgEGCCqGSM49AwEHA0IABEmnmLR1TVpSdHa7
        zAxHCCQ26k1s0zubWfSaP7QolmNw8iogP62s+NNKhu24h2lE98ZnyFT+chS9
        6rDKhgjwE9ujgYcwgYQwHQYDVR0OBBYEFB0xFmG2EVCbPPoTthVfOQvtdkMq
        MAkGA1UdEwQCMAAwKwYDVR0RBCQwIqAgBgkrBgEEAYLuUgGgEwwRMDAtRDAt
        RTUtRjItMDAtMDIwKwYJKwYBBAGC7lICBB4MHGh0dHBzOi8vaGlnaGdheeS5z
        YW5kZWxtYW4uY2EwCgYIKoZIzj0EAwIDaQAwZgIxAOEnU355qdbVT97mqgxI
        a9S9YdHu6JzxwluHu9fLnzScGzxuk2frST/4jO8RR60zMgIxAKvW7G91h4qr
        uZtFcJHhkImzDrt8nuPJdlsJRKKv7fAFPb6VaCDM8NGBgHkAFvuwDDGCAaYw
        ggGiAgEBMFIwTTESMBAGCgmSJomT8ixkARkwAmNhMRkwFwYKCZImiZPyLGQB
        GRYJc2FuZGVsbWFuMRwwGgYDVQQDDBNVbnN0cnVuZyBIaWdod2F5IENBAgEM
        MA0GCWCGSAFlAwQCAQUAoIHkMBgGCSqGSIb3DQEJAzELBgkqhkiG9w0BBwEw
        HAYJKoZIhvcNAQkFMQ8XDTE3MTAxMjEzNTgyM1owLwYJKoZIhvcNAQkEMSIE
        IP59cuKVAPkKOOlQIaIV/W1AsWKbmVmBd9wFSuD5yLafMHkGCSqGSIb3DQEJ
        DzFsMGowCwYJYIZIAWUDBAEqMAsGCWCGSAFlAwQBFjALBglghkgBZQMEAQIw
        CgYIKoZIhvcNAwcwDgYIKoZIhvcNAwICAgCAMA0GCCqGSIb3DQMCAgFAMAcG
        BSsOAwIHMA0GCCqGSIb3DQMCAgEoMAoGCCqGSM49BAMCBEcwRQIgEMg1dJL7
        FcdtrVDx8qCazoe9+22Nz4ZwRB9gATGL7MMCIQDjssUlZzJqp2/kCd4WhxUh
        saCpTFwPrnNew5wCkYUF8Q=="}}

```
 2685:d=3  hl=4 l= 434 cons: cont [ 0 ]
 2689:d=4  hl=4 l= 430 cons: SEQUENCE
 2693:d=5  hl=4 l= 307 cons: SEQUENCE
 2697:d=6  hl=2 l=   3 cons: cont [ 0 ]
 2699:d=7  hl=2 l=   1 prim: INTEGER           :02
 2702:d=6  hl=2 l=   1 prim: INTEGER           :03
 2705:d=6  hl=2 l=  10 cons: SEQUENCE
 2707:d=7  hl=2 l=   8 prim: OBJECT            :ecdsa-with-S
HA384
 2717:d=6  hl=2 l=  78 cons: SEQUENCE
 2719:d=7  hl=2 l=  18 cons: SET
 2721:d=8  hl=2 l=  16 cons: SEQUENCE
 2723:d=9  hl=2 l=  10 prim: OBJECT            :domainCompon
ent
 2735:d=9  hl=2 l=   2 prim: IA5STRING         :ca
 2739:d=7  hl=2 l=  25 cons: SET
 2741:d=8  hl=2 l=  23 cons: SEQUENCE
 2743:d=9  hl=2 l=  10 prim: OBJECT            :domainCompon
ent
 2755:d=9  hl=2 l=   9 prim: IA5STRING         :sandelman
 2766:d=7  hl=2 l=  29 cons: SET
 2768:d=8  hl=2 l=  27 cons: SEQUENCE
 2770:d=9  hl=2 l=   3 prim: OBJECT            :commonName
 2775:d=9  hl=2 l=  20 prim: UTF8STRING        :Unstrung Fou
ntain CA
 2797:d=6  hl=2 l=  30 cons: SEQUENCE
 2799:d=7  hl=2 l=  13 prim: UTCTIME           :170905011245
Z
 2814:d=7  hl=2 l=  13 prim: UTCTIME           :190905011245
Z
```

```
 2829:d=6  hl=2 l=  67 cons: SEQUENCE
 2831:d=7  hl=2 l=  18 cons: SET
