ANIMA WG M. Pritikin
Internet-Draft Cisco
Intended status: Standards Track M. Richardson
Expires: January 22, 2020 Sandelman
M. Behringer

K. Watsen
Watsen Networks
July 21, 2019

Bootstrapping Remote Secure Key Infrastructures (BRSKI)
draft-ietf-anima-bootstrapping-keyinfra-24

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 certificates, 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.

Copyright Notice

Table of Contents

1. Introduction 4
1.1. Prior Bootstrapping Approaches 6
1.2. Terminology 7
1.3. Scope of solution 10
1.3.1. Support environment 10
1.3.2. Constrained environments 10
1.3.3. Network Access Controls 11
1.3.4. Bootstrapping is not Booting 11
1.4. Leveraging the new key infrastructure / next steps . . . 11
1.5. Requirements for Autonomic Network Infrastructure (ANI) devices 12
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 22
2.7. Cloud Registrar 22
2.8. Determining the MASA to contact 23
3. Voucher-Request artifact 23
3.1. Nonceless Voucher Requests 24

3.2. Tree Diagram 24
3.3. Examples 25
3.4. YANG Module 26
4. Proxying details (Pledge - Proxy - Registrar) 29
4.1. Pledge discovery of Proxy 30
4.1.1. Proxy GRASP announcements 32
4.2. CoAP connection to Registrar 33
4.3. Proxy discovery and communication of Registrar 33
5. Protocol Details (Pledge - Registrar - MASA) 34
5.1. BRSKI-EST TLS establishment details 36
5.2. Pledge Requests Voucher from the Registrar 37
5.3. Registrar Authorization of Pledge 38
5.4. BRSKI-MASA TLS establishment details 38
5.5. Registrar Requests Voucher from MASA 39
5.5.1. MASA renewal of expired vouchers 41
5.5.2. MASA verification of voucher-request signature consistency 41
5.5.3. MASA authentication of registrar (certificate) . . . 41
5.5.4. MASA revocation checking of registrar (certificate) . 42
5.5.5. MASA verification of pledge prior-signed-voucher-request 42
5.5.6. MASA pinning of registrar 42
5.5.7. MASA nonce handling 42
5.6. MASA and Registrar Voucher Response 43
5.6.1. Pledge voucher verification 45
5.6.2. Pledge authentication of provisional TLS connection . 46
5.7. Pledge BRSKI Status Telemetry 47
5.8. Registrar audit log request 48
5.8.1. MASA audit log response 49
5.8.2. Registrar audit log verification 50
5.9. EST Integration for PKI bootstrapping 51
5.9.1. EST Distribution of CA Certificates 52
5.9.2. EST CSR Attributes 52
5.9.3. EST Client Certificate Request 53
5.9.4. Enrollment Status Telemetry 53
5.9.5. Multiple certificates 54
5.9.6. EST over CoAP 54
6. Clarification of transfer-encoding 55
7. Reduced security operational modes 55
7.1. Trust Model 55
7.2. Pledge security reductions 56
7.3. Registrar security reductions 57
7.4. MASA security reductions 58
8. IANA Considerations 59
8.1. Well-known EST registration 59
8.2. PKIX Registry 59
8.3. Pledge BRSKI Status Telemetry 59

8.4. DNS Service Names 59
8.5. MUD File Extension for the MASA 60
9. Applicability to the Autonomic
Control Plane 60
10. Privacy Considerations 61
10.1. MASA audit log 61
10.2. What BRSKI-MASA reveals to the manufacturer 62
10.3. Manufacturers and Used or Stolen Equipment 63
10.4. Manufacturers and Grey market equipment 64
10.5. Some mitigations for meddling by manufacturers 65
11. Security Considerations 66
11.1. DoS against MASA 67
11.2. Freshness in Voucher-Requests 68
11.3. Trusting manufacturers 69
11.4. Manufacturer Maintenance of trust anchors 70
12. Acknowledgements 71
13. References 71
13.1. Normative References 72
13.2. Informative References 74
Appendix A. IPv4 and non-ANI operations 77
A.1. IPv4 Link Local addresses 78
A.2. Use of DHCPv4 78
Appendix B. mDNS / DNSSD proxy discovery options 78
Appendix C. MUD Extension 79
Appendix D. Example Vouchers 81
D.1. Keys involved 81
D.1.1. MASA key pair for voucher signatures 81
D.1.2. Manufacturer key pair for IDevID signatures 81
D.1.3. Registrar key pair 82
D.1.4. Pledge key pair 84
D.2. Example process 85
D.2.1. Pledge to Registrar 86
D.2.2. Registrar to MASA 89
D.2.3. MASA to Registrar 94
Authors' Addresses 98

1. Introduction

BRSKI provides a solution for secure zero-touch (automated) bootstrap of new (unconfigured) 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 9 explains the details applicability for this the ACP usage.

This document describes how pledges discover (or are 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" (MASA) and passes to the pledge to answer points 3 and 4.

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

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

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

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

1.1. Prior Bootstrapping Approaches

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

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

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

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

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

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

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

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

o 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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

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+ [RFC7228]) devices operating on a non-Challenged network. The entire solution as described here is not intended to be useable as-is by constrained devices operating on challenged networks (such as 802.15.4 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 securely deploy functionalities across the domain, e.g:

o Device management.

o Routing authentication.

o Service discovery.

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

1.5. Requirements for Autonomic Network Infrastructure (ANI) devices

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

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

The pledge must perform discovery of the proxy as described in Section 4.1 using GRASP [I-D.ietf-anima-grasp] M_FLOOD announcements.

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

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

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

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

2. Architectural Overview

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

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

Figure 1

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

Figure 2: pledge state diagram

State descriptions for the pledge are as follows:

1. Discover a communication channel to a registrar.

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

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

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

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

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

This specification details integration with EST enrollment so that pledges can optionally obtain a locally issued certificate, although any 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 enables:

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

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

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

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

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

Section 7.2.13 of [IDevID] discusses keyUsage and extendedKeyUsage extensions in the IDevID certificate. Any restrictions included reduce the utility of the IDevID and so this specification RECOMMENDS that no key usage restrictions be included. Additionally, [RFC5280] section 4.2.1.3 does not require key usage restrictions for end entity certificates.

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]:

o 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)

o 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.

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

2.3.2. MASA URI extension

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

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 IRI provides the authority

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

As explained in [RFC5280] section 7.4, a complete IRI SHOULD be in this extension, including the scheme, iauthority, and ipath. As a consideration to constrained systems, this MAY be reduced to only the iauthority, in which case a scheme of "https://" ([RFC7230] section 2.7.3) and ipath of "/.well-known/est" is to be assumed, as explained in Section 5.

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

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

The new extension is identified as follows:

```
<CODE BEGINS>

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

DEFINITIONS IMPLICIT TAGS ::= BEGIN

-- EXPORTS ALL --

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

id-pe
FROM PKIX1Explicit-2009
{ iso(1) identified-organization(3) dod(6) internet(1)
security(5) mechanisms(5) pkix(7) id-mod(0)
id-mod-pkix1-explicit-02(51) } ;
MASACertExtensions EXTENSION ::= { ext-MASAURL, ... }
ext-MASAURL EXTENSION ::= { SYNTAX MASAURLSyntax
IDENTIFIED BY id-pe-masa-url }

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

MASAURLSyntax ::= IA5String

END

<CODE ENDS>
```

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

2.4. Protocol Flow

A representative flow is shown in Figure 3:

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

Figure 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. A pledge MAY ignore all time stamps in the voucher and in the certificate validity periods if it does not know the current time.

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

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

2.6.2. Infinite Lifetime of IDevID

[RFC5280] explains that long lived pledge certificates "SHOULD be assigned the GeneralizedTime value of 99991231235959Z". 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 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 (IDevID) 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 signed 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.

