Abstract

This document specifies automated bootstrapping of an Autonomic Control Plane. To do this a remote secure key infrastructure (BRSKI) is created using manufacturer installed X.509 certificate, in combination with a manufacturer’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

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on July 21, 2019.
Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction .................................. 4
   1.1. Prior Bootstrapping Approaches .......... 6
2. Architectural Overview .......................... 12
   2.1. Behavior of a Pledge .................... 14
   2.2. Secure Imprinting using Vouchers ....... 15
   2.3. Initial Device Identifier ............... 16
      2.3.1. Identification of the Pledge .......... 16
      2.3.2. MASA URI extension ................. 17
   2.4. Protocol Flow ........................... 19
   2.5. Architectural Components ............... 21
      2.5.1. Pledge .......................... 21
      2.5.2. Join Proxy ........................ 21
      2.5.3. Domain Registrar .................... 21
      2.5.4. Manufacturer Service ................. 21
      2.5.5. Public Key Infrastructure (PKI) .... 21
   2.6. Certificate Time Validation .............. 22
      2.6.1. Lack of realtime clock ............... 22
      2.6.2. Infinite Lifetime of IDevID .......... 24
   2.7. Cloud Registrar ........................ 24
   2.8. Determining the MASA to contact ....... 24
3. Voucher-Request artifact ..................... 25
   3.1. Tree Diagram ........................... 26
1. Introduction

BRSKI provides a solution for secure zero-touch (automated) bootstrap of virgin (untouched) devices that are called pledges in this document.

This document primarily provides for the needs of the ISP and Enterprise focused ANIMA Autonomic Control Plane (ACP) [I-D.ietf-anima-autonomic-control-plane]. Other users of the BRSKI protocol will need to provide separate applicability statements that include privacy and security considerations appropriate to that deployment. Section 8 explains the details applicability for this the ACP usage.
This document describes how pledges discover (or be discovered by) an element of the network domain to which the pledge belongs to perform the bootstrap. This element (device) is called the registrar. Before any other operation, pledge and registrar need to establish mutual trust:

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

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

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

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

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

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

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

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

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

To literally "pull yourself up by the bootstraps" is an impossible action. Similarly the secure establishment of a key infrastructure without external help is also an impossibility. Today it is commonly accepted that the initial connections between nodes are insecure, until key distribution is complete, or that domain-specific keying material (often pre-shared keys, including mechanisms like SIM cards) is pre-provisioned on each new device in a costly and non-scalable manner. Existing automated mechanisms are known as non-secured 'Trust on First Use' (TOFU) [RFC7435], 'resurrecting duckling' [Stajano99there resurrecting] or 'pre-staging'.

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

- using the Implicit Trust Anchor database to authenticate an owner specific service (not an autonomic solution because the URL must be securely distributed),
- engaging a human user to authorize the CA certificate using out-of-band data (not an autonomic solution because the human user is involved),
- using a configured Explicit TA database (not an autonomic solution because the distribution of an explicit TA database is not autonomic),
- and using a Certificate-Less TLS mutual authentication method (not an autonomic solution because the distribution of symmetric key material is not autonomic).

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

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

- the pledge requires realtime connectivity to the manufacturer service,
o the domain identity is exposed to the manufacturer service (this is a privacy concern),

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

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

1.2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [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 pinned-domain-cert leaf, i.e. the Registrars' certificate. 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 at the factory.
Voucher: A signed artifact from the MASA that indicates to a pledge the cryptographic identity of the registrar it should trust. There are different types of vouchers depending on how that trust is asserted. Multiple voucher types are defined in [RFC8366].

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

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

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

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

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

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

IPIP Proxy: A stateless proxy alternative.

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 [RFC8366]

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.

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

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

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

1.3.1. Support environment

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

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

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

1.3.2. Constrained environments

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

The solution described in this document is aimed in general at non-constrained (i.e., class 2+) devices operating on a non-Challenged network. The entire solution as described here is not intended to be usable as-is by constrained devices operating on challenged networks (such as 802.15.4 LLNs).

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

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

1.3.3. Network Access Controls

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

1.3.4. Bootstrapping is not Booting

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

1.4. Leveraging the new key infrastructure / next steps

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

Services that benefit from this:
o Device management.

o Routing authentication.

o Service discovery.

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

1.5. Requirements for Autonomic Network Infrastructure (ANI) devices

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

For devices that intend to become part of an Autonomic Network Infrastructure (ANI) ([I-D.ietf-anima-reference-model]) that includes an Autonomic Control Plane ([I-D.ietf-anima-autonomic-control-plane]), the following actions are required and MUST be performed by the pledge:

o BRSKI: Request Voucher

o EST: CA Certificates Request

o EST: CSR Attributes

o EST: Client Certificate Request

o BRSKI: Enrollment status Telemetry

The ANI Join Registrar ASA MUST support all the BRSKI and above listed EST operations.

All ANI devices SHOULD support the BRSKI proxy function, using circuit proxies. Other proxy methods are optional, and MUST NOT enabled unless the Join Registrar ASA indicates support for them in its announcement. (See Section 4.3)

2. Architectural Overview

The logical elements of the bootstrapping framework are described in this section. Figure 1 provides a simplified overview of the components.
We assume a multi-vendor network. In such an environment there could be a Manufacturer Service for each manufacturer that supports devices following this document’s specification, or an integrator could provide a generic service authorized by multiple manufacturers. It is unlikely that an integrator could provide Ownership Tracking services for multiple manufacturers due to the required sales channel integrations necessary to track ownership.

The domain is the managed network infrastructure with a Key Infrastructure the pledge is joining. The domain provides initial device connectivity sufficient for bootstrapping through a proxy. The domain registrar authenticates the pledge, makes authorization decisions, and distributes vouchers obtained from the Manufacturer Service. Optionally the registrar also acts as a PKI 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    |
+--------------+  
|       (1) Discover |
+---------------+  
|                   |
|       (2) Identity |
+-----------------+  
| rejected        |
|                   |
|       (3) Request |
|     Join         |
+-----------------+  
|                   |
|       (4) Imprint |
+-----------------+  
| Bad MASA response |
|   send Voucher Status Telemetry |
+-----------------+  
|       (5) Enroll |
+-----------------+  
| Enroll Failure |
|                   |
|       (6) 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. Request 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 manufacturer service provided voucher. A voucher contains sufficient information for the pledge to complete authentication of a registrar. (The embedded ‘pinned-domain-certificate’ 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.

After imprint a secure transport exists between pledge and registrar. This specification details integration with EST enrollment so that pledges can optionally obtain a locally issued certificate, although any REST interface could be integrated in future work.

2.2. Secure Imprinting using Vouchers

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

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

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

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

2.3. Initial Device Identifier

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

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

2. Securely authenticating the pledge’s identity via TLS connection to registrar. This provides protection against cloned/fake pledged.

3. Secure auto-discovery of the pledge’s MASA by the registrar via the MASA URI in IDevID as explained below.

4. (Optionally) communicating the MUD URL (see Appendix C).

5. (Optional) Signing of voucher-request by the pledge’s IDevID to enable MASA to generate voucher only to a registrar that has a connection to the pledge.

6. Authorizing pledge (via registrar) to receive certificate from domain CA, by signing the Certificate Signing Request (CSR).

2.3.1. Identification of the Pledge

In the context of BRSKI, pledges are uniquely identified by a "serial-number". This serial-number is used both in the "serial-number" field of voucher or voucher-requests (see Section 3) and in local policies on registrar or MASA (see Section 5).

The following fields are defined in [IDevID] and [RFC5280]:

The subject field’s DN encoding MUST include the "serialNumber" attribute with the device’s unique serial number. (from [IDevID] section 7.2.8, and [RFC5280] section 4.1.2.4’s list of standard attributes)

The subject-alt field’s encoding MAY include a non-critical version of the RFC4108 defined HardwareModuleName. (from [IDevID] section 7.2.9) If the IDevID is stored in a Trusted Platform Module (TPM), then this field MAY contain the TPM identification rather than the device’s serial number. If both fields are present, then the subject field takes precedence.

and they are used as follows by the pledge to build the "serial-number" that is placed in the voucher-request. In order to build it, the fields need to be converted into a serial-number of "type string". The following methods are used depending on the first available IDevID certificate field (attempted in this order):

1. [RFC4519] section 2.31 provides an example ("WI-3005") of the Distinguished Name "serialNumber" attribute. [RFC4514] indicates this is a printable string so no encoding is necessary.

2. The HardwareModuleName hwSerialNum OCTET STRING. This value is base64 encoded to convert it to a printable string format.

The above process to locate the serial-number MUST be performed by the pledge when filling out the voucher-request. Signed voucher-requests are always passed up to the MASA, and the connection between the serial-number in the voucher-request and the serial number in the IDevID certificate.

As explained in Section 5.5 the Registrar MUST extract the serial-number again itself from the pledge’s TLS certificate. It may consult the serial-number in the pledge-request if there are any possible confusion about the source of the serial-number (hwSerialNum vs serialNumber).