 2833:d=8  hl=2 l=  16 cons: SEQUENCE
 2835:d=9  hl=2 l=  10 prim: OBJECT            :domainCompon
ent
 2847:d=9  hl=2 l=   2 prim: IA5STRING         :ca
 2851:d=7  hl=2 l=  25 cons: SET
 2853:d=8  hl=2 l=  23 cons: SEQUENCE
 2855:d=9  hl=2 l=  10 prim: OBJECT            :domainCompon
ent
 2867:d=9  hl=2 l=   9 prim: IA5STRING         :sandelman
 2878:d=7  hl=2 l=  18 cons: SET
 2880:d=8  hl=2 l=  16 cons: SEQUENCE
 2882:d=9  hl=2 l=   3 prim: OBJECT            :commonName
 2887:d=9  hl=2 l=   9 prim: UTF8STRING        :localhost
 2898:d=6  hl=2 l=  89 cons: SEQUENCE
 2900:d=7  hl=2 l=  19 cons: SEQUENCE
 2902:d=8  hl=2 l=   7 prim: OBJECT            :id-ecPublicK
ey
 2911:d=8  hl=2 l=   8 prim: OBJECT            :prime256v1
 2921:d=7  hl=2 l=  66 prim: BIT STRING
 2989:d=6  hl=2 l=  13 cons: cont [ 3 ]
 2991:d=7  hl=2 l=  11 cons: SEQUENCE
 2993:d=8  hl=2 l=   9 cons: SEQUENCE
 2995:d=9  hl=2 l=   3 prim: OBJECT            :X509v3 Basic
Constraints
 3000:d=9  hl=2 l=   2 prim: OCTET STRING      [HEX DUMP]:30
00
 3004:d=5  hl=2 l=  10 cons: SEQUENCE
 3006:d=6  hl=2 l=   8 prim: OBJECT            :ecdsa-with-S
HA384
 3016:d=5  hl=2 l= 105 prim: BIT STRING
 3123:d=3  hl=4 l= 423 cons: SET
 3127:d=4  hl=4 l= 419 cons: SEQUENCE
 3131:d=5  hl=2 l=   1 prim: INTEGER           :01
 3134:d=5  hl=2 l=  83 cons: SEQUENCE
 3136:d=6  hl=2 l=  78 cons: SEQUENCE
 3138:d=7  hl=2 l=  18 cons: SET
 3140:d=8  hl=2 l=  16 cons: SEQUENCE
 3142:d=9  hl=2 l=  10 prim: OBJECT            :domainCompon
ent
 3154:d=9  hl=2 l=   2 prim: IA5STRING         :ca
 3158:d=7  hl=2 l=  25 cons: SET
 3160:d=8  hl=2 l=  23 cons: SEQUENCE
 3162:d=9  hl=2 l=  10 prim: OBJECT            :domainCompon
ent
 3174:d=9  hl=2 l=   9 prim: IA5STRING         :sandelman
 3185:d=7  hl=2 l=  29 cons: SET
```

```
 3187:d=8  hl=2 l=  27 cons: SEQUENCE
 3189:d=9  hl=2 l=   3 prim: OBJECT            :commonName
 3194:d=9  hl=2 l=  20 prim: UTF8STRING        :Unstrung Fou
ntain CA
 3216:d=6  hl=2 l=   1 prim: INTEGER           :03
 3219:d=5  hl=2 l=  13 cons: SEQUENCE
 3221:d=6  hl=2 l=   9 prim: OBJECT            :sha256
 3232:d=6  hl=2 l=   0 prim: NULL
 3234:d=5  hl=3 l= 228 cons: cont [ 0 ]
 3237:d=6  hl=2 l=  24 cons: SEQUENCE
 3239:d=7  hl=2 l=   9 prim: OBJECT            :contentType
 3250:d=7  hl=2 l=  11 cons: SET
 3252:d=8  hl=2 l=   9 prim: OBJECT            :pkcs7-data
 3263:d=6  hl=2 l=  28 cons: SEQUENCE