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

3.1. Nonceless Voucher Requests

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

3.2. Tree Diagram

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

```
module: ietf-voucher-request

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

3.3. Examples

This section provides voucher-request examples for illustration purposes. The contents of the certificate have been elided to save space. For detailed examples, see Appendix D.2. These examples conform to the encoding rules defined in [RFC7951].

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


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


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

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


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


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


3.4. YANG Module

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

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

  namespace
    "urn:ietf:params:xml:ns:yang:ietf-voucher-request";
  prefix "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 8366: Voucher Profile for Bootstrapping Protocols";
  }

  organization
   "IETF ANIMA Working Group";

  contact
   "WG Web:   <http://tools.ietf.org/wg/anima/>
    WG List:  <mailto:anima@ietf.org>
    Author:   Kent Watsen
              <mailto:kwatsen@juniper.net>
    Author:   Michael H. Behringer
              <mailto:Michael.H.Behringer@gmail.com>
    Author:   Steinthor Bjarnason
              <mailto:sbjarnason@arbor.net>
    Author:   Max Pritikin
              <mailto:pritikin@cisco.com>
    Author:   Michael Richardson
              <mailto:mcr+ietf@sandelman.ca>";

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

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

    Copyright (c) 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;
      }

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

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

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

        leaf prior-signed-voucher-request {
          type binary;
          description
```

```
            "If it is necessary to change a voucher, or re-sign and
             forward a voucher that was previously provided along a
             protocol path, then the previously signed voucher SHOULD be
             included in this field.

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

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

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

}

<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 MUST NOT assume any specific TLS version.

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

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

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

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

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

4.1. Pledge discovery of Proxy

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

To discover the proxy the pledge performs the following actions:

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

and discovery of proxy will be by Link-Local mechanisms. IPv4 methods are described in Appendix A

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

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

While the GRASP M_FLOOD mechanism is passive for the pledge, the 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. Connection attempts to different proxies SHOULD be sent with an interval of 3 to 5s. The pledge SHOULD continue to listen to for additional GRASP M_FLOOD messages during the connection attempts.

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, ""],
            [O_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,
                                       objective-value]

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

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

Figure 6c: AN_Proxy CDDL

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

The $transport-proto above indicates the method that the pledge-proxy-registrar will use. The TCP method described here is

mandatory, and other proxy methods, such as CoAP methods not defined in this document are optional. Other methods MUST NOT be enabled unless the Join Registrar ASA indicates support for them in it's own announcement.

4.2. CoAP connection to Registrar

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

4.3. Proxy discovery and communication of Registrar

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

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

The M_FLOOD is formatted as follows:

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

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

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

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

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

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

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

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

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

5. Protocol Details (Pledge - Registrar - MASA)

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

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

The communication channel between the registrar and MASA is similarly described as extensions to EST within the same "/.well-known" tree.

For clarity this channel is referred to as "BRSKI-MASA". (See Figure 1).

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

BRSKI uses existing CMS message formats for existing EST operations. BRSKI uses JSON [RFC8259] for all new operations defined here, and voucher formats.

While EST section 3.2 does not insist upon use of HTTP 1.1 persistent connections, ([RFC7230] section 6.3) BRSKI-EST connections SHOULD use persistent connections. The intention of this guidance is to ensure the provisional TLS state occurs only once, and that the subsequent resolution of the provision state is not subject to a MITM attack during a critical phase.

Summarized automation extensions for the BRSKI-EST flow are:

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

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

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

o 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.

o 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:

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

o This extends the informal set of "identifer type" values defined in [RFC6125] to include a SERIALNUM-ID category of identifier that can be included in a certificate and therefore that can also be used for matching purposes. As noted in that document this is not

a formal definition as the underlying types have been previously defined elsewhere. 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 performs input validation of all data received until a voucher is verified as specified in Section 5.6.1 and the TLS connection leaves the provisional state. Until these operations are complete the pledge could be communicating with an attacker.

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

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

5.2. Pledge Requests Voucher from the Registrar

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

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

The pledge voucher-request Content-Type is:

application/voucher-cms+json 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]. This voucher media type is defined in [RFC8366] and is also used for the pledge voucher-request. The pledge SHOULD sign the request using the Section 2.3 credential.

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

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

The pledge populates the voucher-request fields as follows:

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

nonce: The pledge voucher-request MUST contain a cryptographically strong random or pseudo-random number nonce. (see [RFC4086]) Doing so ensures Section 2.6.1 functionality. The nonce MUST NOT be reused for multiple bootstrapping attempts. (The registrar voucher-request MAY omit the nonce as per Section 3.1)

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 3.3 Example 1.

The registrar validates the client identity as described in EST [RFC7030] section 3.3.2. 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 the HTTP 404 error code. If the voucher-request is in an unknown format, then an HTTP 406 error code is more appropriate. A situation that could be resolved with administrative action (such as adding a vendor to a whitelist) MAY be responded with an 403 HTTP error code.

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

5.4. BRSKI-MASA TLS establishment details

The BRSKI-MASA TLS connection is a 'normal' TLS connection appropriate for HTTPS REST interfaces. The registrar initiates the connection and uses the MASA URL obtained as described in Section 2.8. The mechanisms in [RFC6125] SHOULD be used authentication of the MASA. Some vendors will establish explicit (or private) trust anchors for validating their MASA; this will typically done as part of a sales channel integration. Registars SHOULD permit trust anchors to be pre-configured on a per-vendor basis.

The primary method of registrar "authentication" by the MASA is detailed in Section 5.5. As detailed in Section 11 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 7.4)

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

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

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

The registrar populates the voucher-request fields as follows:

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

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

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

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

prior-signed-voucher-request: The signed pledge voucher-request 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 and appropriate 'Accept header field' values from the physical device labeling or from the sales channel (out-of-scope for this document).

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

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

The MASA verifies that the registrar voucher-request is internally consistent but does not necessarily authenticate the registrar certificate since the registrar 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 leafs 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. If the registrar voucher-request contains a nonce, and the prior-signed-voucher-request exists, then the MASA MUST verify that the nonce is consistent. (Recall from above that the voucher-request might not contain a nonce, see Section 5.5 and Section 5.5.3).

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). Registrar evaluation of the voucher itself is purely for transparency and audit purposes to further inform log verification (see Section 5.8.2) and therefore a registrar could accept future voucher formats that are opaque to the registrar.

If the voucher-request is successful, the server (MASA responding to registrar or registrar responding to pledge) response MUST contain an HTTP 200 response code. The server MUST answer with a suitable 4xx or 5xx HTTP [RFC7230] error code when a problem occurs. In this case, the response data from the MASA MUST be a plaintext human-readable (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 (blinked LED etc) during this wait cycle if mechanisms for this are available. To prevent an attacker registrar from significantly delaying bootstrapping the pledge MUST limit the 'Retry-After' time to 60 seconds. Ideally the pledge would keep track of the appropriate Retry-After header field values for any number of outstanding registrars but this would involve a state table on the pledge. Instead the pledge MAY ignore the exact Retry-After value in favor of a single hard coded value (a registrar that is unable to complete the transaction after the first 60 seconds has another chance a minute later). A pledge SHOULD only maintain a 202 retry-state for up to 4 days, which is longer than a long weekend, after which time the enrollment attempt fails and the pledge returns to discovery state.

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

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

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

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

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

The voucher response format is as indicated in the submitted Accept header fields or based on the MASA's prior understanding of proper format for this Pledge. Only the [RFC8366] "application/voucher-cms+json" media type is defined at this time. The syntactic details of vouchers are described in detail in [RFC8366]. Figure 8 shows a sample of the contents of a voucher.


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


Figure 8: An example voucher

Figure 1: An example voucher

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. This figure is illustrative, for an example, see Appendix D.2

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

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

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

Whenever a voucher is issued the MASA MUST update the audit log 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(s) associated with the manufacturer's MASA (this is likely included in the pledge's firmware). Management of the manufacturer installed trust anchor(s) is out-of-scope of this document; this protocol does not update these trust anchor(s).

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

The pledge MUST verify 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 7.2).

The pledge MUST be prepared to parse and fail gracefully from a voucher response that does not contain a 'pinned-domain-cert' field. 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 inconsistently (an inconsistently rolled registrar key, or some other mis-configuration). The inconsistently could also be the result an active MITM attack on the EST connection.

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

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

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

5.7. Pledge BRSKI Status Telemetry

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

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

The posted data media type: application/json

The client HTTP POSTs the following to the server at the EST well known URI "/voucher_status".

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

The Status field indicates if the voucher was acceptable. Boolean values are acceptable.

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

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

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

The keys to this JSON hash are case-insensitive. Figure 2 shows an example JSON.

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


Figure 2: Example Status Telemetry

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

Additional standard JSON fields in this POST MAY be added, see Section 8.3. A server that sees unknown fields should log them, but otherwise ignore them.

5.8. Registrar audit log request

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

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

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

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

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

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] sections 6.3.2 and 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.

5.8.1. MASA audit log response

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


```
{
  "version":"1",
  "events":[
    {
        "date":"<date/time of the entry>",
        "domainID":"<domainID extracted from voucher-request>",
        "nonce":"<any nonce if supplied (or the exact string 'NULL')>",
        "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>"
     }
}
```


Figure 3: Example of audit-log response

Distribution of a large log is less than ideal. This structure can be optimized as follows: Nonced or Nonceless entries for the same domainID MAY be abridged from the log leaving only the single most

recent nonced or nonceless entry for that domainID. In the case of truncation the 'event' truncation value SHOULD contain a count of the number of events for this domainID that were omitted. The log SHOULD NOT be further reduced but there could exist operational situation where maintaining the full log is not possible. In such situations the log MAY be arbitrarily abridged for length, with the number of removed entries indicated as 'arbitrary'.

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

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

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 appends details of the assignment to an internal audit log for that device. The internal audit log is processed when responding to requests for details as described in Section 5.8. The contents of the audit log can express a variety of trust levels, and this section explains what kind of trust a registrar can derive from the entries.

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

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

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

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

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

assertion: The assertion leaf in the voucher and audit log indicates why the MASA issued the voucher. A "verified" entry means that the MASA issued the associated voucher as a result of positive verification of ownership 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.

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,
  "Reason":"Informative human readable message",
  "reason-context": "Additional information"
}
```


Figure 4: Example of enrollment status POST

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

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

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

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

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

7. 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 in the generality of the protocol. Use of the suggested mechanism here MUST be detailed in specific profiles of BRSKI, such as in Section 9.

7.1. Trust Model

This section explains the trust relationships detailed in Section 2.4:

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

Figure 10

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

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

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

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

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

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

4. A craft/serial console COULD include a command such as "est-enroll [2001:db8:0:1]:443" that begins the EST process from the point after the voucher is validated. This process SHOULD

include server certificate verification using an on-screen fingerprint.

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

7.3. Registrar security reductions

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

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

2. A registrar MAY choose to accept devices that claim a unique identity without the benefit of authenticating that claimed identity. This could occur when the pledge does not include an X.509 IDevID factory installed credential. New Entities without an X.509 IDevID credential MAY form the Section 5.2 request using the Section 5.5 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.

5. A registrar MAY accept voucher formats of future types that can not be parsed by the Registrar. This reduces the Registrar's visibility into the exact voucher contents but does not change the protocol operations.

7.4. MASA security reductions

Lower security modes chosen by the MASA service affect all device deployments unless 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.

8. IANA Considerations

This document requires the following IANA actions:

8.1. 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. IANA is asked to change the registration of "est" to include RFC7030 and this document.

8.2. PKIX Registry

IANA is requested to register the following:

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

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

8.3. 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

8.4. 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]

8.5. 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.

9. 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 it's network.

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

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

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

10. Privacy Considerations

10.1. 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 authoritatively bound to the supply chain.

Additionally the domainID captures only the unauthenticated subject key identifier of the domain. A privacy sensitive domain could theoretically generate a new domainID for each device being deployed. Similarly a privacy sensitive domain would likely purchase devices that support proximity assertions from a manufacturer that does not require sales channel integrations. This would result in a

significant level of privacy while maintaining the security characteristics provided by Registrar based audit log inspection.

10.2. What BRSKI-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.

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

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

While the contents of the signed part of the pledge voucher request can not be changed, they are not encrypted at the registrar. The ability to audit the messages by the owner of the network prevents exfiltration of data by a nefarious pledge. 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:

- o the identity of the device being enrolled (down to the serial-number!).

- o 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.

- o the time the device is activated,

- o 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 re-designs, outsourcing, etc.) separate from the MASA, which may need to operate reliably for decades.

10.3. 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.

10.4. Manufacturers and Grey market equipment

Manufacturers of devices often sell different products into different regional markets. Which product is available in which market can be driven by price differentials, support issues (some markets may

require manuals and tech-support to be done in the local language), government export regulation (such as whether strong crypto is permitted to be exported, or permitted to be used in a particular market). When an domain owner obtains a device from a different market (they can be new) and transfers it to a different location, this is called a Grey Market.

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

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

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

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

10.5. 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 gratuitiously.

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 to 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 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 warrantee is terminated. For others, the privacy requirements of some deployments might consider this a standard operating practice.

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

11. Security Considerations

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

To facilitate logging and administrative oversight, in addition to triggering 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 7 to a statement that the registrar "MAY" choose to accept devices that fail cryptographic authentication. This reflects current (poor) practices in shipping devices without a cryptographic identity that are NOT RECOMMENDED.

