

ANIMA WG
Internet-Draft
Intended status: Standards Track
Expires: October 15, 2018

M. Pritikin
Cisco
M. Richardson
Sandelman
M. Behringer

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

**Bootstrapping Remote Secure Key Infrastructures (BRSKI)
draft-ietf-anima-bootstrapping-keyinfra-14**

Abstract

This document specifies automated bootstrapping of a remote secure key infrastructure (BRSKI) using manufacturer installed X.509 certificate, in combination with a manufacturer's authorizing service, both online and offline. Bootstrapping a new device can occur using a routable address and a cloud service, or using only link-local connectivity, or on limited/disconnected networks. Support for lower security models, including devices with minimal identity, is described for legacy reasons but not encouraged. Bootstrapping is complete when the cryptographic identity of the new key infrastructure is successfully deployed to the device but the established secure connection can be used to deploy a locally issued certificate to the device as well.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

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

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

This Internet-Draft will expire on October 15, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

- [1. Introduction](#) [4](#)
- [1.1. Prior Bootstrapping Approaches](#) [5](#)
- [1.2. Terminology](#) [6](#)
- [1.3. Scope of solution](#) [9](#)
- [1.3.1. Support environment](#) [9](#)
- [1.3.2. Constrained environments](#) [10](#)
- [1.3.3. Network Access Controls](#) [11](#)
- [1.4. Leveraging the new key infrastructure / next steps](#) . . . [11](#)
- 1.5. Requirements for Autonomic Network Infrastructure (ANI) devices [11](#)
- [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](#) [18](#)
- [2.5. Architectural Components](#) [20](#)
- [2.5.1. Pledge](#) [20](#)
- [2.5.2. Join Proxy](#) [20](#)
- [2.5.3. Domain Registrar](#) [20](#)
- [2.5.4. Manufacturer Service](#) [20](#)
- [2.5.5. Public Key Infrastructure \(PKI\)](#) [20](#)
- [2.6. Certificate Time Validation](#) [21](#)
- [2.6.1. Lack of realtime clock](#) [21](#)
- [2.6.2. Infinite Lifetime of IDevID](#) [23](#)
- [2.7. Cloud Registrar](#) [23](#)
- [2.8. Determining the MASA to contact](#) [23](#)
- [3. Voucher-Request artifact](#) [24](#)
- [3.1. Tree Diagram](#) [25](#)
- [3.2. Examples](#) [25](#)

3.3.	YANG Module	27
4.	Proxying details (Pledge - Proxy - Registrar)	30
4.1.	Pledge discovery of Proxy	31
4.1.1.	Proxy GRASP announcements	32
4.2.	CoAP connection to Registrar	33
4.3.	Proxy discovery of Registrar	33
5.	Protocol Details (Pledge - Registrar - MASA)	35
5.1.	BRSKI-EST TLS establishment details	36
5.2.	Pledge Requests Voucher from the Registrar	37
5.3.	BRSKI-MASA TLS establishment details	38
5.4.	Registrar Requests Voucher from MASA	39
5.4.1.	MASA renewal of expired vouchers	40
5.4.2.	MASA verification of voucher-request signature consistency	40
5.4.3.	MASA authentication of registrar (certificate)	41
5.4.4.	MASA revocation checking of registrar (certificate)	41
5.4.5.	MASA verification of pledge prior-signed-voucher- request	42
5.4.6.	MASA pinning of registrar	42
5.4.7.	MASA nonce handling	42
5.5.	MASA Voucher Response	42
5.5.1.	Pledge voucher verification	44
5.5.2.	Pledge authentication of provisional TLS connection	44
5.6.	Pledge Voucher Status Telemetry	45
5.7.	Registrar audit log request	46
5.7.1.	MASA audit log response	47
5.7.2.	Registrar audit log verification	49
5.8.	EST Integration for PKI bootstrapping	50
5.8.1.	EST Distribution of CA Certificates	50
5.8.2.	EST CSR Attributes	51
5.8.3.	EST Client Certificate Request	52
5.8.4.	Enrollment Status Telemetry	52
5.8.5.	Multiple certificates	53
5.8.6.	EST over CoAP	53
6.	Reduced security operational modes	53
6.1.	Trust Model	53
6.2.	Pledge security reductions	54
6.3.	Registrar security reductions	55
6.4.	MASA security reductions	56
7.	IANA Considerations	56
7.1.	Well-known EST registration	56
7.2.	PKIX Registry	57
7.3.	Voucher Status Telemetry	57
7.4.	DNS Service Names	57
7.5.	MUD File Extension for the MASA server	58
8.	Privacy Considerations	58
8.1.	MASA audit log	58
9.	Security Considerations	58

- [9.1. Freshness in Voucher-Requests](#) [60](#)
- [9.2. Trusting manufacturers](#) [61](#)
- [10. Acknowledgements](#) [62](#)
- [11. References](#) [62](#)
- [11.1. Normative References](#) [62](#)
- [11.2. Informative References](#) [65](#)
- [Appendix A. IPv4 and non-ANI operations](#) [67](#)
- [A.1. IPv4 Link Local addresses](#) [67](#)
- [A.2. Use of DHCPv4](#) [67](#)
- [Appendix B. mDNS / DNSSD proxy discovery options](#) [67](#)
- [Appendix C. IPIP Join Proxy mechanism](#) [68](#)
- [C.1. Multiple Join networks on the Join Proxy side](#) [69](#)
- [C.2. Automatic configuration of tunnels on Registrar](#) [69](#)
- [C.3. Proxy Neighbor Discovery by Join Proxy](#) [69](#)
- [C.4. Use of connected sockets; or IP_PKTINFO for CoAP on Registrar](#) [70](#)
- [C.5. Use of socket extension rather than virtual interface](#) [70](#)
- [Appendix D. MUD Extension](#) [71](#)
- [Appendix E. Example Vouchers](#) [73](#)
- [E.1. Keys involved](#) [73](#)
- [E.1.1. MASA key pair for voucher signatures](#) [73](#)
- [E.1.2. Manufacturer key pair for IDevID signatures](#) [73](#)
- [E.1.3. Registrar key pair](#) [74](#)
- [E.1.4. Pledge key pair](#) [76](#)
- [E.2. Example process](#) [77](#)
- [E.2.1. Pledge to Registrar](#) [77](#)
- [E.2.2. Registrar to MASA](#) [83](#)
- [E.2.3. MASA to Registrar](#) [89](#)
- [Authors' Addresses](#) [94](#)

1. Introduction

BRSKI provides a solution for secure zero-touch (automated) bootstrap of virgin (untouched) devices that are called pledges in this document. These pledges need to discover (or be discovered by) an element of the network domain to which the pledge belongs to perform the bootstrap. This element (device) is called the registrar. Before any other operation, pledge and registrar need to establish mutual trust:

1. Registrar authenticating the pledge: "Who is this device? What is its identity?"
2. Registrar authorizing the pledge: "Is it mine? Do I want it? What are the chances it has been compromised?"
3. Pledge authenticating the registrar: "What is this registrar's identity?"

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

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

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

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

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 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 [[RFC2119](#)].

The following terms are defined for clarity:

- domainID:** The domain IDentity is the 160-bit SHA-1 hash of the BIT STRING of the subjectPublicKey of the pinned-domain-cert leaf, i.e. the Registrars' certificate. This is consistent with the subject key identifier ([Section 4.2.1.2 \[RFC5280\]](#)).
- drop ship:** The physical distribution of equipment containing the "factory default" configuration to a final destination. In zero-touch scenarios there is no staging or pre-configuration during drop-ship.
- imprint:** The process where a device obtains the cryptographic key material to identify and trust future interactions with a network. This term is taken from Konrad Lorenz's work in biology with new ducklings: during a critical period, the duckling would assume that anything that looks like a mother duck is in fact their mother. An equivalent for a device is to obtain the fingerprint of the network's root certification authority certificate. A device that imprints on an attacker suffers a similar fate to a duckling that imprints on a hungry wolf. Securely imprinting is a primary focus of this document [[imprinting](#)]. The analogy to Lorenz's work was first noted in [[Stajano99theresurrecting](#)].
- enrollment:** The process where a device presents key material to a network and acquires a network specific identity. For example when a certificate signing request is presented to a certification authority and a certificate is obtained in response.
- Pledge:** The prospective device, which has an identity installed at the factory.
- Voucher:** A signed artifact from the MASA that indicates to a pledge the cryptographic identity of the registrar it should trust. There are different types of vouchers depending on how that trust is asserted. Multiple voucher types are defined in [[I-D.ietf-anima-voucher](#)]
- 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 [[I-D.ietf-anima-voucher](#)]

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

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

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

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

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 [\[I-D.ietf-anima-autonomic-control-plane\]](#). This document details specific requirements for pledges, proxies and registrars when they are part of an ANI.

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

[1.3.](#) Scope of solution

[1.3.1.](#) Support environment

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

owners, and the eventual domain owner will not know a-priori which device will go to which location.

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

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

1.3.2. Constrained environments

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

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

1.4. Leveraging the new key infrastructure / next steps

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

Services that benefit from this:

- o Device management.
- o Routing authentication.
- o Service discovery.

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

1.5. Requirements for Autonomic Network Infrastructure (ANI) devices

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

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

- o BRSKI: Request Voucher
- o EST: CA Certificates Request
- o EST: CSR Attributes
- o EST: Client Certificate Request
- o BRSKI: Enrollment status Telemetry

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

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

2. Architectural Overview

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

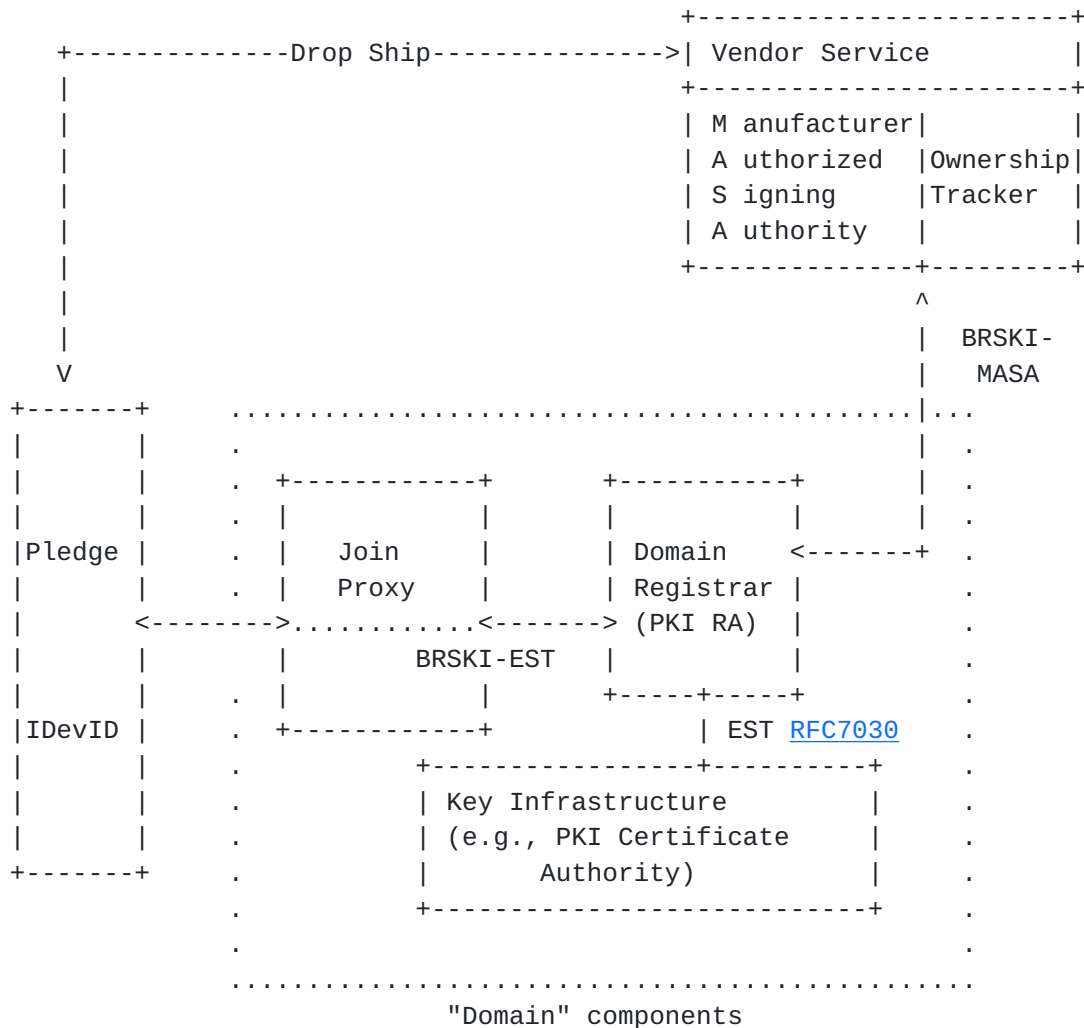


Figure 1

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.