 3265:d=7  hl=2 l=   9 prim: OBJECT            :signingTime
 3276:d=7  hl=2 l=  15 cons: SET
 3278:d=8  hl=2 l=  13 prim: UTCTIME           :171026013618
Z
 3293:d=6  hl=2 l=  47 cons: SEQUENCE
 3295:d=7  hl=2 l=   9 prim: OBJECT            :messageDiges
t
 3306:d=7  hl=2 l=  34 cons: SET
 3308:d=8  hl=2 l=  32 prim: OCTET STRING      [HEX DUMP]:44
0133BDCF6733E8EED13D323F2042F69A61E3103ACC65002696FC77A702A3
70
 3342:d=6  hl=2 l= 121 cons: SEQUENCE
 3344:d=7  hl=2 l=   9 prim: OBJECT            :S/MIME Capab
ilities
 3355:d=7  hl=2 l= 108 cons: SET
 3357:d=8  hl=2 l= 106 cons: SEQUENCE
 3359:d=9  hl=2 l=  11 cons: SEQUENCE
 3361:d=10 hl=2 l=   9 prim: OBJECT            :aes-256-cbc
 3372:d=9  hl=2 l=  11 cons: SEQUENCE
 3374:d=10 hl=2 l=   9 prim: OBJECT            :aes-192-cbc
 3385:d=9  hl=2 l=  11 cons: SEQUENCE
 3387:d=10 hl=2 l=   9 prim: OBJECT            :aes-128-cbc
 3398:d=9  hl=2 l=  10 cons: SEQUENCE
 3400:d=10 hl=2 l=   8 prim: OBJECT            :des-ede3-cbc
 3410:d=9  hl=2 l=  14 cons: SEQUENCE
 3412:d=10 hl=2 l=   8 prim: OBJECT            :rc2-cbc
 3422:d=10 hl=2 l=   2 prim: INTEGER           :80
 3426:d=9  hl=2 l=  13 cons: SEQUENCE
 3428:d=10 hl=2 l=   8 prim: OBJECT            :rc2-cbc
 3438:d=10 hl=2 l=   1 prim: INTEGER           :40
 3441:d=9  hl=2 l=   7 cons: SEQUENCE
 3443:d=10 hl=2 l=   5 prim: OBJECT            :des-cbc
 3450:d=9  hl=2 l=  13 cons: SEQUENCE
 3452:d=10 hl=2 l=   8 prim: OBJECT            :rc2-cbc
```

```
 3462:d=10 hl=2 l=   1 prim: INTEGER            :28
 3465:d=5  hl=2 l=  10 cons: SEQUENCE
 3467:d=6  hl=2 l=   8 prim: OBJECT             :ecdsa-with-S
HA256
 3477:d=5  hl=2 l=  71 prim: OCTET STRING       [HEX DUMP]:30
4502200DDA79B8F52530AA7B1854000FBCA9020A85BFCABE2A426DE9CDCE
EE2569548F02210083D6EF019318A9BE2830BC80E659F8E561D27172FA33
3637DFAB98F750783B46
```

E.2.3. MASA to Registrar

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

MIIG3AYJKoZIhvcNAQcCoIIGzTCCBskCAQExDzANBglghkgBZQMEAgEFADCC
AxAGCSqGSIb3DQEHAaCCAwEEggL9eyJpZXRmLXZvdWNoZXI6dm91Y2hlciI6
eyJhc3NlcnRpb24iOiJsb2dnZWQiLCJjcmVhdGVkLW9uIjoiMjAxNy0xMC0x
MlQxMzo1NDozMS40MzktMDQ6MDAiLCJzZXJpYWwtbnVtYmVyIjoiMDAtRDAt
RTUtRjItMDAtMDIiLCJub25jZSI6IkRzczk5c0JyM3BOTU9BQ2UtTFlZN3ci
LCJwaW5uZWQtZG9tYWluLWNlcnQiOiJNSUlCcmpDQ0FUZgBd0lCQWdJQkF6
QUtCZ2dxaGtqT1BRUURBekJPTVJJd0VBWUtDWkltaVpQeUxHUUJHUllFDWTJF
eEdUQQVhCZ29Ka2lhSmsvSXNaQUVaRmdsellXNXpaWkhMXWtaV3h0WVc0eEhUQWJCZ05W