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

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

11.1. DoS against MASA

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

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

bootstrapping itself. Registrar control over requesting and auditing nonceless vouchers allows device owners to choose an appropriate balance.

The MASA is exposed to DoS attacks wherein attackers claim an unbounded number of devices. Ensuring a registrar is representative of a valid manufacturer customer, even without validating ownership of specific pledge devices, helps to mitigate this. Pledge signatures on the pledge voucher-request, as forwarded by the registrar in the prior-signed-voucher-request field of the registrar voucher-request, significantly reduce this risk by ensuring the MASA can confirm proximity between the pledge and the registrar making the request. 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.

11.2. 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:

- o Ongoing network monitoring for unexpected bootstrapping attempts by pledges.

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

11.3. 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 adopted including:

- o Manually configuring each manufacturer's trust anchor.

- o 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, dem0.example. vs demO.example.

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

- o 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.

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

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

11.4. Manufacturer Maintenance of trust anchors

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

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

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

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

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

12. 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, Peter van der Stok, and Thomas Werner

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

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

13. References

13.1. Normative References

[I-D.ietf-anima-autonomic-control-plane]
Eckert, T., Behringer, M., and S. Bjarnason, "An Autonomic Control Plane (ACP)", draft-ietf-anima-autonomic-control-plane-19 (work in progress), March 2019.

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

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

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

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

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

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

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

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

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

[RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007, <https://www.rfc-editor.org/info/rfc4941>.

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

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

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

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

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

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

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

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

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

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

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <https://www.rfc-editor.org/info/rfc7230>.

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

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

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

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

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

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

13.2. Informative References

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

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

[I-D.ietf-ace-coap-est]
Stok, P., Kampanakis, P., Richardson, M., and S. Raza, "EST over secure CoAP (EST-coaps)", draft-ietf-ace-coap-est-12 (work in progress), June 2019.

[I-D.ietf-anima-constrained-voucher]
Richardson, M., Stok, P., and P. Kampanakis, "Constrained Voucher Artifacts for Bootstrapping Protocols", draft-ietf-anima-constrained-voucher-05 (work in progress), July 2019.

[I-D.ietf-anima-reference-model]
Behringer, M., Carpenter, B., Eckert, T., Ciavaglia, L., and J. Nobre, "A Reference Model for Autonomic Networking", draft-ietf-anima-reference-model-10 (work in progress), November 2018.

[I-D.ietf-anima-stable-connectivity]
Eckert, T. and M. Behringer, "Using Autonomic Control Plane for Stable Connectivity of Network OAM", draft-ietf-anima-stable-connectivity-10 (work in progress), February 2018.

[I-D.ietf-cbor-cddl]
Birkholz, H., Vigano, C., and C. Bormann, "Concise data definition language (CDDL): a notational convention to express CBOR and JSON data structures", draft-ietf-cbor-cddl-08 (work in progress), March 2019.

[I-D.ietf-netconf-zerotouch]
Watsen, K., Abrahamsson, M., and I. Farrer, "Secure Zero Touch Provisioning (SZTP)", draft-ietf-netconf-zerotouch-29 (work in progress), January 2019.

[I-D.ietf-opsawg-mud]
Lear, E., Droms, R., and D. Romascanu, "Manufacturer Usage Description Specification", draft-ietf-opsawg-mud-25 (work in progress), June 2018.

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

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

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

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

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

[RFC2663] Srisuresh, P. and M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations", RFC 2663, DOI 10.17487/RFC2663, August 1999, <https://www.rfc-editor.org/info/rfc2663>.

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

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

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

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

[RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, DOI 10.17487/RFC7228, May 2014, <https://www.rfc-editor.org/info/rfc7228>.

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

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

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

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

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

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

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

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

Appendix A. IPv4 and non-ANI operations

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

Appendix B. mDNS / DNSSD proxy discovery options

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

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

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

Discovery of registrar MAY also be performed with DNS-based service discovery by searching for the service "_brski-registrar._tcp.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:

```
<CODE BEGINS> file "ietf-mud-extension@2018-02-14.yang"
module ietf-mud-brski-masa {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-mud-brski-masa";
  prefix ietf-mud-brski-masa;
  import ietf-mud {
    prefix ietf-mud;
  }
  import ietf-inet-types {
    prefix inet;
  }

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

  revision 2018-02-14 {
    description
    "Initial revision.";
    reference
    "RFC XXXX: Manufacturer Usage Description
    Specification";
  }

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

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


This public key validates vouchers:


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


D.1.2. Manufacturer key pair for IDevID signatures

This private key signs IDevID certificates:

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


This public key validates IDevID certificates:


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


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


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


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

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

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

D.1.4. Pledge key pair

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


```
-----BEGIN EC PRIVATE KEY-----
MHcCAQEEIBgR6SV+uEvWfl5zCQWZxWjYbMhXPyNqdHJ3KPh11mm4oAoGCCqGSM49
AwEHoUQDQgAEWi/jqPpRJ0JgWghZRgeZlLKutbXVjmnHb+1AYaEF/YQjE2g5FZV8
KjiR/bkEl+l8M4onIC7KHaXKKkuag9S6Tw==
-----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.


```
-----BEGIN CERTIFICATE-----
MIICBDCCAYugAwIBAgIECe20qTAKBggqhkjOPQQDAjBNMRIwEAYKCZImiZPyLGQB
GRYCY2ExGTAXBgoJkiaJk/IsZAEZFglzYW5kZWxtYW4xHDAaBgNVBAMME1Vuc3Ry
dW5nIEhpZ2h3YXkgQ0EwIBcNMTkwNDI0MDIxNjU4WhgPMjk5OTEyMzEwMDAwMDBa
MBwxGjAYBgNVBAUMETAwLWQwLWU1LTAyLTAwLTJkMFkwEwYHKoZIzj0CAQYIKoZI
zj0DAQcDQgAEWi/jqPpRJ0JgWghZRgeZlLKutbXVjmnHb+1AYaEF/YQjE2g5FZV8
KjiR/bkEl+l8M4onIC7KHaXKKkuag9S6T6OBhzCBhDAdBgNVHQ4EFgQUj8KYdUoE
OvJ0kcOIbjEWwgWdDYkwCQYDVR0TBAIwADArBgNVHREEJDAioCAGCSsGAQQBgu5S
AaATDBEwMC1EMC1FNS0wMi0wMC0yRDArBgkrBgEEAYLuUgIEHgwcbWFzYS5ob25l
eWR1a2VzLnNhbmRlbG1hbi5jYTAKBggqhkjOPQQDAgNnADBkAjAmvMjmNgjypDhc
fynMV3kMuIpSKrYzRWr4g3PtTwXDsAe0oitTTj4QtU1bajhOfTkCMGMNbsW2Q41F
z9t6PDVdtOKabBbAP1RVoFTlDQuO9nmLzb5kU+cUqCtPRFZBUXP3kg==
-----END CERTIFICATE-----
```


The pledge public certificate as decoded by openssl's x509 utility so that the extensions can be seen. There is a second Custom Extension is included to provided to contain the EUI48/EUI64 that the pledge will configure as it's layer-2 address (this is non-normative).

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 166573225 (0x9edb4a9)
        Signature Algorithm: ecdsa-with-SHA256
        Issuer: DC = ca, DC = sandelman, CN = Unstrung Highway CA
        Validity
            Not Before: Apr 24 02:16:58 2019 GMT
            Not After : Dec 31 00:00:00 2999 GMT
        Subject: serialNumber = 00-d0-e5-02-00-2d
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    04:5a:2f:e3:a8:fa:51:27:42:60:5a:08:59:46:07:
                    99:94:b2:ae:b5:b5:d5:8e:69:c7:6f:ed:40:61:a1:
                    05:fd:84:23:13:68:39:15:95:7c:2a:38:91:fd:b9:
                    04:97:e9:7c:33:8a:27:20:2e:ca:1d:a5:ca:2a:4b:
                    9a:83:d4:ba:4f
                ASN1 OID: prime256v1
                NIST CURVE: P-256
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                8F:C2:98:75:4A:04:3A:F2:74:91:C3:88:6E:31:16:C2:05:9D:0D:89
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Subject Alternative Name:
                othername:<unsupported>
            1.3.6.1.4.1.46930.2:
                ..masa.honeydukes.sandelman.ca
    Signature Algorithm: ecdsa-with-SHA256
         30:64:02:30:26:bc:c8:e6:36:08:f2:a4:38:5c:7f:29:cc:57:
         79:0c:b8:8a:52:2a:b6:33:45:6a:f8:83:73:ed:4f:05:c3:b0:
         07:b4:a2:2b:53:4e:3e:10:b5:4d:5b:6a:38:4e:7d:39:02:30:
         63:0d:6e:c5:b6:43:8d:45:cf:db:7a:3c:35:5d:b4:e2:9a:6c:
         16:c0:3f:54:55:a0:54:e5:0d:0b:8e:f6:79:8b:cd:be:64:53:
         e7:14:a8:2b:4f:44:56:41:51:73:f7:92
```

D.2. Example process

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

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.


