Workgroup: ANIMA WG
Internet-Draft: draft-ietf-anima-brski-prm-00
Published: 25 October 2021
Intended Status: Standards Track
Expires: 28 April 2022
Authors: S. Fries T. Werner E. Lear
Siemens Siemens Cisco Systems
M. Richardson
Sandelman Software Works

BRSKI with Pledge in Responder Mode (BRSKI-PRM)

Abstract

This document defines enhancements to the bootstrapping a remote secure key infrastructure (BRSKI, [RFC8995]) to facilitate bootstrapping in domains featuring no or only timely limited connectivity between a pledge and the domain registrar. This specifically targets situations, in which the interaction model changes from a pledge-initiator-mode as in BRSKI to a pledge-responder-mode as desribed here. To support this functionality BRSKI-PRM introduces a new registrar-agent component, which facilitates the communication between pledge and registrar during the bootstrapping phase. To support the establishment of a trust relation between a pledge and the domain registrar, BRSKI-PRM relies on the exchange of authenticated self-contained objects (signature-wrapped objects). The defined approach is agnostic regarding the utilized enrollment protocol, deployed by the registrar to communicate with the Domain CA.

Table of Contents

1. Introduction

BRSKI as defined in [RFC8995] specifies a solution for secure zero-touch (automated) bootstrapping of devices (pledges) in a (customer) site domain. This includes the discovery of network elements in the target domain, time synchronization, and the exchange of security information necessary to establish trust between a pledge and the domain. Security information about the target domain, specifically the target domain certificate, is exchanged utilizing voucher objects as defined in [RFC8366]. These vouchers are signed objects, which are provided via the domain registrar to the pledge and originate from a Manufacturer's Authorized Signing Authority (MASA).

BRSKI addresses scenarios in which the pledge acts as client for the bootstrapping and is the initiator of the bootstrapping. In industrial environments the pledge may behave as a server and thus does not initiate the bootstrapping with the domain registrar. In this scenarios it is expected that the pledge will be triggered to generate request objects to be bootstrapped in the registrar's domain. For this, an additional component is introduced acting as an agent for the domain registrar (registrar-agent) towards the pledge. This may be a functionality of a commissioning tool or it may be even co-located with the registrar. In contrast to BRSKI the registrar-agent performs the object exchange with the pledge and provides/retrieves data objects to/from the domain registrar. For the interaction with the domain registrar the registrar agent will use existing BRSKI endpoints.

The goal is to enhance BRSKI to be usable also for a pledge in responder mode. This is addressed by

* introducing the registrar-agent as new component to facilitate the communication between the pledge and the registrar, when the pledge is in responder mode (acting as server).

* handling the security on application layer only to enable application of arbitrary transport means between the pledge and the domain registrar, by keeping the registrar-agent in the communication path. Examples may be connectivity via IP based networks (wired or wireless) but also connectivity via Bluetooth or NFC between the pledge and the registrar-agent.

* allowing to utilize credentials different from the pledge's IDevID to establish a TLS connection to the domain registrar, which is necessary in case of using a registrar-agent.

* defining the interaction (data exchange and data objects) between a pledge acting as server an a registrar-agent and the domain registrar.

For the enrollment of devices BRSKI relies on EST [RFC7030] to request and distribute target domain specific device certificates. EST in turn relies on a binding of the certification request to an underlying TLS connection between the EST client and the EST server. According to BRSKI the domain registrar acts as EST server and is also acting as registration authority (RA).

To be done: * include reasoning for not using TLS (IDevID does not contain SAN, TLS server flag) between the pledge and the registrar-agent. * Enhancements to EST state machine necessary to process self-contained objects on the registrar-agent and domain-registrar * accepting

2. Terminology

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

This document relies on the terminology defined in [RFC8995]. The following terms are defined additionally:

CA: Certification authority, issues certificates.

RA: Registration authority, an optional system component to which a CA delegates certificate management functions such as authorization checks.

IED: Intelligent Electronic Device (in essence a pledge).

on-site: Describes a component or service or functionality available in the target deployment domain.

off-site: Describes a component or service or functionality available in an operator domain different from the target deployment domain. This may be a central site or a cloud service, to which only a temporary connection is available, or which is in a different administrative domain.

asynchronous communication: Describes a timely interrupted communication between an end entity and a PKI component.

synchronous communication: Describes a timely uninterrupted communication between an end entity and a PKI component.

authenticated self-contained object: Describes an object, which is cryptographically bound to the EE certificate (IDevID certificate or LDEVID certificate) of a pledge. The binding is assumed to be

provided through a digital signature of the actual object using the corresponding private key of the EE certificate.

3. Scope of Solution

3.1. Supported Environment

The solution is intended to be applicable in domains in which pledges have no direct connection to the domain registrar, but are expected to be managed by the registrar. This can be motivated by pledges featuring a different technology stack or by pledges without an existing connection to the domain registrar during bootstrapping. These pledges are likely to act in a server role. Therefore, the pledge has to offer endpoints on which it can be triggered for the generation of voucher-request objects and certification objects as well as to provide the response objects to the pledge.

3.2. Application Examples

The following examples are intended to motivate the support of different enrollment approaches in general and asynchronous enrollment specifically, by introducing industrial applications cases, which could leverage BRSKI as such but also require support of in situation, in which the pledge acts as a server and only answers specific requests.