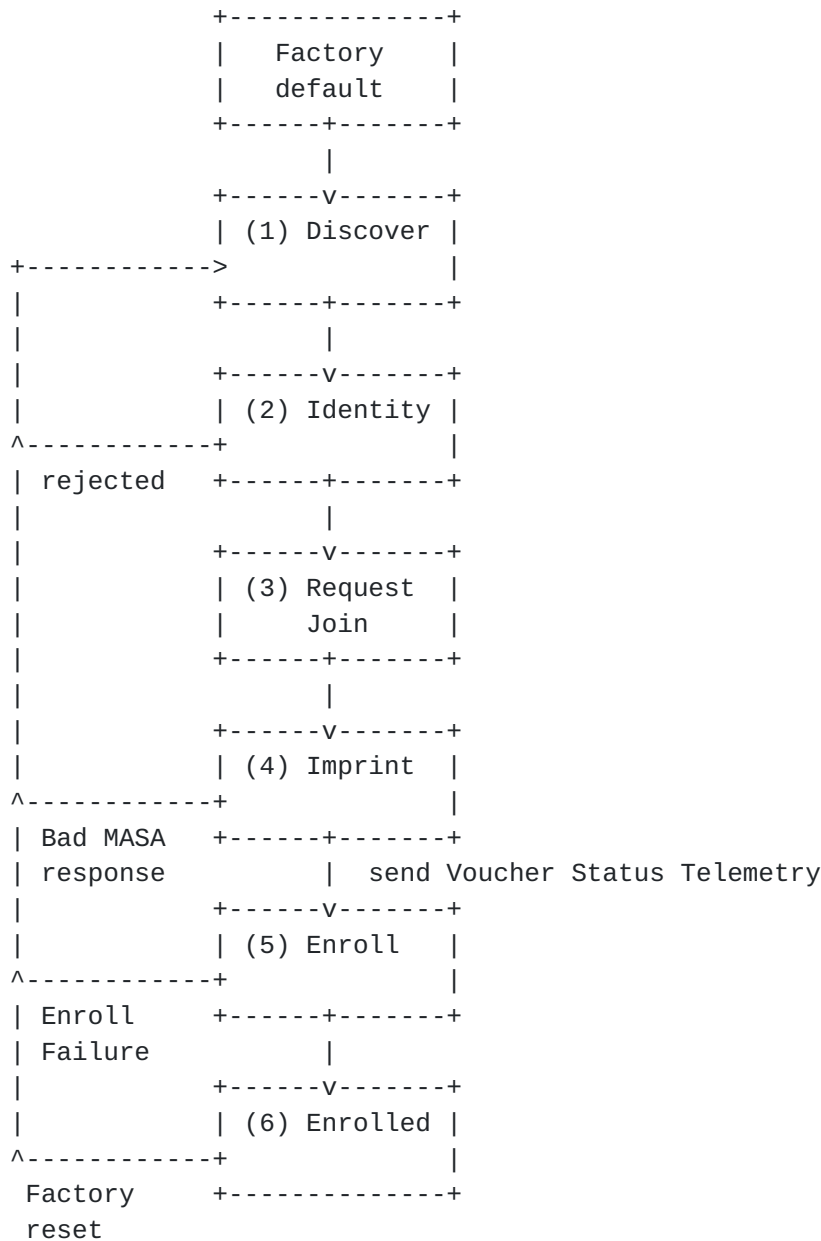


Figure 2

State descriptions for the pledge are as follows:

1. Discover a communication channel to a registrar.

2. Identify itself. This is done by presenting an X.509 IDevID credential to the discovered registrar (via the proxy) in a TLS handshake. (The registrar credentials are only provisionally accepted at this time).
3. Request to join the discovered registrar. A unique nonce can be included ensuring that any responses can be associated with this particular bootstrapping attempt.
4. Imprint on the registrar. This requires verification of the manufacturer service provided voucher. A voucher contains sufficient information for the pledge to complete authentication of a registrar. (The embedded 'pinned-domain-certificate' enables the pledge to finish authentication of the registrar TLS server certificate).
5. Enroll. By accepting the domain specific information from a registrar, and by obtaining a domain certificate from a registrar using a standard enrollment protocol, e.g. Enrollment over Secure Transport (EST) [[RFC7030](#)].
6. The pledge is now a member of, and can be managed by, the domain and will only repeat the discovery aspects of bootstrapping if it is returned to factory default settings.

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

2.2. Secure Imprinting using Vouchers

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

The format and cryptographic mechanism of vouchers is described in detail in [[I-D.ietf-anima-voucher](#)].

Vouchers provide a flexible mechanism to secure imprinting: the pledge device only imprints when a voucher can be validated. At the lowest security levels the MASA server can indiscriminately issue vouchers and log claims of ownership by domains. At the highest security levels issuance of vouchers can be integrated with complex sales channel integrations that are beyond the scope of this document. The sales channel integration would verify actual (legal) ownership of the pledge by the domain. This provides the flexibility for a number of use cases via a single common protocol mechanism on

the pledge and registrar devices that are to be widely deployed in the field. The MASA services have the flexibility to leverage either the currently defined claim mechanisms or to experiment with higher or lower security levels.

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

2.3. Initial Device Identifier

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

1. Uniquely identifying the pledge by the Distinguished Name (DN) and subjectAltName (SAN) parameters in the IDeVID. The unique identification of a pledge in the voucher objects are derived from those parameters as described below.
2. Securely authenticating the pledges identity via TLS connection to registrar. This provides protection against cloned/fake pledged.
3. Secure auto-discovery of the pledges MASA by the registrar via the MASA URI in IDeVID as explained below.
4. (Optionally) communicating the MUD URL (see [Appendix D](#)).
5. (Optional) Signing of voucher-request by the pledges IDeVID to enable MASA to generate voucher only to a registrar that has a connection to the pledge.
6. Authorizing pledge (via registrar) to receive certificate from domain CA, by signing the Certificate Signing Request (CSR).

2.3.1. Identification of the Pledge

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

The following fields are defined in [[IDeVID](#)] and [[RFC5280](#)]:

- 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, formatted according to [RFC4514](#) rules.
2. The HardwareModuleName hwSerialNum OCTET STRING, base64 encoded.

[2.3.2](#). MASA URI extension

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

Any Internationalized Resource Identifiers (IRIs) MUST be mapped to URIs as specified in [Section 3.1 of RFC3987](#) before they are placed in the certificate extension. The URI provides the authority information. The BRSKI "/.well-known" tree ([RFC5785](#)) is described in [Section 5](#).

The new extension is identified as follows:

<CODE BEGINS>

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

DEFINITIONS IMPLICIT TAGS ::= BEGIN

-- EXPORTS ALL --

IMPORTS

EXTENSION

FROM PKIX-CommonTypes-2009

```
{ iso(1) identified-organization(3) dod(6) internet(1)
security(5) mechanisms(5) pkix(7) id-mod(0)
id-mod-pkixCommon-02(57) }
```

id-pe

FROM PKIX1Explicit-2009

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

MASACertExtensions EXTENSION ::= { ext-MASAURL, ... }

ext-MASAURL EXTENSION ::= { SYNTAX MASAUURLSyntax

IDENTIFIED BY id-pe-masa-url }

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

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

```

Figure 3

[2.5.](#) 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](#), with an optional stateless IPIP proxy mechanism described in [Appendix C](#).

[2.5.3.](#) Domain Registrar

The domain's registrar operates as the BRSKI-MASA client when requesting vouchers from the MASA (see [Section 5.3](#)). 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 5.8](#)).

The registrar uses an Implicit Trust Anchor database for authenticating the BRSKI-MASA TLS connection MASA server 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 5.4](#) and [Section 5.5](#), and an ownership tracking/auditing function described in [Section 5.6](#) and [Section 5.7](#).

[2.5.5.](#) Public Key Infrastructure (PKI)

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

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

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

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

[2.6.](#) Certificate Time Validation

[2.6.1.](#) Lack of realtime clock

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

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

- o Initially the pledge does not know the current time.
- o Bootstrapping pledges that have a Realtime Clock (RTC), SHOULD use it to verify certificate validity. However, they MUST be prepared for the recognize that the RTC might be completely wrong when a RTC battery fails and resets to an origin time (e.g., Jan. 1, 1970)
- o If the pledge has any stable storage (such as from where firmware is loaded) then it SHOULD assume that the clock CAN NOT be before the date at which the firmware or the the storage was last time stamped. The pledge SHOULD NOT update the timestamps in any file systems until it has a secure time source. This provides an earliest date which is reasonable. Call this the current reasonable date (CRD). This value MUST NOT be used for any future Registration attempt. The current reasonable date (CRD) may only increase during a single attempt.
- o The pledge is exposed to dates in the following five places (registrar certificate, notBefore and notAfter. Voucher created-on, and expires-on. Additionally, CMS signatures contain a signingTime)
- o During the initial connection with the registrar, the pledge sees the registrar's Certificate. It has an inception date (notBefore)

and an expiry date (notAfter). It is reasonable that the notBefore date be after the pledge's current working reasonable date. It is however, suspicious for the notAfter date to be before the pledge's current reasonable date. No action is recommended, other than an internal audit entry for this.

- o If the notBefore date of the registrar's certificate is newer than the pledge's reasonable date, then it MAY update it's current reasonable date to the notBefore value.
- o After the voucher request process, the pledge will have a voucher. It can validate the signature on the voucher, as it has been (by literal construction) provided with the MASA's key as a trust anchor. The time values (created-on, expires-on) in the voucher can not in general be validated as the pledge has no certain real time clock. There are some reasonable assumptions that can be made: the voucher's expires-on time can not be prior to the pledge's current reasonable date. For nonceless vouchers, the voucher's created-on time COULD be earlier if the as well if a long-lived voucher was obtained some time in the past, and the pledge has since gone through a firmware update and factory reset.
- o If the voucher contains a nonce then the pledge MUST confirm the nonce matches the original pledge voucher-request. This ensures the voucher is fresh. See / ([Section 5.2](#)). In that case, the voucher's created-on date MUST NOT be prior to the pledge's current reasonable date. In addition, when there is a valid nonce, the current reasonable date MAY be incremented to that of the CMS signingTime.
- o Once the voucher is accepted the validity period of the pinned-domain-cert in the voucher now serves as a valid time window. As explained in [Section 5.4.4](#), the MASA has checked the registrar's certificate against real clocks, the endorsement of the MASA allows the pledge to treat the notBefore and notAfter dates as being constraints on any subsequent certificate validity periods that may need to be checked: for instance, validating peer certificates during ANIMA ACP setup.
- o When accepting an enrollment certificate the validity period within the new certificate is assumed to be valid by the pledge. The pledge is now willing to use this credential for client authentication.

2.6.2. 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 $N \leq 3$ years, rendering replacement pledges from storage useless after N years unless registrars support ignoring such a lifetime.

2.7. Cloud Registrar

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

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

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

If the pledge uses a well known URI for contacting a cloud registrar an Implicit Trust Anchor database (see [\[RFC7030\]](#)) MUST be used to authenticate service as described in [\[RFC6125\]](#). This is consistent with the human user configuration of an EST server URI in [\[RFC7030\]](#) which also depends on [RFC6125](#).

2.8. Determining the MASA to contact

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

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

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

It can be operationally difficult to ensure the necessary X.509 extensions are in the pledge's IDevID due to the difficulty of aligning current pledge manufacturing with software releases and development. As a final fallback the registrar MAY be manually configured or distributed with a MASA URL for each 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 server.

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

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

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

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

3.1. Tree Diagram

The following tree diagram illustrates a high-level view of a voucher-request document. The notation used in this diagram is described in [[I-D.ietf-anima-voucher](#)]. Each node in the diagram is fully described by the YANG module in [Section 3.3](#). Please review the YANG module for a detailed description of the voucher-request format.

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

```

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

```

3.2. Examples

This section provides voucher-request examples for illustration purposes. These examples conform to the encoding rules defined in [[RFC7951](#)].

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

```

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

```

Example (2) The following example illustrates a registrar voucher-request. The 'prior-signed-voucher-request' leaf is populated with the pledge's voucher-request (such as the

prior example). The pledge's voucher-request, if a signed artifact with a CMS format signature is a binary object. In the JSON encoding used here it must be base64 encoded. The nonce, created-on and assertion is carried forward. serial-number is extracted from the pledge's Client Certificate from the TLS connection. See [Section 5.4](#).

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

Example (3) The following example illustrates a registrar voucher-request. The 'prior-signed-voucher-request' leaf is not populated with the pledge's voucher-request nor is the nonce leaf. This form might be used by a registrar requesting a voucher when the pledge 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 5.4](#).

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

Example (4) The following example illustrates a registrar voucher-request. The 'prior-signed-voucher-request' leaf is not populated with the pledge voucher-request because the pledge did not sign its own request. This form might be used when more constrained pledges are being deployed. The nonce is populated from the pledge's request. See [Section 5.4](#).


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

3.3. YANG Module

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

```
<CODE BEGINS> file "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 YYYY: Voucher Profile for Bootstrapping Protocols";
  }

  organization
    "IETF ANIMA Working Group";

  contact
    "WG Web: <http://tools.ietf.org/wg/anima/>
    WG List: <mailto:anima@ietf.org>
    Author: Kent Watsen
```



```
<mailto:kwatsen@juniper.net>
Author: Max Pritikin
       <mailto:pritikin@cisco.com>
Author: Michael Richardson
       <mailto:mcr+ietf@sandelman.ca>
Author: Toerless Eckert
       <mailto:tte+ietf@cs.fau.de>";
```

description

"This module module defines the format for a voucher request.
It is a superset of the voucher itself.
This artifact may be optionally signed.
It provides content to the MASA for consideration
during a voucher request.

The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL NOT',
'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'MAY', and 'OPTIONAL' in
the module text are to be interpreted as described in [RFC 2119](#).

Copyright (c) 2017 IETF Trust and the persons identified as
authors of the code. All rights reserved.

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

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

```
revision "2018-02-14" {
  description
    "Initial version";
  reference
    "RFC XXXX: Voucher Profile for Bootstrapping Protocols";
}

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

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



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

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

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

    leaf prior-signed-voucher-request {
      type binary;
      description
        "If it is necessary to change a voucher, or re-sign and
        forward a voucher that was previously provided along a
        protocol path, then the previously signed voucher SHOULD be
        included in this field.

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

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

    leaf proximity-registrar-cert {
      type binary;
      description
        "An X.509 v3 certificate structure as specified by RFC 5280,
        Section 4 encoded using the ASN.1 distinguished encoding
        rules (DER), as specified in ITU-T X.690.

        The first certificate in the Registrar TLS server
        certificate_list sequence (see [RFC5246]) presented by
        the Registrar to the Pledge. This MUST be populated in a
```


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

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

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

4.1. Pledge discovery of Proxy

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

To discover the proxy the pledge performs the following actions:

1. MUST: Obtains a local address using IPv6 methods as described in [[RFC4862](#)] IPv6 Stateless Address AutoConfiguration. Use of [[RFC4941](#)] temporary addresses is encouraged. A new temporary address SHOULD be allocated whenever the discovery process is forced to restart due to failures. Pledges will generally prefer use of IPv6 Link-Local addresses, and discovery of proxy will be by Link-Local mechanisms. IPv4 methods are described in [Appendix A](#)
2. MUST: Listen for GRASP M_FLOOD ([[I-D.ietf-anima-grasp](#)]) announcements of the objective: "AN_Proxy". See section [Section 4.1.1](#) for the details of the objective. The pledge MAY listen concurrently for other sources of information, see [Appendix B](#).

Once a proxy is discovered the pledge communicates with a registrar through the proxy using the bootstrapping protocol defined in [Section 5](#).

While the GRASP M_FLOOD mechanism is passive for the pledge, the optional other methods (mDNS, and IPv4 methods) are active. The pledge SHOULD run those methods in parallel with listening to for the M_FLOOD. The active methods SHOULD exponentially back-off to a maximum of one hour to avoid overloading the network with discovery

attempts. Detection of change of physical link status (ethernet carrier for instance) SHOULD reset the exponential back off.

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

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

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

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

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

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

4.1.1. Proxy GRASP announcements

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

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

Figure 6b: Proxy Discovery

The formal CDDL definition is:


```

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

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

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

locator      = [ O_IPv6_LOCATOR, ipv6-address,
                 transport-proto, port-number ]
ipv6-address = the v6 LL of the Proxy
transport-proto = IPPROTO_TCP / IPPROTO_UDP / IPPROTO_IPV6
port-number   = selected by Proxy

```

Figure 6c: AN_Proxy CDDL

[4.2. CoAP connection to Registrar](#)

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

[4.3. Proxy discovery of Registrar](#)

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

The registrar announces itself using ACP instance of GRASP using M_FLOOD messages. They MUST support ANI TLS circuit proxy and therefore BRSKI across HTTPS/TLS native across the ACP. ANI registrars MAY support the IPIP proxy method by implementing IPIP tunneling for their HTTPS/TLS traffic across the ACP. 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, 80]]

```

Figure 7a: Registrar Discovery

The formal CDDL definition is:


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

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

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

Figure 7: AN_join_registrar CDDL

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

The locators are to be interpreted as follows:

```
locator1 = [0_IPv6_LOCATOR, fd45:1345::6789, 6, 443]
locator2 = [0_IPv6_LOCATOR, fd45:1345::6789, 17, 5683]
locator3 = [0_IPv6_LOCATOR, fe80::1234, 41, nil]
```

A protocol of 6 indicates that TCP proxying on the indicated port is desired. A protocol of 17 indicates that UDP proxying on the indicated port is desired. In each case, the traffic SHOULD be proxied to the same port at the ULA address provided.