```
-----BEGIN CMS-----
MIIGtQYJKoZIhvcNAQcCoIIGpjCCBqICAQExDTALBglghkgBZQMEAgEwggNRBgkq
hkiG9w0BBwGgggNCBIIDPnsiaWV0Zi12b3VjaGVyLXJlcXVlc3Q6dm91Y2hlciI6
eyJhc3NlcnRpb24iOiJwcm94aW1pdHkiLCJjcmVhdGVkLW9uIjoiMjAxOS0wNS0x
NVQxNzoyNTo1NS42NDQtMDQ6MDAiLCJzZXJpYWwtbnVtYmVyIjoiMDAtZDAtZTUt
MDItMDAtMmQiLCJub25jZSI6IlZPVUZULVd3ckV2ME51QVFFSG9WN1EiLCJwcm94
aW1pdHktcmVnaXN0cmFyLWNlcnQiOiJNSUlCMFRDQ0FWYWdBd0lCQWdJQkFqQUtC
Z2dxaGtqT1BRUURBekJ4TVJJd0VBWUtDWkltaVpQeUxHUUJHUllDWTJFeEdUQVhC
Z29Ka2lhSmsvSXNaQUVaRmdsellXNWtaV3h0WVc0eFFEQStCZ05WQkFNTU55TThV
M2x6ZEdWdFZtRnlhV0ZpYkdVNk1IZ3dNREF3TURBd05HWTVNVEZoTUQ0Z1ZXNXpk
SEoxYm1jZ1JtOTFibloyVc0Z1EwRXdIaGNOTVRjeE1UQTNNak0wTlRJNFdoY05N
VGt4TVRBM01qTTBOVEk0V2pCRE1SSXdFQVlLQ1pJbWlaUHlMR1FCR1JZQ1kyRXhH
VEFYQmdvSmtpYUprL0lzWkFFWkZnbHpZVzVrWld4dFlXNHhFakFRQmdOVkJBTU1D
V3h2WTJGc2FHOXpkREJaTUJNR0J5cUdTTTQ5QWdFR0NDcUdTTTQ5QXdFSEEwSUFC
SlpsVUhJMHVwL2wzZVpmOXZDQmIrbElub0VNRWdjN1JvK1haQ3RqQUkwQ0QxZkpm
SlIvaEl5eURtSFd5WWlORmJSQ0g5ZnlhcmZremdYNHAwelRpenFRRFBTE1Ba0dB
MVVkRXdRQ01BQXdDZ1lJS29aSXpqMEVBd01EYVFBd1pnSXhBTFFFNTnVyZjh0djUw
bFJPRDVEUVhIRU9KSk5XM1FWMmc5UUVkRFNrMk1ZK0FvU3JCU21HU05qaDRvbEVP
aEV1TGdJeEFKNG5XZk53K0JqYlptS2lJaVVFY1R3SE1oR1ZYYU1IWS9GN24zOXd3
S2NCQlNPbmROUHFDcE9FTGw2YnEzQ1pxUT09In19oIICCDCCAgQwggGLoAMCAQIC
BAnttKkwCgYIKoZIzj0EAwIwTTESMBAGCgmSJomT8ixkARkWAmNhMRkwFwYKCZIm
iZPyLGQBGRYJc2FuZGVsbWFuMRwwGgYDVQQDDBNVbnN0cnVuZyBIaWdod2F5IENB
MCAXDTE5MDQyNDAyMTY1OFoYDzI5OTkxMjMxMDAwMDAwWjAcMRowGAYDVQQFDBEw
MC1kMC1lNS0wMi0wMC0yZDBZMBMGByqGSM49AgEGCCqGSM49AwEHA0IABFov46j6
USdCYFoIWUYHmZSyrrW11Y5px2/tQGGhBf2EIxNoORWVfCo4kf25BJfpfDOKJyAu
yh2lyipLmoPUuk+jgYcwgYQwHQYDVR0OBBYEFI/CmHVKBDrydJHDiG4xFsIFnQ2J
MAkGA1UdEwQCMAAwKwYDVR0RBCQwIqAgBgkrBgEEAYLuUgGgEwwRMDAtRDAtRTUt
MDItMDAtMkQwKwYJKwYBBAGC7lICBB4MHG1hc2EuaG9uZXlkdWtlcy5zYW5kZWxt
YW4uY2EwCgYIKoZIzj0EAwIDZwAwZAIwJrzI5jYI8qQ4XH8pzFd5DLiKUiq2M0Vq
+INz7U8Fw7AHtKIrU04+ELVNW2o4Tn05AjBjDW7FtkONRc/bejw1XbTimmwWwD9U
VaBU5Q0LjvZ5i82+ZFPnFKgrT0RWQVFz95IxggErMIIBJwIBATBVME0xEjAQBgoJ
kiaJk/IsZAEZFgJjYTEZMBcGCgmSJomT8ixkARkWCXNhbmRlbG1hbjEcMBoGA1UE
AwwTVW5zdHJ1bmcgSGlnaHdheSBDQQIECe20qTALBglghkgBZQMEAgGgaTAYBgkq
hkiG9w0BCQMxCwYJKoZIhvcNAQcBMBwGCSqGSIb3DQEJBTEPFw0xOTA1MTUyMTI1
NTVaMC8GCSqGSIb3DQEJBDEiBCAQN2lP7aqwyhmj9qUHt6Qk/SbOTOPXFOwn1wv2
5YGYgDAKBggqhkjOPQQDAgRHMEUCIEYQhHToU0rrhPyQv2fR0TwWePTx2Z1DEhR4
tTl/Dr/ZAiEA47u9+bIz/p6nFJ+wctKHER+ycUzYQF56h9odMo+Ilkc=
-----END CMS-----
```


file: examples/vr_00-D0-E5-02-00-2D.pkcs

The ASN1 decoding of the artifact:

```
    0:d=0  hl=4 l=1717 cons: SEQUENCE
    4:d=1  hl=2 l=   9 prim: OBJECT            :pkcs7-signedData
   15:d=1  hl=4 l=1702 cons: cont [ 0 ]
   19:d=2  hl=4 l=1698 cons: SEQUENCE
   23:d=3  hl=2 l=   1 prim: INTEGER           :01
   26:d=3  hl=2 l=  13 cons: SET
   28:d=4  hl=2 l=  11 cons: SEQUENCE
   30:d=5  hl=2 l=   9 prim: OBJECT            :sha256
   41:d=3  hl=4 l= 849 cons: SEQUENCE
   45:d=4  hl=2 l=   9 prim: OBJECT            :pkcs7-data
   56:d=4  hl=4 l= 834 cons: cont [ 0 ]
   60:d=5  hl=4 l= 830 prim: OCTET STRING      :{"ietf-voucher-request:v
  894:d=3  hl=4 l= 520 cons: cont [ 0 ]
  898:d=4  hl=4 l= 516 cons: SEQUENCE
  902:d=5  hl=4 l= 395 cons: SEQUENCE
  906:d=6  hl=2 l=   3 cons: cont [ 0 ]
  908:d=7  hl=2 l=   1 prim: INTEGER           :02
  911:d=6  hl=2 l=   4 prim: INTEGER           :09EDB4A9
  917:d=6  hl=2 l=  10 cons: SEQUENCE
  919:d=7  hl=2 l=   8 prim: OBJECT            :ecdsa-with-SHA256
  929:d=6  hl=2 l=  77 cons: SEQUENCE
  931:d=7  hl=2 l=  18 cons: SET
  933:d=8  hl=2 l=  16 cons: SEQUENCE
  935:d=9  hl=2 l=  10 prim: OBJECT            :domainComponent
  947:d=9  hl=2 l=   2 prim: IA5STRING         :ca
  951:d=7  hl=2 l=  25 cons: SET
  953:d=8  hl=2 l=  23 cons: SEQUENCE
  955:d=9  hl=2 l=  10 prim: OBJECT            :domainComponent
  967:d=9  hl=2 l=   9 prim: IA5STRING         :sandelman
  978:d=7  hl=2 l=  28 cons: SET
  980:d=8  hl=2 l=  26 cons: SEQUENCE
  982:d=9  hl=2 l=   3 prim: OBJECT            :commonName
  987:d=9  hl=2 l=  19 prim: UTF8STRING        :Unstrung Highway CA
 1008:d=6  hl=2 l=  32 cons: SEQUENCE
 1010:d=7  hl=2 l=  13 prim: UTCTIME           :190424021658Z
 1025:d=7  hl=2 l=  15 prim: GENERALIZEDTIME   :29991231000000Z
 1042:d=6  hl=2 l=  28 cons: SEQUENCE
 1044:d=7  hl=2 l=  26 cons: SET
 1046:d=8  hl=2 l=  24 cons: SEQUENCE
 1048:d=9  hl=2 l=   3 prim: OBJECT            :serialNumber
 1053:d=9  hl=2 l=  17 prim: UTF8STRING        :00-d0-e5-02-00-2d
 1072:d=6  hl=2 l=  89 cons: SEQUENCE
 1074:d=7  hl=2 l=  19 cons: SEQUENCE
 1076:d=8  hl=2 l=   7 prim: OBJECT            :id-ecPublicKey
 1085:d=8  hl=2 l=   8 prim: OBJECT            :prime256v1
 1095:d=7  hl=2 l=  66 prim: BIT STRING
```

```
 1163:d=6  hl=3 l= 135 cons: cont [ 3 ]
 1166:d=7  hl=3 l= 132 cons: SEQUENCE
 1169:d=8  hl=2 l=  29 cons: SEQUENCE
 1171:d=9  hl=2 l=   3 prim: OBJECT            :X509v3 Subject Key Ident
 1176:d=9  hl=2 l=  22 prim: OCTET STRING      [HEX DUMP]:04148FC298754A
 1200:d=8  hl=2 l=   9 cons: SEQUENCE
 1202:d=9  hl=2 l=   3 prim: OBJECT            :X509v3 Basic Constraints
 1207:d=9  hl=2 l=   2 prim: OCTET STRING      [HEX DUMP]:3000
 1211:d=8  hl=2 l=  43 cons: SEQUENCE
 1213:d=9  hl=2 l=   3 prim: OBJECT            :X509v3 Subject Alternati
 1218:d=9  hl=2 l=  36 prim: OCTET STRING      [HEX DUMP]:3022A02006092B
 1256:d=8  hl=2 l=  43 cons: SEQUENCE
 1258:d=9  hl=2 l=   9 prim: OBJECT            :1.3.6.1.4.1.46930.2
 1269:d=9  hl=2 l=  30 prim: OCTET STRING      [HEX DUMP]:0C1C6D6173612E
 1301:d=5  hl=2 l=  10 cons: SEQUENCE
 1303:d=6  hl=2 l=   8 prim: OBJECT            :ecdsa-with-SHA256
 1313:d=5  hl=2 l= 103 prim: BIT STRING
 1418:d=3  hl=4 l= 299 cons: SET
 1422:d=4  hl=4 l= 295 cons: SEQUENCE
 1426:d=5  hl=2 l=   1 prim: INTEGER           :01
 1429:d=5  hl=2 l=  85 cons: SEQUENCE
 1431:d=6  hl=2 l=  77 cons: SEQUENCE
 1433:d=7  hl=2 l=  18 cons: SET
 1435:d=8  hl=2 l=  16 cons: SEQUENCE
 1437:d=9  hl=2 l=  10 prim: OBJECT            :domainComponent
 1449:d=9  hl=2 l=   2 prim: IA5STRING         :ca
 1453:d=7  hl=2 l=  25 cons: SET
 1455:d=8  hl=2 l=  23 cons: SEQUENCE
 1457:d=9  hl=2 l=  10 prim: OBJECT            :domainComponent
 1469:d=9  hl=2 l=   9 prim: IA5STRING         :sandelman
 1480:d=7  hl=2 l=  28 cons: SET
 1482:d=8  hl=2 l=  26 cons: SEQUENCE
 1484:d=9  hl=2 l=   3 prim: OBJECT            :commonName
 1489:d=9  hl=2 l=  19 prim: UTF8STRING        :Unstrung Highway CA
 1510:d=6  hl=2 l=   4 prim: INTEGER           :09EDB4A9
 1516:d=5  hl=2 l=  11 cons: SEQUENCE
 1518:d=6  hl=2 l=   9 prim: OBJECT            :sha256
 1529:d=5  hl=2 l= 105 cons: cont [ 0 ]
 1531:d=6  hl=2 l=  24 cons: SEQUENCE
 1533:d=7  hl=2 l=   9 prim: OBJECT            :contentType
 1544:d=7  hl=2 l=  11 cons: SET
 1546:d=8  hl=2 l=   9 prim: OBJECT            :pkcs7-data
 1557:d=6  hl=2 l=  28 cons: SEQUENCE
 1559:d=7  hl=2 l=   9 prim: OBJECT            :signingTime
 1570:d=7  hl=2 l=  15 cons: SET
 1572:d=8  hl=2 l=  13 prim: UTCTIME           :190515212555Z
 1587:d=6  hl=2 l=  47 cons: SEQUENCE
 1589:d=7  hl=2 l=   9 prim: OBJECT            :messageDigest
```