3.2.1. Building Automation

In building automation, a use case can be described by a detached building or the basement of a building equipped with sensor, actuators, and controllers connected, but with only limited or no connection to the centralized building management system. This limited connectivity may be during the installation time but also during operation time. During the installation in the basement, a service technician collects the necessary information from the basement network and provides them to the central building management system, e.g., using a laptop or even a mobile phone to transport the information. This information may comprise parameters and settings required in the operational phase of the sensors/ actuators, like a certificate issued by the operator to authenticate against other components and services.

The collected information may be provided by a domain registrar already existing in the installation network. In this case connectivity to the backend PKI may be facilitated by the service technician's laptop.

Contrary, the information can also be collected from the pledges directly and provided to a domain registrar deployed in a different

network. In this cases connectivity to the domain registrar may be facilitated by the service technician's laptop.

3.2.2. Infrastructure Isolation Policy

This refers to any case in which network infrastructure is normally isolated from the Internet as a matter of policy, most likely for security reasons. In such a case, limited access to a domain registrar may be allowed in carefully controlled short periods of time, for example when a batch of new devices are deployed, but impossible at other times.

3.2.3. Less Operational Security in the Target-Domain

The registration point performing the authorization of a certificate request is a critical PKI component and therefore implicates higher operational security than other components utilizing the issued certificates for their security features. CAs may also demand higher security in the registration procedures. Especially the CA/Browser forum currently increases the security requirements in the certificate issuance procedures for publicly trusted certificates. There may be the situation that the target domain does not offer enough security to operate a registration point and therefore wants to transfer this service to a backend that offers a higher level of operational security.

4. Requirements Discussion and Mapping to Solution-Elements

Based on the intended target environment described in Section 3.1 and the motivated application examples described in Section 3.2 the following base requirements are derived to support authenticated self-contained objects as container carrying the request and response messages to support the communication over a registrar-agent.

At least the following properties are required:

* Proof of Possession: proves to possess and control the private key corresponding to the public key contained in the certification request, typically by adding a signature using the private key.

* Proof of Identity: provides data-origin authentication of a data object, e.g., a certificate request, utilizing an existing IDevID. Certificate updates may utilize the certificate that is to be updated.

Solution examples (not complete) based on existing technology are provided with the focus on existing IETF documents:

* Certification request objects: Certification requests are structures protecting only the integrity of the contained data providing a proof-of-private-key-possession for locally generated key pairs. Examples in scope for certification requests are:

  - PKCS#10 [RFC2986]: Defines a structure for a certification request. The structure is signed to ensure integrity protection and proof of possession of the private key of the requester that corresponds to the contained public key.

Note that the integrity of the certification request is bound to the public key contained in the certification request by performing the signature operation with the corresponding private key. In the considered application examples, this is not sufficient to provide data origin authentication and needs to be bound to the existing credential of the pledge (IDevID) additionally. This binding supports the authorization decision for the certification request through the provisioning of a proof of identity. The binding of data origin authentication to the certification request may be delegated to the protocol used for certificate management.

5. Architectural Overview and Communication Exchanges

To BRSKI wirh pledge in responder mode, the base system architecture defined in BRSKI [RFC8995] is enhanced to facilitate the use case. The pledge-responder-mode) allows delegated bootstrapping using a registrar-agent instead a direct connection from the pledge to the domain registrar. The communication model between registrar-agent and pledge assumes that the pledge is acting as server and responds to requests.

Necessary enhancements to support authenticated self-contained objects for certificate enrollment are kept on a minimum to ensure reuse of already defined architecture elements and interactions.

For the authenticated self-contained objects used for the certification request, BRSKI-PRM relies on the defined message wrapping mechanisms of the enrollment protocols stated in Section 4 above.

5.1. Pledge-responder-mode (PRM): Registrar-agent Communication with Pledges

To support mutual trust establishment of pledges, not directly connected to the domain registrar this approach relies on the exchange of authenticated self-contained objects (the voucher

request/response objects as known from BRSKI and the enrollment request/response objects as introduced by BRSKI-PRM). This allows independence of a potential protection provided by the used transport protocol.

In contrast to BRSKI, the object exchanges performed with the help of a registrar-agent component, supporting the interaction of the pledge with the domain registrar. It may be an integrated functionality of a commissioning tool. This leads to enhancements of the logical elements in the BRSKI architecture as shown in Figure 1. The registrar-agent interacts with the pledge to acquire and to supply the required data objects for bootstrapping, which are also exchanged between the registrar-agent and the domain registrar. Moreover, the addition of the registrar-agent also influences the sequences for the data exchange between the pledge and the domain registrar described in [RFC8995]. The general goal for the registrar-agent application is the reuse of already defined endpoints of the domain registrar side. The functionality of the already existing registrar endpoints may need small enhancements.