2.3.2. MASA URI extension

The following newly defined field SHOULD be in the PKIX IDevID certificate: A PKIX 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 ([RFC5785]) 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:
Figure 3
2.5. Architectural Components

2.5.1. Pledge

The pledge is the device that is attempting to join. Until the pledge completes the enrollment process, it has link-local network connectivity only to the proxy.

2.5.2. Join Proxy

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

2.5.3. Domain Registrar

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

The registrar uses an Implicit Trust Anchor database for authenticating the BRSKI-MASA TLS connection MASA certificate. The registrar uses a different Implicit Trust Anchor database for authenticating the BRSKI-EST TLS connection pledge client certificate. Configuration or distribution of these trust anchor databases is out-of-scope of this specification.

2.5.4. Manufacturer Service

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

2.5.5. Public Key Infrastructure (PKI)

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

The voucher provides a method for the distribution of a single PKI trust anchor (as the "pinned-domain-cert"). A distribution of the full set of current trust anchors is possible using the optional EST integration.
The domain’s registrar acts as an [RFC5272] Registration Authority, requesting certificates for pledges from the Key Infrastructure.

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

2.6. Certificate Time Validation

2.6.1. Lack of realtime clock

Many devices when bootstrapping do not have knowledge of the current time. Mechanisms such as Network Time Protocols cannot be secured until bootstrapping is complete. Therefore bootstrapping is defined 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:

- Initially the pledge does not know the current time.
- Bootstrapping pledges that have a Realtime Clock (RTC), SHOULD use it to verify certificate validity. However, they MUST be prepared for the recognize that the RTC might be completely wrong when a RTC battery fails and resets to an origin time (e.g., Jan. 1, 1970)
- If the pledge has any stable storage (such as from where firmware is loaded) then it SHOULD assume that the clock CAN NOT be before the date at which the firmware or the the storage was last time stamped. The pledge SHOULD NOT update the timestamps in any file systems until it has a secure time source. This provides an earliest date which is reasonable. Call this the current reasonable date (CRD). This value MUST NOT be used for any future Registration attempt. The current reasonable date (CRD) may only increase during a single attempt.
- The pledge is exposed to dates in the following five places (registrar certificate, notBefore and notAfter. Voucher created-on, and expires-on. Additionally, CMS signatures contain a signingTime)
- During the initial connection with the registrar, the pledge sees the registrar’s Certificate. It has an inception date (notBefore)
and an expiry date (notAfter). It is reasonable that the notBefore date be after the pledge’s current working reasonable date. It is however, suspicious for the notAfter date to be before the pledge’s current reasonable date. No action is recommended, other than an internal audit entry for this.

- If the notBefore date of the registrar’s certificate is newer than the pledge’s reasonable date, then it MAY update it’s current reasonable date to the notBefore value.

- After the voucher request process, the pledge will have a voucher. It can validate the signature on the voucher, as it has been (by literal construction) provided with the MASA’s key as a trust anchor. The time values (created-on, expires-on) in the voucher can not in general be validated as the pledge has no certain real time clock. There are some reasonable assumptions that can be made: the voucher’s expires-on time can not be prior to the pledge’s current reasonable date. For nonceless vouchers, the voucher’s created-on time COULD be earlier if the as well if a long-lived voucher was obtained some time in the past, and the pledge has since gone through a firmware update and factory reset.

- 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. In that case, the voucher’s created-on date MUST NOT be prior to the pledge’s current reasonable date. In addition, when there is a valid nonce, the current reasonable date MAY be incremented to that of the CMS signingTime.

- Once the voucher is accepted the validity period of the pinned-domain-cert in the voucher now serves as a valid time window. As explained in Section 5.5.4, the MASA has checked the registrar’s certificate against real clocks, the endorsement of the MASA allows the pledge to treat the notBefore and notAfter dates as being constraints on any subsequent certificate validity periods that may need to be checked: for instance, validating peer certificates during ANIMA ACP setup.

- 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.
2.6.2. Infinite Lifetime of IDevID

[RFC5280] explains that long lived pledge certificates "SHOULD be assigned the GeneralizedTime value of 99991231235959Z". Registrars MUST support such lifetimes and SHOULD support ignoring pledge lifetimes if they did not follow the RFC5280 recommendations.

For example, IDevID may have incorrect lifetime of N <= 3 years, rendering replacement pledges from storage useless after N years unless registrars support ignoring such a lifetime.

2.7. Cloud Registrar

There exist operationally open network wherein devices gain unauthenticated access to the internet at large. In these use cases the management domain for the device needs to be discovered within the larger internet. These are less likely within the anima scope but may be more important in the future.

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

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

If the pledge uses a well known URI for contacting a cloud registrar 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.

2.8. Determining the MASA to contact

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

The device’s Initial Device Identifier 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 C.

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

3. Voucher-Request artifact

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

A pledge forms the "pledge voucher-request" and submits it to the registrar.

The registrar in turn forms the "registrar voucher-request", and submits it to the MASA.

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

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

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

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

The following tree diagram illustrates a high-level view of a voucher-request document. The voucher-request builds upon the voucher artifact described in [RFC8366]. The tree diagram is described in [RFC8340]. Each node in the diagram is fully described by the YANG module in Section 3.3. 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-request examples for illustration purposes. 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": "62a2e7693d82fcd2624de58fb6722e5",
      "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
Example (3) The following example illustrates a registrar voucher-request. The ‘prior-signed-voucher-request’ leaf is not populated with the pledge’s voucher-request nor is the nonce leaf. This form might be used by a registrar requesting a voucher when the pledge can not communicate with the registrar (such as when it is powered down, or still in packaging), and therefore could not submit a nonce. This scenario is most useful when the registrar is aware that it will not be able to reach the MASA during deployment. See Section 5.5.

{  "ietf-voucher-request:voucher": {  "nonce": "62a2e7693d82fcd2a568f6722e5",  "created-on": "2017-01-01T00:00:02.000Z",  "assertion": "proximity",  "idevid-issuer": "base64encodedvalue=="  "serial-number": "JADA123456789"  "prior-signed-voucher": "base64encodedvalue=="  }
}

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 its 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 5.5.

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


```yaml
{
  "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 [RFC8366] voucher into a voucher-request.

```yaml
<CODE BEGINS> file "ietf-voucher-request@2018-02-14.yang"
module ietf-voucher-request {
  yang-version 1.1;

  namespace
  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 "This module defines the format for a voucher,
      which is produced by a pledge’s manufacturer or
      delegate (MASA) to securely assign a pledge to
      an ‘owner’, so that the pledge may establish a secure
      connection to the owner’s network infrastructure";
    reference "RFC YYYY: 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"
This module defines the format for a voucher request. It is a superset of the voucher itself. This artifact may be optionally signed. It provides content to the MASA for consideration during a voucher request.

The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', '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 "2018-02-14" {
    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-request information when signing a voucher for imprinting so as to minimize the final voucher size.";
    }

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

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

This section defines a stateful proxy mechanism which is referred to as a "circuit" 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.

Registrars are assumed to have logically a locally integrated circuit proxy to support directly (subnet) connected pledges - because registrars themselves do not define any functions for pledges to discover them. Such a logical local proxy does not need to provide actual TCP proxying (just discovery) as long as the registrar can operate with subnet (link) local addresses on the interfaces where pledges may connect to.

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

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

During the discovery of the Registrar by the Join Proxy, the Join Proxy will also learn which kinds of proxy mechanisms are available. This will allow the Join Proxy to use the lowest impact mechanism which the Join Proxy and Registrar have in common.
In order to permit the proxy functionality to be implemented on the maximum variety of devices the chosen mechanism SHOULD use the minimum amount of state on the proxy device. While many devices in the ANIMA target space will be rather large routers, the proxy function is likely to be implemented in the control plane CPU of such a device, with available capabilities for the proxy function similar to many class 2 IoT devices.

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

4.1. Pledge discovery of Proxy

The result of discovery is a logical communication with a registrar, through a proxy. The proxy is transparent to the pledge 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.

While the GRASP M_FLOOD mechanism is passive for the pledge, the optional other methods (mDNS, and IPv4 methods) are active. The pledge SHOULD run those methods in parallel with listening to for the M_FLOOD. The active methods SHOULD exponentially back-off to a maximum of one hour to avoid overloading the network with discovery attempts. Detection of change of physical link status (ethernet carrier for instance) SHOULD reset the exponential back off.
The pledge could discover more than one proxy on a given physical interface. The pledge can have a multitude of physical interfaces as well: a layer-2/3 ethernet switch may have hundreds of physical ports.

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

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

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

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

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

4.1.1. Proxy GRASP announcements

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

```
[M_FLOOD, 12340815, h'fe800000000000000000000000000001', 180000,
 ["AN_Proxy", 4, 1, ""],
 [Q.IPv6_LOCATOR,
  h'fe800000000000000000000000000001', IPPROTO_TCP, 4443]]
```

Figure 6b: Proxy Discovery

The formal CDDL [I-D.ietf-cbor-cddl] definition is:
flood-message = [M_FLOOD, session-id, initiator, ttl,  
+ [objective, (locator-option / [])]]

objective = ["AN_Proxy", objective-flags, loop-count,  
objectives-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
objectives-value = any ; none

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

Figure 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. 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 ACP instance of GRASP using  
M_FLOOD messages. They MUST support ANI TLS circuit proxy and  
therefore BRSKI across HTTPS/TLS native across the ACP. ANI proxies  
MUST support GRASP discovery of registrars.