QkFNTUZGVnJ5ZFc1bklFWnZkZVzUwWVdVdsdUlFTkJNQjRYRFRFRFFM01Ea3dO
VEF4TVRJME5Wb1hEVEU1TURrd05UQXhNVEkwTlZvd1F6RVNNQkFHQ2dtU0pv
bVQ4aXhrQVJrV0FtTmhNUmt3RndZS0NaSW1pWlB5TEdRQkdSWUpjMkZ1WkdW
c2JXRnVUUkl3RUZRRFZRUUREQWxzYjJJOaGJHaHZjM1F1RBVEJnY3Foa2pP
UFFJQkjnZ3Foa2pPUUFNQkJ3TkNBQVExWkE3TncweFNNL1EydTE5NEZ6UU1r
dFo5NHdhQUlWMGkvb1ZUUUddPSjh6VzZNd0Y1eitEcGI4L3B1aE9iSk1aMFU2
SC93ZkFwUjZzdmx1bWQ0cnl5b3cw0N6QUpCZ05WSFJNRUFqQUFNQW9HQ0Nx
R1NNNDlCQU1EQTJrQU1HWUNNUUMzL2lUUUozZXZZWWNnYlhooYm16cnA2NHQz
UUM2cWpJZVVkyamtEeDA2Mm51TmlmVkt0eWFhcmEzRjMwQUlrS1NFQ01RRGky
OWVmYlRMYmR0RGszdGVjWS9yRDddWNzdYYUo2bllDbwREQ1I1NFRyU0ZOTGd4
dnQxbHlGTSswZllwwWVJjM289In19oIIB0zCCAc8wggFWoAMCAQICAQEwCgYI
KoZIzj0EAwIwTTESMBAGCgmSJomT8ixkARkWAmNhMRkwFwYKCZImiZPyLGQB
GRYJc2FuZGVsbWFuMRwwGgYDVQQDDBNVbnN0cnVuZyBIaWdod2F5IENBMB4X
DTE3MDMyNjE2MTk0MFoXDTE5MDMyNjE2MTk0MFowRzESMBAGCgmSJomT8ixk
ARkWAmNhMRkwFwYKCZImiZPyLGQBGRYJc2FuZGVsbWFuMRYwFAYDVQQDDA1V
bnN0cnVuZyBNQVNBMHYwEAYHKoZIzj0CAQYFK4EEACIDYgAE2QB90W9hbyCT
p7bPr17llt+aH8jWwh84wMzotpFmRRNQcrqyiJjXDTBRoqxp0VyFxqlgn8OS
AoCfArjN71ebcvW3+ylJTpHo8077/uT1fvnpZD/R0PN76kwMLNlsFk8SoxAw
DjAMBgNVHRMBAf8EAjAAMAoGCCqGSM49BAMCA2cAMGQCMBm9KMjNHaD+rd/y
0jy+Tg7mrRMDGIe1hjviGExwvCuxMhwTpgmEXik9vhoVfwi1swIwTculDCU7
dbbMSbCanTD1CBY/uMGYNQDiG/yaAOjO6996cC0E6x0cRM1TBn1jpGFMMYIB
xjCCAcICAQEwUjBNMRIwEAYKCZImiZPyLGQBGRYCY2ExGTAXBgoJkiaJk/Is
ZAEZFglzYW5kZWxtYW4xHDAaBgNVBAMME1Vuc3RydW5nIEhpZ2h3YXkgQ0EC
AQEwDQYJYIZIAWUDBAIBBQCggeQwGAYJKoZIhvcNAQkDMQsGCSqGSIb3DQEH
ATAcBgkqhkiG9w0BCQUxDxcNMTcxMDEyMTc1NDMxWjAvBgkqhkiG9w0BCQQx
IgQgQXnG628cIW8MoYfB1ljDDlLlJQlxED2tnjcvkLEfix0weQYJKoZIhvcN
AQkPMWwwajALBglghkgBZQMEASowCwYJYIZIAWUDBAEWMAsGCWCGSAFlAwQB
AjAKBggqhkiG9w0DBzAOBggqhkiG9w0DAgICAIAwDQYIKoZIhvcNAwICAUAw
BwYFKw4DAgcwDQYIKoZIhvcNAwICASgwCgYIKoZIzj0EAwIEZzBlAjEAhzid
/AkNjttpSP1rflNppdHsi324Z2+TXJxueewnJ8z/2NXb+Tf3DsThv7du00Oz
AjBjyOnmkkSKHsPR2JluA5c6wovuPEnNKP32daGGeFKGEHMkTInbrqipC881
/5K9Q+k=

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

The ASN1 decoding of the artifact:

```
    0:d=0  hl=4 l=1756 cons: SEQUENCE
    4:d=1  hl=2 l=   9 prim: OBJECT            :pkcs7-signed
Data
```

```
    15:d=1  hl=4 l=1741 cons: cont [ 0 ]
    19:d=2  hl=4 l=1737 cons: SEQUENCE
    23:d=3  hl=2 l=   1 prim: INTEGER           :01
    26:d=3  hl=2 l=  15 cons: SET
    28:d=4  hl=2 l=  13 cons: SEQUENCE
    30:d=5  hl=2 l=   9 prim: OBJECT            :sha256