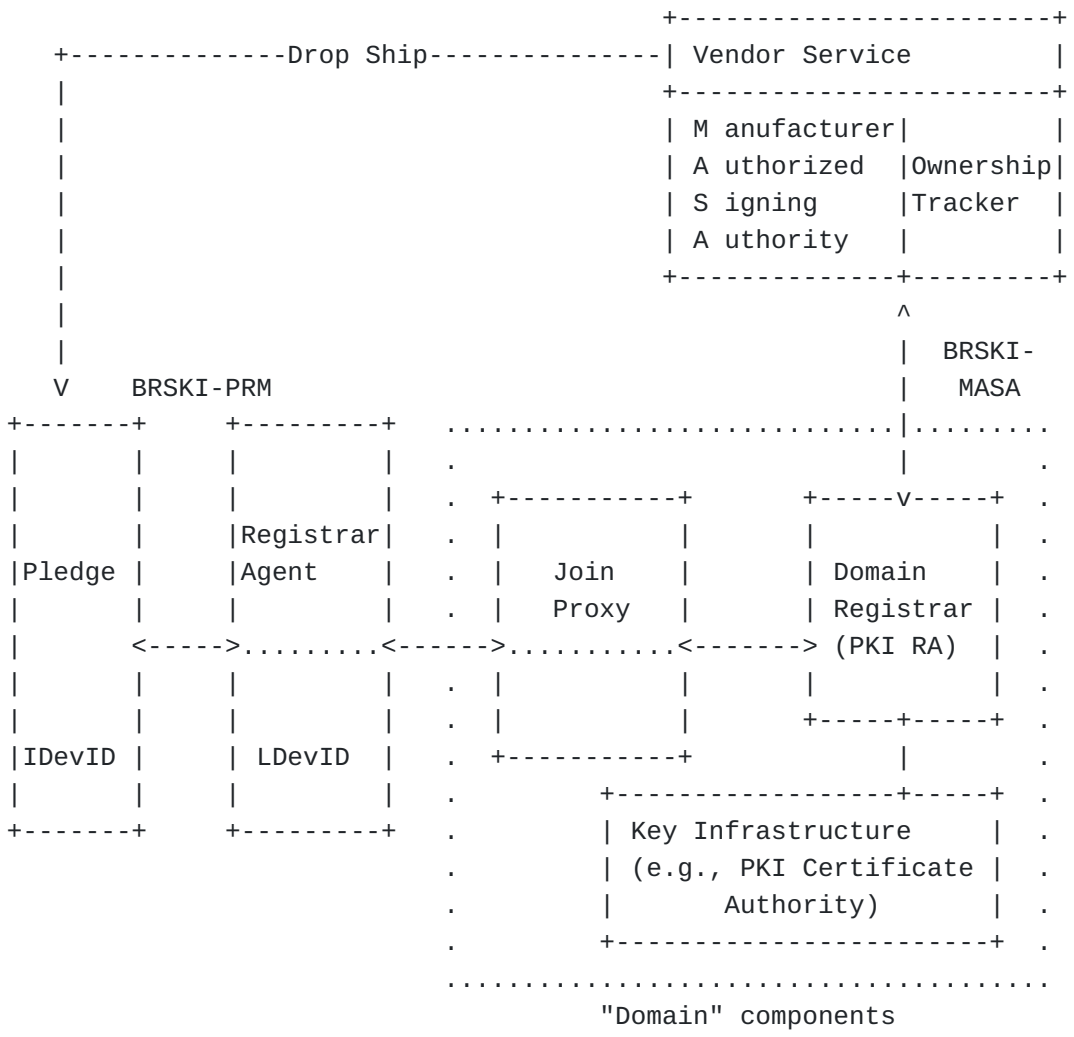


Figure 1: Architecture overview using registrar-agent

The architecture overview in Figure 1 utilizes the same logical components as BRSKI with the registrar-agent component in addition.

For authentication towards the domain registrar, the registrar-agent uses its LDevID. The provisioning of the registrar-agent LDevID may be done by a separate BRSKI run or other means in advance. It is recommended to use short lived registrar-agent LDevIDs in the range of days or weeks.

If a registrar detects a request originates from a registrar-agent it is able to switch the operational mode from BRSKI to BRSKI-PRM.

In addition, the domain registrar may authenticate the user operating the registrar-agent to perform additional authorization of a pledge enrollment action. Examples for such user level authentication are the application of HTTP authentication or the usage of authorization tokens or other. This is out of scope of this document.

The following list describes the components in a (customer) site domain:

* Pledge: The pledge is expected to respond with the necessary data objects for bootstrapping to the registrar-agent. The transport protocol used between the pledge and the registrar-agent is assumed to be HTTP in the context of this document. Other transport protocols may be used but are out of scope of this document. As the pledge is acting as a server during bootstrapping it leads to some differences to BRSKI:

  - Discovery of the domain registrar by the pledge is not needed as the pledge will be triggered by the registrar-agent.

  - Discovery of the pledge by the registrar-agent must be possible.

  - As the registrar-agent must be able to request data objects for bootstrapping of the pledge, the pledge must offer corresponding endpoints.

  - The registrar-agent may provide additional data to the pledge, in the context of the triggering request.

  - Order of exchanges in the call flow may be different as the registrar-agent collects both objects, pledge-voucher-request objects and pledge-enrollment-request objects, at once and provides them to the registrar. This approach may also be used to perform a bulk bootstrapping of several devices.

-The data objects utilized for the data exchange between the pledge and the registrar are self-contained authenticated objects (signature-wrapped objects).

*Registrar-agent: provides a communication path to exchange data objects between the pledge and the domain registrar. The registrar-agent facilitates situations, in which the domain registrar is not directly reachable by the pledge, either due to a different technology stack or due to missing connectivity. The registrar-agent triggers the pledge to create bootstrapping information such as voucher request objects and enrollment request objects from one or multiple pledges at once and performs a bulk bootstrapping based on the collected data. The registrar-agent is expected to possess information of the domain registrar, either by configuration or by using the discovery mechanism defined in [RFC8995]. There is no trust assumption between the pledge and the registrar-agent as only authenticated self-contained objects are applied, which are transported via the registrar-agent and provided either by the pledge or the registrar. The trust assumption between the registrar-agent and the registrar bases on an own LDevID of the registrar-agent, acting as registrar component. This allows the registrar-agent to authenticate towards the registrar. The registrar can utilize this authentication to distinguish communication with a pledge from a registrar-agent based on the exchanged objects.

