ANIMA WG
Internet-Draft
Intended status: Standards Track
Expires: November 24, 2017

M. Pritikin
Cisco
M. Richardson
SSW
M. Behringer
S. Bjarnason
Cisco
K. Watsen
Juniper Networks
May 23, 2017

Bootstrapping Remote Secure Key Infrastructures (BRSKI)
draft-ietf-anima-bootstrapping-keyinfra-06

Abstract

This document specifies automated bootstrapping of a remote secure key infrastructure (BRSKI) using vendor installed X.509 certificate, in combination with a vendor's authorizing service, both online the Internet, 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

Table of Contents

1. Introduction

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

- o Registrar authenticating the Pledge: "Who is this device? What is its identity?"
- o Registrar authorization the Pledge: "Is it mine? Do I want it? What are the chances it has been compromised?"

- o Pledge authenticating the Registrar/Domain: "What is this domain's identity?"
- o Pledge authorization the Registrar: "Should I join it?"

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

The syntactic details of vouchers are described in detail in [I-D.ietf-anima-voucher]. This document details automated protocol mechanisms to obtain vouchers.

BRSKI results in the Pledge storing an X.509 root certificate sufficient for verifying the Registrar identity. In the process a TLS connection is established which can be directly used for Enrollment over Secure Transport (EST). The Pledge can use these credentials to secure additional protocol exchanges.

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

1.1. Other Bootstrapping Approaches

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

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

- o using the Implicit Trust Anchor database (not an autonomic solution because the URL must be securely distributed),
- o engaging a human user to authorize the CA certificate using out-of-band data (not an autonomic solution because the human user is involved),

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

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

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

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

- o the pledge requires realtime connectivity to the vendor service,

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

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

BRSKI addresses these issues by defining "voucher" and automation extensions to the EST protocol.

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 domain trust anchor that is stored by the Domain CA. This is consistent with the Certification Authority subject key identifier (Section 4.2.1.2 [RFC5280]) of the Domain CA's self signed root certificate. (A string value bound to the Domain CA's self signed root certificate subject and issuer fields is often colloquially used as a humanized identity value but during protocol discussions the more exact term as defined here is used).

drop ship: The physical distribution of equipment containing the "factory default" configuration to a final destination. In zero-

touch scenarios there is no staging or pre-configuration during drop-ship.

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

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

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

Voucher A signed statement from the MASA service that indicates to a Pledge the cryptographic identity of the Registrar it should trust. There are different types of vouchers depending on how that trust asserted. Multiple voucher types are defined in [I-D.ietf-anima-voucher]

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

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

Join Registrar (and Coordinator): A representative of the domain that is configured, perhaps autonomically, to decide whether a new device is allowed to join the domain. The administrator of the domain interfaces with a Join Registrar (and Coordinator) to control this process. Typically a Join Registrar is "inside" its domain. For simplicity this document often refers to this as just "Registrar". The term JRC is used in common with other bootstrap mechanisms.

Join Proxy: A domain entity that helps the pledge join the domain. A Proxy facilitates communication for devices that find themselves in an environment where they are not provided connectivity until after they are validated as members of the domain. The pledge is unaware that they are communicating with a proxy rather than directly with a Registrar.

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

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

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

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

1.3. Scope of solution

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

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

In many target applications, the systems involved are large router platforms with multi-gigabit inter-connections, mounted in controlled access data centers. But this solution is not exclusive to the large, it is intended to scale to thousands of devices located in hostile environments, such as ISP provided CPE devices which are

drop-shipped to the end user. The situation where an order is fulfilled from distributed warehouse from a common stock and shipped directly to the target location at the request of the domain owner is explicitly supported. That stock ("SKU") could be provided to a number of potential domain owners, and the eventual domain owner will not know a-priori which device will go to which location.

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

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

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

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

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

2. Architectural Overview

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

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

Figure 1

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

The domain is the managed network infrastructure with a Key Infrastructure the Pledge is joining. The a domain provides initial device connectivity sufficient for bootstrapping with a Circuit Proxy. The Domain registrar authenticates the Pledge, makes authorization decisions, and distributes vouchers obtained from the Vendor Service. Optionally the Registrar also acts as a PKI Registration Authority.

2.1. Secure Imprinting using Vouchers

A voucher is a cryptographically protected statement to the Pledge device authorizing a zero-touch imprint on the Registrar domain.

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

Vouchers provide a flexible mechanism to secure imprinting: the Pledge device only imprints when a voucher can be validated. At the lowest security levels the MASA server can indiscriminately issue vouchers. At the highest security levels issuance of vouchers can be integrated with complex sales channel integrations that are beyond the scope of this document. This provides the flexibility for a number of use cases via a single common protocol mechanism on the Pledge and Registrar devices that are to be widely deployed in the field. The MASA vendor services have the flexibility to leverage either the currently defined claim mechanisms or to experiment with higher or lower security levels.

2.2. Initial Device Identifier

Pledge authentication is via an X.509 certificate installed during the manufacturing process. This Initial Device Identifier provides a basis for authenticating the Pledge during subsequent protocol exchanges and informing the Registrar of the MASA URI. There is no requirement for a common root PKI hierarchy. Each device vendor can generate their own root certificate.

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

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

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

The following newly defined field SHOULD be in the X.509 IDevID certificate: An X.509 non-critical certificate extension that

contains a single Uniform Resource Identifier (URI) that points to an on-line Manufacturer Authorized Signing Authority. The URI is represented as described in Section 7.4 of [RFC5280].

Any Internationalized Resource Identifiers (IRIs) MUST be mapped to URIs as specified in Section 3.1 of [RFC3987] before they are placed in the certificate extension. The URI provides the authority information. The BRSKI .well-known tree is described in Section 3

The new extension is identified as follows:

```
<CODE BEGINS>

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

DEFINITIONS IMPLICIT TAGS ::= BEGIN

-- EXPORTS ALL --

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

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

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

MASAURLSyntax ::= IA5String

END

<CODE ENDS>
```

The choice of id-pe is based on guidance found in Section 4.2.2 of [RFC5280], "These extensions may be used to direct applications to

on-line information about the issuer or the subject". The MASA URL
is precisely that: online information about the particular subject.

2.3. Protocol Flow

A representative flow is shown in Figure 2:

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

```
     P                      |       \---->      |                    |
     P                      |                   |                    |
     P                      |       [verify audit log and voucher]   |
     P                      |                   |                    |
     P<------voucher---------------------------|                    |
   [verify voucher ]        |                   |                    |
   [verify provisional cert|                   |                    |
     |                      |                   |                    |
     |<--------------------------------------->|                    |
     | Continue with RFC7030 enrollment         |                    |
     | using now bidirectionally authenticated |                    |
     | TLS session.         |                   |                    |
     |                      |                   |                    |
     |                      |                   |                    |
     |                      |                   |                    |
```

Figure 2

2.4. Lack of realtime clock

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

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

o Initially the Pledge does not know the current time.

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

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

- If the voucher contains a nonce then the Pledge MUST confirm the nonce matches the original voucher request. This ensures the voucher is fresh. See / (Section 3.2).

- Once the voucher is accepted the validity period of the domainCAcert in the voucher (see Section 3.4) now serves as a valid time window. Any subsequent certificate validity periods checked during RFC5280 path validation MUST occur within this window.