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

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

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

5. Protocol Details (Pledge - Registrar - MASA)

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

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

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

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

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

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

Summarized automation extensions for the BRSKI-EST flow are:

- o The pledge provisionally accepts the registrar certificate during the TLS handshake as detailed in [Section 5.1](#).
- o The pledge either attempts concurrent connections, or it times out quickly and tries connections in series.
- 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 5.5.1](#). This moves the BRSKI-EST TLS connection out of the provisional state.
- o Mandatory bootstrap steps conclude with voucher status telemetry (see [Section 5.6](#)).

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 In the language of [[RFC6125](#)] this provides for a SERIALNUM-ID category of identifier that can be included in a certificate and therefore that can also be used for matching purposes. The SERIALNUM-ID whitelist is collated according to manufacturer trust anchor since serial numbers are not globally unique.
- o The registrar requests and validates the voucher from the MASA.
- o The registrar forwards the voucher to the pledge when requested.
- o The registrar performs log verifications in addition to local authorization checks before accepting optional pledge device enrollment requests.

5.1. BRSKI-EST TLS establishment details

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

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

The pledge maintains a security paranoia concerning the provisional state, and all data received, until a voucher is received and verified as specified in [Section 5.5.1](#)

To avoid blocking on a single erroneous registrar the pledge MUST drop the connection after 5 seconds in which there has been no progress on the TCP connection. It should proceed to connect to any other registrar's via any other discovered proxies if there are any. If there were no other proxies discovered, the pledge MAY continue to

wait, as long as it is concurrently listening for new proxy announcements.

5.2. Pledge Requests Voucher from the Registrar

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

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

The request media types are:

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

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

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

The pledge populates the voucher-request fields as follows:

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

`nonce`: The pledge voucher-request MUST contain a cryptographically strong random or pseudo-random number nonce. Doing so ensures [Section 2.6.1](#) functionality. The nonce MUST NOT be reused for multiple bootstrapping attempts.

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

`proximity-registrar-cert`: In a pledge voucher-request this is the first certificate in the TLS server 'certificate_list' sequence (see [[RFC5246](#)]) presented by the registrar to the pledge. This MUST be populated in a pledge voucher-request if the "proximity" assertion is populated.

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

An example JSON payload of a pledge voucher-request is in [Section 3.2 Example 1](#).

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

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

[5.3](#). BRSKI-MASA TLS establishment details

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

The primary method of registrar "authentication" by the MASA is detailed in [Section 5.4](#). As detailed in [Section 9](#) the MASA might find it necessary to request additional registrar authentication.

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

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

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

5.4. Registrar Requests Voucher from MASA

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

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

The request media type is defined in [\[I-D.ietf-anima-voucher\]](#) and is `application/voucher-cms+json`. It is a JSON document that has been signed using a CMS structure. The registrar **MUST** sign the registrar voucher-request. The entire registrar certificate chain, up to and including the Domain CA, **MUST** be included in the CMS structure.

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

The registrar populates the voucher-request fields as follows:

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

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

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

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

`prior-signed-voucher-request`: If a signed pledge voucher-request was received then it **SHOULD** be included in the registrar voucher-request. (NOTE: what is included is the complete pledge voucher-request, inclusive of the 'assertion', 'proximity-registrar-cert', etc wrapped by the pledge's original signature). If a signed voucher-request was not received from the pledge then this leaf is omitted from the registrar voucher request.

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

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

Example JSON payloads of registrar voucher-requests are in [Section 3.2](#) Examples 2 through 4.

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

5.4.1. MASA renewal of expired vouchers

As described in [[I-D.ietf-anima-voucher](#)] vouchers are normally short lived to avoid revocation issues. If the request is for a previous (expired) voucher using the same registrar (as determined by the registrar pinned-domain-cert) and the MASA has not been informed that the claim is invalid then the request for a renewed voucher SHOULD be automatically authorized.

5.4.2. 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 5.4](#).

5.4.3. MASA authentication of registrar (certificate)

If a nonceless voucher-request is submitted the MASA server MUST authenticate the registrar as described in either EST [\[RFC7030\]](#) [section 3.2](#), [section 3.3](#), or by validating the registrar's certificate used to sign the registrar voucher-request. Any of these methods reduce the risk of DDoS attacks and provide an authenticated identity as an input to sales channel integration and authorizations (the actual sale-channel integration is also out-of-scope of this document).

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

5.4.4. MASA revocation checking of registrar (certificate)

Note that this section is about ensuring consistency of the registrar. These checks are not equivalent to [\[RFC5280\]](#) Certificate Revocation Checks that the pledge would ideally do if it had a real time clock.

If the registrar uses a CA for which the MASA is able to obtain revocation information, then the MASA SHOULD check for the maximum validity period of the registrar's certificate.

A registrar which has done these checks may set the "domain-cert-revocation-checks" attribute in the voucher to false (or omit it).

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 resulting voucher should have a lifetime (expires-on field) which is 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.

By limiting the voucher lifetime in this way, the MASA is ensuring the voucher is consistent with any CRL and lifetime checks. See section [Section 2.6.1](#).

If CRL information is unavailable to the MASA, then the MASA SHOULD rely on the validity information. Because the registrar certificate authority is unknown to the MASA in advance this is only an extended consistency check and is not required. The maximum lifetime of the voucher issued SHOULD NOT exceed the lifetime of the registrar's

revocation validation (for example if the registrar revocation status is indicated in a CRL that is valid for two weeks then that is an appropriate lifetime for the voucher.)

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

The MASA server MAY verify that the registrar voucher-request includes the 'prior-signed-voucher-request' field populated with a signed pledge voucher-request that includes a 'proximity-registrar-cert' that is consistent with the certificate used to sign the registrar voucher-request. This ensures that the BRSKI-EST TLS connection has no man-in-the-middle. The MASA server 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.

If this check succeeds the MASA updates the voucher and audit log assertion leafs with the "proximity" assertion.

5.4.6. 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 5.4](#). 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.

5.4.7. MASA nonce handling

The MASA does not verify the nonce itself. It MAY perform a simple consistency check: If the registrar voucher-request contains a nonce and the prior-signed-voucher-request exists then the nonce in both MUST be consistent. (Recall from above that the voucher-request might not contain a nonce, see [Section 5.4](#) and [Section 5.4.3](#)).

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.

5.5. MASA Voucher Response

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

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

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

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

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

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

A 406 (Not Acceptable) response is appropriate if a voucher of the desired type, or using the desired algorithms (as indicated by the Accept: headers, and algorithms used in the signature) cannot be issued, such as because the MASA knows the pledge cannot process that type.

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

The response media type is:

application/voucher-cms+json The response is a "YANG-defined JSON document that has been signed using a CMS structure" as described in [\[I-D.ietf-anima-voucher\]](#) using the JSON encoded described in [\[RFC7951\]](#). The MASA MUST sign the request.

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

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

[5.5.1.](#) Pledge voucher verification

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

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

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

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

[5.5.2.](#) Pledge authentication of provisional TLS connection

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

If a registrar's credentials cannot be verified using the pinned-domain-cert trust anchor from the voucher then the TLS connection is immediately discarded and the pledge abandons attempts to bootstrap with this discovered registrar. The pledge SHOULD send voucher status telemetry (described below) before closing the TLS connection. The pledge MUST attempt to enroll using any other proxies it has

found. It SHOULD return to the same proxy again after attempting with other proxies. Attempts should be attempted in the exponential backoff described earlier. Attempts SHOULD be repeated as failure may be the result of a temporary inconsistency (an inconsistently rolled registrar key, or some other mis-configuration). The inconsistency could also be the result of 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 a trust anchor for future operations. It can therefore be used to authenticate any dynamically discovered EST server that contains 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.

[5.6. Pledge Voucher Status Telemetry](#)

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

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

The posted data media type: application/json

The client HTTP POSTs the following to the server at the EST well known URI `"/voucher_status"`. The Status field indicates if the voucher was acceptable. If it was not acceptable the Reason string indicates why. In the failure case this message may be sent to an unauthenticated, potentially malicious registrar and therefore the Reason string SHOULD NOT provide information beneficial to an attacker. The operational benefit of this telemetry information is

balanced against the operational costs of not recording that an voucher was ignored by a client the registrar expected to continue joining the domain.

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

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

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

Additional standard responses MAY be added via Specification Required.

5.7. Registrar audit log request

After receiving the voucher status telemetry [Section 5.6](#), the registrar SHOULD request the MASA audit log from the MASA service.

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

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

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

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

Rather than returning the audit log as a response to the POST (with a return code 200), the MASA MAY instead return a 201 ("Created") RESTful response ([\[RFC7231\] section 7.1](#)) containing a URL to the prepared (and easily cachable) audit response.

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

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

The request media type is:

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

[5.7.1](#). MASA audit log response

A log data file is returned consisting of all log entries. 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 that was placed in the voucher assertion leaf>"
    },
    {
      "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 that was placed in the voucher assertion leaf>"
    }
  ],
  "truncation": {
    "nonced duplicates": <number of entries truncated>,
    "nonceless duplicates": <number of entries truncated>,
    "arbitrary": <number of entries truncated>
  }
}

```

Distribution of a large log is less than ideal. This structure can be optimized as follows: Nonced or Nonceless entries for the same domainID MAY be truncated from the log leaving only the single most recent nonced or nonceless entry. The log SHOULD NOT be further reduced but there could exist operational situation where maintaining the full log is not possible. In such situations the log MAY be arbitrarily truncated for length. The truncation method(s) used MUST be indicated in the JSON truncation dictionary using "nonced duplicates", "nonceless duplicates", and "arbitrary" where the number of entries that have been truncation is indicated. If the truncation count exceeds 1024 then the MASA MAY use this value without further incrementing it.

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

5.7.2. Registrar audit log verification

Each time the Manufacturer Authorized Signing Authority (MASA) issues a voucher, it places it into the audit log for that device. The details are described in [Section 5.7](#). 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 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.

5.8. EST Integration for PKI bootstrapping

The pledge SHOULD follow the BRSKI operations with EST enrollment operations including "CA Certificates Request", "CSR Attributes" and "Client Certificate Request" or "Server-Side Key Generation", etc. This is a relatively seamless integration since BRSKI REST calls provide an automated alternative to the manual bootstrapping method described in [[RFC7030](#)]. As noted above, use of HTTP 1.1 persistent connections simplifies the pledge state machine.

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

Although EST allows clients to obtain multiple certificates by sending multiple CSR requests BRSKI mandates use of the CSR Attributes request and mandates that the registrar validate the CSR against the expected attributes. This implies that client requests will "look the same" and therefore result in a single logical certificate being issued even if the client were to make multiple requests. Registrars MAY contain more complex logic but doing so is out-of-scope of this specification. BRSKI does not signal any enhancement or restriction to this capability.

5.8.1. 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 5.5.1](#) for a discussion of the limitations inherent in having a single certificate

instead of a full CA Certificates response.) Although these limitations are acceptable during initial bootstrapping, they are not appropriate for ongoing PKIX end entity certificate validation.

5.8.2. EST CSR Attributes

Automated bootstrapping occurs without local administrative configuration of the pledge. In some deployments 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.

5.8.3. EST Client Certificate Request

The pledge MUST request a new client certificate. See [RFC7030, section 4.2](#).

5.8.4. Enrollment Status Telemetry

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

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

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

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

Status media type: application/json

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

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

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

Within the server logs the server MUST capture if this message was received over an TLS session with a matching client certificate. This allows for clients that wish to minimize their crypto operations to simply POST this response without renegotiating the TLS session - at the cost of the server not being able to accurately verify that enrollment was truly successful.

5.8.5. Multiple certificates

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

5.8.6. 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.vanderstok-ace-coap-est](#)] and that CoAP mappings for BRSKI will be discussed either there or in future work.

6. Reduced security operational modes

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

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

6.1. Trust Model

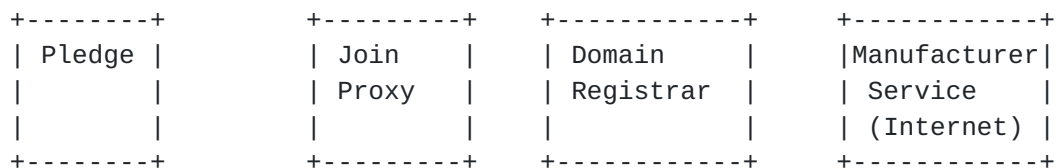


Figure 10

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 server a registrar makes all decisions. For Ownership Audit Vouchers (see [\[I-D.ietf-anima-voucher\]](#)) the registrar is provided an opportunity to accept MASA server decisions.

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

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

6.2. Pledge security reductions

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

1. The pledge MUST accept nonceless vouchers. This allows for a use case where the registrar can not connect to the MASA at the deployment time. Logging and validity periods address the security considerations of supporting these use cases.
2. The pledge MAY support "trust on first use" for physical interfaces such as a local console port or physical user interface but MUST NOT support "trust on first use" on network interfaces. This is because "trust on first use" permanently degrades the security for all use cases.
3. The pledge MAY have an operational mode where it skips voucher validation one time. For example if a physical button is depressed during the bootstrapping operation. This can be useful if the manufacturer service is unavailable. This behavior SHOULD be available via local configuration or physical presence methods (such as use of a serial/craft console) to ensure new entities can always be deployed even when autonomic methods fail. This allows for unsecured imprint.

It is RECOMMENDED that "trust on first use" or skipping voucher validation 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.

6.3. Registrar security reductions

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

1. A registrar MAY choose to accept all devices, or all devices of a particular type, at the administrator's discretion. This could occur when informing all registrars of unique identifiers of new entities might be operationally difficult.
2. A registrar MAY choose to accept devices that claim a unique identity without the benefit of authenticating that claimed identity. This could occur when the pledge does not include an X.509 IDevID factory installed credential. New Entities without an X.509 IDevID credential MAY form the [Section 5.2](#) request using the [Section 5.4](#) format to ensure the pledge's serial number information is provided to the registrar (this includes the IDevID AuthorityKeyIdentifier value, which would be statically configured on the pledge.) The pledge MAY refuse to provide a TLS client certificate (as one is not available.) The pledge SHOULD support HTTP-based or certificate-less TLS authentication as described in EST [RFC7030 section 3.3.2](#). A registrar MUST NOT accept unauthenticated New Entities unless it has been configured to do so by an administrator that has verified that only expected new entities can communicate with a registrar (presumably via a physically secured perimeter.)
3. A registrar MAY submit a nonceless voucher-requests to the MASA service (by not including a nonce in the voucher-request.) The resulting vouchers can then be stored by the registrar until they are needed during bootstrapping operations. This is for use cases where the target network is protected by an air gap and therefore cannot contact the MASA service during pledge deployment.
4. A registrar MAY ignore unrecognized nonceless log entries. This could occur when used equipment is purchased with a valid history being deployed in air gap networks that required permanent vouchers.

6.4. MASA security reductions

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

1. Not enforcing that a nonce is in the voucher. This results in distribution of a voucher that never expires and in effect makes the Domain an always trusted entity to the pledge during any subsequent bootstrapping attempts. That this occurred is captured in the log information so that the registrar can make appropriate security decisions when a pledge joins the Domain. This is useful to support use cases where registrars might not be online during actual device deployment. Because this results in a long lived voucher and does not require the proof that the device is online, this is only accepted when the registrar is authenticated by the MASA server and authorized to provide this functionality. The MASA server is RECOMMENDED to use this functionality only in concert with an enhanced level of ownership tracking (out-of-scope.) If the pledge device is known to have a real-time-clock that is set from the factory, use of a voucher validity period is RECOMMENDED.
2. Not verifying ownership before responding with a voucher. This is expected to be a common operational model because doing so relieves the manufacturer providing MASA services from having to track ownership during shipping and supply chain and allows for a very low overhead MASA service. A registrar uses the audit log information as a defense in depth strategy to ensure that this does not occur unexpectedly (for example when purchasing new equipment the registrar would throw an error if any audit log information is reported.) The MASA SHOULD verify the 'prior-signed-voucher-request' information for pledges that support that functionality. This provides a proof-of-proximity check that reduces the need for ownership verification.

7. IANA Considerations

This document requires the following IANA actions:

7.1. Well-known EST registration

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

- o add /.well-known/est/requestvoucher (see [Section 5.4](#))
- o add /.well-known/est/requestauditlog (see [Section 5.6](#))

[7.2.](#) 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. [[EDNOTE: fix names]]

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

[7.3.](#) Voucher Status Telemetry

IANA is requested to create a registry entitled: `_Voucher Status Telemetry Attributes_`. New items can be added using the Specification Required. The following items are to be in the initial registration, with this document as the reference:

- o version
- o Status
- o Reason
- o reason-context

[7.4.](#) DNS Service Names

IANA is requested to register the following Service Names:

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

7.5. MUD File Extension for the MASA server

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 D](#).

8. Privacy Considerations

8.1. 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.

9. Security Considerations

There are uses cases where the MASA could be unavailable or uncooperative to the Registrar. They include planned and unplanned network partitions, changes to MASA policy, or other instances where MASA policy rejects a claim. These introduce an operational risk to the Registrar owner that MASA behavior might limit the ability to re-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 repeat bootstrapping for 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. This reduces the resale value of the equipment because future owners will detect the lowered security inherent in the existence of a nonceless voucher that would be trusted by their pledge. This reflects a balance between partition resistant recovery and security of future bootstrapping. Registrars take the pledge's audit history into account when applying policy to new devices.

The MASA server is exposed to DoS attacks wherein attackers claim an unbounded number of devices. Ensuring a registrar is representative of a valid 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.

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 but this is mandated anyway because of the operational benefits of an informed administrator in cases where the failure is indicative of a problem. The registrar is RECOMMENDED to verify MASA logs if voucher status telemetry is not received.

To facilitate truly limited clients EST [RFC7030 section 3.3.2](#) requirements that the client MUST support a client authentication model have been reduced in [Section 6](#) to a statement that the registrar "MAY" choose to accept devices that fail cryptographic authentication. This reflects current (poor) practices in shipping devices without a cryptographic identity that are NOT RECOMMENDED.

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

Pledges might chose to engage in protocol operations with multiple discovered registrars in parallel. As noted above they will only do so with distinct nonce values, but the end result could be multiple vouchers issued from the MASA if all registrars attempt to claim the device. This is not a failure and the pledge choses whichever voucher to accept based on internal logic. The registrar's verifying

log information will see multiple entries and take this into account for their analytics purposes.

9.1. Freshness in Voucher-Requests

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

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

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

The fake registrar (Rm) can obtain a voucher signed by the MASA either directly or through arbitrary intermediaries. Assuming that the MASA accepts the 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 5.7](#). Rm might chose to wait until after the intended registrar completes the authorization process before submitting the now-stale pledge voucher-request. The Rm would need to remove the pledge's nonce.

In order to successfully use the resulting "stale voucher" Rm would have to attack the pledge and return it to a bootstrapping enabled

state. This would require wiping the pledge of current configuration and triggering a re-bootstrapping of the pledge. This is no more likely than simply taking control of the pledge directly but if this is a consideration the target network is RECOMMENDED to take the following steps:

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

9.2. Trusting manufacturers

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

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

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

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

- 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, c1sco.com, ..
- 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.

[10.](#) Acknowledgements

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

[11.](#) References

[11.1.](#) Normative References

- [I-D.ietf-anima-autonomic-control-plane]
Eckert, T., Behringer, M., and S. Bjarnason, "An Autonomic Control Plane (ACP)", [draft-ietf-anima-autonomic-control-plane-13](#) (work in progress), December 2017.
- [I-D.ietf-anima-grasp]
Bormann, C., Carpenter, B., and B. Liu, "A Generic Autonomic Signaling Protocol (GRASP)", [draft-ietf-anima-grasp-15](#) (work in progress), July 2017.
- [I-D.ietf-anima-voucher]
Watsen, K., Richardson, M., Pritikin, M., and T. Eckert, "Voucher Profile for Bootstrapping Protocols", [draft-ietf-anima-voucher-07](#) (work in progress), January 2018.
- [IDevID] IEEE Standard, "IEEE 802.1AR Secure Device Identifier", December 2009, <<http://standards.ieee.org/findstds/standard/802.1AR-2009.html>>.

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3542] Stevens, W., Thomas, M., Nordmark, E., and T. Jinmei, "Advanced Sockets Application Program Interface (API) for IPv6", [RFC 3542](#), DOI 10.17487/RFC3542, May 2003, <<https://www.rfc-editor.org/info/rfc3542>>.
- [RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, Ed., "Extensible Authentication Protocol (EAP)", [RFC 3748](#), DOI 10.17487/RFC3748, June 2004, <<https://www.rfc-editor.org/info/rfc3748>>.
- [RFC3927] Cheshire, S., Aboba, B., and E. Guttman, "Dynamic Configuration of IPv4 Link-Local Addresses", [RFC 3927](#), DOI 10.17487/RFC3927, May 2005, <<https://www.rfc-editor.org/info/rfc3927>>.
- [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>>.
- [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>>.
- [RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", [RFC 7159](#), DOI 10.17487/RFC7159, March 2014, <<https://www.rfc-editor.org/info/rfc7159>>.
- [RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", [RFC 7228](#), DOI 10.17487/RFC7228, May 2014, <<https://www.rfc-editor.org/info/rfc7228>>.

- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", [RFC 7950](#), DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG", [RFC 7951](#), DOI 10.17487/RFC7951, August 2016, <<https://www.rfc-editor.org/info/rfc7951>>.

11.2. Informative References

- [docsisroot]
CableLabs, "CableLabs Digital Certificate Issuance Service", February 2018, <<https://www.cablelabs.com/resources/digital-certificate-issuance-service/>>.
- [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-06](#) (work in progress), February 2018.
- [I-D.ietf-netconf-zerotouch]
Watsen, K., Abrahamsson, M., and I. Farrer, "Zero Touch Provisioning for Networking Devices", [draft-ietf-netconf-zerotouch-21](#) (work in progress), March 2018.
- [I-D.ietf-opsawg-mud]
Lear, E., Droms, R., and D. Romascanu, "Manufacturer Usage Description Specification", [draft-ietf-opsawg-mud-20](#) (work in progress), April 2018.
- [I-D.richardson-anima-state-for-joinrouter]
Richardson, M., "Considerations for stateful vs stateless join router in ANIMA bootstrap", [draft-richardson-anima-state-for-joinrouter-02](#) (work in progress), January 2018.
- [I-D.vanderstok-ace-coap-est]
Stok, P., Kampanakis, P., Kumar, S., Richardson, M., Furuhed, M., and S. Raza, "EST over secure CoAP (EST-coaps)", [draft-vanderstok-ace-coap-est-04](#) (work in progress), January 2018.
- [imprinting]
Wikipedia, "Wikipedia article: Imprinting", July 2015, <[https://en.wikipedia.org/wiki/Imprinting_\(psychology\)](https://en.wikipedia.org/wiki/Imprinting_(psychology))>.

- [RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in IPv6 Specification", [RFC 2473](#), DOI 10.17487/RFC2473, December 1998, <<https://www.rfc-editor.org/info/rfc2473>>.
- [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>>.
- [RFC7217] Gont, F., "A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)", [RFC 7217](#), DOI 10.17487/RFC7217, April 2014, <<https://www.rfc-editor.org/info/rfc7217>>.
- [RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", [RFC 7231](#), DOI 10.17487/RFC7231, June 2014, <<https://www.rfc-editor.org/info/rfc7231>>.
- [RFC7435] Dukhovni, V., "Opportunistic Security: Some Protection Most of the Time", [RFC 7435](#), DOI 10.17487/RFC7435, December 2014, <<https://www.rfc-editor.org/info/rfc7435>>.
- [RFC7575] Behringer, M., Pritikin, M., Bjarnason, S., Clemm, A., Carpenter, B., Jiang, S., and L. Ciavaglia, "Autonomic Networking: Definitions and Design Goals", [RFC 7575](#), DOI 10.17487/RFC7575, June 2015, <<https://www.rfc-editor.org/info/rfc7575>>.
- [Stajano99theresurrecting] Stajano, F. and R. Anderson, "The resurrecting duckling: security issues for ad-hoc wireless networks", 1999, <<https://www.cl.cam.ac.uk/~fms27/papers/1999-StajanoAnd-duckling.pdf>>.

[TR069] Broadband Forum, "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 specification of BRSKI in [Section 4](#) intentionally only covers the mechanisms for an IPv6 pledge using Link-Local addresses. This section describes non-normative extensions that can be used in other environments.

A.1. IPv4 Link Local addresses

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

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

A.2. Use of DHCPv4

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

Appendix B. mDNS / DNSSD proxy discovery options

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

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

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

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

registrar._tcp.example.com". In this case the domain "example.com" is discovered as described in [\[RFC6763\] section 11](#) (Appendix A.2 suggests the use of DHCP parameters).

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

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

[Appendix C](#). IPIP Join Proxy mechanism

The circuit proxy mechanism suffers from requiring a state on the proxy for each connection that is relayed. The proxy can be considered a kind of Algorithm Gateway (see [\[RFC2663\], section 2.9](#)).

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

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

One way to look at this construct is to consider that the registrar is extending an interface to attaching to the network on which the proxy is physically present. The registrar then interacts as if it were present on that network using link-local (fe80::) addresses. The registrar is unaware that the traffic is being proxied through a tunnel, and does not need any special routing.

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

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

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

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

The simplest way to get this additional key is to allocate an additional ACP address; one address for each network on which join traffic is occurring.

C.2. Automatic configuration of tunnels on Registrar

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

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

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

C.3. Proxy Neighbor Discovery by Join Proxy

The proxy **MUST** answer neighbor discovery messages for the address given by the registrar as being its link-local address. The proxy

must also advertise this address as the address to which to connect when advertising its existence.

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

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

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

When using UDP sockets with CoAP, the application will have to pay attention to the incoming ifindex on the socket. Access to this information is available using the IP_PKTINFO auxiliary extension, which is a standard part of the IPv6 sockets API [[RFC3542](#)].

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

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

Some operating systems on which a registrar needs to be implemented may find need for a virtual interface per proxy to be problematic. There are other mechanisms which can be implemented.

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

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

[Appendix D](#). MUD Extension

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

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

The model is defined as follows:

```
<CODE BEGINS> 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 E. Example Vouchers

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

E.1. Keys involved

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

E.1.1. MASA key pair for voucher signatures

This private key signs vouchers:

```
-----BEGIN EC PRIVATE KEY-----
MIGkAgEBBDAGiRoYqKoEcf0fvRvmZ5P5Azn58tuI7nSnIy70gFnCeINo+BmbgMho
r6lcU60gwVagBwYFK4EEACKhZANiAATZAH3Rb2FvIJOnts+vXuWW35ofyNbCHzjA
z0i2kWZFE1ByurKImNcNMFGirGnRXIXGqWCFw5ICgJ8CuM3vV5ty9bf7KUl0kejz
Tvv+5PV++elkP9HQ83vqTaws2WwWTxI=
-----END EC PRIVATE KEY-----
```

This public key validates vouchers:

```
-----BEGIN CERTIFICATE-----
MIIBzzCCAVagAwIBAgIBATAKBggqhkJOPQDAjBNMRIwEAYKCZImiZPyLGQBGRYC
Y2ExGTAXBgoJkiaJk/IsZAEZFglzYW5kZWxtYW4xHDAaBgNVBAMME1Vuc3RydW5n
IEhpZ2h3YXkgQ0EwHhcNMTCwMzI2MTYxOTQwWhcNMTCwMzI2MTYxOTQwWjBHMRIw
EAYKCZImiZPyLGQBGRYCY2ExGTAXBgoJkiaJk/IsZAEZFglzYW5kZWxtYW4xZjAU
BgNVBAMMDVuc3RydW5nIE1BU0EwdjAQBgcqhkJOPQIBBgUrgQQAIGNiAATZAH3R
b2FvIJOnts+vXuWW35ofyNbCHzjAz0i2kWZFE1ByurKImNcNMFGirGnRXIXGqWCF
w5ICgJ8CuM3vV5ty9bf7KUl0kejzTvv+5PV++elkP9HQ83vqTaws2WwWTxKjEDA0
MAwGA1UdEwEB/wQMAAwCgYIKoZizj0EAWIDZwAwZAIwGb0oyM0doP6t3/LSPL50
DuatEwMYh7WGO+IYTHC8K7EyHB0mCYReKT2+GhV/CLWzAjBNy6UMJtt1tsxJsJqd
MPUIFj+4wZg1A0Ib/JoA6M7r33pwLQTrHRxEzVMGfW0kYUw=
-----END CERTIFICATE-----
```