*Join Proxy: same functionality as described in [RFC8995]. Note that it may be used by the registrar-agent instead of the pledge to find the registrar, if not configured.

*Domain Registrar: In general the domain registrar fulfills the same functionality regarding the bootstrapping of the pledge in a (customer) site domain by facilitating the communication of the pledge with the MASA service and the domain PKI service. In contrast to [RFC8995], the domain registrar does not interact with a pledge directly but through the registrar-agent. The registrar detects if the bootstrapping is performed by the pledge directly or by the registrar-agent. The manufacturer provided components/services (MASA and Ownership tracker) are used as defined in [RFC8995]. For issuing a voucher, the MASA may perform additional checks on voucher-request objects, to issue a voucher indicating agent-proximity instead of registrar-proximity.

[RFC Editor: please delete] /*

Open Issues: The voucher defined in [RFC8366] defines the leaf assertion as enum, which cannot be extended. To define an additional assertion RFC 8366 may be revised. There is currently ongoing work for a RFC8366bis. */

"Agent-proximity" is a weaker assertion then "proximity". In case of "agent-proximity" it is a statement, that the proximity-registrar-certificate was provided via the registrar-agent and not directly. This can be verified by the registrar and also by the MASA through voucher-request processing. Note that at the time of creating the voucher-request, the pledge cannot verify the LDevID(Reg) EE certificate and has no proof-of-possession of the corresponding private key for the certificate. Trust handover to the domain is established via the "pinned-domain-certificate" in the voucher.

In contrast, "proximity" provides a statement, that the pledge was in direct contact with the registrar and was able to verify proof-of-possession of the private key in the context of the TLS handshake. The provisionally accepted LDevID(Reg) EE certificate can be verified after the voucher has been processed by the pledge.

5.1.1. Behavior of Pledge in Pledge-Responder-Mode

In contrast to BRSKI the pledge acts as a server component. It is triggered by the registrar-agent for the generation of pledge-voucher-request and pledge-enrollment-request objects as well as for the processing of the response objects and the generation of status information. Due to the use of the registrar-agent, the interaction with the domain registrar is changed as shown in Figure 3. To enable interaction with the registrar-agent, the pledge provides endpoints using the BRSKI interface based on the "/.well-known/brski" URI tree. The following endpoints are defined for the pledge in this document:

* /.well-known/brski/pledge-voucher-request: trigger pledge to create voucher request. It returns the pledge-voucher-request.

* /.well-known/brski/pledge-enrollment-request: trigger pledge to create enrollment request. it returns the pledge-enrollment-request.

* /.well-known/brski/pledge-voucher: supply MASA provided voucher to pledge. It returns the pledge-voucher-status.

* /.well-known/brski/pledge-enrollment: supply enroll response (certificate) to pledge. It returns the pledge-enrollment-status.

* /.well-known/brski/pledge-CACerts: supply CACerts to pledge (optional).

5.1.2. Behavior of Registrar-Agent

The registrar-agent is a new component in the BRSKI context. It provides connectivity between the pledge and the domain registrar and reuses the endpoints of the domain registrar side already

specified in [RFC8995]. It facilitates the exchange of data objects between the pledge and the domain registrar, which are the voucher request/response objects, the enrollment request/response objects, as well as related status objects. For the communication the registrar-agent utilizes communication endpoints provided by the pledge. The transport in this specification is based on HTTP but may also be done using other transport mechanisms. This new component changes the general interaction between the pledge and the domain registrar as shown in Figure 7.

The registrar-agent is expected to already possess an LDevID(RegAgt) to authenticate towards the domain registrar. The registrar-agent will use this LDevID(RegAgt) when establishing the TLS session with the domain registrar in the context of for TLS client-side authentication. The LDevID(RegAgt) certificate MUST include a SubjectKeyIdentifier (SKID), which is used as reference in the context of an agent-signed-data object. Note that this is an additional requirement for issuing the certificate, as [IEEE-802.1AR] only requires the SKID to be included for intermediate CA certificates. In the specific application of BRSKI-PRM, it is used in favor of a certificate fingerprint to avoid additional computations.

Using an LDevID for TLS client-side authentication is a deviation from [RFC8995], in which the pledge's IDevID credential is used to perform TLS client authentication. The use of the LDevID(RegAgt) allows the domain registrar to distinguish, if bootstrapping is initiated from a pledge or from a registrar-agent and adopt the internal handling accordingly. As BRSKI-PRM uses authenticated self-contained data objects between the pledge and the domain registrar, the binding of the pledge identity to the request object is provided by the data object signature employing the pledge's IDevID. The objects exchanged between the pledge and the domain registrar used in the context of this specifications are JOSE objects

In addition to the LDevID(RegAgt), the registrar-agent is provided with the product-serial-numbers of the pledges to be bootstrapped. This is necessary to allow the discovery of pledges by the registrar-agent using mDNS. The list may be provided by administrative means or the registrar agent may get the information via an interaction with the pledge, like scanning of product-serial-number information using a QR code or similar.

According to [RFC8995] section 5.3, the domain registrar performs the pledge authorization for bootstrapping within his domain based on the pledge voucher-request object.