```
 1600:d=7  hl=2 l=  34 cons: SET
 1602:d=8  hl=2 l=  32 prim: OCTET STRING      [HEX DUMP]:1037694FEDAAB0
 1636:d=5  hl=2 l=  10 cons: SEQUENCE
 1638:d=6  hl=2 l=   8 prim: OBJECT            :ecdsa-with-SHA256
 1648:d=5  hl=2 l=  71 prim: OCTET STRING      [HEX DUMP]:30450220461084
```

The JSON contained in the voucher request:

{"ietf-voucher-request:voucher":{"assertion":"proximity","cr
eated-on":"2019-05-15T17:25:55.644-04:00","serial-number":"0
0-d0-e5-02-00-2d","nonce":"VOUFT-WwrEv0NuAQEHoV7Q","proximit
y-registrar-cert":"MIIB0TCCAVagAwIBAgIBAjAKBggqhkjOPQQDAzBxM
RIwEAYKCZImiZPyLGQBGRYCY2ExGTAXBgoJkiaJk/IsZAEZFglzYW5kZWxtY
W4xQDA+BgNVBAMMNyM8U3lzdGVtVmFyaWFibGU6MHgwMDAwMDAwNGY5MTFhM
D4gVW5zdHJ1bmcgRm91bnRhaW4gQ0EwHhcNMTcxMTA3MjM0NTI4WhcNMTkxM
TA3MjM0NTI4WjBDMRIwEAYKCZImiZPyLGQBGRYCY2ExGTAXBgoJkiaJk/IsZ
AEZFglzYW5kZWxtYW4xEjAQBgNVBAMMCWxvY2FsaG9zdDBZMBMGByqGSM49A
gEGCCqGSM49AwEHA0IABJZlUHI0up/l3eZf9vCBb+lInoEMEgc7Ro+XZCtjA
I0CD1fJfJR/hIyyDmHWyYiNFbRCH9fyarfkzgX4p0zTizqjDTALMAkGA1UdE
wQCMAAwCgYIKoZIzj0EAwMDaQAwZgIxALQMNurf8tv50lROD5DQXHEOJJNW3
QV2g9QEdDSk2MY+AoSrBSmGSNjh4olEOhEuLgIxAJ4nWfNw+BjbZmKiIiUEc
TwHMhGVXaMHY/F7n39wwKcBBSOndNPqCpOELl6bq3CZqQ=="}}

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.

-----BEGIN CMS-----
MIIPkwYJKoZIhvcNAQcCoIIPhDCCD4ACAQExDTALBglghkgBZQMEAgEwggnUBgkq
hkiG9w0BBwGgggnFBIIJwXsiaWV0Zi12b3VjaGVyLXJlcXVlc3Q6dm91Y2hlciI6
eyJhc3NlcnRpb24iOiJwcm94aW1pdHkiLCJjcmVhdGVkLW9uIjoiMjAxOS0wNS0x
NVQyMToyNTo1NS43NThaIiwic2VyaWFsLW51bWJlciI6IjAwLWQwLWU1LTAyLTAw
LTJkIiwibm9uY2UiOiJWT1VGVC1Xd3JFdjBOdUFRRUhvVjdRIiwicHJpb3Itc2ln
bmVkLXZvdWNoZXItcmVxdWVzdCI6Ik1JSUd0UVlKS29aSWh2Y05BUWNDb0lJR3Bq
Q0NCcUlDQVFFeERUQUxCZ2xnaGtnQlpRTUVBZ0V3Z2dOUkJna3Foa2lHOXcwQkJ3
R2dnZ05DQklJRFBuc2lhV1YwWmkxMmIzVmphR1Z5TFhKbGNYVmxjM1E2ZG05MVky
aGxjaUk2ZXlKaGMzTmxjblJwYjI0aU9pSndjbTk0YVcxcGRIa2lMQ0pqY21WaGRH
VmtMVzl1SWpvaU1qQXhPUzB3TlMweE5WUXhOem95TlRvMU5TNDJORFF0TURRNk1E
QWlMQ0p6WlhKcFlXd3RiblZ0WW1WeUlqb2lNREF0WkRBdFpUVXRNREl0TURBdE1t
UWlMQ0p1YjI1alpTSTZJbFpQVlVaVUxWZDNja1YyTUU1MVFWRkRkZ2ZTRzlXTjFFaUxD
SndjbTk0YVcxcGRIa3RjbVZuYVhOMGNtRnlMV05sY25RaU9pSk5TVWxDTUZSRFEw
RldZV2RCZDBsQ1FXZEpRa0ZxUVV0Q1oyZHhhR3RxVDFCUlVVUkJla0o0VFZKSmQw
VkJXVXRESFlsVXhIVVVKSFVsbERXVEpGZUVkVVFWaENaMjlLYTJsaFNrc3ZTWE5h
c3ZTWE5hUVVWYVJteDZXVmMxYTFwWGVEQlhWWE5hVkZvelZuZWNZV0pHYTNOVWtXTGRV
NTVUUGhWTTJ4NlpFZFdkRlp0Um5saFYwWnBZa2RWTmxSWTNoZ0ZE1ESFlsVFVSQmQw
NUhXVFZOVkVab1RVUTBaMVpYTlhwa1NGSXhZbTFqWjFKdE9URmlibEpvWVZjMFox

RXdSWGRJYUdOT1RWUmplRTFVUVROTmFrMHdUbFJKTkZkb1kwNU5WR3Q0VFZSQk0w
MXFUVEJPVkVrMFYycENNSRTFTU1hkRlFWbExRMXBKYldsYVVIbE1SMUZDUjFKWlEx
a3lSWGhIVkVGWVFtZHZTbXRwWVVwckwwbHpXa0ZGV2tabmJIcFpWelZyV2xkNGRG
bFhOSGhGYWtGUlFtZE9Wa0pCVFUxRFYzaDJXVEpHYzJGSE9YcGtSRUphVFVKTlIw
SjVjVWRUVFRRNVFXZEZSME5EY1VkVFRUTVRWGRGU0VFd1NVRkNTbHBzVlVoSk1I
VndMMnd6WlZwbU9YWkRRbUlyYkVsdWIwVk5SV2RqTjFKdksxaGFRM1JxUVVrd1Ew
UXhaa3BtU2xKdmFFbDVlVJ0U0ZkNVdXbE9SbUpTUTBnNVpubGhjbVpyZW1kWU5I
QXdlbFJwWW5GcVJGUkJURFFCYTBkQk1WVmtSWGRSUTAxQlFYZENaRVRaMWxKUzI5YVNY
cHFNRVZCZDAxRVlWRkJkMXBuU1hoQlRGRk5UblZ5WmpoMGRqVXdiRkpQUkRWRVVW
aElSVTlLU2s1WE0xRldNbWNWVVWa1JGTnJNazFaSzBGdlUzSkNVMjFIVTA1cWFE
UnZiRVZQYUVWMVRHEplRUZLTkc1WFprNTNLMEpxWW1wdFMybEpTVlZGWTFSM1NF
MW9SMVpZWVUxSVdTOUdOMjR6T1hkM1MxaENRbE5TT1VSRGRjRTlGVEd3Mllu
RXpRMXB4VVQwOUluMD0RDQ0FnUXdnZ0dMb0FNQ0FRSUNCQW50dEtrd0Nn
WUlLb1pJemowRUF3SXdUVEVTTUJBR0NnbVNKb21UOGl4a0FSa1dBbBU5oTVJrd0Z3
WUtDWkltaVpQeUxHUUJHUllKYzJGdVpHVnNiV0Z1TVJ3d0dnWURWUURWUVFERREJOVmJu
TjBjblZ1Z1WnlCSWFXZG9kMkY1SUVOQk1DQVhEVEU1TURReU5EQXlNVFkxT0ZvWUR6
STVPVGt4TWpNeE1EQXdNREF3V2pBY01Sb3dHQVlEVlEQVlEVlFRRkRkRCRXdNQzFrTUxbE5T
MHdNaTB3TURBeE1HQnlxR1NNNDlBZ0VHQ0NxR1NNNDlBd0VIQTBJQUJJQUJG
b3Y0Nmo2VVNkQ1lGb0lXVVlIbVpTeXJyVzExWTVweDIvdFFHR2hCZjJFSXhOb09S
V1ZmQ280a2YyYyNUJKZnBmRE9LSnlBdXloMmx5aXBMbW9QVXVrK2pnWWN3Z1lRd0hR
WURWUjBPQkJZRUZJL0NtSFZLQkRyeeWRKSERpRzR4RnNJRm5RMkpNQWtHQTFVZEV3
UUNNQUF3S3dZRFZSMFJCQ1F3SXFBZ0Jna3JCZ0VFQVlMZVd2VnR2dFd3dSTURBdFJE
QXRSVFV0TURJdE1EEQXRNa1F3S3dZRVJJQUdDQUdDQUdDN2xJQ0NNE1IRzFoYzJFdWFH
OXVaWGxrZFd4NXZVVZGd0dFlXd4dFlXNHhFN3dZS1p0dEY1QXdDR2pBRFFXdJRFp3
QXdaQUl3SnJ6STVqWUk4cVE0WEg4cHpGZDVETGlLVWlxMk0wwVnErSU56N1U4Rnc3
QUh0S0lyVTA0K0VMVk5XMm80VG4wNUFqQmpEVEpsVEdtMGtPTlJjZEhlVUVdlR0I1d1t
d1d3RDlVVmFCVTVRMExxdlo1aTgyK1pGUUZ5SGS2dyVDBSV1FWRno5NUl4Z2dGck1J
SUJKd0lCQVRCVk1FMHhFakFRQmdvSmtpYUprL0lzWkFFWkZnZ2ZSmpZZVEVaTUJjR0Nn
bVNKb21UOGl4a0FSa1dDWE5oYm1SbGJHMWhiakVjTUJvR0ExVUVBd3dUVlVvc1emRI
SjFibWNnSGlnaFdBR0VBRGVOVEFMQmdsZHrZ0JaUU1FQWdHZ2FU
QVlCZ2txaGtpRzl3MEJDUWN4Q3dZSktvWklodmNOQVFjQkVTZk9BYjBaaTkwM0RS
RUpCVEVQRncweE9UQTBNalF3TWpFMlREMWxNNxR1NJYjNEUUVVVKQkRFaUJD
QVFFOMmxQN2Fxd3lobWo5cVVIdDZRay9TYk9UUT1BYRk93bjF3I1WUdZZ0RBS0Jn
Z3Foa2pPUFFFREFnUkhNRVVDSVFDaHd5YXB5UXYyUXVBeHZBQUVXdkZEVGFCWEtB
MURFaFI0dFRsL0RyL1pBaUNOSXovcDZuRkord2N0S0hFUit5Y1V6WVFGWVFG
NTZoOW9kTW8rSWxrYzYzRmqggRCMIIB0TCCAVagAwIBAgIBAjAKBggqhkjOPQQD
AzBxMRIwEAYKCZImiZPyLGQBGRYCY2ExGTAXBgoJkiaJk/IsZAEZFglzYW5kZWxt
YW4xQDA+BgNVBAMMNyM8U3lzdGVtVmFyaWFibGU6MHgwMDAwMDAwNGY5MTFhMD4g
VW5zdHJ1bmcgRm91bnRhaW4gQ0EwHhcNMTcxMTA3MjM0NTI4WhcNMTkxMTA3MjM0
NTI4WjBDMRIwEAYKCZImiZPyLGQBGRYCY2ExGTAXBgoJkiaJk/IsZAEZFglzYW5k
ZWxtYW4xEjAQBgNVBAMMCWxvY2FsaG9zdDBZMBMGByqGSM49AgEGCCqGSM49AwEH
A0IABJZlUHI0up/l3eZf9vCBb+lInoEMEgc7Ro+XZCtjAI0CD1fJfJR/hIyyDmHW
yYiNFbRCH9fyarfkzgX4p0zTizqjDTALMAkGA1UdEwQCMAAwCgYIKoZIzj0EAwMD
aQAwZgIxALQMNurf8tv50lROD5DQXHEOJJNW3QV2g9QEdDSk2MY+AoSrBSmGSNjh
4olEOhEuLgIxAJ4nWfNw+BjbZmKiIiUEcTwHMhGVXaMHY/F7n39wwKcBBSOndNPq
CpOELl6bq3CZqTCCAmkwggHvoAMCAQICAQMwCgYIKoZIzj0EAwIwbTESMBAGCgmS
JomT8ixkARkWAmNhMRkwFwYKCZImiZPyLGQBGRYJc2FuZGVsbWFuMTwwOgYDVQQD
DDNmb3VudGFpbi10ZXN0LmV4YW1wbGUuY29tIFVuc3RydW5nIEZvdW50YWluIFJv

b3QgQ0EwHhcNMTkwMTEzMjI1NDQ0WhcNMjEwMTEyMjI1NDQ0WjBtMRIwEAYKCZIm
iZPyLGQBGRYCY2ExGTAXBgoJkiaJk/IsZAEZFglzYW5kZWxtYW4xPDA6BgNVBAMM
M2ZvdW50YWluLXRlc3QuZXhhbXBsZS5jb20gVW5zdHJ1bmcgRm91bnRhaW4gUm9v
dCBDQTB2MBAGByqGSM49AgEGBSuBBAAiA2IABBt/WboXwxq8Zo2MbODD+jFxD2X2
IpG9t1aAB9vfuHqlRU15ikaXGVmWMbGPaX0yvjzIPltjtUb2qNVvm/nA89O5FD9y
R1Gkdt3S8L/1yo8wAX/4wl/T9SADRIuL8gdstKNjMGEwDwYDVR0TAQH/BAUwAwEB
/zAOBgNVHQ8BAf8EBAMCAQYwHQYDVR0OBBYEFLml9ssR4QekSSynCMZ8ELyHs3Qm
MB8GA1UdIwQYMBaAFLml9ssR4QekSSynCMZ8ELyHs3QmMAoGCCqGSM49BAMCA2gA
MGUCMAviLdbfd6AZdsOxNgf7D15WFmGC1JkHeEbT/0w4UXz6q/48S71/IMbSXRWH
aNxiJwIxAOCRjtlN+VSmCLTvWwMTxnSpIuqMr/O1y2Z8rl459VRFphWPdbf4i0qE
cwu0u4JzpDGCAUwwggFIAgEBMHYwcTESMBAGCgmSJomT8ixkARkWAmNhMRkwFwYK
CZImiZPyLGQBGRYJc2FuZGVsbWFuMUAwPgYDVQQDDDcjPFN5c3RlbVZhcmlhYmxl
OjB4MDAwMDAwMDRmOTExYTA+IFVuc3RydW5nIEZvdW50YWluIENBAgECMAsGCWCG
SAFlAwQCAaBpMBgGCSqGSIb3DQEJAzELBgkqhkiG9w0BBwEwHAYJKoZIhvcNAQkF
MQ8XDTE5MDUxNTIxMjU1NVowLwYJKoZIhvcNAQkEMSIEIFBQjMmWzZOEkRHXrVAS
snJwgQ26goyvOAtUFYs3MstMMAoGCCqGSM49BAMCBEcwRQIgBthbhEmgbqZbYDkD
zxHXLzJ5eusWplzHKqZyxNpzaR8CIQC3UtMu0QsXoUpYL016iTsbd7Eedi8IfnwQ
akExfhh0ew==
-----END CMS-----

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

The ASN1 decoding of the artifact:

```
  0:d=0  hl=4 l=3987 cons: SEQUENCE
  4:d=1  hl=2 l=   9 prim: OBJECT            :pkcs7-signedData
 15:d=1  hl=4 l=3972 cons: cont [ 0 ]
 19:d=2  hl=4 l=3968 cons: SEQUENCE
 23:d=3  hl=2 l=   1 prim: INTEGER           :01
 26:d=3  hl=2 l=  13 cons: SET
 28:d=4  hl=2 l=  11 cons: SEQUENCE
 30:d=5  hl=2 l=   9 prim: OBJECT            :sha256
 41:d=3  hl=4 l=2516 cons: SEQUENCE
 45:d=4  hl=2 l=   9 prim: OBJECT            :pkcs7-data
 56:d=4  hl=4 l=2501 cons: cont [ 0 ]
 60:d=5  hl=4 l=2497 prim: OCTET STRING      :{"ietf-voucher-request:v
2561:d=3  hl=4 l=1090 cons: cont [ 0 ]
2565:d=4  hl=4 l= 465 cons: SEQUENCE
2569:d=5  hl=4 l= 342 cons: SEQUENCE
2573:d=6  hl=2 l=   3 cons: cont [ 0 ]
2575:d=7  hl=2 l=   1 prim: INTEGER           :02
2578:d=6  hl=2 l=   1 prim: INTEGER           :02
2581:d=6  hl=2 l=  10 cons: SEQUENCE
2583:d=7  hl=2 l=   8 prim: OBJECT            :ecdsa-with-SHA384
2593:d=6  hl=2 l= 113 cons: SEQUENCE
2595:d=7  hl=2 l=  18 cons: SET
2597:d=8  hl=2 l=  16 cons: SEQUENCE
2599:d=9  hl=2 l=  10 prim: OBJECT            :domainComponent
```

```
 2611:d=9  hl=2 l=   2 prim: IA5STRING         :ca
 2615:d=7  hl=2 l=  25 cons: SET
 2617:d=8  hl=2 l=  23 cons: SEQUENCE
 2619:d=9  hl=2 l=  10 prim: OBJECT            :domainComponent
 2631:d=9  hl=2 l=   9 prim: IA5STRING         :sandelman
 2642:d=7  hl=2 l=  64 cons: SET
 2644:d=8  hl=2 l=  62 cons: SEQUENCE
 2646:d=9  hl=2 l=   3 prim: OBJECT            :commonName
 2651:d=9  hl=2 l=  55 prim: UTF8STRING        :#<SystemVariable:0x00000
 2708:d=6  hl=2 l=  30 cons: SEQUENCE
 2710:d=7  hl=2 l=  13 prim: UTCTIME           :171107234528Z
 2725:d=7  hl=2 l=  13 prim: UTCTIME           :191107234528Z
 2740:d=6  hl=2 l=  67 cons: SEQUENCE
 2742:d=7  hl=2 l=  18 cons: SET
 2744:d=8  hl=2 l=  16 cons: SEQUENCE
 2746:d=9  hl=2 l=  10 prim: OBJECT            :domainComponent
 2758:d=9  hl=2 l=   2 prim: IA5STRING         :ca
 2762:d=7  hl=2 l=  25 cons: SET
 2764:d=8  hl=2 l=  23 cons: SEQUENCE
 2766:d=9  hl=2 l=  10 prim: OBJECT            :domainComponent
 2778:d=9  hl=2 l=   9 prim: IA5STRING         :sandelman
 2789:d=7  hl=2 l=  18 cons: SET
 2791:d=8  hl=2 l=  16 cons: SEQUENCE
 2793:d=9  hl=2 l=   3 prim: OBJECT            :commonName
 2798:d=9  hl=2 l=   9 prim: UTF8STRING        :localhost
 2809:d=6  hl=2 l=  89 cons: SEQUENCE
 2811:d=7  hl=2 l=  19 cons: SEQUENCE
 2813:d=8  hl=2 l=   7 prim: OBJECT            :id-ecPublicKey
 2822:d=8  hl=2 l=   8 prim: OBJECT            :prime256v1
 2832:d=7  hl=2 l=  66 prim: BIT STRING
 2900:d=6  hl=2 l=  13 cons: cont [ 3 ]
 2902:d=7  hl=2 l=  11 cons: SEQUENCE
 2904:d=8  hl=2 l=   9 cons: SEQUENCE
 2906:d=9  hl=2 l=   3 prim: OBJECT            :X509v3 Basic Constraints
 2911:d=9  hl=2 l=   2 prim: OCTET STRING      [HEX DUMP]:3000
 2915:d=5  hl=2 l=  10 cons: SEQUENCE
 2917:d=6  hl=2 l=   8 prim: OBJECT            :ecdsa-with-SHA384
 2927:d=5  hl=2 l= 105 prim: BIT STRING
 3034:d=4  hl=4 l= 617 cons: SEQUENCE
 3038:d=5  hl=4 l= 495 cons: SEQUENCE
 3042:d=6  hl=2 l=   3 cons: cont [ 0 ]
 3044:d=7  hl=2 l=   1 prim: INTEGER           :02
 3047:d=6  hl=2 l=   1 prim: INTEGER           :03
 3050:d=6  hl=2 l=  10 cons: SEQUENCE
 3052:d=7  hl=2 l=   8 prim: OBJECT            :ecdsa-with-SHA256
 3062:d=6  hl=2 l= 109 cons: SEQUENCE
 3064:d=7  hl=2 l=  18 cons: SET
 3066:d=8  hl=2 l=  16 cons: SEQUENCE
```