E.1.2. Manufacturer key pair for IDevID signatures

This private key signs IDevID certificates:


```

-----BEGIN EC PRIVATE KEY-----
MIGkAgEBBDAGiRoYqKoEcF0fvRvmZ5P5Azn58tuI7nSnIy70gFnCeINo+BmbgMho
r6lcU60gwVagBwYFK4EEACKhZANiAATZAH3Rb2FvIJ0nts+vXuWw35ofyNbCHzjA
z0i2kWZFE1ByurKImNcNMFgirGnRXIXGqWCfw5ICgJ8CuM3vV5ty9bf7KUL0kejz
Tvv+5PV++elkP9HQ83vqTaws2WwWTxI=
-----END EC PRIVATE KEY-----

```

This public key validates IDevID certificates:

```

-----BEGIN CERTIFICATE-----
MIIBzzCCAVagAwIBAgIBATAKBggqhkJOPQQAjBNMRIwEAYKCZImiZPyLgQBGRYC
Y2ExGTAXBgoJkiaJk/IsZAEZFglzYW5kZWxtYW4xHDAaBgNVBAMME1Vuc3RydW5n
IEhpZ2h3YXkgQ0EwHhcNMTCwMzI2MTYxOTQwWhcNMTCwMzI2MTYxOTQwWjBHMRIw
EAYKCZImiZPyLgQBGRYCY2ExGTAXBgoJkiaJk/IsZAEZFglzYW5kZWxtYW4xZjAU
BgNVBAMMDVuc3RydW5nIE1BU0EwdjAQBgcqhkJOPQIBBgUrgQQAIGNiAATZAH3R
b2FvIJ0nts+vXuWw35ofyNbCHzjAz0i2kWZFE1ByurKImNcNMFgirGnRXIXGqWCf
w5ICgJ8CuM3vV5ty9bf7KUL0kejzTvv+5PV++elkP9HQ83vqTaws2WwWTxKjEDA0
MAwGA1UdEwEB/wQCMAAwCgYIKoZizj0EAWIDZwAwZAIwGbo0yM0doP6t3/LSP150
DuatEwMYh7WGO+IYTHC8K7EyHB0mCYReKT2+GhV/CLWzAjBNy6UMJTt1tsxJsJqd
MPUIFj+4wZg1A0Ib/JoA6M7r33pwLQTrHRxEzVMGfW0kYUw=
-----END CERTIFICATE-----

```

E.1.3. Registrar key pair

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

```

-----BEGIN EC PRIVATE KEY-----
MHcCAQEEIF+obiToYYYemifPsZvrjWJ0yFsCJwIFhpokmT/TULmXoAoGCCqGSM49
AwEHoUQDQgAENWQ0zcNMUjP0NrtfeBc0DJLWfeMGgCFdIv6FUz4DifM1ujMBec/g
6W/P6boTmyTGdF0h/8HwKUerL5bpneK8sg==
-----END EC PRIVATE KEY-----

```

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

```

-----BEGIN CERTIFICATE-----
MIIBrjCCAT0gAwIBAgIBAZAKBggqhkJOPQQAzb0MRIwEAYKCZImiZPyLgQBGRYC
Y2ExGTAXBgoJkiaJk/IsZAEZFglzYW5kZWxtYW4xHTAbBgNVBAMMFFVuc3RydW5n
IEZvdW50YWluIENBMB4XDTE3MDkwNTAxMTI0NV0xOTQwWhcNMTCwMzI2MTYxOTQw
WjBHMRIwEAYKZm90eSMBAGCgmsJomT8ixkARKwAmNhMRkwFwYKCZImiZPyLgQBGRYJc2FuZGVsbWVuc3RydW5n
w0xSM/Q2u194FzQmktZ94waAIV0i/oVTPg0J8zW6MwF5z+Dpb8/puhObJMZ0U6H/
wfApr6svlumd4ryyow0wCzAJBgNVHRMEAjAAMAoGCCqGSM49BAMDA2kAMGYCMQC3
/iTQJ3evYYcgbXhbmzrp64t3QC6qjIeY2jkDx062nuNifVKtyaara3F30AIkKSEC
MQDi29efbTLbdtDk3tecY/rD7V77XaJ6nYCmdDCR54TrSFNLgxvt1lyFM+0fYpYR
c3o=
-----END CERTIFICATE-----

```


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

Certificate:

Data:

Version: 3 (0x2)

Serial Number: 3 (0x3)

Signature Algorithm: ecdsa-with-SHA384

Issuer: DC = ca, DC = sandelman, CN = Unstrung Fount

ain CA

Validity

Not Before: Sep 5 01:12:45 2017 GMT

Not After : Sep 5 01:12:45 2019 GMT

Subject: DC = ca, DC = sandelman, CN = localhost

Subject Public Key Info:

Public Key Algorithm: id-ecPublicKey

Public-Key: (256 bit)

pub:

04:35:64:0e:cd:c3:4c:52:33:f4:36:bb:5f:7

8:17:

34:0c:92:d6:7d:e3:06:80:21:5d:22:fe:85:5

3:3e:

03:89:f3:35:ba:33:01:79:cf:e0:e9:6f:cf:e

9:ba:

13:9b:24:c6:74:53:a1:ff:c1:f0:29:47:ab:2

f:96:

e9:9d:e2:bc:b2

ASN1 OID: prime256v1

NIST CURVE: P-256

X509v3 extensions:

X509v3 Basic Constraints:

CA:FALSE

Signature Algorithm: ecdsa-with-SHA384

30:66:02:31:00:b7:fe:24:d0:27:77:af:61:87:20:6d:78:

5b:

9b:3a:e9:eb:8b:77:40:2e:aa:8c:87:98:da:39:03:c7:4e:

b6:

9e:e3:62:7d:52:ad:c9:a6:ab:6b:71:77:d0:02:24:29:21:

02:

31:00:e2:db:d7:9f:6d:32:db:76:d0:e4:de:d7:9c:63:fa:

c3:

ed:5e:fb:5d:a2:7a:9d:80:a6:74:30:91:e7:84:eb:48:53:

4b:

83:1b:ed:d6:5c:85:33:ed:1f:62:96:11:73:7a

E.1.4. Pledge key pair

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

```

-----BEGIN EC PRIVATE KEY-----
MHcCAQEEIL+ue8PQcN+M7LFBGPsompYwobI/rsoHnTb2a+0h0+8joAoGCCqGSM49
AwEHoUQDQgAEumBVaDlX87WyME8CJToyt9Nwy6sYw0DTbjjJIn79pgr7ALa//Y8p
r70WpK1SIaiUeeFw7e+lCzTp1Z+wJu14Bg==
-----END EC PRIVATE KEY-----

```

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

```

-----BEGIN CERTIFICATE-----
MIICmjCCAbegAwIBAgIBDDAKBggqhkJOPQQAjBNMRIwEAYKZImiZPyLGQBGRYC
Y2ExGTAXBgoJkiaJk/IsZAEZFglzYW5kZWxtYW4xHDAaBgNVBAMME1Vuc3RydW5n
IEhpZ2h3YXkgQ0EwIBcNMTCxMDEyMTMjUyWhgPMjk5OTEyMzEwMDAwMDBaMEsX
EjAQBgoJkiaJk/IsZAEZFgJjYTEZMBcGCgmsJomT8ixkARKWCXNhbmlbG1hbjEa
MBGGA1UEAwRMDAtRDAtRTUtRjItMDAtMDIwWTATBgcqhkJOPQIBBggqhkJOPQMB
BwNCAARJp5i0dU1aUnR2u8wMRwgkNupNbnNM7m1n0mj+0KJZjcPIqID+trPjTSobt
uIdpRPfGZ8hU/nIUveqwyoYI8BPbo4GHMIGEMB0GA1UdDgQWBQdMRZthFQmzz6
E7YVXzkL7XZDKjAJBgNVHRMEAjAAMCsGA1UdEQQkMCKgIAYJKwYBBAGC7lIBoBMM
ETAwLUQwLUU1LUYyLTAWLTAYMcGCSsGAQQBgu5SAGQeDBxodHRwczovL2hpZ2h3
YXkuc2FuZGVsbWVufLmNhMAoGCCqGSM49BAMCA2kAMGYCMQDhJ1N+eanW1U/e5qoM
SGvUvWHR7uic8cJbh7vXy580nBs8bpNn60k/+IzveUetMzICMQCr1uxvdYeKq7mb
RXCR4ZCJsw67fJ7jyXZbcUSir+3wBT2+lWggzPDRgYB5ABb7sAw=
-----END CERTIFICATE-----

```

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

Certificate:

```

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

```



```

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

```

E.2. Example process

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

E.2.1. Pledge to Registrar

As described in [Section 5.2](#), the pledge will sign a pledge voucher-request containing the registrar's public key in the proximity-

registrar-cert field. The base64 has been wrapped at 60 characters for presentation reasons.

MIIHHAYJKoZIhvcNAQcCoIIHDTCCBwkCAQExDzANBgIghkgBZQMEAgEFADCC
Aw4GCSqGSIB3DQEHAaCCAv8EggL7eyJpZXRMlXZvdWNoZXItcmVxdWVzdDp2
b3VjaGVyIjpwImFzc2VydGlvbiI6InByb3hpbWl0eSIsImNyZWZ0ZWQtY24i
OiiyMDE3LTA5LTAxIiwic2VyaWFsLW51bWJlciI6IjAwLUQwLUU1LUYyLTAW
LTAYIiwibm9uY2U0iJEC3M50XNCcjNwTk1PQUNLLUxZWtd3IiwicHJveGlt
aXR5LXJlZ2lzdHJhc11jZXJ0IjoitULJQnJqQ0NBVE9nQXdJQkFnSUJBekFL
QmdncW hrak9QUVFEQXpCT01SSXdFQVLLQ1pJbWlaUHLMR1FCR1JZQ1kyRXhh
VEFYQmdvSmtYUprL0LzWkFFwKZnbHpZVzVrWld4dFlXNHhIVEFiQmd0VkJB
TU1GRlZ1YzNSEwRXNW5JRVP2ZFc1MF1XbHVJRJU5CTUI0WERURTNNRgt3TLRB
eE1USTBOVm9YRFRFNU1Ea3d0VEF4TVRJM5Wb3dRekVTTUJBR0NnbVNBk21U
0GL4a0FSa1dBbU5oTVJrd0Z3WUtDwkltaVpQeUxHUUJHUllkYzJGdVpHVnNi
V0Z1TVJJd0VBWURWUVFEREFsc2IyTmhir2h2YzNRd1dUQVRCZ2NxaGtqt1BR
SUJJCZ2dxaGtqt1BRTUJCd05DQUFRMVpBN053MHhTTS9RMnUx0TRGe1FN3Ra
OTR3YUFJVjBpL29wVFBnT0o4elc2TXdGNXorRHBi0C9wdwhPYkpNwjBVNkgv
d2ZBcFI2c3Zsdw1kNHJ5eW93MHdDekFKQmd0VkhSTUVBakFBTUFvR0NDcUdT
TTQ5QkFNREEya0FNR1LDTVFDMy9pVFFKM2V2WVljZ2JYaGJtenJwNjR0M1FD
NnFqSWZVMmprRHgwnJJudU5pZlZLdHlhYXJhM0YzMEFJa0tTRUNNUURpmjll
ZmJUTGJkdERrM3RlY1kvckQ3Vjc3WGFKNm5ZQ21kRENSNTRUcLNGTKxneHZ0
MWx5Rk0rMGZZcFlSYzNvPSJ9faCCAjYwggIyMIIBt6ADAgEAgEMMAoGCCqG
SM49BAMCME0xExAQBg0JkiaJk/IsZAEZFgJjYTEZMBcGCgmSJomT8ixkARkK
CXNhbmlRlG1hbG1hbjEcmBoGA1UEAwTVW5zdHJ1bmcgSGlnaHdheSBDQTAgFw0x
NzEwMTIxMzUyNTJaGA8yOTk5MTIzMTAwMDAwMFowSzESMBAGCgmSJomT8ixk
ARkKAmNhMRkwFwYKcZImiZPyLGOBGRYJc2FuZGVsbWwFUMRowGAYDVQQDDBEW
MC1EMC1FNS1GMi0wMC0wMjBZBMGBYqGSM49AgEGCCqGSM49AwEHA0IABEmn
mLR1TVpSdHa7zAxHCCQ26k1s0zubWfSaP7QoLmNw8iogP62s+NNKhu24h2LE
98ZnyFT+chS96rDKhgjwE9ujgYcwgYQwHQYDVR00BBYEFB0xFmG2EVCbPPoT
thVf0QvtdkMqMAKGA1UdEwQCAAwKwYDVR0RBCQwIqAgBgkrBgEEAYLuUgGg
EwwRMDAtRDAtRTUtRjItMDAtMDIwKwYJKwYBBAGC7LICBB4MHGhdHBz0i8v
aGlnaHdheS5zYW5kZxwYw4uY2EwCgYIKoZiZj0EAwIDAQAwwZG1xIAwdo
d2F5IENBAGEMMA0GCWCGSAFLAwQCAQAoIHKMBGCSqGSIB3DQEJAZELBgkq
hkiG9w0BBwEwHAYJKoZIhvcNAQkFMQ8XDTE3MTAxMjE3NTQzMFowLWYJKoZI
hvcNAQkEMSIIEIP59cuKVAPkK00lQIaIV/W1AsWkVmBd9wFSu5yLafMHKG
CSqGSIB3DQEDzFsMGowCwYJYIZIAWUDBAEqMASGCWCGSAFLAwQBFjALBglg
hkgBZQMEAIwCgYIKoZIhvcNAwCwDgYIKoZIhvcNAwICAgCAMA0GCCqGSIB3
DQMCAGFAMACGBS0AwIHMA0GCCqGSIB3DQMCAGe0MAoGCCqGSM49BAMCBEYw
RAIgyUy0NTdP+xTKm/Et69eI++S/2z3dQwPK0wdL0cDCSvACIAh3jJbybMnK
cf7DKKnsn2G/006HeB/8imMI+hnA7CfN