The following information is therefore available at the registrar-agent:

*LDevID(RegAgt): own operational key pair.

*LDevID(reg) certificate: certificate of the domain registrar.

*Serial-number(s): product-serial-number(s) of pledge(s) to be bootstrapped.

5.1.2.1. Discovery of Registrar by Registrar-Agent

The discovery of the domain registrar may be done as specified in [RFC8995] with the deviation that it is done between the registrar-agent and the domain registrar. Alternatively, the registrar-agent may be configured with the address of the domain registrar and the certificate of the domain registrar.

5.1.2.2. Discovery of Pledge by Registrar-Agent

The discovery of the pledge by registrar-agent should be done by using DNS-based Service Discovery [RFC6763] over Multicast DNS [RFC6762] to discover the pledge at "product-serial-number.brski-pledge._tcp.local." The pledge constructs a local host name based on device local information (product-serial-number), which results in "product-serial-number.brski-pledge._tcp.local.". It can then be discovered by the registrar-agent via mDNS. Note that other mechanisms for discovery may be used.

The registrar-agent is able to build the same information based on the provided list of product-serial-number.

5.1.3. Bootstrapping Objects and Corresponding Exchanges

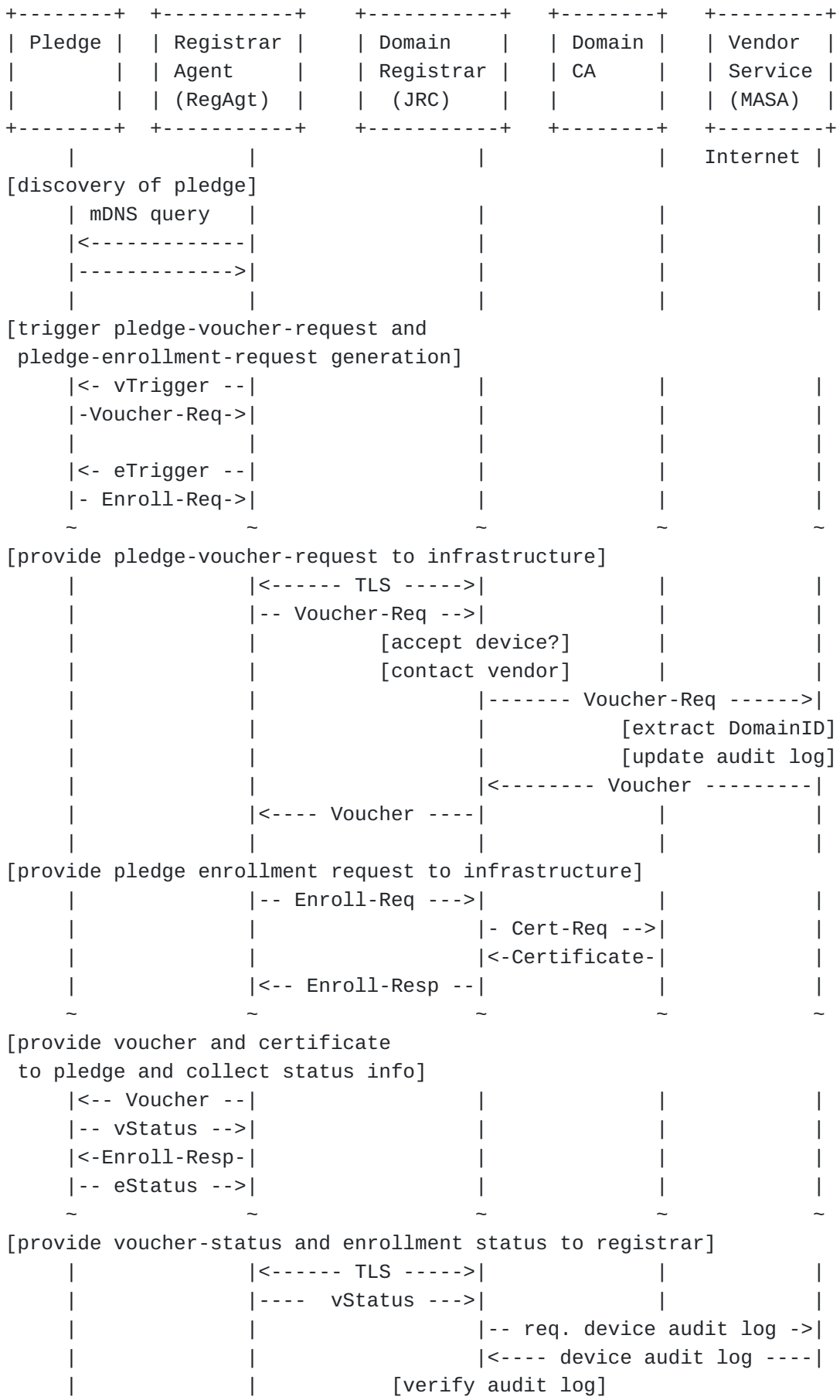
The interaction of the pledge with the registrar-agent may be accomplished using different transport means (protocols and or network technologies). For this document the usage of HTTP is targeted as in BRSKI. Alternatives may be CoAP, Bluetooth Low Energy (BLE), or Nearfield Communication (NFC). This requires independence of the exchanged data objects between the pledge and the registrar from transport security. Therefore, authenticated self-contained objects (here: signature-wrapped objects) are applied in the data exchange between the pledge and the registrar.