- 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.

2.5. Cloud Registrar

The Pledge MAY contact a well known URI of a cloud Registrar if a local Registrar can not be discovered or if the Pledge's target use cases do not include a local Registrar.

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

3. Protocol Details

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

BRSKI is described as extensions to EST [RFC7030] to reduce the number of TLS connections and crypto operations required on the Pledge. The Registrar implements the BRSKI REST interface within the same .well-known URI tree as the existing EST URIs as described in EST [RFC7030] section 3.2.2. A MASA URI is therefore "https:// authority "./well-known/est".

Establishment of the TLS connection for bootstrapping is as specified in EST [RFC7030] section 4.1.1 "Bootstrap Distribution of CA Certificates" [RFC7030] with the following extensions for automation:

Automation extensions for the Pledge (equivalent to EST client) are:

- The Pledge provisionally accepts the Registrar certificate during the TLS handshake as detailed in EST.

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

- o The Registar MAY respond with an HTTP 202 ("the request has been accepted for processing, but the processing has not been completed") as described in EST [RFC7030] section 4.2.3 wherein the client "MUST wait at least the specified 'retry-after' time before repeating the same request". The Pledge is RECOMMENDED to provide local feed (blinked LED etc) during this wait cycle if mechanisms for this are available. To prevent an attacker Registrar from significantly delaying bootstrapping the Pledge MUST limit the 'retry-after' time to 60 seconds. To avoid waiting on a single erroneous Registrar the Pledge MUST drop the connection after 5 seconds and proceed to other discovered Registrars. Ideally the Pledge could keep track of the appropriate retry-after value for any number of outstanding Registrars but this would involve a large state table on the Pledge. Instead the Pledge MAY ignore the exact retry-after value in favor of a single hard coded value that takes effect between discovery (Appendix D.1.1.1) attempts. A Registrar that is unable to complete the transaction the first time due to timing reasons will have future chances.

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

- o If necessary the Pledge calls the EST defined /cacerts method to obtain the current CA certificate. These are validated using the Voucher.

- o The Pledge completes authentication of the server certificate as detailed in Section 3.4.1. This moves the TLS connection out of the provisional state. Optionally the TLS connection can now be used for EST enrollment.

The Pledge establishes the TLS connection with the Registrar through the circuit proxy (see Appendix D.1.2) but the TLS connection is with the Registar; so in the above section the "Pledge" is the TLS client and the "Registrar" is the TLS server. All security associations established are between the new device and the Registrar regardless of proxy operations.

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

- o Client authentication is automated using Initial Device Identity. The subject field's DN encoding MUST include the "serialNumber"

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

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

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

- The Registar performs log verifications in addition to local authorization checks before accepting optional Pledge device enrollment requests.

3.1. Discovery

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 Registrar the Pledge performs the following actions:

a. MUST: Obtains a local address using IPv6 methods as described in [RFC4862] IPv6 Stateless Address AutoConfiguration. [RFC7217] is encouraged. Pledges will generally prefer use of IPv6 Link-Local addresses, and discovery of Proxy will be by Link-Local mechanisms. [[EDNOTE: In some environments, a routable public address may be obtained, should it be? Should it be used?]] IPv4 methods are described in Appendix A

b. MUST: Listen for GRASP M_FLOOD ([I-D.ietf-anima-grasp]) announcements of the objective: "ACP+Proxy". See section Section 3.1.1 for the details of the 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 3.

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

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

Once a connection to a Registrar is established (e.g. establishment of a TLS session key) there are expectations of more timely responses, see Section 3.2.

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

3.1.1. Proxy Discovery Protocol Details

The proxy uses the GRASP M_FLOOD mechanism to announce itself. This announcement is done with the same message as the ACP announcement detailed in [I-D.ietf-anima-autonomic-control-plane].

```
 proxy-objective = ["Proxy", [ O_IPv6_LOCATOR, ipv6-address,
 transport-proto, port-number ] ]

 ipv6-address       - the v6 LL of the proxy
 transport-proto    - 6, for TCP 17 for UDP
 port-number        - the TCP or UDP port number to find the proxy
```

Figure 5

3.1.2. Registrar Discovery Protocol Details

A Registrar is typically configured manually. When the Registrar joins an Autonomic Control Plane ([I-D.ietf-anima-autonomic-control-plane]) it MUST respond to GRASP ([I-D.ietf-anima-grasp]) M_NEG_SYN message.

The registrar responds to discovery messages from the proxy (or GRASP caches between them) as follows: (XXX changed from M_DISCOVERY)

```
 objective         = ["AN_registrar", F_DISC, 255 ]
 discovery-message = [M_NEG_SYN, session-id, initiator, objective]
```

Figure 6: Registrar Discovery

The response from the registrar (or cache) will be a M_RESPONSE with the following parameters:

```
 response-message = [M_RESPONSE, session-id, initiator, ttl,
 (+locator-option // divert-option), ?objective)]
 initiator = ACP address of Registrar
 locator1  = [O_IPv6_LOCATOR, fd45:1345::6789, 6,  443]
 locator2  = [O_IPv6_LOCATOR, fd45:1345::6789, 17, 5683]
 locator3  = [O_IPv6_LOCATOR, fe80::1234, 41, nil]
```

Figure 7: Registrar Response

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

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

Registrars MUST accept TCP / UDP traffic on the ports given at the ACP address of the Registrar. If the Registrar supports IPIP 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 in addition to TCP traffic.

3.2. **Request 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 "/requestvoucher".

The request is itself a voucher [I-D.ietf-anima-voucher]. The Pledge populates the voucher fields as follows:

assertion: The voucher request MUST contain an assertion of "proximity". [[EDNOTE: this is a placeholder as this commit is not the correct place to expand this list. Using proximity and channel binding: if both the Pledge and the Registrar sign the

channel binding statement then these provide vital proximity information to the MASA. To be expanded on in a future commit.]]

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

nonce: The voucher request MUST contain a cryptographically strong random or pseudo-random number nonce. Doing so ensures Section 2.4 functionality. The nonce MUST NOT be reused for multiple bootstrapping attempts.

All other fields MAY be ommitted in a voucher request. [[EDNOTE: An issue has been created for the voucher document to ensure normative language supports this]]

Signing the request is RECOMMENDED if the Pledge has sufficient processing to perform the crypto operations. Doing so allows the Registrar and MASA to confirm the "proximity" assertion of the Pledge.

Request media type: application/voucherrequest

An example JWS payload of the voucher request:

```
{
  "ietf-voucher:voucher": {
    "nonce": "62a2e7693d82fcda2624de58fb6722e5",
    "created-on": "2017-01-01T00:00:00.000Z",
    "assertion": "proximity"
  }
}
```

[[EDNOTE: The move to JWT allows for relatively simple signing operations. One possibility here is to carry the original signed JWT as an optional part of the payload sent to the MASA (e.g. "original request"). this would be a BRSKI addition to the voucher?]]

The Registrar validates the client identity as described in EST [RFC7030] section 3.3.2. The registrar performs authorization as detailed in Section 3.3.2. If authorization is successful the Registrar obtains an Voucher from the MASA service (see Section 3.3).

The received voucher request is forwarded to the Pledge.