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

The ASN1 decoding of the artifact:


```

    0:d=0  hl=4  l=1820  cons: SEQUENCE
    4:d=1  hl=2  l=    9  prim: OBJECT           :pkcs7-signed
Data
  15:d=1  hl=4  l=1805  cons: cont [ 0 ]
  19:d=2  hl=4  l=1801  cons: SEQUENCE
  23:d=3  hl=2  l=    1  prim: INTEGER           :01
  26:d=3  hl=2  l=   15  cons: SET
  28:d=4  hl=2  l=   13  cons: SEQUENCE
  30:d=5  hl=2  l=    9  prim: OBJECT           :sha256
  41:d=5  hl=2  l=    0  prim: NULL
  43:d=3  hl=4  l=   782  cons: SEQUENCE
  47:d=4  hl=2  l=    9  prim: OBJECT           :pkcs7-data
  58:d=4  hl=4  l=   767  cons: cont [ 0 ]
  62:d=5  hl=4  l=   763  prim: OCTET STRING     :{"ietf-voucher-request:voucher":{"assertion":"proximity","created-on":"2017-09-01","serial-number":"00-D0-E5-F2-00-02","nonce":"Dss99sBr3pNMOACe-LYY7w","proximity-registrar-cert":"MIIBrjCCAT0gAwIBAgIBAzAKBggqhkJOPQQDAZBOMRIwEAYKCCZImiZPyLQBGGRYCY2ExGTAXBgoJkiaJk/IsZAEZFglzYW5kZWxtYW4xHTAbBgNVBAMMFVuc3RydW5nIEZvdW50YWluIENBMB4XDTE3MDkwNTAxMTI0NVoXDTE3MDkwNTAxMTI0NVowQzESMBAGCgMSJomT8ixkARKwAmNhMRkwFwYKCCZImiZPyLQBGGRYJc2FuZGVsbWwFuMRIwEAYDVQQDDAlsb2NhbgHvc3QwWTATBgcqhkJOPQIBBggqhkJOPQMBBwNCAAQ1ZA7Nw0xSM/Q2u194FzQMktZ94waAIV0i/oVTPg0J8zW6MwF5z+Dpb8/puh0bJmZ0U6H/wfApr6svlumd4ryyow0wCzAJBgNVHRMEAjAAMAoGCCqGSM49BAMDA2kAMGYCMQC3/iTQJ3evYYcgbXhbmzrp64t3QC6qjIeY2jkDx062nuNifVKtyaara3F30AIkkSECMQDi29efbTLbdtDk3tecY/rD7V77XaJ6nYCmdDCR54TrSFNLgxvt1lyFM+0fYpYRc3o="}}
  829:d=3  hl=4  l=   566  cons: cont [ 0 ]
  833:d=4  hl=4  l=   562  cons: SEQUENCE
  837:d=5  hl=4  l=   439  cons: SEQUENCE
  841:d=6  hl=2  l=    3  cons: cont [ 0 ]
  843:d=7  hl=2  l=    1  prim: INTEGER           :02
  846:d=6  hl=2  l=    1  prim: INTEGER           :0C
  849:d=6  hl=2  l=   10  cons: SEQUENCE
  851:d=7  hl=2  l=    8  prim: OBJECT           :ecdsa-with-S
HA256
  861:d=6  hl=2  l=    77  cons: SEQUENCE
  863:d=7  hl=2  l=   18  cons: SET
  865:d=8  hl=2  l=   16  cons: SEQUENCE
  867:d=9  hl=2  l=   10  prim: OBJECT           :domainCompon
ent
  879:d=9  hl=2  l=    2  prim: IA5STRING        :ca
  883:d=7  hl=2  l=   25  cons: SET
  885:d=8  hl=2  l=   23  cons: SEQUENCE
  887:d=9  hl=2  l=   10  prim: OBJECT           :domainCompon
ent
  899:d=9  hl=2  l=    9  prim: IA5STRING        :sandelman
  910:d=7  hl=2  l=   28  cons: SET

```



```

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

```


30302D3032

1235:d=8 hl=2 l= 43 cons: SEQUENCE
1237:d=9 hl=2 l= 9 prim: OBJECT :1.3.6.1.4.1.

46930.2

1248:d=9 hl=2 l= 30 prim: OCTET STRING [HEX DUMP]:0C
1C68747470733A2F2F686967687761792E73616E64656C6D616E2E6361

1280:d=5 hl=2 l= 10 cons: SEQUENCE
1282:d=6 hl=2 l= 8 prim: OBJECT :ecdsa-with-S

HA256

1292:d=5 hl=2 l= 105 prim: BIT STRING
1399:d=3 hl=4 l= 421 cons: SET
1403:d=4 hl=4 l= 417 cons: SEQUENCE
1407:d=5 hl=2 l= 1 prim: INTEGER :01
1410:d=5 hl=2 l= 82 cons: SEQUENCE
1412:d=6 hl=2 l= 77 cons: SEQUENCE
1414:d=7 hl=2 l= 18 cons: SET
1416:d=8 hl=2 l= 16 cons: SEQUENCE
1418:d=9 hl=2 l= 10 prim: OBJECT :domainCompon

ent

1430:d=9 hl=2 l= 2 prim: IA5STRING :ca
1434:d=7 hl=2 l= 25 cons: SET
1436:d=8 hl=2 l= 23 cons: SEQUENCE
1438:d=9 hl=2 l= 10 prim: OBJECT :domainCompon

ent

1450:d=9 hl=2 l= 9 prim: IA5STRING :sandelman
1461:d=7 hl=2 l= 28 cons: SET
1463:d=8 hl=2 l= 26 cons: SEQUENCE
1465:d=9 hl=2 l= 3 prim: OBJECT :commonName
1470:d=9 hl=2 l= 19 prim: UTF8STRING :Unstrung Hig

hway CA

1491:d=6 hl=2 l= 1 prim: INTEGER :0C
1494:d=5 hl=2 l= 13 cons: SEQUENCE
1496:d=6 hl=2 l= 9 prim: OBJECT :sha256
1507:d=6 hl=2 l= 0 prim: NULL
1509:d=5 hl=3 l= 228 cons: cont [0]
1512:d=6 hl=2 l= 24 cons: SEQUENCE
1514:d=7 hl=2 l= 9 prim: OBJECT :contentType
1525:d=7 hl=2 l= 11 cons: SET
1527:d=8 hl=2 l= 9 prim: OBJECT :pkcs7-data
1538:d=6 hl=2 l= 28 cons: SEQUENCE
1540:d=7 hl=2 l= 9 prim: OBJECT :signingTime
1551:d=7 hl=2 l= 15 cons: SET
1553:d=8 hl=2 l= 13 prim: UTCTIME :171012175430

Z

1568:d=6 hl=2 l= 47 cons: SEQUENCE
1570:d=7 hl=2 l= 9 prim: OBJECT :messageDiges

t

1581:d=7 hl=2 l= 34 cons: SET


```

1583:d=8 hl=2 l= 32 prim: OCTET STRING [HEX DUMP]:FE
7D72E29500F90A38E95021A215FD6D40B1629B99598177DC054AE0F9C8B6
9F
1617:d=6 hl=2 l= 121 cons: SEQUENCE
1619:d=7 hl=2 l= 9 prim: OBJECT :S/MIME Capab
ilities
1630:d=7 hl=2 l= 108 cons: SET
1632:d=8 hl=2 l= 106 cons: SEQUENCE
1634:d=9 hl=2 l= 11 cons: SEQUENCE
1636:d=10 hl=2 l= 9 prim: OBJECT :aes-256-cbc
1647:d=9 hl=2 l= 11 cons: SEQUENCE
1649:d=10 hl=2 l= 9 prim: OBJECT :aes-192-cbc
1660:d=9 hl=2 l= 11 cons: SEQUENCE
1662:d=10 hl=2 l= 9 prim: OBJECT :aes-128-cbc
1673:d=9 hl=2 l= 10 cons: SEQUENCE
1675:d=10 hl=2 l= 8 prim: OBJECT :des-edes3-cbc
1685:d=9 hl=2 l= 14 cons: SEQUENCE
1687:d=10 hl=2 l= 8 prim: OBJECT :rc2-cbc
1697:d=10 hl=2 l= 2 prim: INTEGER :80
1701:d=9 hl=2 l= 13 cons: SEQUENCE
1703:d=10 hl=2 l= 8 prim: OBJECT :rc2-cbc
1713:d=10 hl=2 l= 1 prim: INTEGER :40
1716:d=9 hl=2 l= 7 cons: SEQUENCE
1718:d=10 hl=2 l= 5 prim: OBJECT :des-cbc
1725:d=9 hl=2 l= 13 cons: SEQUENCE
1727:d=10 hl=2 l= 8 prim: OBJECT :rc2-cbc
1737:d=10 hl=2 l= 1 prim: INTEGER :28
1740:d=5 hl=2 l= 10 cons: SEQUENCE
1742:d=6 hl=2 l= 8 prim: OBJECT :ecdsa-with-S
HA256
1752:d=5 hl=2 l= 70 prim: OCTET STRING [HEX DUMP]:30
440220614CB435374FFB14E49BF12DEBD788FBE4BFDB3DDD4303CA3B074B
D1C0C24AF0022008778C96F26CC9CA71FEC328A9EC9F61BF3B4E87781FFC
8A6308FA19C0EC27CD

```

The JSON contained in the voucher request:


```
{
  "ietf-voucher-request:voucher": {
    "assertion": "proximity",
    "created-on": "2017-09-01",
    "serial-number": "00-D0-E5-F2-00-02",
    "nonce": "Dss99sBr3pNMOAcE-LYY7w",
    "proximity-registrar-cert": "
    MIIIBrjCCAT0gAwIBAgIBAzAKBggqhkJOPQQDAzBOMRIwEAYKZImiZPyLGQB
    GRYCY2ExGTAXBgoJkiaJk/IsZAEZFglzYW5kZWxtYW4xHTAbBgNVBAMMFFVu
    c3RydW5nIEZvdW50YWluIENBMB4XDTE3MDkwNTAxMTI0NVoXDTE5MDkwNTAx
    MTI0NVowQzESMBAGCgmSjomT8ixkARKwAmNhMRkwFwYKZImiZPyLGQBGRYJ
    c2FuZGVsbWwFuMRiWEAYDVQDDAlsb2NhbGhvc3QwWTATBgqcqhkJOPQIBBggq
    hkjOPQMBBwNCAAQ1ZA7Nw0xSM/Q2u194FzQMktZ94waAIV0i/oVTPgOJ8zW6
    MwF5z+Dpb8/puhObJMz0U6H/wfApR6svlumd4ryyow0wCzAJBgNVHRMEAjAA
    MAoGCCqGSM49BAMDA2KAMGYCMQC3/iTQJ3evYYcgbXhbmzrp64t3QC6qjIeY
    2jkDx062nuNifVKtyaara3F30AIkKSECMQDi29efbTLbdtDk3tecY/rD7V77
    XaJ6nYCmdDCR54TrSFNLgxvt1lyFM+0fYpYRc3o="}}

```

E.2.2. Registrar to MASA

As described in [Section 5.4](#) the registrar will sign a registrar voucher-request, and will include pledge's voucher request in the prior-signed-voucher-request.

```
MIIN2gYJKoZIhvcNAQcCoIINyzCCDccCAQExDzANBgIghkgBZQMEAgEFADCC
Ck4GCSqGSIB3DQEHAaCCCj8Eggo7eyJpZXRMlXZvdWNoZXItcmVxdWVzdDp2
b3VjaGVyIj7ImFzc2VydGlvbiI6InByb3hpbWl0eSIsImNyZWF0ZWQtb24i
0iIyMDE3LTA5LTE1VDAwOjAwOjAwLjAwMFoiLCJzZXJpYWwtbnVtYmVyIjoj
SkFEQTEyMzQ1Njc4OSIsIm5vbmNlIjojYmVjZDEyMzQ1Njc4OSIsImNp
ZWQtdm91Y2hlcil1yZXF1ZXN0IjojTUJSEhRWUpLb1pJaHZjTkFRY0NvSUI
RGpDQ0J3b0NBVV4RHpbTKJnbGdoa2dCWlFNRUFnRUZBRENDQxc0R0NTcudT
SWIzRFFSEFhQ0NBdjhFZ2dMN2V5SnBaWFJtTFhadmRXtm9aWEloY21WeGRX
VnpkRHAYjNwamFHVnlJanA3SW1GemMyVnlkR2x2YmLJNklunliM2hwYlds
MGVTSXNjBU55WldGMFpUXRimjRpT2lJeU1ERTNMVEE1TFRBeIpd2ljMLZ5
YVdGc0xXNTFiV0psY2lJNklqQXdmVVF3TFVVMUxVWXLmVEF3TFRBeIpd2li
bTl1WTJVaU9pSkVjM001T1h0Q2NqTndUazFQUVV0bExVeFpXVGQzSWl3aWNI
SnZlR2x0YVhSNuXYSmxaMmx6ZEhKaGNpMwpaWEowSwpvaVRVbEpRbkpxUTBO
QLZFOW5RWGRKUwtGbLNVSKJla0ZMUW1kbmNXaHJhazlRVVZGRVFYcENUMDFT
U1hkrLfwBEXRMBKYldSYVVIbE1SMUZDUjFKWlExa3lSWGhIVkVGVVftZHZT
bXRwVWVvcKwwbHpxa0ZGV2tabmJicFpWELZyV2xkNGRgBfhOSghJVkVGAVft
ZE9wa0pCVFUXR1JswjFZek5TZVdSWE5XNUpSVnAyWkZjMU1GbFhiSFZKULU1
Q1RVSTBXRvJVULROTlJHdDNubFJCZUUXVVNUQk9wbTLZUkZSRk5VMUvhM2RP
VkvGNFRWUkpNRTVXYjNkUmVrVlRUVUpCUjB0bmJWtktimjFVT0dsNGEwRlNh
MWRCYlU1b1RWSnJkMFozV1V0RFdrbHRhVnBRZVV4SFVVSkhVbGxLWxpKR2RW
cEhWbk5pVjBaMVRWskpMFZCV1VSV1VWRkVSRUZZyZJJJeVRtaGLSMmgyWxpO
UmQxZFRVRLJDWjJ0eGFHdHFUMUJSU1VKQ1oyZHhhR3RvVDFCUlRVskNkMDVE
UVVGuk1wceJOMDUzTUhoVFRTOVJNbLV4T1RSR2VsRk5hM1JhT1RSM1LVRkpW
akJwTDI5V1ZGQm5UMG80ZwxjmlRYZEd0Wg9yUkhCaU9D0XdkV2hQWwtwTldq
QLZ0a2d2ZDJaQmNGSTJjM1pzZFcxa05ISjVlVzkzTUhkRGVrRktRbWRPVmtO
U1RVVkJha0ZCVFVGdliWtkRjVWRUVFRNvFrRk5SRUV5YTBGTlixbERUVkZE
TXk5cFZGRktNMLYyV1ZsaloysllhR0p0ZW5Kd05qUjBNMUZETm5GcVNXVlpN
bXByUkhnd05qSnVkvTVwMxaTGRIBghZWEpoTTBZek1FRkphMHRUULVOTLVV