The M_FLOOD is formatted as follows:

[M_FLOOD, 12340815, h’fda379a6f6ee0000002000000640000001’, 180000,  
{"AN_join_registrar", 4, 255, "EST-TLS"}],  
[O_IPv6_LOCATOR,  
h’fda379a6f6ee0000002000000640000001’, IPPROTO_TCP, 80}]

Figure 7a: 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 sufficiently low that the aggregate amount of periodic M_FLOODs 
from all EST servers causes negligible traffic across the ACP.

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

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

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

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

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

5. Protocol Details (Pledge - Registrar - MASA)

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

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

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

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

BRSKI uses existing CMS message formats for existing EST operations. BRSKI 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 state occurs only once, and that the subsequent resolution of the provision state is not subject to a MITM attack during a critical phase.

Summarized automation extensions for the BRSKI-EST flow are:

- The pledge provisionally accepts the registrar certificate during the TLS handshake as detailed in Section 5.1.
- The pledge either attempts concurrent connections, or it times out quickly and tries connections in series.
- The pledge requests and validates a voucher using the new REST calls described below.
- The pledge completes authentication of the server certificate as detailed in Section 5.6.1. This moves the BRSKI-EST TLS connection out of the provisional state.
- Mandatory bootstrap steps conclude with voucher status telemetry (see Section 5.7).

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

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

- Client authentication is automated using Initial Device Identity (IDevID) as per the EST certificate based client authentication. The subject field’s DN encoding MUST include the "serialNumber" attribute with the device’s unique serial number.
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 manufacturer trust anchor since serial numbers are not globally unique.

The registrar requests and validates the voucher from the MASA.

The registrar forwards the voucher to the pledge when requested.

The registrar performs log verifications in addition to local authorization checks before accepting optional pledge device enrollment requests.

5.1. BRSKI-EST TLS establishment details

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

Establishment of the BRSKI-EST TLS connection is as specified in EST [RFC7030] section 4.1.1 "Bootstrap Distribution of CA Certificates" [RFC7030] wherein the client is authenticated with the IDevID certificate, and the EST server (the registrar) is provisionally authenticated with an unverified server certificate.

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 5.6.1

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 connect to any other registrar’s via any other discovered proxies if there are any. If there were no other proxies discovered, the pledge MAY continue to wait, as long as it is concurrently listening for new proxy announcements.

5.2. Pledge Requests Voucher from the Registrar

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

This is done with an HTTPS POST using the operation path value of "/.well-known/est/requestvoucher".
The 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 the exception that it is not within a CMS 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 implementations 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 2.6.1 functionality. The nonce MUST NOT be reused for multiple 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 that the ‘proximity’ assertion and associated ‘proximity-registrar-cert’ are correct.
5.3. Registrar Authorization of Pledge

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

A Registrar accepts or declines a request to join the domain, based on the authenticated identity presented. Automated acceptance criteria include:

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

If 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 5.5) and returns that MASA signed voucher to the pledge as described in Section 5.6.

5.4. BRSKI-MASA TLS establishment details

The BRSKI-MASA TLS connection is a 'normal' TLS connection appropriate for HTTPS REST interfaces. The registrar initiates the connection and uses the MASA URL obtained as described in Section 2.8 for [RFC6125] authentication of the MASA.

The primary method of registrar "authentication" by the MASA is detailed in Section 5.5. As detailed in Section 10 the MASA might find it necessary to request additional registrar authentication.

The MASA and the registrars SHOULD be prepared to support TLS client certificate authentication and/or HTTP Basic or Digest authentication as described in RFC7030 for EST clients. This connection MAY also have no client authentication at all (Section 6.4)

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

5.5. Registrar Requests Voucher from MASA

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

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

The request media type is defined in [RFC8366] 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 CMS structure.

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

idevid-issuer: The idevid-issuer value from the pledge certificate is included to ensure a statistically unique identity.

prior-signed-voucher-request: 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). If a signed voucher-request was not received from the pledge then this leaf is omitted from the registrar voucher request.

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

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

Example JSON payloads of registrar voucher-requests are in Section 3.2 Examples 2 through 4.

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

5.5.1. MASA renewal of expired vouchers

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

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

5.5.2. MASA verification of voucher-request 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 the voucher-request signer to be a registrar. Performing this check provides value to the domain PKI by assuring the domain administrator that the MASA service will only respect claims from authorized Registration Authorities of the domain.

The MASA verifies that the domain CA certificate is included in the CMS structure as detailed in Section 5.5.

5.5.3. MASA authentication of registrar (certificate)

If a nonceless voucher-request is submitted the MASA 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 (details are out-of-scope of this document).

In the nonced case, validation of the registrar MAY be omitted if the device policy is to accept audit-only vouchers.

5.5.4. MASA revocation checking of registrar (certificate)

As noted in Section 5.5.3 the MASA performs registrar authentication in a subset of situations (e.g. nonceless voucher requests). Normal PKIX revocation checking is assumed during either EST client authentication or voucher-request signature validation. Similarly, as noted in Section 5.5.2, the MASA performs normal PKIX revocation checking during signature consistency checks (a signature by a registrar certificate that has been revoked is an inconsistency).

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

The MASA MAY verify that the registrar voucher-request includes the 'prior-signed-voucher-request' field. If so the prior-signed-voucher-request MUST include a 'proximity-registrar-cert' that is consistent with the certificate used to sign the registrar voucher-request. Additionally the voucher-request serial-number leaf MUST match the pledge serial-number that the MASA extracts from the signing certificate of the prior-signed-voucher-request. The MASA is aware of which pledges support signing of their voucher requests and can use this information to confirm proximity of the pledge with the registrar, thus ensuring that the BRSKI-EST TLS connection has no man-in-the-middle.

If these checks succeed the MASA updates the voucher and audit log assertion leaves with the "proximity" assertion.
5.5.6. MASA pinning of registrar

The registrar’s certificate chain is extracted from the signature method. The chain includes the domain CA certificate as specified in Section 5.5. This 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.

5.5.7. MASA nonce handling

The MASA does not verify the nonce itself. It MAY perform a simple consistency check: If the registrar voucher-request contains a nonce and the prior-signed-voucher-request exists then the nonce in both MUST be consistent. (Recall from above that the voucher-request might not contain a nonce, see Section 5.5 and Section 5.5.3).

The MASA MUST use the nonce from the registrar voucher-request for the resulting voucher and audit log. The prior-signed-voucher-request nonce is ignored during this operation.

5.6. MASA and Registrar Voucher Response

The MASA voucher response to the registrar is forwarded without changes to the pledge; therefore this section applies to both the MASA and the registrar. The HTTP signaling described applies to both the MASA and registrar responses. A registrar either caches prior MASA responses or dynamically requests a new voucher based on local policy (it does not generate or sign a voucher).

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

The registrar MAY respond with an HTTP 202 ("the request has been accepted for processing, but the processing has not been completed") as described in EST [RFC7030] section 4.2.3 wherein the client "MUST wait at least the specified 'Retry-After' time before repeating the same request". (see [RFC7231] section 6.6.4) The pledge is RECOMMENDED to provide local feedback (blinking LED etc) during this wait cycle if mechanisms for this are available. To prevent an attacker registrar from significantly delaying bootstrapping the pledge MUST limit the 'Retry-After' time to 60 seconds. Ideally the pledge would keep track of the appropriate Retry-After header values
for any number of outstanding registrars but this would involve a state table on the pledge. Instead the pledge MAY ignore the exact Retry-After value in favor of a single hard coded value. A registrar that is unable to complete the transaction the first time due to timing reasons will have future chances.

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

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

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

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

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

The response media type is:

application/voucher-cms+json The response is a "YANG-defined JSON document that has been signed using a CMS structure" as described in [RFC8366] using the JSON encoded described in [RFC7951]. The MASA MUST sign the response.

The syntactic details of vouchers are described in detail in [RFC8366]. For example, the voucher consists of:
`{  
  "ietf-voucher:voucher": {  
    "nonce": "62a2e7693d82fcda2624de58fb6722e5",  
    "assertion": "logging"  
    "pinned-domain-cert": "base64encodedvalue=="  
    "serial-number": "JADA123456789"  
  }  
}

The MASA populates the voucher fields as follows:

nonce: The nonce from the pledge if available. See Section 5.5.7.

assertion: The method used to verify assertion. See Section 5.5.5.

pinned-domain-cert: The domain CA cert. See Section 5.5.6.

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

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

expires-on: This is set for nonceless vouchers. The MASA ensures the voucher lifetime is consistent with any revocation or pinned-domain-cert consistency checks the pledge might perform. See section Section 2.6.1. There are three times to consider: (a) a configured voucher lifetime in the MASA, (b) the expiry time for the registrar’s certificate, (c) any certificate revocation information (CRL) lifetime. The expires-on field SHOULD be before the earliest of these three values. Typically (b) will be some significant time in the future, but (c) will typically be short (on the order of a week or less). The RECOMMENDED period for (a) is on the order of 20 minutes, so it will typically determine the lifespan of the resulting voucher.