3.3. Request Voucher from MASA

A Registrar requests a Voucher from the MASA service using a REST interface. For simplicity this is defined as an optional EST message between a Registrar and an EST server running on the MASA service although the Registrar is not required to make use of any other EST functionality when communicating with the MASA service. (The MASA service MUST properly reject any EST functionality requests it does not wish to service; a requirement that holds for any REST interface).

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

Request media type: application/voucherrequest+cms

The request format is a JSON object optionally containing the nonce value (as obtained from the bootstrap request) and the X.509 IDevID extracted serial number (the full certificate is not needed and no proof-of-possession information for the device identity is included). The AuthorityKeyIdentifier value from the certificate is included to ensure a statistically unique identity. The Pledge's serial number is extracted from the X.509 IDevID. See Section 2.2.

```
{
  "ietf-voucher:voucher": {
    "nonce": "62a2e7693d82fcda2624de58fb6722e5",
    "created-on": "2017-01-01T00:00:00.000Z",
    "assertion": "proximity"
    "device-identifier-aki": "[[EDNOTE:authority key identifier field
    from the IDevID. Voucher draft text that device-identifier is "A
    unique identifier (e.g., serial number) within the scope of the
    MASA" is insufficient because it prevents vendors from sharing
    a MASA]]"
    "device-identifier": "JADA123456789"
  }
}
```

A Registrar MAY exclude the nonce from the request. Doing so allows the Registrar to request a Voucher when the Pledge is not online, or when the target bootstrapping environment is not on the same network as the MASA server (this requires the Registrar to learn the appropriate DevIDSerialNumber field from the physical device labeling or from the sales channel -- how this occurs is out-of-scope of this document). If a nonce is not provided the MASA server MUST authenticate the Registrar as described in EST [RFC7030] section 3.3.2 to reduce the risk of DDoS attacks. The MASA performs authorization as detailed in Appendix D.1.3.2.

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 Registrar's' domainID) and the MASA has not been informed that the claim is invalid - the request for a renewed voucher SHOULD be automatically authorized. If authorization is successful the MASA responds with a [I-D.ietf-anima-voucher] voucher. The MASA SHOULD check for revocation of the Registrar certificate. The maximum lifetime of the voucher issued SHOULD NOT exceed the lifetime of the Registrar's revocation validation (for example if the Registrar revocation status is indicated in a CRL that is valid for two weeks then that is an appropriate lifetime for the voucher).

The voucher request is signed by the Registrar as indicated in [I-D.ietf-anima-voucher] voucher. The entire certificate chain, up to and including the Domain CA, MUST be included. The MASA service checks the internal consistency of the voucher request but does not authenticate the domain identity information since the domain is not know to the MASA server in advance. The MASA server MUST verify that the voucher request is signed by a Registrar certificate (by checking for the cmc-idRA field) that was issued by the self signed root certificate included in the request. [[EDNOTE: can we simplify the above sentence?]] This ensures that the Registrar making the claim is an authorized Registrar of the unauthenticated domain.

The root certificate is extracted and used to populate the Voucher. The domain ID (e.g. hash of the public key of the domain) is extracted from the root certificate and is used to update the audit log.

3.4. Voucher Response

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

Response media type: application/voucher+cms

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"
    "trusted-ca-certificate": "<base64 encoded certificate>"
    "device-identifier": "JADA123456789"
  }
}
```

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

The 'trusted-ca-certificate' element of the voucher contains the domain CA's public key. This is useful for bootstrapping a public key infrastructure but to support bootstrapping other key infrastructures additional domain identity types might be defined in the future.

The pledge's EST clients MUST be prepared to ignore additional fields they do not recognize.

the pledge MUST be prepared to parse and fail gracefully from an Voucher response that does not contain a 'trusted-ca-certificate' field at all.

The Pledge MUST use the 'trusted-ca-certificate' trust anchor to immediately complete authentication of the provisional TLS connection.

3.4.1. Completing authentication of Provisional TLS connection

If a Registrar's credentials can not be verified using the trusted-ca-certificate trust anchor from the voucher then the TLS connection is immediately discarded and the Pledge abandons attempts to bootstrap with this discovered registrar. The pledge SHOULD send voucher status telemetry (described below) before closing the TLS connection. The pledge MUST attempt to enroll using any other proxies it has found. It SHOULD return to the same proxy again after attempting with other proxies. Attempts should be attempted in the exponential backoff described earlier. Attempts SHOULD be repeated as failure may be the result of a temporary inconsistently (an inconsistently rolled Registrar key, or some other mis-configuration). The inconsistently could also be the result an active MITM attack on the EST connection.

To ensure that the trusted-ca-certificate provide chain is able to verify, the Registrar MUST use a certificate that chains to the trusted-ca-certificate.

The Pledge's PKIX path validation of a Registrar certificate's validity period information is as described in Section 2.4. Beyond that once PKIX path validation is successful the TLS connection is no longer provisional.

The trusted-ca-certificate is installed as an Explicit Trust Anchor for future operations. It can therefore can be used to authenticate any dynamically discovered EST server that contain the id-kp-cmcRA extended key usage extension as detailed in EST RFC7030 section 3.6.1; but to reduce system complexity the Pledge SHOULD avoid additional discovery operations. Instead the Pledge SHOULD communicate directly with the Registrar as the EST server for future key management operations. The 'trusted-ca-certificate' is not a complete distribution of the EST section 4.1.3 CA Certificate Response which is an additional justification for the recommendation to proceed with EST key management operations.

3.5. 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 client SHOULD post a status message.

The posted data media type: application/json

The client HTTP POSTs the following to the server at the EST well known URI /voucher_status. The Status field indicates if the Voucher was acceptable. If it was not acceptable the Reason string indicates why. In the failure case this message is being sent to an unauthenticated, potentially malicious Registrar and therefore the Reason string SHOULD NOT provide information beneficial to an attacker. The operational benefit of this telemetry information is balanced against the operational costs of not recording that an Voucher was ignored by a client the registar expected to continue joining the domain.

[[EDNOTE: the server can know which pledge failed by the previous voucher, I think. Is this worth noting?]]

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

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.

3.6. MASA authorization log Request

A registrar requests the MASA authorization log from the MASA service using this EST extension. If a device had previously registered with another domain, a Registrar of that domain would show in the log.

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

The client MUST HTTP POSTs the same Voucher Request as for requesting a Voucher. It is posted to the /requestauditlog URI instead. The IDevIDAuthorityKeyIdentifier and DevIDSerialNumber 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.

Request media type: application/voucherrequest+cms

3.7. MASA authorization log Response

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