```


UnBNamxswm1KVVRRHSmtkRVJyTTNSbFkxa3Zja1EzVmpjM1dHRktObTVaUTIx
a1JFTlNOVFJVY2xOR1RreG5lSFowTVd4NVJrMHJNR1paY0ZsU1l6TnZQU0o5
ZmFDQ0FqWXdZ0l5TULJQnQ2QURBZ0VDQWdFTU1Bb0dDQ3FHU000OUJBTUNN
RTB4RwPBUUJnb0prawFKay9Jc1pBRVpGZ0pqwVRFwk1CY0dDZ21Tsm9tVDhp
eGtBUmtXQ1h0aGJtUmxirZFoYmpFY01Cb0dBMMVFQXd3VFZXNpkSEoxYm1j
Z1NHbG5hSGRoZVNCRRFFUQWdGdzB4TnpFd01USXhNeLV5TLRKYUdBOHLPVGS1
TVRJek1UQXdNREF3TUZvd1N6RVNNQkFHQ2dtU0pVbVQ4aXhrQVJrV0FtTmhN
Umt3RNdZS0NaSW1pWLB5TEdRQkdSWUpjMkZ1WkdWc2JXRnVNUm93R0FZRFZR
UUREQkV3TUMxRU1DMUZOUzFHTWkwd01DMHdNakJaTUJNR0J5cUdTTTQ5QWdF
R0NDcUdTTTQ5QXdFSEEwSUFcRW1ubUxSMVRwcfNkSGE3ekF4SENDUTI2azFz
MHp1YldmU2FQN1FvbG10dzhpb2dQnJjZk050S2h1MjRoMmxFOThabnlGVCtj
aFM5NnJES2hnandFOXVqZ1lj2dZUXdIUUVLEVLiWt0JCWUVGQjB4Rm1HMkVW
Q2JQUG9UdGhwZk9RdnRka01xTUFrR0ExVWRFd1FDTUFBd0t3WURWUjBSQkNR
d0lxQWdCZ2tyQmdFRUFZTHVZ0dnRXd3Uk1EQXRSREF0ULRVdFJqSXRNREF0
TURJd0t3WUPLd1LCQkFHQzdsSUNCQjRNSEdoMGRiQnpPaTh2YUdsbmFIZGhL
UzV6Wvc1a1pXeHRZVzR1WTJFd0NnWULLb1pJemowRUF3SURhUUF3WmdJeEFP
RW5VMzU1cWRiVLQ5N21xZ3hJYTLTOVlkSHU2Snp4d2x1SHU5ZkxueLnjR3p4
dWsyZnJTVc80ak84ULI2MHPNZ0L4QUt2VzdH0TFoNHfydVp0RmNKSghrSW16
RHJ00G51UEpkbHNKUKtLdjdmQUZQYjZwYUNETTh0R0JnSGtBRnZ1d0RER0NB
YVl3Z2dHaUFnRUJNRkL3VFRFU01CQUdDZ21Tsm9tVDhpEgTBUmtXQW10aE1S
a3dGd1LLQ1pJbWlaUHLMR1FCR1JZSmMyRnVar1ZzYldGdU1Sd3dHZ1LEVLFR
RERCTLzibk4wY25WdVp5QkLhV2RvZDJGnuLFTkJBZ0VNTUEwR0NXQ0dTQUZs
QXDRQ0FRVUFvSUhrTUJnR0NTcUdTSWIZRFFFSkF6RUxCZ2txaGtpRzL3MEJC
d0V3SEFZSkvtwklodmNOQVFrRk1R0FhEVEUzTVRBeE1qRXpOVGd5TTFvd0x3
WUpLb1pJaHzjTkFRa0VNU0lFSVA10WN1S1ZBUGtLT09sUUhSVYvVzFbc1dL
Ym1wbUJkOXDGU3VENXlMYWZNSGtHQ1NxR1NjYjNEUUVKRHpGc01Hb3dDd1Lk
WUlaSUFxVURCQUVxTUFzR0NXQ0dTQUZsQXDRQkZqQUxCZ2xnaGtnQlpRTUVB
UUL3Q2dZSutvWklodmNOQXdjd0RnWULLb1pJaHzjTkF3SUNBZ0NBtUEwR0ND
cUdTSWIZRFFNQ0FnRkFNQWNHQlnzT0F3SUhNQTbHQ0NxR1NjYjNEU1DQWdF
b01Bb0dDQ3FHU000OUJBTUNCRWN3ULFJZ0VNZzFkSkw3RmNkdHJWRHg4cUNh
em9lOSsyMk56NFp3UKI5Z0FUR0w3TU1DSVFEanZVWxaeKpxcDIva0NkNFdo
eFVoc2FDcFRGd1Bybk5ldzV3Q2tZVUY4UT09In19oIIBsjCCAA4wggEzoAMC
AQICAQMwCgYIKoZiZj0EAwMwTjESMBAGCgmSJomT8ixkARkWAmmMRkWFwYK
CZImiZPyLGQBGRYJc2FuZGVsbWwFUMR0wGwYDVQQDDBRVbnN0cnVuZyBGb3Vu
dGFpbjBDQTAeFw0xNA5MDUwMTEyNDVaFw0xOTA5MDUwMTEyNDVaMEMxEjAQ
BgoJkiaJk/IsZAEZFgJjYTEZMBcGCgmSJomT8ixkARkWCXNhbMRlbG1hbG1ES
MBAGA1UEAwJbG9jYXZ0b3N0MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAE
NWQ0zcnMUjP0NrtfeBc0DJLWfeMGgCFdIv6FUz4DifM1ujMBec/g6W/P6boT
myTgDF0h/8HwKuerL5bpnek8sqMMAaswCQYDVR0TBAlwADAKBggqhkJOPQQD
AwNpADBMAjEAT/4k0Cd3r2GHIG14W5s66euLd0AuqoyHmNo5A8d0tp7jYn1S
rcmmq2txd9ACJckhAjEA4tvXn20y23bQ5N7XnGP6w+1e+12iep2ApnQwkeeE
60hTS4Mb7dZchTPtH2KWEXN6MYIBpzCCAAMCAQEwUzBOMRIwEAYKZImiZPy
LGQBGRYCY2ExGTAXBgoJkiaJk/IsZAEZFglzYW5kZWxtYW4xHTAbBgNVBAMM
FFVuc3RydW5nIEZvdW50YWluIENBAGEDMA0GCWCGSAFLAwQCAQUAoIHkMBGg
CSqGSIb3DQEJAZELBgkqhkiG9w0BBwEwHAYJKoZIhvcNAQkFMQ8XDTE3MTAy
NjAxMzYxOFowLWYJKoZIhvcNAQkEMSIIEQBM73PZzPo7tE9Mj8gQvaaYeMQ
OslLACaw/HenAqNwMHkGCSqGSIb3DQEJdzFsMGowCwYJYIZIAWUDBAEqMASG
CWCGSAFLAwQBFjALBgLghkgBZQMEAAQIwCgYIKoZIhvcNAwcwDgYIKoZIhvcN

AwICAgCAMA0GCCqGSib3DQMCAGFAMAcGBSS0AwIHMA0GCCqGSib3DQMCAGeO
MAoGCCqGSM49BAMCBECwRQIGDdp5uPULMKp7GFQAD7ypAgqFv8q+KkJt6c30
7iVpVI8CIQCD1u8BkxipvigwvIDmWfjLYdJxcvozNjffq5j3UHg7Rg==

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

The ASN1 decoding of the artifact:

```

0:d=0 hl=4 l=3546 cons: SEQUENCE
4:d=1 hl=2 l= 9 prim: OBJECT          :pkcs7-signed
Data
15:d=1 hl=4 l=3531 cons: cont [ 0 ]
19:d=2 hl=4 l=3527 cons: SEQUENCE
23:d=3 hl=2 l= 1 prim: INTEGER          :01
26:d=3 hl=2 l= 15 cons: SET
28:d=4 hl=2 l= 13 cons: SEQUENCE
30:d=5 hl=2 l= 9 prim: OBJECT          :sha256
41:d=5 hl=2 l= 0 prim: NULL
43:d=3 hl=4 l=2638 cons: SEQUENCE
47:d=4 hl=2 l= 9 prim: OBJECT          :pkcs7-data
58:d=4 hl=4 l=2623 cons: cont [ 0 ]
62:d=5 hl=4 l=2619 prim: OCTET STRING  :{"ietf-vouch
er-request:voucher":{"assertion":"proximity","created-on":"2
017-09-15T00:00:00.000Z","serial-number":"JADA123456789","no
nce":"abcd1234","prior-signed-voucher-request":"MIIHHQYJKoZI
hvcNAQcCoIIHdjCCBwoCAQExDzANBgIghkgBZQMEAgEFADCCAw4GCSqGSib3
DQEHAaCCAv8Eggl7eyJpZXRMlXZvdWNoZXItcmVxdWVzdDp2b3VjaGVyIjp7
ImFzc2VydGlvbiI6InByb3hpbWl0eSIsImNyZWZ0ZWQtb24iOiIyMDE3LTA5
LTAxIiwic2VyaWFsLW51bWJlciI6IjAwLUQwLUU1LUYyLTAwLTAyIiwibm9u
Y2UiOiIjEC3M5OXNccjNwTk1PQUNLLUxZWtd3IiwicHJveGltaxR5LXJlZ2lz
dHJhciIjZXJ0IjoitULjQnJqQ0NBVE9nQXdJQkFnSUJBekFLQmdncWhrak9Q
UVFEQXpCT01SSXdFQVLLQ1pJbWlaUHLMR1FCR1JZQ1kyRXhhVFEFYQmdvSmtP
YUprL0lzWkFFWkZnbHpvZVZvVrWld4dFlXNHhIVEFiQmd0VkJBTU1GRlZ1YzNS
eWRXNW5JRvP2Zfc1MF1xBHVJRu5CTUI0WERURTNRRgt3TLRBeE1USTBOVm9Y
RFRFNU1Ea3d0VEF4TVRjME5wb3dRekVTTUJBR0NnbVNBk21UOGL4a0FSa1dB
bu5oTVJrd0Z3WUtDwkltavPqEuxHUUJHULLKYzJGdVpHVnNiV0Z1TVJJd0VB
WURWUVFEREFsc2IyTmhiR2h2YzNRd1dUQVRCZ2NxaGtqT1BR5JCZ2dxaGtq
T1BRTUJCd05DQUFRMvPBN053MhHTTS9RMnUx0TRGeLFNa3Ra0TR3YUFJVjBp
L29WVFBnT0o4e1c2TXdGNXorRHBi0C9wdWhPYkpNwJbVnkgvd2ZBcFI2c3Zs
dW1kNHJ5eW93MHdDekFKQmd0VkhSTUVBakFBTUFvR0NDcUdTTTQ5QkFNREEy
a0FNR1LDTVFDMy9pVFFKM2V2WVljZ2JYaGJtenJwNjR0M1FDNnFqSWVZMmpr
RHgwNjJudU5pZlZLdHlhYXJhM0YzMEFJa0tTRUNNUURpmjllZmJUTGJkdERR
M3RlY1kvckQ3Vjc3WGFKNm5ZQ21kRENSNTRUc1NGTKxneHZ0MwX5Rk0rMGZZ
cFlSYzNvPSJ9faCCAjYwggIyMIIBT6ADAgECAgEMMAoGCCqGSM49BAMCME0x
EjAQBgoJkiaJk/IsZAEZFgJjYTEZMBcGCgmsJomT8ixkARKwCXNhbMRlbG1h
bjEcMBoGA1UEAwTVW5zdHJ1bmcgSGlnaHdheSBDQTAqFw0xNzEwMTIxMzUy
NTJaGA8yOTk5MTIzMTAwMDAwMFowSzESMBAGCgmsJomT8ixkARKwAmNhMRkw
FwYKZCImiZPyLGQBGryJc2FuZGVzbnBwFUMRowGAYDVQQDBDEwMC1EMC1FNS1G

```



```

Mi0wMC0wMjBZMBMGBYqGSM49AgEGCCqGSM49AwEHA0IABEmnmLR1TVpSdHa7
zAxHCCQ26k1s0zubWfSaP7QoLmNw8iogP62s+NNKhu24h2lE98ZnyFT+chS9
6rDKhgjwE9ujgYcwgYQwHQYDVR00BBYEFB0xFmG2EVCbPPoTthVfOQvtdkMq
MAKGA1UdEwQCAAwKwYDVR0RBCQwIqAgBgkrBgEEAYLuUgGgEwwRMDAtRDAt
RTUtRjItMDAtMDIwKwYJKwYBBAGC7LICBB4MHGh0dHBz0i8vaGlnaHdheS5z
YW5kZWxtYW4uY2EwCgYIKoZIZj0EAWIDaQAwZgIXA0EnU355qdbVT97mqgxI
a9S9YdHu6JzxwluHu9fLnzScGzxuk2frST/4j08RR60zMgIXAKvW7G91h4qr
uZtFcJHhkImzDrt8nuPJdlsJRKKv7fAFPb6VaCDM8NGBgHkAFvuwDDGCAaYw
ggGiAgEBMFIwTTESMBAGCgmSJomT8ixkARkWAmmMRkwFwYKcZImiZPyLGQB
GRYJc2FuZGVsbWwFuMRwwGgYDVQVQDBNBvbnN0cnVuZyBIAwdod2F5IENBAGEM
MA0GCWCGSAFlAwQCAUAoIHKMBGCSqGSIb3DQEJAzELBGlhkgBZQMEAAQIw
HAYJKoZIhvcNAQkFMQ8XDTE3MTAxMjEzNTgyM1owLWYJKoZIhvcNAQkEMSI
EIP59cuKVAPkK00lQIaIV/W1AsWKbmVmBd9wFSu5yLafMHkGCSqGSIb3DQEJ
DzFsMGowCwYJYIZIAWUDBAEqMAsGCWCGSAFlAwQBFjALBglghkgBZQMEAAQIw
CgYIKoZIhvcNAwcwDgYIKoZIhvcNAwICAgCAMA0GCqGSIb3DQMCAGFAMAcG
BSs0AwIHMA0GCCqGSIb3DQMCAGeOMAoGCCqGSM49BAMCBECwRQIgemg1dJL7
FcdtrVDx8qCazoe9+22Nz4ZwRB9gATGL7MMCIQDjssULZzJqp2/kCd4WhxUh
saCpTFwPrnNew5wCkYUF8Q=="}}

```

```

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

```


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


```

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