Whenever a voucher is issued the MASA MUST update the audit log appropriately. The internal state requirements to maintain the audit log are out-of-scope. See Section 5.8.1 for a discussion of reporting the log to a registrar.
5.6.1. Pledge voucher verification

The pledge MUST verify the voucher signature using the manufacturer installed trust anchor associated with the manufacturer’s MASA (this is likely included in the pledge’s firmware).

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

The pledge MUST verify that the voucher nonce field is accurate and matches the nonce the pledge submitted to this registrar, or that the voucher is nonceless (see Section 6.2).

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 that it does not recognize.

5.6.2. Pledge authentication of provisional TLS connection

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

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

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

The pledge’s PKIX path validation of a registrar certificate’s validity period information is as described in Section 2.6.1. Once the PKIX path validation is successful the TLS connection is no longer provisional.
The pinned-domain-cert MAY be installed as an trust anchor for future operations. 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 [RFC7030] section 4.1.3 CA Certificate Response, which is an additional justification for the recommendation to proceed with EST key management operations. Once a full CA Certificate Response is obtained it is more authoritative for the domain than the limited ‘pinned-domain-cert’ response.

5.7. Pledge BRSKI Status Telemetry

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

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

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

```
{
  "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 JSON fields in this POST MAY be added, see Section 7.3.

5.8. Registrar audit log request

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

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

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

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

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

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.

In order to avoid enumeration of device audit logs, MASA that return URLs SHOULD take care to make the returned URL unguessable. For instance, rather than returning URLs containing a database number such as https://example.com/auditlog/1234 or the EUI of the device such https://example.com/auditlog/10-00-00-11-22-33, the MASA SHOULD return a randomly generated value (a "slug" in web parlance). The value is used to find the relevant database entry.
A MASA that returns a code 200 MAY also include a Location: header for future reference by the registrar.

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.

5.8.1. MASA audit log response

A log data file is returned consisting of all log entries associated with the the device selected by the IDevID presented in the request. The audit log may be truncated of old or repeated values as explained below. The returned data is in JSON format ([RFC7951]), and the Content-Type SHOULD be "application/json". For example:

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

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 for that domainID. In the case of
truncation the ‘event’ truncation value SHOULD contain a count of the number of events for this domainID that were truncated. 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, with the number of removed entries indicated as ‘arbitrary’.

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

A log where duplicate entries for the same domain have been 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 manufacturer transparency is better than hidden truncations.

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

5.8.2. Registrar audit log verification

Each time the Manufacturer Authorized Signing Authority (MASA) issues a voucher, it places it into the audit log for that device. The details are described in Section 5.8. The contents of the audit log can express a variety of trust levels, and this section explains what kind of trust a registrar can derive from the entries.

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

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

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

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

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

assertion: The assertion leaf in the voucher and audit log indicates why the MASA issued the voucher. A "verified" entry means that the MASA issued the associated voucher as a result of positive verification of ownership but this can still be problematic for registrar’s that expected only new (not pre-owned) pledges. A "logged" assertion informs the registrar that the prior vouchers were issued with minimal verification. A "proximity" assertion assures the registrar that the pledge was truly communicating with the prior domain and thus provides assurance that the prior domain really has deployed the pledge.

A relatively simple policy is to white list known (internal or external) domainIDs and to require all vouchers to have a nonce and/or require that all nonceless vouchers be from a subset (e.g. only internal) domainIDs. A simple action is to revoke any locally issued credentials for the pledge in question or to refuse to forward the voucher. 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.

5.9. EST Integration for PKI bootstrapping

The pledge SHOULD follow the BRSKI operations with EST enrollment operations including "CA Certificates Request", "CSR Attributes" and "Client Certificate Request" or "Server-Side Key Generation", etc. This is a relatively seamless integration since BRSKI 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.

An ANIMA ANI pledge MUST implement the EST automation extensions described below. 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.

5.9.1. EST Distribution of CA Certificates

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

This ensures that the pledge has the complete set of current CA certificates beyond the pinned-domain-cert (see Section 5.6.1 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.

5.9.2. EST CSR Attributes

Automated bootstrapping occurs without local administrative configuration of the pledge. In some deployments it is plausible that the pledge generates a certificate request containing only identity information known to the pledge (essentially the X.509 IDevID information) and ultimately receives a certificate containing domain specific identity information. Conceptually the CA has complete control over all fields issued in the end entity certificate. Realistically this is operationally difficult with the current status of PKI certificate authority deployments, where the CSR is submitted to the CA via a number of non-standard protocols. Even with all standardized protocols used, it could operationally be problematic to expect that service specific certificate fields can be created by a CA that is likely operated by a group that has no insight into different network services/protocols used. For example, the CA could even be outsourced.
To alleviate these operational difficulties, the pledge MUST request the EST "CSR Attributes" from the EST server and the EST server needs to be able to reply with the attributes necessary for use of the certificate in its intended protocols/services. This approach allows for minimal CA integrations and instead the local infrastructure (EST server) informs the pledge of the proper fields to include in the generated CSR. This approach is beneficial to automated bootstrapping 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 registrar MUST also confirm that the resulting CSR is formatted as indicated before forwarding the request to a CA. If the registrar is communicating with the CA using a protocol such as full CMC, which provides mechanisms to override the CSR attributes, then these mechanisms MAY be used even if the client ignores CSR Attribute guidance.

5.9.3. EST Client Certificate Request

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

5.9.4. Enrollment Status Telemetry

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

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.

5.9.5. Multiple certificates

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

5.9.6. EST over CoAP

This document describes extensions to EST for the purposes of bootstrapping of remote key infrastructures. Bootstrapping is relevant for CoAP enrollment discussions as well. The definition of EST and BRSKI over CoAP is not discussed within this document beyond ensuring proxy support for CoAP operations. Instead it is anticipated that a definition of CoAP mappings will occur in subsequent documents such as [I-D.ietf-ace-coap-est] and that CoAP mappings for BRSKI will be discussed either there or in future work.
6. 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.

This section is considered non-normative: use suggested methods MUST be detailed in specific profiles of BRSKI. This is the subject for future work.

6.1. Trust Model

This section explains the trust relationships detailed in Section 2.4:

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

Figure 10

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

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

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

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

Vendor Service, Ownership Validation: This form of manufacturer 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 a use case where the registrar can not connect to the MASA at the deployment time. Logging and validity periods address the security considerations of supporting these use cases.

2. 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 manufacturer service is unavailable. This behavior SHOULD be available via local configuration or physical presence methods (such as use of a serial/craft console) to ensure new entities can always be deployed even when autonomic methods fail. This allows for unsecured imprint.

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

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

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

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

6.4. MASA security reductions

Lower security modes chosen by the MASA service affect 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 a 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 a 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 and authorized to provide this functionality. The MASA 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 a voucher. This is expected to be a common operational model because doing so relieves the manufacturer providing MASA services from having to track ownership during shipping and supply chain and allows for a very low overhead MASA service. A registrar uses the audit log information as a defense in depth strategy to ensure that this does not occur unexpectedly (for example when purchasing new equipment the registrar would throw an error if any audit log information is reported.) The MASA SHOULD verify the ‘prior-signed-voucher-request’ information for pledges that support that functionality. This provides a proof-of-proximity check that reduces the need for ownership verification.

7. IANA Considerations

This document requires the following IANA actions:

7.1. Well-known EST registration

This document extends the definitions of "est" (so far defined via RFC7030) in the "https://www.iana.org/assignments/well-known-uris/ well-known-uris.xhtml" registry as follows:

- add /.well-known/est/requestvoucher (see Section 5.5)
- add /.well-known/est/requestauditlog (see Section 5.7)

7.2. PKIX Registry

IANA is requested to register the following:

This document requests a number for id-mod-MASAUReXtn2016(TBD) from the pkix(7) id-mod(0) Registry.

This document requests a number from the id-pe registry (SMI Security for PKIX Certificate Extension) for id-pe-masa-url.

7.3. Pledge BRSKI Status Telemetry

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

- version
- Status
- Reason
- reason-context

7.4. DNS Service Names

IANA is requested to register the following Service Names:

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

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

7.5. MUD File Extension for the MASA

The IANA is requested to list the name "masa" in the MUD extensions registry defined in [I-D.ietf-opsawg-mud]. Its use is documented in Appendix C.

8. Applicability to the Autonomic Control Plane

This document provides a solution to the requirements for secure bootstrap set out in Using an Autonomic Control Plane for Stable Connectivity of Network Operations, Administration, and Maintenance [RFC8368], A Reference Model for Autonomic Networking [I-D.ietf-anima-reference-model] and specifically the An Autonomic Control Plane (ACP) [I-D.ietf-anima-autonomic-control-plane], section 3.2 (Secure Bootstrap), and section 6.1 (ACP Domain, Certificate and Network).
The protocol described in this document has appeal in a number of other non-ANIMA use cases. Such uses of the protocol will be deploying into other environments with different tradeoffs of privacy, security, reliability and autonomy from manufacturers. As such those use cases will need to provide their own applicability statements, and will need to address unique privacy and security considerations for the environments in which they are used.

The autonomic control plane that this document provides bootstrap for is typically a medium to large Internet Service Provider organization, or an equivalent Enterprise that has significant layer-3 router connectivity. (A network consisting of primarily layer-2 is not excluded, but the adjacencies that the ACP will create and maintain will not reflect the topology until all devices participate in the ACP).

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

Such an operator also is capable of performing the traditional (craft serial-console) based bootstrap of devices. The zero-touch mechanism presented in this and the ACP document represents a significant efficiency: in particular it reduces the need to put senior experts on airplanes to configure devices in person. There is a recognition as the technology evolves that not every situation may work out, and occasionally a human still have to visit.

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

9. Privacy Considerations

9.1. MASA audit log

The MASA audit 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 authoritative bound to the supply chain.

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

9.2. Used, Stolen or Grey Market equipment

9.2.1. What BRSKI-MASA reveals to the manufacturer

The so-called "call-home" mechanism that occurs as part of the BRSKI-MASA connection standardizes what has been deemed by some as a sinister mechanism for corporate oversight of individuals. ([livingwithIoT] and [IoTstrangeThings] for a small sample).

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

First, it needs to be re-iterated that the BRSKI-MASA mechanism only occurs once during the commissioning of the device. It is well defined, and although encrypted with TLS, it could in theory be made auditable as the contents are well defined.

This connection does not occur when the device powers on or is restarted for normal routines. It is possible with low-quality implementations that it might occur after certain firmware upgrades, for instance if the upgrade requires repartitioning of firmware space and this affects the credential storage. The usage of a Trusted Platform Module (TPM) to store credentials should completely eliminate this fault, however. Therefore this situation is clearly a quality of implementation issue, and even in the worse case, it results in additional enrollments being recorded when the significant firmware update occurs.

(Some manufacturers might regard such an additional call-home a significant feature as it might tell them how many of their customers have performed this major upgrade, and therefore how many are still vulnerable to some serious issue)
Second, the BRSKI call-home mechanism is mediated via the owner’s Registrar, and the information that is transmitted is directly auditable by the device owner. This is in stark constrast to many "call-home" protocols where the device autonomously calls home and uses an undocumented protocol. While the contents of the signed part of the pledge voucher request can not be changed, they are not encrypted at the registrar. The contents of an unsigned voucher request are, however, completely changeable by the Registrar. Both are, to re-iterate, encrypted by TLS while in transit.