```
{
  "version":"1",
  "events":[
    {
     "date":"<date/time of the entry>",
     "domainID":"<domainID as extracted from the domain CA certificate
                  within the CMS of the audit voucher request>",
     "nonce":"<any nonce if supplied (or the exact string 'NULL')>"
    },
    {
     "date":"<date/time of the entry>",
     "domainID":"<domainID as extracted from the domain CA certificate
                  within the CMS of the audit voucher request>",
     "nonce":"<any nonce if supplied (or the exact string 'NULL')>"
    }
  ]
}
```

Distribution of a large log is less than ideal. This structure can be optimized as follows: All nonce-less entries for the same domainID MAY be condensed into the single most recent nonceless entry.

A Registrar SHOULD use this log information to make an informed decision regarding the continued bootstrapping of the Pledge. For example if the log includes unexpected domainIDs this is indicative of problematic imprints by the Pledge. If the log includes nonce-less entries this is indicative of the permanent ability for the indicated domain to trigger a reset of the device and take over management of it. Equipment that is purchased pre-owned can be expected to have an extensive history. A Registrar MAY request logs at future times [[EDNOTE: we need to ensure MASA server is not slammed with too many requests]]. A Registrar MAY be configured to ignore the history of the device but it is RECOMMENDED that this only be configured if hardware assisted NEA [RFC5209] is supported.

Log entries containing the Domain's ID can be compared against local history logs in search of discrepancies.

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

3.8. EST Integration for PKI bootstrapping

This section describes EST extensions necessary to enable fully automated bootstrapping. Although the Voucher request/response structure members IDevIDAuthorityKeyIdentifier and DevIDSerialNumber are specific to PKI bootstrapping these are the only PKI specific aspects of the extensions and future work might replace them with non-PKI structures.

Once the Voucher is received, as specified in this document, the client has sufficient information to leverage the existing communication channel with a Registrar to continue an EST RFC7030 enrollment. The Pledge SHOULD use the existing current TLS connection to proceed with EST enrollment, thus reducing the total amount of cryptographic and round trip operations required during bootstrapping (enrollment picks up after EST RFC7030 "Bootstrap Distribution of CA Certificates" and the client continues with EST enrollment operations including "CA Certificates Request", "CSR Attributes" and "Client Certificate Request" or "Server-Side Key Generation").

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

[[EDNOTE:might be best to discuss in CSR attributes?]]For the purposes of creating the ANIMA Autonomic Control Plane, the contents of the new certificate MUST be carefully specified. [I-D.ietf-anima-autonomic-control-plane] section 5.1.1 contains details. The Registrar MUST provide the the correct ACP information to populate the subjectAltName / rfc822Name field in the "CSR Attributes" step.

3.8.1. EST Distribution of CA Certificates

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

This ensures that the Pledge has the complete set of current CA certificates beyond the domainCAcert (see Section 3.4 for a discussion of the limitations). Although these restrictions are acceptable for a Registrar integrated with initial bootstrapping they are not appropriate for ongoing PKIX end entity certificate validation.

3.8.2. EST CSR Attributes

Automated bootstrapping occurs without local administrative configuration of the Pledge. In some deployments its plausible that the Pledge generates a certificate request containing only identity information known to the Pledge (essentially the X.509 IDevID information) and ultimately receives a certificate containing domain specific identity information. Conceptually the CA has complete control over all fields issued in the end entity certificate. Realistically this is operationally difficult with the current status of PKI certificate authority deployments where the CSR is submitted to the CA via a number of non-standard protocols.

To alleviate operational difficulty the Pledge MUST request the EST "CSR Attributes" from the EST server. This allows the local infrastructure to inform the Pledge of the proper fields to include in the generated CSR.

[[EDNOTE: The following is specific to anima purposes and should be moved to an appropriate anima document so as to keep bootstrapping as generic as possible: What we want are a 'domain name' stored in [TBD] and an 'ACP IPv6 address' stored in the iPAddress field as specified in RFC5208 s4.2.1.6. ref ACP draft where certificate verification [TBD]. These should go into the subjectaltname in the [TBD] fields.]]. 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 registar SHOULD support local policy concerning this functionality. [[EDNOTE: extensive use of EST CSR Attributes might need an new OID definition]].]]

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

3.8.3. EST Client Certificate Request

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

3.8.4. Enrollment Status Telemetry

For automated bootstrapping of devices the adminstrative elements providing bootstrapping also provide indications to the system administrators concerning device lifecycle status. This might include information concerning attempted bootstrapping messages seen

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

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

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

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

Status media type: application/json

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

```
{
  "version":"1",
  "Status":TRUE /* TRUE=Success, FALSE=Fail"
  "Reason":"Informative human readable message"
  "SubjectKeyIdentifier":"<base64 encoded subjectkeyidentifier for the
                           enrollment that failed>"
}
```

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.

3.8.5. EST over CoAP

[[EDNOTE: In order to support smaller devices the above section on Proxy behavior introduces mandatory to implement support for CoAP support by the Proxy. This implies similar support by the Pledge and Registrar and means that the EST protocol operation encapsulation into CoAP needs to be described. EST is HTTP based and "CoaP is designed to easily interface with HTTP for integration" [RFC7252]. Use of CoAP implies Datagram TLS (DTLS) wherever this document describes TLS handshake specifics. A complexity is that the large message sizes necessary for bootstrapping will require support for [draft-ietf-core-block].]]

4. 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.

4.1. Trust Model