    41:d=5  hl=2 l=   0 prim: NULL
    43:d=3  hl=4 l= 784 cons: SEQUENCE
    47:d=4  hl=2 l=   9 prim: OBJECT            :pkcs7-data
    58:d=4  hl=4 l= 769 cons: cont [ 0 ]
    62:d=5  hl=4 l= 765 prim: OCTET STRING      :{"ietf-vouch
er:voucher":{"assertion":"logged","created-on":"2017-10-12T1
3:54:31.439-04:00","serial-number":"00-D0-E5-F2-00-02","nonc
e":"Dss99sBr3pNMOACe-LYY7w","pinned-domain-cert":"MIIBrjCCAT
OgAwIBAgIBAzAKBggqhkjOPQQDAzBOMRIwEAYKCZImiZPyLGQBGRYCY2ExGT
AXBgoJkiaJk/IsZAEZFglzYW5kZWxtYW4xHTAbBgNVBAMMFFVuc3RydW5nIE
ZvdW50YWluIENBMB4XDTE3MDkwNTAxMTI0NVoXDTE5MDkwNTAxMTI0NVowQz
ESMBAGCgmSJomT8ixkARkWAmNhMRkwFwYKCZImiZPyLGQBGRYJc2FuZGVsbW
FuMRIwEAYDVQQDDAlsb2NhbGhvc3QwWTATBgcqhkjOPQIBBggqhkjOPQMBBw
NCAAQ1ZA7Nw0xSM/Q2u194FzQMktZ94waAIV0i/oVTPgOJ8zW6MwF5z+Dpb8
/puhObJMZ0U6H/wfApR6svlumd4ryyow0wCzAJBgNVHRMEAjAAMAoGCCqGSM
49BAMDA2kAMGYCMQC3/iTQJ3evYYcgbXhbmzrp64t3QC6qjIeY2jkDx062nu
NifVKtyaara3F30AIkKSECMQDi29efbTLbdtDk3tecY/rD7V77XaJ6nYCmdD
CR54TrSFNLgxvt1lyFM+0fYpYRc3o="}}
   831:d=3  hl=4 l= 467 cons: cont [ 0 ]
   835:d=4  hl=4 l= 463 cons: SEQUENCE
   839:d=5  hl=4 l= 342 cons: SEQUENCE
   843:d=6  hl=2 l=   3 cons: cont [ 0 ]
   845:d=7  hl=2 l=   1 prim: INTEGER           :02
   848:d=6  hl=2 l=   1 prim: INTEGER           :01
   851:d=6  hl=2 l=  10 cons: SEQUENCE
   853:d=7  hl=2 l=   8 prim: OBJECT            :ecdsa-with-S
HA256
   863:d=6  hl=2 l=  77 cons: SEQUENCE
   865:d=7  hl=2 l=  18 cons: SET
   867:d=8  hl=2 l=  16 cons: SEQUENCE
   869:d=9  hl=2 l=  10 prim: OBJECT            :domainCompon
ent
   881:d=9  hl=2 l=   2 prim: IA5STRING         :ca
   885:d=7  hl=2 l=  25 cons: SET
   887:d=8  hl=2 l=  23 cons: SEQUENCE
   889:d=9  hl=2 l=  10 prim: OBJECT            :domainCompon
ent
   901:d=9  hl=2 l=   9 prim: IA5STRING         :sandelman
   912:d=7  hl=2 l=  28 cons: SET
   914:d=8  hl=2 l=  26 cons: SEQUENCE
   916:d=9  hl=2 l=   3 prim: OBJECT            :commonName
   921:d=9  hl=2 l=  19 prim: UTF8STRING        :Unstrung Hig
```

```
    hway CA
      942:d=6  hl=2 l=  30 cons: SEQUENCE
      944:d=7  hl=2 l=  13 prim: UTCTIME          :170326161940
    Z
      959:d=7  hl=2 l=  13 prim: UTCTIME          :190326161940
    Z
      974:d=6  hl=2 l=  71 cons: SEQUENCE
      976:d=7  hl=2 l=  18 cons: SET
      978:d=8  hl=2 l=  16 cons: SEQUENCE
      980:d=9  hl=2 l=  10 prim: OBJECT           :domainCompon
    ent