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

- the identity of the device being enrolled (down to the serial-number!).
- an identity of the domain owner in the form of the domain trust anchor. However, this is not a global PKI anchored name within the WebPKI, so this identity could be pseudonymous. If there is sales channel integration, then the MASA will have authenticated the domain owner, either via pinned certificate, or perhaps another HTTP authentication method, as per Section 5.5.3.
- the time the device is activated,
- the IP address of the domain Owner’s Registrar. For ISPs and Enterprises, the IP address provides very clear geolocation of the owner. No amount of IP address privacy extensions ([RFC4941]) can do anything about this, as a simple whois lookup likely identifies the ISP or Enterprise from the upper bits anyway. A passive attacker who observes the connection definitely may conclude that the given enterprise/ISP is a customer of the particular equipment vendor. The precise model that is being enrolled will remain private.

The above situation is to be distinguished from a residential/individual person who registers a device from a manufacturer: that an enterprise/ISP purchases routing products is hardly worth mentioning. Deviations would, however, be notable.

The situation is not improved by the enterprise/ISP using anonymization services such as ToR [Dingledine2004], as a TLS 1.2 connection will reveal the ClientCertificate used, clearly identifying the enterprise/ISP involved. TLS 1.3 is better in this regard, but an active attacker can still discover the parties involved by performing a Man-In-The-Middle-Attack on the first attempt (breaking/killing it with a TCP RST), and then letting subsequent connection pass through.
A manufacturer could attempt to mix the BRSKI-MASA traffic in with general traffic their site by hosting the MASA behind the same (set) of load balancers that the companies normal marketing site is hosted behind. This makes lots of sense from a straight capacity planning point of view as the same set of services (and the same set of Distributed Denial of Service mitigations) may be used. Unfortunately, as the BRSKI-MASA connections include TLS ClientCertificate exchanges, this may easily be observed in TLS 1.2, and a traffic analysis may reveal it even in TLS 1.3. This does not make such a plan irrelevant. There may be other organizational reasons to keep the marketing site (which is often subject to frequent redesigns, outsourcing, etc.) separate from the MASA, which may need to operate reliably for decades.

9.2.2. Manufacturers and Used or Stolen Equipment

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

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

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

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

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

4. There is a fourth case, if the manufacturer is providing protection against stolen devices. The manufacturer then has a responsibility to protect the legitimate owner against fraudulent claims that the equipment was stolen. Such a claim would cause the manufacturer to refuse to issue a new voucher. Should the device go through a deep factory reset (for instance, replacement of a damaged main board component, the device would not bootstrap.
5. Finally, there is a fifth case: the manufacturer has decided to end-of-line the device, or the owner has not paid a yearly support amount, and the manufacturer refuses to issue new vouchers at that point. This last case is not new to the industry: many license systems are already deployed that have significantly worse effect.

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

9.2.3. Manufacturers and Grey market equipment

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

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

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

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

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

9.2.4. Some mitigations for meddling by manufacturers

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

A manufacturer could provide a mechanism to manage the trust anchors and built-in certificates (IDevID) as an extension. This is a substantial amount of work, and may be an area for future standardization work.

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

In order to automatically find the new MASA, the mechanism describe in this document is to look for the MASA URL extension in the IDevID. A new owner could override this in their Registrar, or the manufacturer could provide a mechanism to update or replace the IDevID prior to sale.

Once the voucher trust anchor and the IDevID is replaced, then the device will no longer trust the manufacturer in any way. When a new owner performs a bootstrap, the device will point to a MASA that has been chosen, and will validate vouchers from this new entity.

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

Some manufacturers may wish to consider replacement of the IDevID as an indication that the device’s warantee is terminated. For others, the privacy requiments of some deployments might consider this a standard operating practice.
As discussed at the end of Section 5.8.1, new work could be done to use a distributed consensus technology for the audit log. This would permit the audit log to continue to be useful, even when there is a chain of MASA due to changes of ownership.

10. Security Considerations

This document details a protocol for bootstrapping that balances operational concerns against security concerns. As detailed in the introduction, and touched on again in Section 6, the protocol allows for reduced security modes. These attempt to deliver additional control to the local administrator and owner in cases where less security provides operational benefits. This section goes into more detail about a variety of specific considerations.

To facilitate logging and administrative oversight, in addition to triggering 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 registrar. This is mandated anyway because of the operational benefits of an informed administrator in cases where the failure is indicative of a problem. The registrar is RECOMMENDED to verify MASA logs if voucher status telemetry is not received.

To facilitate truly limited clients EST RFC7030 section 3.3.2 requirements that the client MUST support a client authentication model have been reduced in Section 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.

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

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

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

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

10.2. Freshness in Voucher-Requests

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

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

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

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

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

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

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

- Ongoing network monitoring for unexpected bootstrapping attempts by pledges.
Retrieval and examination of MASA log information upon the occurrence of any such unexpected events. Rm will be listed in the logs along with nonce information for analysis.

10.3. Trusting manufacturers

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

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

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

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

- Manually configuring each manufacturer’s trust anchor.
- A Trust-On-First-Use (TOFU) mechanism. A human would be queried upon seeing a manufacturer’s trust anchor for the first time, and then the trust anchor would be installed to the trusted store. There are risks with this; even if the key to name is validated using something like the WebPKI, there remains the possibility that the name is a look alike: e.g, clsco.com, ..
- scanning the trust anchor from a QR code that came with the packaging (this is really a manual TOFU mechanism)
- some sales integration process where trust anchors are provided as part of the sales process, probably included in a digital packing "slip", or a sales invoice.
- consortium membership, where all manufacturers of a particular device category (e.g, a light bulb, or a cable-modem) are signed by an certificate authority specifically for this. This is done by CableLabs today. It is used for authentication and authorization as part of TR-79: [docsisroot] and [TR069].
The existing WebPKI provides a reasonable anchor between manufacturer name and public key. It authenticates the key. It does not provide a reasonable authorization for the manufacturer, so it is not directly useable on its own.

11. Acknowledgements

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

12. References

12.1. Normative References

[I-D.ietf-anima-autonomic-control-plane]

[I-D.ietf-anima-grasp]


DOI 10.17487/RFC4519, June 2006,

DOI 10.17487/RFC4862, September 2007,

DOI 10.17487/RFC4941, September 2007,

DOI 10.17487/RFC5246, August 2008,

[RFC5272] Schaad, J. and M. Myers, "Certificate Management over CMS (CMC)", RFC 5272,
DOI 10.17487/RFC5272, June 2008,

DOI 10.17487/RFC5280, May 2008,

DOI 10.17487/RFC5386, November 2008,

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652,
DOI 10.17487/RFC5652, September 2009,

[RFC5660] Williams, N., "IPsec Channels: Connection Latching", RFC 5660,
DOI 10.17487/RFC5660, October 2009,
12.2. Informative References


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


Appendix A. IPv4 and non-ANI operations

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

A.1. IPv4 Link Local addresses

Instead of an IPv6 link-local address, an IPv4 address may be generated using [RFC3927] Dynamic Configuration of IPv4 Link-Local Addresses.
In the case that an IPv4 Link-Local address is formed, then the bootstrap process would continue as in the IPv6 case by looking for a (circuit) proxy.

A.2. Use of DHCPv4

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

Appendix B. mDNS / DNSSD proxy discovery options

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

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

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

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

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

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

Appendix C. 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:
The MUD extensions string "masa" is defined, and MUST be included in the extensions array of the mud container of a MUD file when this extension is used.
Appendix D. 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 reproducible examples the public and private keys for an example MASA and registrar are first listed.

D.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.

D.1.1. MASA key pair for voucher signatures

This private key signs vouchers:

-----BEGIN EC PRIVATE KEY-----
MIGkAgEBBDAgiRoYqKoEcfOfvRvmZ5P5Azns58tu7nSnIy7qFmCeiNo+BmbgMho
r61cU6ogwVagBwYFK4EEACHkZANIAATZAH3Rb2FvIbOntsv+xUuW35ofyNbcHJzJa
zOi2kWzFE1ByurKImNcNMFGirGnRXIGQwVCw5ICgJ8CuM3vV5ty9bf7KUL0kejz
TvV+5PV++e1kP9HQ8s3vqTaws2WwWTxI=
-----END EC PRIVATE KEY-----

This public key validates vouchers:

-----BEGIN CERTIFICATE-----
MIIBzzCCAVagAwIBAgIBATAKBggqhkjOPQQDAjBNMRIwEAYKCZImlZPyLGQBRGYC
Y2ExGTXBgoJkiaJk/IJaZAEZFGJzYW5kZwxtYW4xHDAaBgNBAMME1Vuc3RYydW5n
IEhpZ2h3YXkgQ0EwHhcNMTcwMzI2NTYxOTQwWhcNMTkwMzI2NTYxOTQwWjBHMRiw
EAYKCZImlZPyLGQBRGYCZExEToGTXBgoJkiaJk/IsZAEZFGJzYW5kZwxtYW4xfJAU
BgNBAMMDVuc3RYydW5nIE1BU0EwdaJABucghjikOQPQIIBgUrgQQAiNIAATZAH3R
b2FvIbOntvs+xUuW35ofyNbcHJzJaOic2kWzFE1ByurKImNcNMFGirGnRXIGQwVCw
5ICgJ8CuM3vV5ty9bf7KUL0kejzTvV+5PV++e1kP9HQ8s3vqTaws2WwWTxKjEDAO
MAwGAlUdeWEVB/wQCMAdaCgYIKoZIzj0EAWIDZwA2AIwGb0oyM0d0P6t3/LSP1jO
DuatEyMHy7WGo+IYTHC8kt7EyHOMCYReKT2+Gh/V/CLWzAjBPY60UMMTt1tsxJsJq
dMPUIFj+4Wz1ALoIb/JoA6M7r33pLqTrHRxHzVMEFWOKYUw=
-----END CERTIFICATE-----

D.1.2. Manufacturer key pair for IDevID signatures

This private key signs IDevID certificates:
-----BEGIN EC PRIVATE KEY-----
MIGkAgEBBDAgiRoYqKoEcfOvfvRvmZ5P5Azn58tuI7nSnIy70GfNcEiNo+BmbqMho
r61cU60gwVagBwYFK4E3EAckh2ANIAATZAH3r2FvJ0hts+vxuWw35ofyNbcHzjA
z0I2kWZFE1ByurKImNcNMFgIvrGnRX1XQgWCFw51CgJ8CuM3v5ty9bf7KUl0Kejz
Tv+5PV++elkP9H0Q83vqTAsw2WwWxI=
-----END EC PRIVATE KEY-----

This public key validates IDevID certificates:

-----BEGIN CERTIFICATE-----
MIIBzzCCAVagAwIBAgIBATAKBggqhkjOPQQDAjBNMRIwEAYKcZImizPzLGQBGRY
C Y2ExGTAxBgoJkiaJaJK/IsZAEZFg1zY5k5ZkxtYW4xHDAaBqNVBAMME1Vuc3rydW5n
IEHp6zh3XkgQ0EwHhcnNMCtwMzI2MTYxOTQwHhcnNMCtwMzI2MTYxOTQwWyjBHMR1w
EAYKcZImizPzLGQBGRYCY2ExGTAxBgoJkiaJaJK/IsZAEZFg1zY5k5ZkxtYW4x4fjAU
BgNVBAMMV0Lc3rydW5nIEH5Bz0EwEdjJQAQBCqCjhjJOPQIBBgUrgQQAigNiAAZAH3
r2FvJ0hts+vxuWw35ofyNbcHzjAz0I2kWZFE1ByurKImNcNMFgIvrGnRXXGqWCF
w51CgJ8CuM3v5ty9bf7KUl0KejzTv+5PV++elkP9H0Q83vqTAsw2WwWxI=
-----END CERTIFICATE-----

D.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-----
MhCACAQEEIF+obtiCYYYWm1fsZvrjWj0yFsCJwIFhopkmT/TULmX0aoGCCqGSM49
AwEPHQDqQ0ANWQ0zCMUJ0P0nRtfeBCDjKLWfeMGCFdIv6F0z4ifM1ujMBec/g
6W/P6boTmyTGDEo/H8WcUeRgPbneK8sg==
-----END EC PRIVATE KEY-----

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

-----BEGIN CERTIFICATE-----
MIIBjCCAT0gWAgIBAgIBAzAKBggqgkhjJOPQODA5BOMRIwEAYKcZImizPzLGQBGRY
C Y2ExGTAxBgoJkiaJaJK/IsZAEZFg1zY5k5ZkxtYW4xHTAbBqNVBAMME1Vuc3rydW5n
IE2vWDsO5YW1uIENMBI4XDETE3MDkwNTAxMTIONVoXDETE5MDkwNTAxMTIONVoWqE
MBAGCgmSJomT8xwARKbAAMNHMrkwFvYXZCZImizPzLGQBGRYJC2FoUGVSbWFMRIw
EAYVQ0DQDLab2NhbcgCh3vWMTATBgcqhjkJOPQIBBgqghjkJOPQIBBggghjkJOPQIB
wNCQA1ZA7Nh0xSM/Q2u194F3Q0W0xZ4aAlV0/0vTPoHJ8zW6M0F5z+Dp88/pbOHMS2U6H/
wpfPrSlumd4ryyowwCzaJdBGnNVRME4JAAAmoGCCqGSM49BMDA2AKMYCMQ3C
/iTQJ3eYVycbXhbmzr6p4t3QC6qjiEY2jDkX062nnnUfVKyara3F30AkKSE
MQO1Z9efThLdDk3teY/r7V777Xaj6nYCmdDCR54TrSFNLgxt1fYPOM0fYpYR
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 Fountain CA
      Validity
         Not Before: Sep 5 01:12:45 2017 GMT
         Not After: Sep 5 01:12:45 2019 GMT
      Subject Public Key Info:
         Public Key Algorithm: id-ecPublicKey
         Public-Key: (256 bit)
         pub:
            3:3e:
            9:ba:
            e9:9d:e2:bc:b2
      ASN1 OID: prime256v1
      X509v3 extensions:
         X509v3 Basic Constraints:
            CA:FALSE
         Signature Algorithm: ecdsa-with-SHA384
            5b:
            b6:
            02:
            c3:
            4b:
D.1.4. Pledge key pair

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

-----BEGIN EC PRIVATE KEY-----
MHcCAQEEIL+ue8PQcN+M7LFBGPsfompYyobI/rsoHnTb2a+0hO+8joAoGCCqGSM49
AwEHoUQDQgAEumBVAoD1X87WyME8CJToyt9NWy6sYW0DTbjjJIn79pgr7ALa/\Y8p
r70WpKlSai1UeeFw7e+1CztP12+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-----
MIICmjCCAbegAwIBAgIBDDAKBggqhkjOPQDAjBNMRIwEAYKCZiZPyLGQBGRYC
Y2ExGTAXBgoJkiaJk/IszAZ2FglzYW5kZmxwYW4xHDAaBgNVAMME1Vuc3ryW5n
IEhp2zh3YXkgQ0EwIBcNMTc5MDM1MjUyWhgPMj5OETEyMzEwMDAwMDk0MEMs
EjAQBgoJkiaJk/IszAZ2FgljYTE2MBCGCgsmZomT8ixxArkWCCXhbmRllbG1bjeAa
MBgGAIUEAwRMHATRArTUrjITMDAtMDIwWTATBgqckjOPQ1IBggqhkjOPQMB
BwNCAARpj5i0dUaUnR2u8wMRwgkNupNbhNM7m1n0mj+0KJZjPiqID+trPSobt
uIdpPFGZ8+n/niUveqwoxYEBPboGHMIGEMB0GA1udDgQWBQdMDR2hthFQmzz6
E7YVxzkL7ZDKjAJBGNVHRMEAJAAMCsGAIUdEQCMCgIAYJKwYBCqGChlBBoBM
ETAwLUQwLUIU7yLTAwZ1AyMCSCsGAQQBgwu5ASAgQeDBxodHRwczovL2hpZ2h3
YXkuc2Fu2GVsbWFuemNhMA0GCCqGSM49BAMCA2kAMGYCQDhJIn+eawU/EDqoM
SGV0VHR7ui8cBH7hv5Y580nBs8bPn60k/+IzvEUetMzICMQCr1uxvdYeKq7mb
RXCR42CJSw67fJ7jyXZbcUS1r+3wBT2+1WgzzPDrYB5Ab7sAw=
-----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 provide 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 Highway CA
  Validity
  Not After : Dec 31 00:00:00 2999 GMT
  Subject: DC=ca, DC=sandelman, CN=00-D0-E5-F2-00-02
  Subject Public Key Info:
    Public Key Algorithm: id-ecPublicKey
    Public-Key: (256 bit)
      pub:
          c:47:
        4:28:
        d:b8:
      0:ca:
        86:08:f0:13:db
  ASN1 OID: prime256v1
X509v3 extensions:
  X509v3 Subject Key Identifier:
  :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
  0c:
  34:
  02:
  89:
  be:
    95:68:20:cc:f0:d1:81:80:79:00:16:fb:b0:0c
D.2. Example process

RFC-EDITOR: these examples will need to be replaced with CMS versions once IANA has assigned the eContentType in [RFC8366].

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

The ASN1 decoding of the artifact:

```plaintext
0:d=0  hl=4 l=1820 cons: SEQUENCE
  4:d=1  hl=2 l=   9 prim: OBJECT            :pkcs7-signed

```
Data

15:d=1  h1=4  l=1805  cons:  cont [ 0 ]
19:d=2  h1=4  l=1801  cons:  SEQUENCE
23:d=3  h1=2  l= 1  prim:  INTEGER           :01
26:d=3  h1=2  l= 5  cons:  SET
28:d=4  h1=2  l= 3  cons:  SEQUENCE
30:d=5  h1=2  l= 9  prim:  OBJECT           :sha256
41:d=5  h1=2  l= 0  prim:  NULL
43:d=3  h1=4  l= 782  cons:  SEQUENCE
47:d=4  h1=2  l= 9  prim:  OBJECT           :pkcs7-data
58:d=4  h1=4  l= 767  cons:  cont [ 0 ]
62:d=5  h1=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"":"MIIBrjCA
AwIBAgIBAzAKBggqhkjOPQDAQBerOMIRiEAYKCIiZPyLGQBGRYCY2ExGTAX
BgoJkiaJk/Iz2AEZFgLzYW5kZWxtYW4xHTAbBgNVBAMMFVu3RydvW5nI3eVz
dW50YWluENNBMB3MDkwNTE0MTAxMTI0NVowQzESMBACgmgSjonT8ixkArK
wAhMbNRIkkwFwYKCIiZPyLGQBGRYJc2FuZGVsbWF0MRAwEAYDVQQDDAlsb
2NhbGhvc2QzQWTATBgcqghkjOPQIBbgghkjoPOPMBBwNCAAQ1ZAZw0xSM/Q
2u194FzQMktZ94waAIV0i/oVTPgoj8zw6wF5z+Dpbb/p
ubohM2Z0U6H/w/ApR6svum4ryy0w0CzAJBgNVHRMEAjAAMAoGCCqGSM49
BAMDA2kAMGYCMQC3/iTQJ3evYYcgbXhbnvzr64t3QC6qjIeY2jku62nuNi
fVKtyaar3f30AiK5ECMQD129efbTLbdtDk3teCY/r7D7v77XaJ6nYcmdDCR
54TrSFNlgxvtllyFMy+0fYrYRc3o="}}
829:d=3  h1=4  l= 566  cons:  cont [ 0 ]
833:d=4  h1=4  l= 562  cons:  SEQUENCE
837:d=5  h1=4  l= 439  cons:  SEQUENCE
841:d=6  h1=2  l=  3  cons:  cont [ 0 ]
843:d=7  h1=2  l=  1  prim:  INTEGER           :02
846:d=6  h1=2  l=  1  prim:  INTEGER           :0C
849:d=6  h1=2  l= 10  cons:  SEQUENCE
851:d=7  h1=2  l=  8  prim:  OBJECT           :ecdsa-with-S
856:d=7  h1=2  l=  7  prim:  OBJECT           :domainCompon
861:d=6  h1=2  l=  77  cons:  SEQUENCE
863:d=7  h1=2  l=  18  cons:  SET
865:d=8  h1=2  l=  16  cons:  SEQUENCE
867:d=9  h1=2  l= 10  cons:  OBJECT           :domainCompon
ent
879:d=9  h1=2  l=  2  prim:  IA5STRING           :ca
883:d=7  h1=2  l=  25  cons:  SET
885:d=8  h1=2  l=  23  cons:  SEQUENCE
887:d=9  h1=2  l=  2  prim:  OBJECT           :domainCompon
ent
899:d=9  h1=2  l=  9  prim:  IA5STRING           :sandelman
910:d=8  h1=2  l=  26  cons:  SET
912:d=8  h1=2  l=  26  cons:  SEQUENCE
914:d=9  h1=2  l=  3  prim:  OBJECT           :commonName
919:d=9  hl=2  l= 19 prim: UTF8STRING  :Unstrung Highway CA
940:d=6  hl=2  l= 32 cons: SEQUENCE
942:d=7  hl=2  l= 13 prim: UTCTIME  :171012135252Z
957:d=7  hl=2  l= 15 prim: GENERALIZEDTIME :299912310000002
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  :domainComponent
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  :domainComponent
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-ecPublicKey
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 Subject 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]:3000
1190:d=8  hl=2  l= 43 cons: SEQUENCE
1192:d=9  hl=2  l=  3 prim: OBJECT  :X509v3 Subject Alternative Name
1197:d=9  hl=2  l= 36 prim: OCTET STRING  [HEX DUMP]:3000
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
1C68747470733A2F2F68696766776617361626672616E64656C6F672E6361
1280:d=5 hl=2 l= 10 cons: SEQUENCE
1282:d=6 hl=2 l=  8 prim: OBJECT             :ecdsa-with-SHA256
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             :domainComponent
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             :domainComponent
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 Highway 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           :171012175430Z
1568:d=6 hl=2 l= 47 cons: SEQUENCE
1570:d=7 hl=2 l=  9 prim: OBJECT             :messageDigest
1581:d=7 hl=2 l=  34 cons: SET
1583:d=8 hl=2 l=  32 prim: OCTET STRING        [HEX DUMP]:FE
7D72E29500F90A38E95021215FD6D40B1629B9598177DC054AE0F9C8B6
The JSON contained in the voucher request:

```
{"ietf-voucher-request:voucher":{"assertion":"proximity","created-on":"2017-09-01","serial-number":"00-D0-E5-F2-00-02","nonce":"Dss99sBr3pNMOACe-LYY7w","proximity-registrar-cert":"MIIbrjCCATQgAwIBAgIBAzAKBggojibZgJIBA-blbgjQQDDAzBOMRIwEAYKCI...","signature":"MIIBrjCCATQgAwIBAgIBAzAKBggojibZgJIBA-blbgjQQDDAzBOMRIwEAYKCI..."}}
```
D.2.2. Registrar to MASA

As described in Section 5.5 the registrar will sign a registrar voucher-request, and will include pledge’s voucher request in the prior-signed-voucher-request.
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 ]
    23:d=2  hl=2 l=  1 prim: INTEGER           :0
```

file: examples/parboiled_vr_00-D0-E5-F2-00-02.pkcs
| 26:d=3  | hl=2  | l= 15 | cons: SET               |
| 28:d=4  | hl=2  | l= 13 | prim: OBJECT            |
| 41:d=5  | hl=2  | l=  0 | prim: NULL              |
| 43:d=3  | hl=2  | l=2638| cons: SEQUENCE          |
| 47:d=4  | hl=2  | l=  9 | prim: OBJECT            |
| 58:d=4  | hl=2  | l=2623| cons: CONT [ 0 ]        |

26:d=3 l= 15 cons: SET
28:d=4 hl=2 l= 13 prim: OBJECT :sha256
41:d=5 hl=2 l=  0 prim: NULL
43:d=3 hl=2 l=2638 cons: SEQUENCE
47:d=4 hl=2 l=  9 prim: OBJECT :pkcs7-data
58:d=4 hl=2 l=2623 cons: CONT [ 0 ]