The registrar-agent provides the domain-registrar certificate (LDevID(Reg) EE certificate) to the pledge to be included into the "agent-provided-proximity-registrar-certificate" leaf in the pledge-voucher-request object. This enables the registrar to verify, that it is the target registrar for handling the request. The registrar certificate may be configured at the registrar-agent or may be

fetched by the registrar-agent based on a prior TLS connection establishment with the domain registrar. In addition, the registrar-agent provides agent-signed-data containing the product-serial-number in the body, signed with the LDevID(RegAgt). This enables the registrar to verify and log, which registrar-agent was in contact with the pledge. Optionally the registrar-agent may provide its LDevID(RegAgt) certificate to the pledge for inclusion into the pledge-voucher-request as "agent-sign-cert" leaf. Note that this may be omitted in constraint environments to safe bandwidth between the registrar-agent and the pledge. If not contained, the registrar-agent MUST fetch the LDevID(RegAgt) certificate based on the SubjectKeyIdentifier (SKID) in the header of the agent-signed-data. The registrar may include the LDevID(RegAgt) certificate information into the registrar-voucher-request.

The MASA in turn verifies the LDevID(Reg) certificate is included in the pledge-voucher-request (prior-signed-voucher-request) in the "agent-provided-proximity-registrar-certificate" leaf and may assert in the voucher "verified" or "logged" instead of "proximity", as there is no direct connection between the pledge and the registrar. If the LDevID(RegAgt) certificate is included contained in the "agent-sign-cert" leave of the registrar-voucher-request, the MASA can verify the LDevID(RegAgt) certificate and the signature of the registrar-agent in the agent-signed-data provided in the prior-signed-voucher-request. If both can be verified successfully, the MASA can assert "agent-proximity" in the voucher. Otherwise, it may assert "verified" or "logged". The voucher can then be supplied via the registrar to the registrar-agent.

Figure 2 provides an overview of the exchanges detailed in the following sub sections.


Pledge
Registrar Agent (RegAgt)
Domain Registrar (JRC)
Domain CA
Vendor Service (MASA)
Internet
[discovery of pledge]
mDNS query
[trigger pledge-voucher-request and pledge-enrollment-request generation]
vTrigger
Voucher-Req
eTrigger
Enroll-Req
[provide pledge-voucher-request to infrastructure]
TLS
Voucher-Req
[accept device?]
[contact vendor]
Voucher-Req
[extract DomainID]
[update audit log]
Voucher
Voucher
[provide pledge enrollment request to infrastructure]
Enroll-Req
Cert-Req
Certificate
Enroll-Resp
[provide voucher and certificate to pledge and collect status info]
Voucher
vStatus
Enroll-Resp
eStatus
[provide voucher-status and enrollment status to registrar]
TLS
vStatus
req. device audit log
device audit log
[verify audit log]

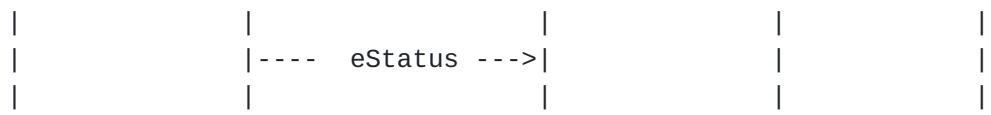

eStatus --->

Figure 2: Overview pledge-responder-mode exchanges

The following sub sections split the interactions between the different components into:

- *Request objects acquisition targets exchanges and objects between the registrar-agent and the pledge.

- *Request handling targets exchanges and objects between the registrar-agent and the registrar and also the interaction of the registrar with the MASA and the domain CA.

- *Response object supply targets the exchanges and objects between the registrar-agent and the pledge including the status objects.

- *Status handling addresses the exchanges between the registrar-agent and the registrar.

5.1.3.1. Request Objects Acquisition by Registrar-Agent from Pledge

The following description assumes that the registrar-agent already discovered the pledge. This may be done as described in Section 5.1.2.2 based on mDNS.

The focus is on the exchange of signature-wrapped objects using endpoints defined for the pledge in Section 5.1.1.

Preconditions:

- *Pledge: possesses IDevID

- *Registrar-agent: possesses IDevID CA certificate and an own LDevID(RegAgt) EE credential for the registrar domain. In addition, the registrar-agent can be configured with the product-serial-number(s) of the pledge(s) to be bootstrapped. Note that the product-serial-number may have been used during the pledge discovery already.

- *Registrar: possesses IDevID CA certificate and an own LDevID/Reg) credential.

- *MASA: possesses own credentials (voucher signing key, TLS server certificate) as well as IDevID CA certificate of pledge vendor / manufacturer and site-specific LDevID CA certificate.