      992:d=9  hl=2 l=   2 prim: IA5STRING        :ca
      996:d=7  hl=2 l=  25 cons: SET
      998:d=8  hl=2 l=  23 cons: SEQUENCE
     1000:d=9  hl=2 l=  10 prim: OBJECT           :domainCompon
    ent
     1012:d=9  hl=2 l=   9 prim: IA5STRING        :sandelman
     1023:d=7  hl=2 l=  22 cons: SET
     1025:d=8  hl=2 l=  20 cons: SEQUENCE
     1027:d=9  hl=2 l=   3 prim: OBJECT           :commonName
     1032:d=9  hl=2 l=  13 prim: UTF8STRING       :Unstrung MAS
    A
     1047:d=6  hl=2 l= 118 cons: SEQUENCE
     1049:d=7  hl=2 l=  16 cons: SEQUENCE
     1051:d=8  hl=2 l=   7 prim: OBJECT           :id-ecPublicK
    ey
     1060:d=8  hl=2 l=   5 prim: OBJECT           :secp384r1
     1067:d=7  hl=2 l=  98 prim: BIT STRING
     1167:d=6  hl=2 l=  16 cons: cont [ 3 ]
     1169:d=7  hl=2 l=  14 cons: SEQUENCE
     1171:d=8  hl=2 l=  12 cons: SEQUENCE
     1173:d=9  hl=2 l=   3 prim: OBJECT           :X509v3 Basic
    Constraints
     1178:d=9  hl=2 l=   1 prim: BOOLEAN          :255
     1181:d=9  hl=2 l=   2 prim: OCTET STRING     [HEX DUMP]:30
    00
     1185:d=5  hl=2 l=  10 cons: SEQUENCE
     1187:d=6  hl=2 l=   8 prim: OBJECT           :ecdsa-with-S
    HA256
     1197:d=5  hl=2 l= 103 prim: BIT STRING
     1302:d=3  hl=4 l= 454 cons: SET
     1306:d=4  hl=4 l= 450 cons: SEQUENCE
     1310:d=5  hl=2 l=   1 prim: INTEGER          :01
     1313:d=5  hl=2 l=  82 cons: SEQUENCE
     1315:d=6  hl=2 l=  77 cons: SEQUENCE
     1317:d=7  hl=2 l=  18 cons: SET
     1319:d=8  hl=2 l=  16 cons: SEQUENCE
     1321:d=9  hl=2 l=  10 prim: OBJECT           :domainCompon
```

```
   ent
    1333:d=9  hl=2 l=   2 prim: IA5STRING          :ca
    1337:d=7  hl=2 l=  25 cons: SET
    1339:d=8  hl=2 l=  23 cons: SEQUENCE
    1341:d=9  hl=2 l=  10 prim: OBJECT             :domainCompon
   ent
    1353:d=9  hl=2 l=   9 prim: IA5STRING          :sandelman
    1364:d=7  hl=2 l=  28 cons: SET
    1366:d=8  hl=2 l=  26 cons: SEQUENCE
    1368:d=9  hl=2 l=   3 prim: OBJECT             :commonName
    1373:d=9  hl=2 l=  19 prim: UTF8STRING         :Unstrung Hig
   hway CA
    1394:d=6  hl=2 l=   1 prim: INTEGER            :01
    1397:d=5  hl=2 l=  13 cons: SEQUENCE
    1399:d=6  hl=2 l=   9 prim: OBJECT             :sha256
    1410:d=6  hl=2 l=   0 prim: NULL
    1412:d=5  hl=3 l= 228 cons: cont [ 0 ]
    1415:d=6  hl=2 l=  24 cons: SEQUENCE
    1417:d=7  hl=2 l=   9 prim: OBJECT             :contentType