```
 3068:d=9  hl=2 l=  10 prim: OBJECT            :domainComponent
 3080:d=9  hl=2 l=   2 prim: IA5STRING         :ca
 3084:d=7  hl=2 l=  25 cons: SET
 3086:d=8  hl=2 l=  23 cons: SEQUENCE
 3088:d=9  hl=2 l=  10 prim: OBJECT            :domainComponent
 3100:d=9  hl=2 l=   9 prim: IA5STRING         :sandelman
 3111:d=7  hl=2 l=  60 cons: SET
 3113:d=8  hl=2 l=  58 cons: SEQUENCE
 3115:d=9  hl=2 l=   3 prim: OBJECT            :commonName
 3120:d=9  hl=2 l=  51 prim: UTF8STRING        :fountain-test.example.co
 3173:d=6  hl=2 l=  30 cons: SEQUENCE
 3175:d=7  hl=2 l=  13 prim: UTCTIME           :190113225444Z
 3190:d=7  hl=2 l=  13 prim: UTCTIME           :210112225444Z
 3205:d=6  hl=2 l= 109 cons: SEQUENCE
 3207:d=7  hl=2 l=  18 cons: SET
 3209:d=8  hl=2 l=  16 cons: SEQUENCE
 3211:d=9  hl=2 l=  10 prim: OBJECT            :domainComponent
 3223:d=9  hl=2 l=   2 prim: IA5STRING         :ca
 3227:d=7  hl=2 l=  25 cons: SET
 3229:d=8  hl=2 l=  23 cons: SEQUENCE
 3231:d=9  hl=2 l=  10 prim: OBJECT            :domainComponent
 3243:d=9  hl=2 l=   9 prim: IA5STRING         :sandelman
 3254:d=7  hl=2 l=  60 cons: SET
 3256:d=8  hl=2 l=  58 cons: SEQUENCE
 3258:d=9  hl=2 l=   3 prim: OBJECT            :commonName
 3263:d=9  hl=2 l=  51 prim: UTF8STRING        :fountain-test.example.co
 3316:d=6  hl=2 l= 118 cons: SEQUENCE
 3318:d=7  hl=2 l=  16 cons: SEQUENCE
 3320:d=8  hl=2 l=   7 prim: OBJECT            :id-ecPublicKey
 3329:d=8  hl=2 l=   5 prim: OBJECT            :secp384r1
 3336:d=7  hl=2 l=  98 prim: BIT STRING
 3436:d=6  hl=2 l=  99 cons: cont [ 3 ]
 3438:d=7  hl=2 l=  97 cons: SEQUENCE
 3440:d=8  hl=2 l=  15 cons: SEQUENCE
 3442:d=9  hl=2 l=   3 prim: OBJECT            :X509v3 Basic Constraints
 3447:d=9  hl=2 l=   1 prim: BOOLEAN           :255
 3450:d=9  hl=2 l=   5 prim: OCTET STRING      [HEX DUMP]:30030101FF
 3457:d=8  hl=2 l=  14 cons: SEQUENCE
 3459:d=9  hl=2 l=   3 prim: OBJECT            :X509v3 Key Usage
 3464:d=9  hl=2 l=   1 prim: BOOLEAN           :255
 3467:d=9  hl=2 l=   4 prim: OCTET STRING      [HEX DUMP]:03020106
 3473:d=8  hl=2 l=  29 cons: SEQUENCE
 3475:d=9  hl=2 l=   3 prim: OBJECT            :X509v3 Subject Key Ident
 3480:d=9  hl=2 l=  22 prim: OCTET STRING      [HEX DUMP]:0414B9A5F6CB11
 3504:d=8  hl=2 l=  31 cons: SEQUENCE
 3506:d=9  hl=2 l=   3 prim: OBJECT            :X509v3 Authority Key Ide
 3511:d=9  hl=2 l=  24 prim: OCTET STRING      [HEX DUMP]:30168014B9A5F6
 3537:d=5  hl=2 l=  10 cons: SEQUENCE
```

```
 3539:d=6  hl=2 l=   8 prim: OBJECT            :ecdsa-with-SHA256
 3549:d=5  hl=2 l= 104 prim: BIT STRING
 3655:d=3  hl=4 l= 332 cons: SET
 3659:d=4  hl=4 l= 328 cons: SEQUENCE
 3663:d=5  hl=2 l=   1 prim: INTEGER           :01
 3666:d=5  hl=2 l= 118 cons: SEQUENCE
 3668:d=6  hl=2 l= 113 cons: SEQUENCE
 3670:d=7  hl=2 l=  18 cons: SET
 3672:d=8  hl=2 l=  16 cons: SEQUENCE
 3674:d=9  hl=2 l=  10 prim: OBJECT            :domainComponent
 3686:d=9  hl=2 l=   2 prim: IA5STRING         :ca
 3690:d=7  hl=2 l=  25 cons: SET
 3692:d=8  hl=2 l=  23 cons: SEQUENCE
 3694:d=9  hl=2 l=  10 prim: OBJECT            :domainComponent
 3706:d=9  hl=2 l=   9 prim: IA5STRING         :sandelman
 3717:d=7  hl=2 l=  64 cons: SET
 3719:d=8  hl=2 l=  62 cons: SEQUENCE
 3721:d=9  hl=2 l=   3 prim: OBJECT            :commonName
 3726:d=9  hl=2 l=  55 prim: UTF8STRING        :#<SystemVariable:0x00000
 3783:d=6  hl=2 l=   1 prim: INTEGER           :02
 3786:d=5  hl=2 l=  11 cons: SEQUENCE
 3788:d=6  hl=2 l=   9 prim: OBJECT            :sha256
 3799:d=5  hl=2 l= 105 cons: cont [ 0 ]
 3801:d=6  hl=2 l=  24 cons: SEQUENCE
 3803:d=7  hl=2 l=   9 prim: OBJECT            :contentType
 3814:d=7  hl=2 l=  11 cons: SET
 3816:d=8  hl=2 l=   9 prim: OBJECT            :pkcs7-data
 3827:d=6  hl=2 l=  28 cons: SEQUENCE
 3829:d=7  hl=2 l=   9 prim: OBJECT            :signingTime
 3840:d=7  hl=2 l=  15 cons: SET
 3842:d=8  hl=2 l=  13 prim: UTCTIME           :190515212555Z
 3857:d=6  hl=2 l=  47 cons: SEQUENCE
 3859:d=7  hl=2 l=   9 prim: OBJECT            :messageDigest
 3870:d=7  hl=2 l=  34 cons: SET
 3872:d=8  hl=2 l=  32 prim: OCTET STRING      [HEX DUMP]:50508CC996CD93
 3906:d=5  hl=2 l=  10 cons: SEQUENCE
 3908:d=6  hl=2 l=   8 prim: OBJECT            :ecdsa-with-SHA256
 3918:d=5  hl=2 l=  71 prim: OCTET STRING      [HEX DUMP]:3045022006D85B
```

D.2.3. MASA to Registrar

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

```
-----BEGIN CMS-----
MIIGsgYJKoZIhvcNAQcCoIIGozCCBp8CAQExDTALBglghkgBZQMEAgEwggNABgkq
hkiG9w0BBwGgggMxBIIDLXsiaWV0Zi12b3VjaGVyOnZvdWNoZXIiOnsiYXNzZXJ0
aW9uIjoibG9nZ2VkIiwiY3JlYXRlZC1vbiI6IjIwMTktMDUtMTZUMDI6NTE6NDIu
Njk3KzAwOjAwIiwic2VyaWFsLW51bWJlciI6IjAwLWQwLWU1LTAyLTAwLTJkIiwi
bm9uY2UiOiJHWmUtT2pvZXJwS0VNNFNNN1N6UzlnIiwicGlubmVkLWRvbWFpbi1j
ZXJ0IjoiTUlJQjBUQ0NBVmFnQXdJQkFnSUJBakFLQmdncWhrak9QUVFEQXpCeE1S
SXdFQVlLQ1pJbWlaUHlMR1FCR1JZQ1kyRXhHVEFYQmdvSmtpYUprL0lzWkFFWkZn
bHpZVzVrWld4dFlXNHhRREErQmdOVkJBTU1OeU04VTNsemRHVnRWbUZ5YVdGaWJH
VTZNSGd3TURBd01EQXdOR1k1TVRGaE1ENGdWVzV6ZEhKMWJtY2dSbTkxYm5SaGFX
NGdRMEV3SGhjTk1UY3hNVEEzTWpNME5USTRXaGNOTVRreE1UQTNNak0wTlRJNFdq
QkRNUkl3RUFZS0NaSW1pWlB5TEdRQkdSWUNZMkV4R1RBWEJnb0praWFKay9Jc1pB
RVpGZ2x6WVc1a1pXeHRZVzR4RWpBUUJnTlZCQU1NQ1d4dlkyRnNhRzl6ZERCWk1C
TUdCeXFHU000OUFnRUdDQ3FHU000OUF3RUhBMElBQkpabFVISTB1cC9sM2VaZjl2
Q0JiK2xJbm9FTUVnYzdSbytYWkN0akFJMENENEMWZKZkpSL2hJeXlEbUhXeVlpTkZi
UkNIOWZ5YXJma3pnWDRwMHpUaXpxakRUQUxNQWtHQTFVZEV3UUNNQUF3Q2dZSUtv
Wkl6ajBFQXdNRGFRQXdaZ0l4QUxRTU51cmY4dHY1MGxST0Q1RFFYSEVPSkpOVzNR
VjJnOVFFZERTazJNWStBb1NyQlNtR1NOamg0b2xFT2hFdUxnSXhBSjRuV2ZOdytC
amJabUtpSWlVRWNUd0hNaEdWWGFNSFkvRjduMzl3d0tjQkJTT25kTlBxQ3BPRUxs
NmJxM0NacVE9PSJ9faCCAfUwggHxMIIBeKADAgECAgQjzIkTMAoGCCqGSM49BAMC
ME0xEjAQBgoJkiaJk/IsZAEZFgJjYTEZMBcGCgmSJomT8ixkARkWCXNhbmRlbG1h
bjEcMBoGA1UEAwwTVW5zdHJ1bmcgSGlnaHdheSBDQTAeFw0xOTA0MjMyMzIxMDda
Fw0xOTA1MjQwOTIxMDdaMGYxDzANBgNVBAYTBkNhbmFkYTESMBAGA1UECgwJU2Fu
ZGVsbWFuMRMwEQYDVQQLDApob25leWR1a2VzMSowKAYDVQQDDCFtYXNhLmhvbmV5
ZHVrZXMuc2FuZGVsbWFuLmNhIE1BU0EwdjAQBgcqhkjOPQIBBgUrgQQAIgNiAAQ1
/2UdVp8zVmgADoBNql7LcPlJsEaaVAogYEqABikNOkoTO3oPjIQfNBxtGfRFzBXx
gihzkTH58r8SW1L/Mej8AFqhB4SZyyjmWURdzD71Ju0M+tRritWf7T+QGaE+fcWj
EDAOMAwGA1UdEwEB/wQCMAAwCgYIKoZIzj0EAwIDZwAwZAIwOMlNOMNYEZo4yLW4
iRltDL8uirmjMdtVmmVYzqYHSindjP0a3pXQkQZ5LLARoSRWAjBTxsnv6ya5HpZI
IWcspDPZGlOSDPm7nuRJSDkgWqevxLI4+9nmIhsfMBsDvz1DJhAxggFMMIIBSAIB
ATBVME0xEjAQBgoJkiaJk/IsZAEZFgJjYTEZMBcGCgmSJomT8ixkARkWCXNhbmRl
bG1hbjEcMBoGA1UEAwwTVW5zdHJ1bmcgSGlnaHdheSBDQQIEI8yJEzALBglghkgB
ZQMEAgGgaTAYBgkqhkiG9w0BCQMxCwYJKoZIhvcNAQcBMBwGCSqGSIb3DQEJBTEP
Fw0xOTA1MTYwMjUxNDJaMC8GCSqGSIb3DQEJBDEiBCCYRh4i21QjEjEk8leRLSVA
x/EVY5g1bM40QM21oR4c2DAKBggqhkjOPQQDAgRoMGYCMQCYYOiSbIlED4nAN0iL
e4S8ixWAZ9SXpGv77bB/G4fTTVTN35mnAeYBfeNfhC6/kOECMQDqlkCmwQJQDdEL
asj1ISinJ/FnZjjgOMz9MXOmGNGIfw9v2VBb9mVyhsOSMcqlVig=
-----END CMS-----
```


file: examples/voucher_00-D0-E5-02-00-2D.pkcs

The ASN1 decoding of the artifact:

```
 0:d=0  hl=4 l=1714 cons: SEQUENCE
 4:d=1  hl=2 l=   9 prim: OBJECT            :pkcs7-signedData
15:d=1  hl=4 l=1699 cons: cont [ 0 ]
19:d=2  hl=4 l=1695 cons: SEQUENCE
23:d=3  hl=2 l=   1 prim: INTEGER           :01
```

```
   26:d=3  hl=2 l=  13 cons: SET
   28:d=4  hl=2 l=  11 cons: SEQUENCE
   30:d=5  hl=2 l=   9 prim: OBJECT            :sha256
   41:d=3  hl=4 l= 832 cons: SEQUENCE
   45:d=4  hl=2 l=   9 prim: OBJECT            :pkcs7-data
   56:d=4  hl=4 l= 817 cons: cont [ 0 ]
   60:d=5  hl=4 l= 813 prim: OCTET STRING      :{"ietf-voucher:voucher":
  877:d=3  hl=4 l= 501 cons: cont [ 0 ]
  881:d=4  hl=4 l= 497 cons: SEQUENCE
  885:d=5  hl=4 l= 376 cons: SEQUENCE
  889:d=6  hl=2 l=   3 cons: cont [ 0 ]
  891:d=7  hl=2 l=   1 prim: INTEGER           :02
  894:d=6  hl=2 l=   4 prim: INTEGER           :23CC8913
  900:d=6  hl=2 l=  10 cons: SEQUENCE
  902:d=7  hl=2 l=   8 prim: OBJECT            :ecdsa-with-SHA256
  912:d=6  hl=2 l=  77 cons: SEQUENCE
  914:d=7  hl=2 l=  18 cons: SET
  916:d=8  hl=2 l=  16 cons: SEQUENCE
  918:d=9  hl=2 l=  10 prim: OBJECT            :domainComponent
  930:d=9  hl=2 l=   2 prim: IA5STRING         :ca
  934:d=7  hl=2 l=  25 cons: SET
  936:d=8  hl=2 l=  23 cons: SEQUENCE
  938:d=9  hl=2 l=  10 prim: OBJECT            :domainComponent
  950:d=9  hl=2 l=   9 prim: IA5STRING         :sandelman
  961:d=7  hl=2 l=  28 cons: SET
  963:d=8  hl=2 l=  26 cons: SEQUENCE
  965:d=9  hl=2 l=   3 prim: OBJECT            :commonName
  970:d=9  hl=2 l=  19 prim: UTF8STRING        :Unstrung Highway CA
  991:d=6  hl=2 l=  30 cons: SEQUENCE
  993:d=7  hl=2 l=  13 prim: UTCTIME           :190423232107Z
 1008:d=7  hl=2 l=  13 prim: UTCTIME           :190524092107Z
 1023:d=6  hl=2 l= 102 cons: SEQUENCE
 1025:d=7  hl=2 l=  15 cons: SET
 1027:d=8  hl=2 l=  13 cons: SEQUENCE
 1029:d=9  hl=2 l=   3 prim: OBJECT            :countryName
 1034:d=9  hl=2 l=   6 prim: PRINTABLESTRING   :Canada
 1042:d=7  hl=2 l=  18 cons: SET
 1044:d=8  hl=2 l=  16 cons: SEQUENCE
 1046:d=9  hl=2 l=   3 prim: OBJECT            :organizationName
 1051:d=9  hl=2 l=   9 prim: UTF8STRING        :Sandelman
 1062:d=7  hl=2 l=  19 cons: SET
 1064:d=8  hl=2 l=  17 cons: SEQUENCE
 1066:d=9  hl=2 l=   3 prim: OBJECT            :organizationalUnitName
 1071:d=9  hl=2 l=  10 prim: UTF8STRING        :honeydukes
 1083:d=7  hl=2 l=  42 cons: SET
 1085:d=8  hl=2 l=  40 cons: SEQUENCE
 1087:d=9  hl=2 l=   3 prim: OBJECT            :commonName
 1092:d=9  hl=2 l=  33 prim: UTF8STRING        :masa.honeydukes.sandelma
```

```
 1127:d=6  hl=2 l= 118 cons: SEQUENCE
 1129:d=7  hl=2 l=  16 cons: SEQUENCE
 1131:d=8  hl=2 l=   7 prim: OBJECT            :id-ecPublicKey
 1140:d=8  hl=2 l=   5 prim: OBJECT            :secp384r1
 1147:d=7  hl=2 l=  98 prim: BIT STRING
 1247:d=6  hl=2 l=  16 cons: cont [ 3 ]
 1249:d=7  hl=2 l=  14 cons: SEQUENCE
 1251:d=8  hl=2 l=  12 cons: SEQUENCE
 1253:d=9  hl=2 l=   3 prim: OBJECT            :X509v3 Basic Constraints
 1258:d=9  hl=2 l=   1 prim: BOOLEAN           :255
 1261:d=9  hl=2 l=   2 prim: OCTET STRING      [HEX DUMP]:3000
 1265:d=5  hl=2 l=  10 cons: SEQUENCE
 1267:d=6  hl=2 l=   8 prim: OBJECT            :ecdsa-with-SHA256
 1277:d=5  hl=2 l= 103 prim: BIT STRING
 1382:d=3  hl=4 l= 332 cons: SET
 1386:d=4  hl=4 l= 328 cons: SEQUENCE
 1390:d=5  hl=2 l=   1 prim: INTEGER           :01
 1393:d=5  hl=2 l=  85 cons: SEQUENCE
 1395:d=6  hl=2 l=  77 cons: SEQUENCE
 1397:d=7  hl=2 l=  18 cons: SET
 1399:d=8  hl=2 l=  16 cons: SEQUENCE
 1401:d=9  hl=2 l=  10 prim: OBJECT            :domainComponent
 1413:d=9  hl=2 l=   2 prim: IA5STRING         :ca
 1417:d=7  hl=2 l=  25 cons: SET
 1419:d=8  hl=2 l=  23 cons: SEQUENCE
 1421:d=9  hl=2 l=  10 prim: OBJECT            :domainComponent
 1433:d=9  hl=2 l=   9 prim: IA5STRING         :sandelman
 1444:d=7  hl=2 l=  28 cons: SET
 1446:d=8  hl=2 l=  26 cons: SEQUENCE
 1448:d=9  hl=2 l=   3 prim: OBJECT            :commonName
 1453:d=9  hl=2 l=  19 prim: UTF8STRING        :Unstrung Highway CA
 1474:d=6  hl=2 l=   4 prim: INTEGER           :23CC8913
 1480:d=5  hl=2 l=  11 cons: SEQUENCE
 1482:d=6  hl=2 l=   9 prim: OBJECT            :sha256
 1493:d=5  hl=2 l= 105 cons: cont [ 0 ]
 1495:d=6  hl=2 l=  24 cons: SEQUENCE
 1497:d=7  hl=2 l=   9 prim: OBJECT            :contentType
 1508:d=7  hl=2 l=  11 cons: SET
 1510:d=8  hl=2 l=   9 prim: OBJECT            :pkcs7-data
 1521:d=6  hl=2 l=  28 cons: SEQUENCE
 1523:d=7  hl=2 l=   9 prim: OBJECT            :signingTime
 1534:d=7  hl=2 l=  15 cons: SET
 1536:d=8  hl=2 l=  13 prim: UTCTIME           :190516025142Z
 1551:d=6  hl=2 l=  47 cons: SEQUENCE
 1553:d=7  hl=2 l=   9 prim: OBJECT            :messageDigest
 1564:d=7  hl=2 l=  34 cons: SET
 1566:d=8  hl=2 l=  32 prim: OCTET STRING      [HEX DUMP]:98461E22DB5423
 1600:d=5  hl=2 l=  10 cons: SEQUENCE
```

```
 1602:d=6  hl=2 l=   8 prim: OBJECT            :ecdsa-with-SHA256
 1612:d=5  hl=2 l= 104 prim: OCTET STRING      [HEX DUMP]:30660231009860
```

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

Kent Watsen
Watsen Networks

Email: kent+ietf@watsen.net