```



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

E.2.3. MASA to Registrar

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

MIIG3AYJKoZIhvcNAQcCoIIgZTCCBskCAQExDzANBgIghkgBZQMEAgEFADCC
AxAGCSqGSib3DQEHAaCCAWEeggL9eyJpZXRmLXZvdWNoZXI6dm91Y2hlcIiE6
eyJhc3NlcnRpb24iOiJsb2dnZWQiLCJjcmVhdGVkLW9uIjoimjAxNy0xMC0x
MlQxMzo1ND0zMS40MzktMDQ6MDAiLCJzZXJpYWwtbnVtYmVyIjoimDAtRDAt
RTUtRjItMDAtMDIiLCJub25jZSI6IkRzczk5c0JyM3B0TU9BQ2UtTFLZ3ci
LCJwaW5uZWQtZG9tYWluLWNlcnQiOiJNSULCcmpDQ0FUT2dBd0lCQWdJQkF6
QUtCZ2dxaGtqT1BRUURBekJPTVJJd0VBWUtDwkltaVpQeUxHUUJHULLDWTJF
eEdUQVhCZ29Ka2lhSmsvSXNaQUVaRmdselLlXNwtaV3h0wVc0eEhUQWJCZ05W
QkFNTUZGVnVjM1J5ZFc1bk1FwnZkVzUwWVdsdU1FTkJNqjRYRFRFM01Ea3d0
VEF4TVRjME5wb1hEVEU1TURrd05UQXhNVEkwTLZvd1F6RVNNQkFHQ2dtU0pv
bVQ4aXhrQVJrV0FtTmhNUmt3RndZS0NaSW1pWlB5TEdRQkdSWUpjMkZ1WkdW
c2JXRnVNUkl3RUFZRFZRUUREQWxzYjJ0aGJHaHZjM1F3V1RBVEJnY3Foa2pP
UFFJQkInZ3Foa2pPUFFNQk3J3TknBQVExWkE3TncweFNnl1EydTE5NEZ6UU1r
dFo5NHdhQUlWmgkvb1ZUUGdPSjh6VzZNd0Y1eitEcGI4L3B1aE9iSk1aMFU2
SC93ZkFwUjZzdmx1bWQ0cnl5b3cWd0N6QUpCZ05WSFJNRUFqQUFNQW9HQ0NX
R1NNNDLCQU1EQTJrQU1HWUNNUUMzL2LUUUozZXZZWwNnYlhoYm16cnA2NHQZ
UUM2cWpJZVkyamtEeDA2Mm51TmlmVkt0eWfHcmEzRjMwQUlrs1NFQ01RRGky
OWVmYlRMYmR0RGszdGVjWS9yRDdWNzdYUo2bllDbwREQ1I1NFRyU0Z0TGd4
dnQxbHlGTSSwZllwVjJm289In19oIIB0zCCAc8wggFwoAMCAQICAQEwCgYI
KoZIZj0EAwIwTTESMBAGCgmSJomT8ixkARkWAmbMRkwFwYKcZImiZPyLGQB
GRYJc2FuZGVsbWFuMRwwGgYDVQDDBNVbnN0cnVuZyBIAwdod2F5IENBMB4X
DTE3MDMyNjE2MTk0MFoXDTE5MDMyNjE2MTk0MfowRzESMBAGCgmSJomT8ixk
ARkWAmbMRkwFwYKcZImiZPyLGQBGRYJc2FuZGVsbWFuMRYwFAYDVQQDDA1V
bnN0cnVuZyBNQVNBMHYwEAYHkoZIZj0CAQYFK4EEACIDYgAE2QB90W9hbyCT
p7bPr17llt+aH8jWwh84wMzotpFmRRNQcrqyiJjXDTBRoqxp0VyFxlgn80S
AoCfArjn71ebcvw3+yLJTpHo8077/ut1fvnpZD/R0PN76kwMLNlsFk8SoxAw
DjAMBGNVHRMBAf8EAJAAMAoGCCqGSM49BAMCA2cAMGQCMBm9KMjNHAd+rd/y
0jy+Tg7mrMDGIe1hjviGExwvCuxMhwTpgmEXik9vhoVfwi1swIwTcuLDCU7
dbbMSbCanTD1CBY/uMGYNQDiG/yaA0j06996cC0E6x0cRM1TBn1jpGFMMYIB
xjCCAcICAQEwUjBNMRiwEAYKcZImiZPyLGQBGRYCY2ExGTAXBgoJkiaJk/Is
ZAEZFglzYW5kZWxtYw4xHDAaBgNVBAMME1Vuc3RydW5nIEhpZ2h3YXkgQ0EC
AQEwDQYJYIZIAWUDBAIBBQCgqeQwGAYJKoZIhvcNAQkDMQsGCsGSIb3DQEHA
ATAcBgkqhkiG9w0BCQUxDxcNMTcxMDEyMTc1NDMxWjAvBgkqhkiG9w0BCQQx
IggQXnG628cIW8MoYfB1ljDDLLlJQlxE2tnjcvkLEfix0weQYJKoZIhvcN
AQkPMwwajALBgIghkgBZQMEASowCwYJYIZIAWUDBAEWMAsgCWCgsAF1AwQB
AjAKBggqhkiG9w0DBZA0BggqhkiG9w0DAGICAIAwDQYIKoZIhvcNAwICAUAw
BwYFKw4DAgcwDQYIKoZIhvcNAwICASgwCgYIKoZIZj0EAwIEZzBlAjEAhZid
/AkNjttSP1rflNppdHsi324Z2+TXJxueewnJ8z/2NXb+Tf3DsThv7du000z
AjBjy0nmkkSKHsPR2JluA5c6wovUPEnNKP32daGGeFKGEHMkTInbrqipC881
/5K9Q+k=

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

The ASN1 decoding of the artifact:

```

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

```



```

15:d=1 hl=4 l=1741 cons: cont [ 0 ]
19:d=2 hl=4 l=1737 cons: SEQUENCE
23:d=3 hl=2 l= 1 prim: INTEGER :01
26:d=3 hl=2 l= 15 cons: SET
28:d=4 hl=2 l= 13 cons: SEQUENCE
30:d=5 hl=2 l= 9 prim: OBJECT :sha256
41:d=5 hl=2 l= 0 prim: NULL
43:d=3 hl=4 l= 784 cons: SEQUENCE
47:d=4 hl=2 l= 9 prim: OBJECT :pkcs7-data
58:d=4 hl=4 l= 769 cons: cont [ 0 ]
62:d=5 hl=4 l= 765 prim: OCTET STRING :{"ietf-vouch
er:voucher":{"assertion":"logged","created-on":"2017-10-12T1
3:54:31.439-04:00","serial-number":"00-D0-E5-F2-00-02","nonc
e":"Dss99sBr3pNM0ACe-LYY7w","pinned-domain-cert":"MIIBrjCCAT
OgAwIBAgIBAzAKBggqhkJOPQQDAZB0MRIWEAYKCZImiZPyLGBGRYCY2ExGT
AXBgoJkiaJk/IsZAEZFglzYw5kZWxtYW4xHTAbBgNVBAMMFVuc3RydW5nIE
ZvdW50YWluIENBMB4XDTE3MDkxMjE0NDUwMDkxMjE0NDUwMDkxMjE0NDUwQz
ESMBAGCgMSJomT8ixkARKwAmNHRkWFwYKCZImiZPyLGBGRYJc2FuZGVsbW
FuMRIWEAYDVQDDAalsb2NhbgGhvc3QwWTATBgcqhkJOPQIBBggqhkJOPQM
BBwNCAAQ1ZA7Nw0xSM/Q2u194FzQMktZ94waAIV0i/oVTPg0J8zW6MwF5z+Dpb8
/puh0bJMZ0U6H/wfApR6svlumd4ryyow0wCzAJBgNVHRMEAjaAMAoGCCqGSM
49BAMDA2kAMGYCMQC3/iTQJ3evYYcgbXhbmzrp64t3QC6qjIeY2jkDx062nu
NifVKtyaara3F30AIkKSECMQDi29efbTLbdtDk3tecY/rD7V77XaJ6nYCmdD
CR54TrSFNLgxvt1lyFM+0fYpYRc3o="}}
831:d=3 hl=4 l= 467 cons: cont [ 0 ]
835:d=4 hl=4 l= 463 cons: SEQUENCE
839:d=5 hl=4 l= 342 cons: SEQUENCE
843:d=6 hl=2 l= 3 cons: cont [ 0 ]
845:d=7 hl=2 l= 1 prim: INTEGER :02
848:d=6 hl=2 l= 1 prim: INTEGER :01
851:d=6 hl=2 l= 10 cons: SEQUENCE
853:d=7 hl=2 l= 8 prim: OBJECT :ecdsa-with-S
HA256
863:d=6 hl=2 l= 77 cons: SEQUENCE
865:d=7 hl=2 l= 18 cons: SET
867:d=8 hl=2 l= 16 cons: SEQUENCE
869:d=9 hl=2 l= 10 prim: OBJECT :domainCompon
ent
881:d=9 hl=2 l= 2 prim: IA5STRING :ca
885:d=7 hl=2 l= 25 cons: SET
887:d=8 hl=2 l= 23 cons: SEQUENCE
889:d=9 hl=2 l= 10 prim: OBJECT :domainCompon
ent
901:d=9 hl=2 l= 9 prim: IA5STRING :sandelman
912:d=7 hl=2 l= 28 cons: SET
914:d=8 hl=2 l= 26 cons: SEQUENCE
916:d=9 hl=2 l= 3 prim: OBJECT :commonName
921:d=9 hl=2 l= 19 prim: UTF8STRING :Unstrung Hig

```


hway CA

942:d=6 hl=2 l= 30 cons: SEQUENCE
 944:d=7 hl=2 l= 13 prim: UTCTIME :170326161940

Z

959:d=7 hl=2 l= 13 prim: UTCTIME :190326161940

Z

974:d=6 hl=2 l= 71 cons: SEQUENCE
 976:d=7 hl=2 l= 18 cons: SET
 978:d=8 hl=2 l= 16 cons: SEQUENCE
 980:d=9 hl=2 l= 10 prim: OBJECT :domainCompon

ent

992:d=9 hl=2 l= 2 prim: IA5STRING :ca
 996:d=7 hl=2 l= 25 cons: SET
 998:d=8 hl=2 l= 23 cons: SEQUENCE
 1000:d=9 hl=2 l= 10 prim: OBJECT :domainCompon

ent

1012:d=9 hl=2 l= 9 prim: IA5STRING :sandelman
 1023:d=7 hl=2 l= 22 cons: SET
 1025:d=8 hl=2 l= 20 cons: SEQUENCE
 1027:d=9 hl=2 l= 3 prim: OBJECT :commonName
 1032:d=9 hl=2 l= 13 prim: UTF8STRING :Unstrung MAS

A

1047:d=6 hl=2 l= 118 cons: SEQUENCE
 1049:d=7 hl=2 l= 16 cons: SEQUENCE
 1051:d=8 hl=2 l= 7 prim: OBJECT :id-ecPublicK

ey

1060:d=8 hl=2 l= 5 prim: OBJECT :secp384r1
 1067:d=7 hl=2 l= 98 prim: BIT STRING
 1167:d=6 hl=2 l= 16 cons: cont [3]
 1169:d=7 hl=2 l= 14 cons: SEQUENCE
 1171:d=8 hl=2 l= 12 cons: SEQUENCE
 1173:d=9 hl=2 l= 3 prim: OBJECT :X509v3 Basic

Constraints

1178:d=9 hl=2 l= 1 prim: BOOLEAN :255
 1181:d=9 hl=2 l= 2 prim: OCTET STRING [HEX DUMP]:30

00

1185:d=5 hl=2 l= 10 cons: SEQUENCE
 1187:d=6 hl=2 l= 8 prim: OBJECT :ecdsa-with-S

HA256

1197:d=5 hl=2 l= 103 prim: BIT STRING
 1302:d=3 hl=4 l= 454 cons: SET
 1306:d=4 hl=4 l= 450 cons: SEQUENCE
 1310:d=5 hl=2 l= 1 prim: INTEGER :01
 1313:d=5 hl=2 l= 82 cons: SEQUENCE
 1315:d=6 hl=2 l= 77 cons: SEQUENCE
 1317:d=7 hl=2 l= 18 cons: SET
 1319:d=8 hl=2 l= 16 cons: SEQUENCE
 1321:d=9 hl=2 l= 10 prim: OBJECT :domainCompon

ent

1333:d=9 hl=2 l= 2 prim: IA5STRING :ca
 1337:d=7 hl=2 l= 25 cons: SET
 1339:d=8 hl=2 l= 23 cons: SEQUENCE
 1341:d=9 hl=2 l= 10 prim: OBJECT :domainCompon

ent

1353:d=9 hl=2 l= 9 prim: IA5STRING :sandelman
 1364:d=7 hl=2 l= 28 cons: SET
 1366:d=8 hl=2 l= 26 cons: SEQUENCE
 1368:d=9 hl=2 l= 3 prim: OBJECT :commonName
 1373:d=9 hl=2 l= 19 prim: UTF8STRING :Unstrung Hig

hway CA

1394:d=6 hl=2 l= 1 prim: INTEGER :01
 1397:d=5 hl=2 l= 13 cons: SEQUENCE
 1399:d=6 hl=2 l= 9 prim: OBJECT :sha256
 1410:d=6 hl=2 l= 0 prim: NULL
 1412:d=5 hl=3 l= 228 cons: cont [0]
 1415:d=6 hl=2 l= 24 cons: SEQUENCE
 1417:d=7 hl=2 l= 9 prim: OBJECT :contentType
 1428:d=7 hl=2 l= 11 cons: SET
 1430:d=8 hl=2 l= 9 prim: OBJECT :pkcs7-data
 1441:d=6 hl=2 l= 28 cons: SEQUENCE
 1443:d=7 hl=2 l= 9 prim: OBJECT :signingTime
 1454:d=7 hl=2 l= 15 cons: SET
 1456:d=8 hl=2 l= 13 prim: UTCTIME :171012175431

Z

1471:d=6 hl=2 l= 47 cons: SEQUENCE
 1473:d=7 hl=2 l= 9 prim: OBJECT :messageDiges

t

1484:d=7 hl=2 l= 34 cons: SET
 1486:d=8 hl=2 l= 32 prim: OCTET STRING [HEX DUMP]:41
 79C6EB6F1C216F0CA187C1D658C30E52E5250971103DAD9E372F90B11F8B

1D

1520:d=6 hl=2 l= 121 cons: SEQUENCE
 1522:d=7 hl=2 l= 9 prim: OBJECT :S/MIME Capab

ilities

1533:d=7 hl=2 l= 108 cons: SET
 1535:d=8 hl=2 l= 106 cons: SEQUENCE
 1537:d=9 hl=2 l= 11 cons: SEQUENCE
 1539:d=10 hl=2 l= 9 prim: OBJECT :aes-256-cbc
 1550:d=9 hl=2 l= 11 cons: SEQUENCE
 1552:d=10 hl=2 l= 9 prim: OBJECT :aes-192-cbc
 1563:d=9 hl=2 l= 11 cons: SEQUENCE
 1565:d=10 hl=2 l= 9 prim: OBJECT :aes-128-cbc
 1576:d=9 hl=2 l= 10 cons: SEQUENCE
 1578:d=10 hl=2 l= 8 prim: OBJECT :des-ed3-cbc
 1588:d=9 hl=2 l= 14 cons: SEQUENCE
 1590:d=10 hl=2 l= 8 prim: OBJECT :rc2-cbc


```

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

```

Authors' Addresses

Max Pritikin
Cisco

Email: pritikin@cisco.com

Michael C. Richardson
Sandelman Software Works

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

Michael H. Behringer

Email: Michael.H.Behringer@gmail.com

Steinthor Bjarnason
Arbor Networks

Email: sbjarnason@arbor.net

Kent Watsen
Juniper Networks

Email: kwatsen@juniper.net