OCET STRING: {"ietf-vouch.er-request:voucher":{"assertion":"proximity","created-on":"2017-09-15T00:00:00.000Z","serial-number":"JADA123456789","no
nce":acbd1234},"prior-signed-voucher-request":"MIIHHQYJKoZI hvcNAQCoIHDwCwrgAwC4x3iXzBqA1uBqkXwzYmEgEFAADCCw4GCsQGSIbD
DQAEhaAACA8eGgj79yEpZKXmLc8vK2zd3WmXYvzFvdpbV3xJvaGyVipj7
Imfc2ZyvGdVb1I6Inmyb3hpW0eSIsImNy2WF0Qtb24i0iyMDE3LTAT
LTAiiwic2ZyvAFsWL51wJkli6Ji7awLUQwLUU1UYLYTAtiLAtiwibm9u
Y2Ui0ioJEC3mOXNCcjNwTk1PQUNLUXzTcd3iwicHJveGltalXR5LXJ22z
DhJhciiljXJ0iuiUJnQjQoNBBvE9nVQxjQkFmSNJbekFmQmndcwrhak9
UVFEOFXcP0I1SSXDFqVI1Qi3pBw14UAHM1RFClJRZ1kryRHvHEFVYQmdvSm
tvpYUpfrL0lzWFKFWkZnbHprZvRsWId4dFLXNHhIVEFQjmdV0kVJBUTU1GR1YzNs
ewRWNWSJ5XRPv2ZfcMI1XVHJVJ5U5TU1U0ERUTNNGr3T31RBeE1USTROYm9Y
RFRFNU1Ea3d0VFEFATVRJME5W3d3dRekVTTJBR0NhNvKBE21U0G14a0Fsd1b
buUSOtvJrdZ3WUIWtdKltuOxUHJHUI1KyzJGdpVhvNiVO2V1TVdJ0DvB
WURWUVUFEREs2IYtmhiRr2hZyNRdldQVRC22XNaGtqT1BRSUJCZ2dxaGtq
T1BRKTUc0d5QDFRMVPbN9053rHmH5TS9RmC3XOtRGeIFNa3aO3YVufvJbp
L29W6F8nT041w2C7yC80mXvRHi1Qc9WdhWhvYxpNWjBVNgvy2ZzCF1c23Zs
dW1kNH5j65WE93Hh1eDKFkFQhHQoKvstUVAV1kBFUvVR0NCDucITTQqFqFNREEy
a0FNrL1ITVFDMy9pVFIFkM2ZVW1j2YJSafeJtenJwNjR0MFDmFqjSVV2Zmpr
RhgWnjJudoPz12LDHh1YJXhM0yMFJea0utrUNnuRUpmlz1ujtgJkEer
M3R1Y1kvcq3Vj3cWFKNm5ZQ21kKRENSNTRUc1NGtXenne20MWx5Rk0zMGZz
cfISyzxNvPSj9faCCCAjyYyriMIIb6dAaCAGEMMAoGCCqGSM49BAMCMExO
EjAQBojKaiyJkrrIzAESzFqjJYET2MBcCGcSmJot8ixkARKwCNhmb1bGih
bbCmcB0AEuAWttV52dHJlbmcqSGLnaHdheSBQTAgFw0x4Z3WtMTIXmZuY
NTJaGAbYotK5MTIzMTAMdAowFoszESEMBAGCgmsJomT8ixkARKwNnhRMrk
FwYKatzCmi2PylQGQbRyYjC2fuZGvsbWfuMrOwGAYDVQDDEuwC1EC1M1FNSG
MioWCC0mWjZBBMmBgqS3M9M94qEGCCqGSM49AwEA0IAEmnnLIH1VpSdha7
zAHCXWQ26kls0ZuWbfAS7QoMN8ioQoP62s+NNKhu24h21E982NYtF+ch3S
D6Khjngw9jjujgycqW9HQMDV00BYEBF8oxF6G2EcvBPcTThFvO9QtdkMq
MAGKAIuKkECwQ0MMaKwYdYRoRBCRcwIaqBgrkrgEEAYzUuGgEwUmAdRDA
RTUtRjItMtAdMlXwKvYWBBACG71ICBG4HMGdHbZoi8vaGlnaHdheS05
YW5kZwXvY4u2EcqY1izqiZj0EawIdaQAw2iqXaoEn530q5gdvT979mqX
as959ydhu6Jzxxwlu4GifrLsGCsZux2frsT/4j08RR60zMqIXAkw97G91h4QR
uzTfCeHJkimDrnm8t8Kv7aFPb6vaCDM8NGBgKAvuWVDGCAaYW
ggAGiEBMFq1WzTTESMBAGCgjsJot8ixkARKwNnhMrkFwYKatzCmi2PylQGB
GryScf2uZGvsbWfuMrOwGAYDVQDDEuwC1EC1M1FNSG
MioWCC0mWjZBBMmBgqS3M9M94qEGCCqGSM49AwEA0IAEmnnLIH1VpSdha7
zAHCXWQ26kls0ZuWbfAS7QoMN8ioQoP62s+NNKhu24h21E982NYtF+ch3S
D6Khjngw9jjujgycqW9HQMDV00BYEBF8oxF6G2EcvBPcTThFvO9QtdkMq
MAGKAIuKkECwQ0MMaKwYdYRoRBCRcwIaqBgrkrgEEAYzUuGgEwUmAdRDA
RTUtRjItMtAdMlXwKvYWBBACG71ICBG4HMGdHbZoi8vaGlnaHdheS05
YW5kZwXvY4u2EcqY1izqiZj0EawIdaQAw2iqXaoEn530q5gdvT979mqX
as959ydhu6Jzxxwlu4GifrLsGCsZux2frsT/4j08RR60zMqIXAkw97G91h4QR
uzTfCeHJkimDrnm8t8Kv7aFPb6vaCDM8NGBgKAvuWVDGCAaYW
}
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-ecPublicKey
2911:d=8  hl=2 l=  8 prim: OBJECT            :prime256v1
2921:d=7  hl=2 l=  11 cons: SEQUENCE
2933:d=8  hl=2 l=  9 cons: SEQUENCE
2955:d=9  hl=2 l=  3 prim: OBJECT            :X509v3 Basic
Constraints
3002: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-SHA384
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            :domainComponent
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            :domainComponent
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
3223: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
D.2.3. MASA to Registrar

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

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

The ASN1 decoding of the artifact:
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-voucher":{"assertion":"logged","created-on":"2017-10-12T13:54:31.439-04:00","serial-number":"00-D0-E5-F2-00-02","nonce":"Dss9sBr3pNMOAcE-LYY7w","pinned-domain-cert":"MIIBrjCCATogAwIBAgIBAAIBAwIBAfIBAgIwEAYKCZIImzPyLQBGCRY2ExGT
AXBgOJKjIAk/IzZAEZFglzYW5kZWxvYXTABQgNBAMMFFVuc3RydW5nIEZvdW50YWRkIE8wMDk0NTMwMDk0MDAgMA0GCSqGSM49BAMDA2IAGYCMQCM0C1gZw0x
SM/Q2u194FzQMktZ94waAIV0i/oVTPgOJz8zW6mW5z+Dpb8/puuhObJMZOu6h/wApR6svlumd4ryyow0wCzaAJBGvNHRMEAjAAMaOGCqGSM
49BAMDA2kAGMYCMQC3/1gJ3cyyYcgbXhmzrp64t3C6qjIeY2jDx062nuNi
ifVKtyaara3F3OAIkKSECMQDj129efbTLbdtDk3tecY/rD7V77XaJ6nYCmD
CR54r3FNlQGxtt17lyFM+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: OCTET STRING :ecdsa-with-SHA256

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

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

901:d=9  hl=2 l=  9 prim: IA5STRING     :sandelman
905:d=7  hl=2 l=  25 cons: SET
907:d=8  hl=2 l=  23 cons: SEQUENCE
909:d=9  hl=2 l=  10 prim: OBJECT       :domainCompon

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 :domainComponent
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 :domainComponent
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-ecPublicKey
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 Constrains
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 :domainComponent
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  :domainComponent
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=  19 prim: OBJECT  :commonName
1373:d=9 hl=2 l=  19 prim: UTF8STRING  :Unstrung High
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=  19 cons: OBJECT  :sha256
1410:d=6 hl=2 l=  13 cons: 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=  24 cons: SEQUENCE
1430:d=8 hl=2 l=  24 cons: OBJECT  :pkcs7-data
1441:d=6 hl=2 l=  34 cons: 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  :messageDigests
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 Capabilities
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-SHA256
1655:d=5 hl=2 l=103 prim: OCTET STRING [HEX DUMP]:
502310087389DFC090D8EDB6948FD6B7E5369A5D1EC8B7DB8676F935C9C
6E79EC2727CFFFD8D5DBF937F70EC4E1BF876ED343B3023063C8E9E69244
8A1EC3D1D8996E03973AC28BEE3C49CD28FDF675A1867852861073244C89
DBAAE8A90BC35FF92BD43E9

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.Behringer@gmail.com

Steinthor Bjarnason
Arbor Networks
Email: sbjarnason@arbor.net

Kent Watsen
Juniper Networks
Email: kwatsen@juniper.net