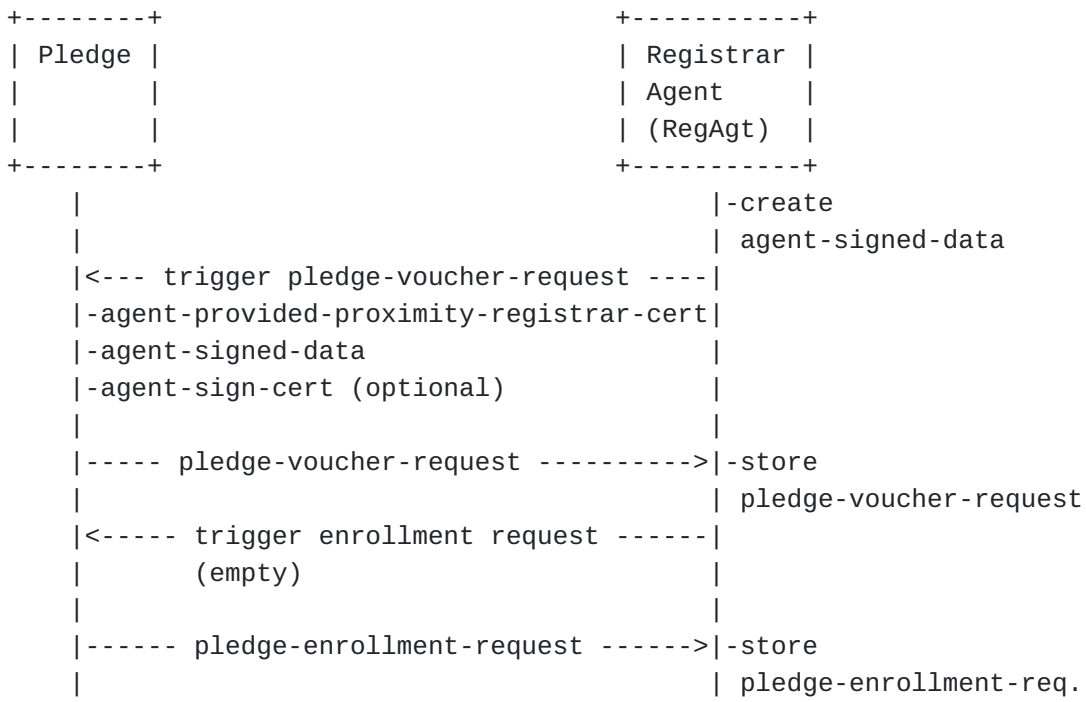


Figure 3: Request collection (registrar-agent - pledge)

Triggering the pledge to create the pledge-voucher-request is done using HTTP POST on the defined pledge endpoint "/.well-known/brski/pledge-voucher-request".

The registrar-agent pledge-voucher-request Content-Type header is:

application/json: defines a JSON document to provide three parameter:

- *agent-provided-proximity-registrar-cert: base64-encoded LDevID(Reg) TLS EE certificate.

- *agent-sign-cert: base64-encoded LDevID(RegAgt) signing certificate (optional).

- *agent-signed-data: base64-encoded JWS-object.

Note that optionally including the agent-sign-cert enables the pledge to verify at least the signature of the agent-signed-data. It may not verify the agent-sign-cert itself due to missing issuing CA information.

The agent-signed-data is a JOSE object and contains the following information:

The header of the agent-signed-data contains:

- *alg: algorithm used for creating the object signature.

*kid: contains the base64-encoded SubjectKeyIdentifier of the LDevID(RegAgt) certificate.

The body of the agent-signed-data contains an ietf-voucher-request-prm:agent-signed-data element (defined in Section 6):

*created-on: MUST contain the creation date and time in yang:date-and-time format.

*serial-number: MUST contain the product-serial-number as type string as defined in [RFC8995], section 2.3.1. The serial-number corresponds with the product-serial-number contained in the X520SerialNumber field of the IDevID certificate of the pledge.

```
{
    "alg": "ES256",
    "kid": "base64encodedvalue=="
}
{
  "ietf-voucher-request-prm:agent-signed-data": {
    "created-on": "2021-04-16T00:00:01.000Z",
    "serial-number": "callee4711"
  }
}
{
    SIGNATURE
}
```

Figure 4: Example of agent-signed-data

Upon receiving the voucher-request trigger, the pledge SHOULD construct the body of the pledge-voucher-request object as defined in [RFC8995]. This object becomes a JSON-in-JWS object as defined in [I-D.ietf-anima-jws-voucher].

The header of the pledge-voucher-request SHALL contain the following parameter as defined in [RFC7515]:

*alg: algorithm used for creating the object signature.

*x5c: contains the base64-encoded pledge IDevID certificate.

The body of the pledge-voucher-request object MUST contain the following parameter as part of the ietf-voucher-request-prm:voucher as defined in [RFC8995]:

*created-on: contains the current date and time in yang:date-and-time format.

*nonce: contains a cryptographically strong random or pseudo-random number.

*serial-number: contains the base64-encoded pledge product-serial-number.

*assertion: contains the requested voucher assertion.

The ietf-voucher-request-prm:voucher is enhanced with additional parameters:

*agent-provided-proximity-registrar-cert: MUST be included and contains the base64-encoded LDevID(Reg) EE certificate (provided as trigger parameter by the registrar-agent).

*agent-signed-data: MUST contain the base64-encoded agent-signed-data (as defined in Figure 4) and provided as trigger parameter.

*agent-sign-cert: May contain the base64-encoded LDevID(RegAgt) EE certificate if provided as trigger parameter.

The enhancements of the YANG module for the ietf-voucher-request with these new leafs are defined in Section 6.