```
+--------+         +---------+    +------------+     +------------+
| New    |         | Circuit |    | Domain     |     | Vendor     |
| Entity |         | Proxy   |    | Registrar  |     | Service    |
|        |         |         |    |            |     | (Internet  |
+--------+         +---------+    +------------+     +------------+
```

Figure 10

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

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

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

Vendor Service, MASA: This form of vendor service is trusted to accurately log all claim attempts and to provide authoritative log information to Registrars. The MASA does not know which devices are associated with which domains. These claims could be

strengthened by using cryptographic log techniques to provide append only, cryptographic assured, publicly auditable logs. Current text provides only for a trusted vendor.

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

4.2. New Entity security reductions

The Pledge MAY support "trust on first use" on physical interfaces but MUST NOT support "trust on first use" on network interfaces. This is because "trust on first use" permanently degrades the security for all other use cases.

The Pledge MAY have an operational mode where it skips Voucher validation one time. For example if a physical button is depressed during the bootstrapping operation. This can be useful if the vendor service is unavailable. This behavior SHOULD be available via local configuration or physical presence methods to ensure new entities can always be deployed even when autonomic methods fail. This allows for unsecured imprint.

It is RECOMMENDED that this only be available if hardware assisted NEA [RFC5209] is supported.

4.3. 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 3.2 request using the Section 3.3 format to ensure the Pledge's serial number information is provided to the Registar (this includes the IDevIDAuthorityKeyIdentifier 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 request nonce-less Vouchers from the MASA service (by not including a nonce in the request). These Vouchers can then be transmitted to the Registrar and stored until they are needed during bootstrapping operations. This is for use cases where target network is protected by an air gap and therefore can not contact the MASA service during Pledge deployment.

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

4.4. MASA security reductions

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

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

2. Not verifying ownership before responding with an Voucher. This is expected to be a common operational model because doing so relieves the vendor providing MASA services from having to track ownership during shipping and supply chain and allows for a very low overhead MASA service. A Registrar uses the audit log

information as a defense in depth strategy to ensure that this does not occur unexpectedly (for example when purchasing new equipment the Registrar would throw an error if any audit log information is reported).

5. IANA Considerations

5.1. PKIX Registry

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

6. Security Considerations

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

The issuance of nonceless vouchers themselves create a security concern. If the Registrar of a previous domain can intercept protocol communications then it can use a previously issued nonceless voucher to establish management control of a pledge device even after having sold it. This risk is mitigated by recording the issuance of such vouchers in the MASA audit log that is verified by the subsequent Registrar. This reduces the resale value of the equipment because future owners will detect the lowered security inherent in the existence of a nonceless voucher that would be trusted by their Pledge. This accurately reflects a balance between partition resistant recovery and security of future bootstrapping. Registrars take the Pledge's audit history into account when applying policy to new devices.

The MASA server is exposed to DoS attacks wherein attackers claim an unbounded number of devices. Ensuring a Registrar is representative of a valid vendor customer, even without validating ownership of specific Pledge devices, helps to mitigate this. Inserting a cryptographic proof-of-possession step to the protocol operations is a possible area of future work. One method that would not introduce

additional round-trips would be for the Registrar to share the Pledge-Registrar TLS handshake with the MASA service when requesting a voucher. Doing so would allow the MASA service to verify that the Registrar's Server Certificate was signed by the Pledge's Certificate Verify message (which covers the entire handshake). [[EDNOTE: Security Considerations should not offer up new protocol ideas without a reason for having not done it...]]

It is possible for an attacker to request a voucher from the MASA service directly after the real Registrar obtains an audit log. If the attacker could also force the bootstrapping protocol to reset there is a theoretical opportunity for the attacker to use their voucher to take control of the Pledge but then proceed to enroll with the target domain. Possible prevention mechanisms include:

- o Per device rate limits on the MASA service ensure such timing attacks are difficult.

- o The Registrar can repeat the request for audit log information at some time after bootstrapping is complete.

To facilitate logging and administrative oversight the Pledge reports on Voucher parsing status to the Registrar. In the case of a failure this information is informative to a potentially malicious Registar but this is RECOMMENDED anyway because of the operational benefits of an informed administrator in cases where the failure is indicative of a problem.

To facilitate truely limited clients EST RFC7030 section 3.3.2 requirements that the client MUST support a client authentication model have been reduced in Section 4 to a statement that clients only "SHOULD" support such a model. This reflects current (poor) practices that are NOT RECOMMENDED.

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

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

7. Acknowledgements

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

8. References

8.1. Normative References

[I-D.ietf-anima-autonomic-control-plane]
Behringer, M., Eckert, T., and S. Bjarnason, "An Autonomic Control Plane", draft-ietf-anima-autonomic-control-plane-06 (work in progress), March 2017.

[I-D.ietf-anima-voucher]
Watsen, K., Richardson, M., Pritikin, M., and T. Eckert, "Voucher Profile for Bootstrapping Protocols", draft-ietf-anima-voucher-02 (work in progress), March 2017.

[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, <http://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, <http://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, <http://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, <http://www.rfc-editor.org/info/rfc3927>.

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

[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, <http://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, <http://www.rfc-editor.org/info/rfc5386>.

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

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

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

[RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013, <http://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, <http://www.rfc-editor.org/info/rfc7030>.

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

8.2. Informative References

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

[I-D.ietf-anima-grasp]
Bormann, C., Carpenter, B., and B. Liu, "A Generic Autonomic Signaling Protocol (GRASP)", draft-ietf-anima-grasp-12 (work in progress), May 2017.

[I-D.ietf-netconf-zerotouch]
Watsen, K. and M. Abrahamsson, "Zero Touch Provisioning for NETCONF or RESTCONF based Management", draft-ietf-netconf-zerotouch-13 (work in progress), March 2017.

[I-D.lear-mud-framework]
Lear, E., "Manufacturer Usage Description Framework", draft-lear-mud-framework-00 (work in progress), January 2016.

[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-01 (work in progress), July 2016.

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

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

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

[RFC7435] Dukhovni, V., "Opportunistic Security: Some Protection Most of the Time", RFC 7435, DOI 10.17487/RFC7435, December 2014, <http://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, <http://www.rfc-editor.org/info/rfc7575>.

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

Appendix A. IPv4 operations

A.1. IPv4 Link Local addresses

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

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

A.2. Use of DHCPv4

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

Appendix B. mDNS / DNSSD proxy discovery options

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

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

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

If no local bootstrapks service is located using the GRASP mechanisms, or the above mentioned DNS-based Service Discovery methods the Pledge MAY contact a well known vendor provided

bootstrapping server by performing a DNS lookup using a well known URI such as "bootstrapks.vendor-example.com". The details of the URI are vendor specific. Vendors that leverage this method on the Pledge are responsible for providing the bootstrapks service.

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

Appendix C. IPIP Join Proxy mechanism

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

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

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

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

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

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

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

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

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

C.2. Automatic configuration of tunnels on Registrar

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

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

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

C.3. Proxy Neighbor Discovery by Join Proxy

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

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

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

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

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

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

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

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

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

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

Appendix D. To be deprecated: Consolidation remnants

[[EDNOTE: As per working group feedback there were multiple instances where this document repeated itself. To address this we have moved all text to this appendix and restored only one copy of each

normative discussion. The next pass will reduce and delete this appendix to '0'; although some may be maintained in a design considerations appendix.]]

D.1. Functional Overview

Entities behave in an autonomic fashion. They discover each other and autonomically bootstrap into a key infrastructure delineating the autonomic domain. See [RFC7575] for more information.

This section details the state machine and operational flow for each of the main three entities. The pledge, the domain (primarily a Registrar) and the MASA service.

A representative flow is shown in Figure 2:

```
   +--------+         +---------+    +------------+     +------------+
   | Pledge |         | Circuit |    | Domain     |     | Vendor     |
   |        |         | Proxy   |    | Registrar  |     | Service    |
   |        |         |         |    |            |     | (Internet  |
   +--------+         +---------+    +------------+     +------------+
     |                     |                   |                    |
     |<-RFC3927 IPv4 adr   | Appendix A        |                    |
   or|<-RFC4862 IPv6 adr   |                   |                    |
     |                     |                   |                    |
     |-------------------->|                   |                    |
     | optional: mDNS query| Appendix B        |                    |
     | RFC6763/RFC6762     |                   |                    |
     |                     |                   |                    |
     |<--------------------|                   |                    |
     | GRASP M_FLOOD       |                   |                    |
     |   periodic broadcast|                   |                    |
     |                     |                   |                    |
     |<------------------->C<----------------->|                    |
     |              TLS via the Circuit Proxy  |                    |
     |<--Registrar TLS server authentication---|                    |
   [PROVISIONAL accept of server cert]         |                    |
     P---X.509 client authentication---------->|                    |
     P                     |                   |                    |
     P---Request Voucher (include nonce)------>|                    |
     P                     |                   |                    |
     P                     |      /--->        |                    |
     P                     |      |      [accept device?]           |
     P                     |      |      [contact Vendor]           |
     P                     |      |            |--Pledge ID-------->|
     P                     |      |            |--Domain ID-------->|
     P                     |      |            |--optional:nonce--->|
     P                     |      |            |     [extract DomainID]
```

```
     P                     |       |           |                    |
     P                     |    optional:      |     [update audit log]
     P                     |       |can        |                    |
     P                     |       |occur      |                    |
     P                     |       |in         |                    |
     P                     |       |advance    |                    |
     P                     |       |           |                    |
     P                     |       |           |<-device audit log--|
     P                     |       |           |<- voucher ---------|
     P                     |       \---->      |                    |
     P                     |                   |                    |
     P                     |       [verify audit log and voucher]   |
     P                     |                   |                    |
     P<------voucher---------------------------|                    |
   [verify voucher ]       |                   |                    |
   [verify provisional cert|                   |                    |
     |                     |                   |                    |
     |<--------------------------------------->|                    |
     | Continue with RFC7030 enrollment        |                    |
     | using now bidirectionally authenticated |                    |
     | TLS session.        |                   |                    |
     |                     |                   |                    |
     |                     |                   |                    |
     |                     |                   |                    |
```

Figure 2

[[UNRESOLVED:need to restore some functional overview section for all these diagrams]]In order to obtain a Voucher and associated logs a Registrar contacts the MASA service Service using REST calls:

```
              +-----------+ +----------+ +-----------+ +----------+
              | New       | | Circuit  | |           | |          |
              | Entity    | | Proxy    | | Registrar | | Vendor   |
              |           | |          | |           | |          |
              ++----------+ +--+-------+ +-----+-----+ +--------+-+
               |               |               |                |
               |               |               |                |
               |   TLS hello   |  TLS hello    |                |
   Establish   +---------------C--------------->                |
   TLS         |               |               |                |
   connection  |               | Server Cert   |                |
               <---------------C---------------+                |
               | Client Cert   |               |                |
               +---------------C--------------->                |
               |               |               |                |
   HTTP REST   | POST /requestvoucher          |                |
   Data        +--------------------nonce------>                |
               |               .               | /requestvoucher|
               |               .               +---------------->
               |                               <----------------+
               |                               | /requestlog    |
               |                               +---------------->
               |            voucher            <----------------+
               <-------------------------------+                |
               | (optional config information) |                |
               |               .               |                |
               |               .               |                |
```

Figure 8

In some use cases the Registrar may need to contact the Vendor in advanced, for example when the target network is air-gapped. The nonceless request format is provided for this and the resulting flow is slightly different. The security differences associated with not knowing the nonce are discussed below:

```
              +-----------+ +----------+ +-----------+ +----------+
              | New       | | Circuit  | |           | |          |
              | Entity    | | Proxy    | | Registrar | | Vendor   |
              |           | |          | |           | |          |
              ++----------+ +--+-------+ +-----+-----+ +--------+-+
               |               |               |                |
               |               |               |                |
               |               |               | /requestvoucher|
               |               |  (nonce       +---------------->
               |               |  unknown)     <----------------+
               |               |               | /requestlog    |
               |               |               +---------------->
               |               |               <----------------+
               |   TLS hello   |  TLS hello    |                |
   Establish   +---------------C--------------->                |
   TLS         |               |               |                |
   connection  |               | Server Cert   |                |
               <---------------C---------------+                |
               | Client Cert   |               |                |
               |               |               |                |
   HTTP REST   | POST /requestvoucher          |                |
   Data        +----------------------nonce---->   (discard     |
               |            voucher            |   nonce)       |
               <-------------------------------+                |
               | (optional config information) |                |
               |               .               |                |
               |               .               |                |
```

Figure 9

D.1.1. Behavior of a Pledge

A pledge that has not yet been bootstrapped attempts to find a local domain and join it. A pledge [[RESOLVED:MUST NOT]] automatically initiate bootstrapping if it has already been configured or is in the process of being configured.

States of a pledge are as follows:

```
                +--------------+
                |   Start      |
                |              |
                +------+-------+
                       |
                +------v-------+
                |  Discover    |
   +------------>              |
   |            +------+-------+
   |                   |
   |            +------v-------+
   |            |  Identity    |
   ^------------+              |
   | rejected   +------+-------+
   |                   |
   |            +------v-------+
   |            | Request      |
   |            | Join         |
   |            +------+-------+
   |                   |
   |            +------v-------+
   |            |  Imprint     |   Optional
   ^------------+              <--+Manual input (Appendix C)
   | Bad Vendor +------+-------+
   | response          |
   |            +------v-------+
   |            |  Enroll      |
   ^------------+              |
   | Enroll     +------+-------+
   | Failure           |
   |            +------v-------+
   |            |  Being       |
   ^------------+  Managed     |
    Factory     +--------------+
    reset
```

Figure 3

State descriptions for the pledge are as follows:

1. Discover a communication channel to a Registrar.

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

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

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

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

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

The following sections describe each of these steps in more detail.

D.1.1.1. Discovery

[[RESOLVED:TEXT moved up into above]]

D.1.1.2. Identity

The Pledge identifies itself during the communication protocol handshake. If the client identity is rejected (that is, the TLS handshake does not complete) the Pledge repeats the Identity process using the next proxy or discovery method available.

[[RESOLVED: need normative statement in protocol section]] The bootstrapping protocol server is not initially authenticated. Thus the connection is provisional and all data received is untrusted until sufficiently validated even though it is over a TLS connection. This is aligned with the existing provisional mode of EST [RFC7030] during s4.1.1 "Bootstrap Distribution of CA Certificates". See Section 3.4 for more information about when the TLS connection authentication is completed.

[[RESOLVED:]]All security associations established are between the new device and the Bootstrapping server regardless of proxy operations.

D.1.1.2.1. Concurrent attempts to join

[[RESOLVED: by dropping this text. the "priority mechanism" is unspecified thus any discussion is unclear. Not only that once an initial request is sent to the registrar the question of multiple MASA interactions has already occurred. Nothing breaks if implementations do this. I've added text to the security considerations indicating the end result (MASA entries that might be ignored by the device but which confuse the end administrator)]] The Pledge MAY attempt multiple mechanisms concurrently, but if it does so, it MUST wait in the provisional state until all mechanisms have either succeeded or failed, and then MUST proceed with the highest priority mechanism which has succeed. To proceed beyond this point, specifically, to provide a nonce, could result in the MASA gratuitously auditing a connection.