    1428:d=7  hl=2 l=  11 cons: SET
    1430:d=8  hl=2 l=   9 prim: OBJECT             :pkcs7-data
    1441:d=6  hl=2 l=  28 cons: SEQUENCE
    1443:d=7  hl=2 l=   9 prim: OBJECT             :signingTime
    1454:d=7  hl=2 l=  15 cons: SET
    1456:d=8  hl=2 l=  13 prim: UTCTIME            :171012175431
   Z
    1471:d=6  hl=2 l=  47 cons: SEQUENCE
    1473:d=7  hl=2 l=   9 prim: OBJECT             :messageDiges
   t
    1484:d=7  hl=2 l=  34 cons: SET
    1486:d=8  hl=2 l=  32 prim: OCTET STRING     [HEX DUMP]:41
   79C6EB6F1C216F0CA187C1D658C30E52E5250971103DAD9E372F90B11F8B
   1D
    1520:d=6  hl=2 l= 121 cons: SEQUENCE
    1522:d=7  hl=2 l=   9 prim: OBJECT             :S/MIME Capab
   ilities
    1533:d=7  hl=2 l= 108 cons: SET
    1535:d=8  hl=2 l= 106 cons: SEQUENCE
    1537:d=9  hl=2 l=  11 cons: SEQUENCE
    1539:d=10 hl=2 l=   9 prim: OBJECT             :aes-256-cbc
    1550:d=9  hl=2 l=  11 cons: SEQUENCE
    1552:d=10 hl=2 l=   9 prim: OBJECT             :aes-192-cbc
    1563:d=9  hl=2 l=  11 cons: SEQUENCE
    1565:d=10 hl=2 l=   9 prim: OBJECT             :aes-128-cbc
    1576:d=9  hl=2 l=  10 cons: SEQUENCE
    1578:d=10 hl=2 l=   8 prim: OBJECT             :des-ede3-cbc
    1588:d=9  hl=2 l=  14 cons: SEQUENCE
    1590:d=10 hl=2 l=   8 prim: OBJECT             :rc2-cbc
```

```
 1600:d=10 hl=2 l=   2 prim: INTEGER            :80
 1604:d=9  hl=2 l=  13 cons: SEQUENCE
 1606:d=10 hl=2 l=   8 prim: OBJECT             :rc2-cbc
 1616:d=10 hl=2 l=   1 prim: INTEGER            :40
 1619:d=9  hl=2 l=   7 cons: SEQUENCE
 1621:d=10 hl=2 l=   5 prim: OBJECT             :des-cbc
 1628:d=9  hl=2 l=  13 cons: SEQUENCE
 1630:d=10 hl=2 l=   8 prim: OBJECT             :rc2-cbc
 1640:d=10 hl=2 l=   1 prim: INTEGER            :28
 1643:d=5  hl=2 l=  10 cons: SEQUENCE
 1645:d=6  hl=2 l=   8 prim: OBJECT             :ecdsa-with-S
HA256
 1655:d=5  hl=2 l= 103 prim: OCTET STRING      [HEX DUMP]:30
6502310087389DFC090D8EDB6948FD6B7E5369A5D1EC8B7DB8676F935C9C
6E79EC2727CCFFD8D5DBF937F70EC4E1BFB76ED343B3023063C8E9E69244
8A1EC3D1D8996E03973AC28BEE3C49CD28FDF675A1867852861073244C89
DBAEA8A90BCF35FF92BD43E9
```

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/

   Michael H. Behringer

   Email: Michael.H.Behring@gmail.com

   Steinthor Bjarnason
   Arbor Networks

   Email: sbjarnason@arbor.net

   Kent Watsen
   Juniper Networks

   Email: kwatsen@juniper.net