D.1.1.3. Request Join

The Pledge POSTs a request to join the domain to the Bootstrapping server. This request contains a Pledge generated nonce and informs the Bootstrapping server which imprint methods the Pledge will accept.

The nonce ensures the Pledge can verify that responses are specific to this bootstrapping attempt. This minimizes the use of global time and provides a substantial benefit for devices without a valid clock.

D.1.1.3.1. Redirects during the Join Process

[[RESOVED via current root protocol discussion. reference to mdnsmethods is dropped]] EST [RFC7030] describes situations where the bootstrapping server MAY redirect the client to an alternate server via a 3xx status code. Such redirects MAY be accepted if the pledge has used the methods described in Appendix B, in combination with an implicit trust anchor. Redirects during the provisional period are otherwise unstrusted, and MUST cause a failure.

D.1.1.4. Imprint

The Pledge validates the voucher and accepts the Registrar ID. The provisional TLS connection is validated using the Registrar ID from the voucher.

D.1.1.5. Lack of realtime clock APPENDIX

[[RESOVED: entire section promoted back into the main text]]

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

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

- o Initially the Pledge does not know the current time.

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

- o The Pledge authenticates the voucher presented to it. During this authentication the Pledge ignores certificate lifetimes (by necessity because it does not have a clock). The voucher itself SHOULD contain the nonce included in the original request which proves the voucher is fresh.

- o Once the voucher is accepted the validity period of the domainCAcert in the voucher (see Section 3.4) now serves as a valid time window. Any subsequent certificate validity periods checked during RFC5280 path validation MUST occur within this window.

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

D.1.1.6. Enrollment

As the final step of bootstrapping a Registrar helps to issue a domain specific credential to the Pledge. For simplicity in this document, a Registrar primarily facilitates issuing a credential by acting as an RFC5280 Registration Authority for the Domain Certification Authority.

Enrollment proceeds as described in [RFC7030]. Authentication of the EST server is done using the Voucher rather than the methods defined in EST.

[[RESOLVED: moved to protocol discussion]]Once the Voucher is received, as specified in this document, the client has sufficient information to leverage the existing communication channel with a Registrar to continue an EST RFC7030 enrollment. Enrollment picks up at RFC7030 section 4.1.1. bootstrapping where the Voucher provides the "out-of-band" CA certificate fingerprint (in this case the full CA certificate) such that the client can now complete the TLS server authentication. At this point the client continues with EST enrollment operations including "CA Certificates Request", "CSR Attributes" and "Client Certificate Request" or "Server-Side Key Generation".

[[RESOLVED: included into EST discussion]]For the purposes of creating the ANIMA Autonomic Control Plane, the contents of the new certificate MUST be carefully specified. [I-D.ietf-anima-autonomic-control-plane] section 5.1.1 contains details. The Registrar MUST provide the the correct ACP information to populate the subjectAltName / rfc822Name field in the "CSR Attributes" step.

D.1.1.7. Being Managed

[[RESOLVED: by slight change to introduction text.]] Functionality to provide generic "configuration" information is supported. The parsing of this data and any subsequent use of the data, for example communications with a Network Management System is out of scope but is expected to occur after bootstrapping enrollment is complete. This ensures that all communications with management systems which can divulge local security information (e.g. network topology or raw key material) is secured using the local credentials issued during enrollment.

The Pledge uses bootstrapping to join only one domain. Management by multiple domains is out-of-scope of bootstrapping. After the device has successfully joined a domain and is being managed it is plausible that the domain can insert credentials for other domains depending on the device capabilities.

See Appendix D.1.5.

D.1.2. Behavior of a Join Proxy

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

[[UNRESOLVED: since proxy behavior is not visible we can limit ourselves to discussion of what the protocol does to enable/faciliate a theoretical proxy]]The Proxy does not terminate the TLS handshake.

[[UNRESOLVED: this is an anima architecture requirement to use BRSKI? move to there?]] A Proxy is always assumed even if it is directly integrated into a Registrar. (In a completely autonomic network, the Registrar MUST provide proxy functionality so that it can be discovered, and the network can grow concentrically around the Registrar)

As a result of the Proxy Discovery process in section Appendix D.1.1.1, the port number exposed by the proxy does not need to be well known, or require an IANA allocation.

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

For the IPIP encapsulation methods, the port announced by the Proxy MUST 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 of the alternative proxy methods.

D.1.2.1. CoAP connection to Registrar

[[RESOLVED:this section thus removed]]The CoAP mechanism was depreciated.

D.1.2.2. HTTPS proxy connection to Registrar

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

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

When the Proxy provides a stateless IPIP encapsulation to a Registrar, then the Registrar will have to perform IPIP decapsulation, remembering the originating outer IPIP source address in order to qualify the inner link-local address. This is a kind of encapsulation and processing which is similar in many ways to how mobile IP works.

Being able to connect a TCP (HTTP) or UDP (CoAP) socket to a link-local address with an encapsulated IPIP header requires API extensions beyond [RFC3542] for UDP use, and requires a form of connection latching (see section 4.1 of [RFC5386] and all of [RFC5660], except that a simple IPIP tunnel is used rather than an IPsec tunnel).

D.1.3. Behavior of the Registrar

A Registrar listens for Pledges and determines if they can join the domain. A Registrar obtains a Voucher from the MASA service and delivers them to the Pledge as well as facilitating enrollment with the domain PKI.

[[RESOLVED: moved to discovery discussion]] A Registrar is typically configured manually. When the Registrar joins an Autonomic Control Plane ([I-D.ietf-anima-autonomic-control-plane]) it MUST respond to GRASP ([I-D.ietf-anima-grasp]) M_DISCOVERY message. See Section 3.1.2

Registrar behavior is as follows:

```
   Contacted by Pledge
           +
           |
   +-------v----------+
   | Entity           | fail?
   | Authentication   +---------+
   +-------+----------+         |
           |                    |
   +-------v----------+         |
   | Entity           | fail?   |
   | Authorization    +--------->
   +-------+----------+         |
           |                    |
   +-------v----------+         |
   | Claiming the     | fail?   |
   | Entity           +--------->
   +-------+----------+         |
           |                    |
   +-------v----------+         |
   | Log Verification | fail?   |
   |                  +--------->
   +-------+----------+         |
           |                    |
   +-------v----------+    +----v-------+
   | Forward          |    |            |
   | Voucher          |    | Reject     |
   | to the Pledge    |    | Device     |
   |                  |    |            |
   +------------------+    +------------+
```

Figure 4

D.1.3.1. **Pledge Authentication**

The applicable authentication methods detailed in EST [RFC7030] are:

o [[RESOLVED:pointed out in protocol details]]the use of an X.509 IDevID credential during the TLS client authentication,

o or the use of a secret that is transmitted out of band between the Pledge and a Registrar (this use case is not autonomic).

In order to validate the X.509 IDevID credential a Registrar maintains a database of vendor trust anchors (e.g. vendor root certificates or keyIdentifiers for vendor root public keys). For user interface purposes this database can be mapped to colloquial vendor names. Registrars can be shipped with the trust anchors of a significant number of third-party vendors within the target market.

D.1.3.2. Pledge Authorization

[[UNRESOLVED: this is referenced above as how the MASA does authorization. That is incorrect]]

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

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

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

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

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

[[RESOLVED: this looks like good text to include in above]]To look the Pledge up in a domain white list a consistent method for extracting device identity from the X.509 certificate is required. RFC6125 describes Domain-Based Application Service identity but here we require Vendor Device-Based identity. The subject field's DN encoding MUST include the "serialNumber" attribute with the device's unique serial number. In the language of RFC6125 this provides for a SERIALNUM-ID category of identifier that can be included in a certificate and therefore that can also be used for matching purposes. The SERIALNUM-ID whitelist is collated according to vendor trust anchor since serial numbers are not globally unique.

[[RESOLVED: into log request]]The Registrar MUST use the vendor provided MASA service to verify that the device's history log does not include unexpected Registrars. If a device had previously registered with another domain, a Registrar of that domain would show in the log.

[[RESOLVED: est integration section used 'SHOULD']]The authorization performed during BRSKI MAY be used for EST enrollment requests by proceeding with EST enrollment using the authenticated and authorized TLS connection. This minimizes the number of cryptographic and protocol operations necessary to complete bootstrapping of the local key infrastructure.

D.1.3.3. Claiming the New Entity

Claiming an entity establishes an audit log at the MASA server and provides a Registrar with proof, in the form of the Voucher, that the log entry has been inserted. As indicated in Appendix D.1.1.4 a Pledge will only proceed with bootstrapping if a Voucher has been received. The Pledge therefore enforces that bootstrapping only occurs if the claim has been logged. There is no requirement for the vendor to definitively know that the device is owned by the Registrar.

The Registrar obtains the MASA URI via static configuration or by extracting it from the X.509 IDevID credential. See Section 2.2.

During initial bootstrapping the Pledge provides a nonce specific to the particular bootstrapping attempt. [[RESOLVED: to resolve this I updated many points where vouchers are referenced]]The Registrar SHOULD include this nonce when claiming the Pledge from the MASA service. Claims from an unauthenticated Registrar are only serviced by the MASA resource if a nonce is provided.

The Registrar can claim a Pledge that is not online by forming the request using the entities unique identifier and not including a nonce in the claim request. Vouchers obtained in this way do not have a lifetime and they provide a permanent method for the domain to claim the device. Evidence of such a claim is provided in the audit log entries available to any future Registrar. Such claims reduce the ability for future domains to secure bootstrapping and therefore the Registrar MUST be authenticated by the MASA service although no requirement is implied that the MASA associates this authentication with ownership.

An Ownership Voucher requires the vendor to definitively know that a device is owned by a specific domain. The method used to "claim" this are out-of-scope. A MASA ignores or reports failures when an attempt is made to claim a device that has a an Ownership Voucher.

D.1.3.4. Log Verification

A Registrar requests the log information for the Pledge from the MASA service. The log is verified to confirm that the following is true to the satisfaction of a Registrar's configured policy:

o Any nonceless entries in the log are associated with domainIDs recognized by the registrar.

o Any nonce'd entries are older than when the domain is known to have physical possession of the Pledge or that the domainIDs are recognized by the registrar.

If any of these criteria are unacceptable to a Registrar the entity is rejected. [[RESOLVED: moved to main body]] 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.

[[RESOLVED: added to main text]]This document specifies a simple log format as provided by the MASA service to the registar. This format could be improved by distributed consensus technologies that integrate vouchers with a technologies such as block-chain or hash trees or the like. Doing so is out of the scope of this document but are anticipated improvements for future work.

D.1.4. Behavior of the MASA Service

[[UNRESOLVED: primary value of keeping this discussion is to distinguish between registrar and masa particularly wrt to the protocol functions provided. perhaps add statements in each protocol entry "provided by masa" etc?]]

The Manufacturer Authorized Signing Authority service is directly provided by the manufacturer, or can be provided by a third party the manufacturer authorizes. It is a cloud resource. The MASA service provides the following functionalities to Registrars:

Issue Vouchers: In response to Registrar requests the MASA service issues vouchers. Depending on the MASA policy the Registrar claim of device ownership is either accepted or verified using out-of-scope methods (that are expected to improve over time).

Log Vouchers Issued: When a voucher is issued the act of issuing it includes updating the certifiable logs. Future work to enhance and distribute these logs is out-of-scope but expected over time.

Provide Logs: As a baseline implementation of the certified logging mechanism the MASA is repsonsible for reporting logged information. The current method involves trusting the MASA. Other logging methods where the MASA is less trusted are expected to be developed over time.

D.1.5. Leveraging the new key infrastructure / next steps

As the devices have a common trust anchor, device identity can be securely established, making it possible to automatically deploy services across the domain in a secure manner.

Examples of services:

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

D.1.5.1. Network boundaries

When a device has joined the domain, it can validate the domain membership of other devices. This makes it possible to create trust boundaries where domain members have higher level of trusted than external devices. Using the autonomic User Interface, specific devices can be grouped into to sub domains and specific trust levels can be implemented between those.

D.1.6. Interactions with Network Access Control

[[RESOLVED: via paragraph in 'scope of solution' discussion.]]

The assumption is that Network Access Control (NAC) completes using the Pledge 's X.509 IDevID credentials and results in the device having sufficient connectivity to discovery and communicate with the proxy. Any additional connectivity or quarantine behavior by the NAC infrastructure is out-of-scope. After the devices has completed bootstrapping the mechanism to trigger NAC to re-authenticate the device and provide updated network privileges is also out-of-scope.

This achieves the goal of a bootstrap architecture that can integrate with NAC but does not require NAC within the network where it wasn't previously required. Future optimizations can be achieved by integrating the bootstrapping protocol directly into an initial EAP exchange.

D.2. Domain Operator Activities

This section describes how an operator interacts with a domain that supports the bootstrapping as described in this document.

D.2.1. Instantiating the Domain Certification Authority

This is a one time step by the domain administrator. This is an "off the shelf" CA with the exception that it is designed to work as an integrated part of the security solution. This precludes the use of 3rd party certification authority services that do not provide support for delegation of certificate issuance decisions to a domain managed Registration Authority.

D.2.2. Instantiating the Registrar

This is a one time step by the domain administrator. One or more devices in the domain are configured take on a Registrar function.

A device can be configured to act as a Registrar or a device can auto-select itself to take on this function, using a detection mechanism to resolve potential conflicts and setup communication with the Domain Certification Authority. Automated Registrar selection is outside scope for this document.

D.2.3. Accepting New Entities

For each Pledge the Registrar is informed of the unique identifier (e.g. serial number) along with the manufacturer's identifying information (e.g. manufacturer root certificate). This can happen in different ways:

1. Default acceptance: In the simplest case, the new device asserts its unique identity to a Registrar. The registrar accepts all devices without authorization checks. This mode does not provide security against intruders and is not recommended.

2. Per device acceptance: The new device asserts its unique identity to a Registrar. A non-technical human validates the identity, for example by comparing the identity displayed by the registrar (for example using a smartphone app) with the identity shown on the packaging of the device. Acceptance may be triggered by a click on a smartphone app "accept this device", or by other forms of pairing. See also [I-D.behringer-homenet-trust-bootstrap] for how the approach could work in a homenet.

3. Whitelist acceptance: In larger networks, neither of the previous approaches is acceptable. Default acceptance is not secure, and a manual per device methods do not scale. Here, the registrar is provided a priori with a list of identifiers of devices that belong to the network. This list can be extracted from an inventory database, or sales records. If a device is detected

that is not on the list of known devices, it can still be manually accepted using the per device acceptance methods.

4. Automated Whitelist: an automated process that builds the necessary whitelists and inserts them into the larger network domain infrastructure is plausible. Once set up, no human intervention is required in this process. Defining the exact mechanisms for this is out of scope although the registrar authorization checks is identified as the logical integration point of any future work in this area.

None of these approaches require the network to have permanent Internet connectivity. Even when the Internet based MASA service is used, it is possible to pre-fetch the required information from the MASA a priori, for example at time of purchase such that devices can enroll later. This supports use cases where the domain network may be entirely isolated during device deployment.

Additional policy can be stored for future authorization decisions. For example an expected deployment time window or that a certain Proxy must be used.

D.2.4. Automatic Enrollment of Devices

The approach outlined in this document provides a secure zero-touch method to enroll new devices without any pre-staged configuration. New devices communicate with already enrolled devices of the domain, which proxy between the new device and a Registrar. As a result of this completely automatic operation, all devices obtain a domain based certificate.

D.2.5. Secure Network Operations

The certificate installed in the previous step can be used for all subsequent operations. For example, to determine the boundaries of the domain: If a neighbor has a certificate from the same trust anchor it can be assumed "inside" the same organization; if not, as outside. See also Appendix D.1.5.1. The certificate can also be used to securely establish a connection between devices and central control functions. Also autonomic transactions can use the domain certificates to authenticate and/or encrypt direct interactions between devices. The usage of the domain certificates is outside scope for this document.

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
Cisco

Email: mbehring@cisco.com

Steinthor Bjarnason
Cisco

Email: sbjarnas@cisco.com

Kent Watsen
Juniper Networks

Email: kwatsen@juniper.net