The object is signed using the pledges IDevID credential contained as x5c parameter of the JOSE header.

```
{
   "alg": "ES256",
   "x5c": ["MIIB2jCC...dA=="]
}
{
  "ietf-voucher-request-prm:voucher": {
   "created-on": "2021-04-16T00:00:02.000Z",
   "nonce": "eDs++/FuDHGUnRxN3E14CQ==",
   "serial-number": "callee4711",
   "assertion": "agent-proximity",
   "agent-provided-proximity-registrar-cert": "base64encodedvalue==",
   "agent-signed-data": "base64encodedvalue==",
   "agent-sign-cert": "base64encodedvalue=="
  }
}
{
    SIGNATURE
}
```

Figure 5: Example of pledge-voucher-request

The pledge-voucher-request Content-Type is defined in [I-D.ietf-anima-jws-voucher] as:

application/voucher-jws+json

The pledge SHOULD include this Content-Type header field indicating the included media type for the voucher response. Note that this is also an indication regarding the acceptable fromat of the voucher response. This format is included by the registrar as described in Section 5.1.3.2.

Once the registrar-agent has received the pledge-voucher-request it can trigger the pledge to generate an enrollment-request object. As in BRSKI the enrollment request object is a PKCS#10, additionally signed by the IDevID. Note, as the initial enrollment aims to request a general certificate, no certificate attributes are provided to the pledge.

Triggering the pledge to create the enrollment-request is done using HTTP POST on the defined pledge endpoint "/.well-known/brski/pledge-enrollment-request".

The registrar-agent pledge-enrollment-request Content-Type header is:

application/json:

with an empty body.

Note that using HTTP POST allows for an empty body, but also to provide additional data, like CSR attributes or information about the enroll type: initial or re-enroll. In the following the enrollment is described as initial enrollment.

Upon receiving the enrollment-trigger, the pledge SHALL construct the pledge-enrollment-request as authenticated self-contained object. The CSR already assures proof of possession of the private key corresponding to the contained public key. In addition, based on the additional signature using the IDevID, proof of identity is provided. Here, a JOSE object is being created in which the body utilizes the YANG module ietf-ztp-types with the grouping for csr-grouping for the CSR as defined in [I-D.ietf-netconf-sztp-csr].

[RFC Editor: please delete] /* Open Issues: Reuse of the sub-tree ietf-sztp-csr:csr may not be possible as it is not the complete module. */

Depending on the capability of the pledge, it constructs the enrollment request as plain PKCS#10. Note that the focus in this use case is placed on PKCS#10 as PKCS#10 can be transmitted in different

enrollment protocols like EST, CMP, CMS, and SCEP. If the pledge is already implementing an enrollment protocol, it may leverage that functionality for the creation of the enrollment request object. Note also that [I-D.ietf-netconf-sztp-csr] also allows for inclusion of certification request objects such as CMP or CMC.

The pledge SHOULD construct the pledge-enrollment-request as PKCS#10 object. In this case it MUST sign it additionally with its IDevID credential to achieve proof-of-identity bound to the PKCS#10 as described below.

A successful enrollment will result in a generic LDevID certificate for the pledge in the new domain, which can be used to request further LDevID certificates if necessary for its operation.

[RFC Editor: please delete] /* Open Issues: Depending on target environment, it may be useful to assume that the pledge may already "know" its functional scope and therefore the number of certificates needed during operation. As a result, multiple CSRs may be generated to provide achieve multiple certificates as a result of the enrollment. This would need further description and potential enhancements also in the enrollment-request object to transport different CSRs. */

[I-D.ietf-netconf-sztp-csr] considers PKCS#10 but also CMP and CMC as certification request format. Note that the wrapping signature is only necessary for plain PKCS#10 as other request formats like CMP and CMS support the signature wrapping as part of their own certificate request format.

The registrar-agent enrollment-request Content-Type header for a wrapped PKCS#10 is:

application/jose:

The header of the pledge enrollment-request SHALL contain the following parameter as defined in [RFC7515]:

*alg: algorithm used for creating the object signature.

*x5c: contains the base64-encoded pledge IDevID certificate.

The body of the pledge enrollment-request object SHOULD contain a P10 parameter (for PKCS#10) as defined for ietf-ztp-types:p10-csr in [I-D.ietf-netconf-sztp-csr]:

*P10: contains the base64-encoded PKCS#10 of the pledge.

The JOSE object is signed using the pledge's IDevID credential, which corresponds to the certificate signaled in the JOSE header.

```
{
    "alg": "ES256",
    "x5c": ["MIIB2jCC...dA=="]
}
{
  "ietf-ztp-types": {
    "p10-csr": "base64encodedvalue=="
  }
}
{
    SIGNATURE
}
```

Figure 6: Example of pledge-enrollment-request

With the collected pledge-voucher-request object and the pledge-enrollment-request object, the registrar-agent starts the interaction with the domain registrar.

[RFC Editor: please delete] /* Open Issues: further description necessary at least for

- *Values to be taken from the IDevID into the PKCS#10 (like product-serial-number or subjectName, or certificate template) */

Once the registrar-agent has collected the pledge-voucher-request and pledge-enrollment-request objects, it connects to the registrar and sends the request objects. As the registrar-agent is intended to work between the pledge and the domain registrar, a collection of requests from more than one pledges is possible, allowing a bulk bootstrapping of multiple pledges using the same connection between the registrar-agent and the domain registrar.

5.1.3.2. Request Handling - Registrar-Agent (Infrastructure)

The bootstrapping exchange between the registrar-agent and the domain registrar resembles the exchanges between the pledge and the domain registrar from BRSKI in the pledge-initiator-mode with some deviations.

Preconditions:

- *Registrar-agent: possesses IDevID CA certificate and own LDevID(RegAgt) EE credential of registrar domain. It knows the address of the domain registrar through configuration or discovery by, e.g., mDNS/DNSSD. The registrar-agent has acquired pledge-voucher-request and pledge-enrollment-request objects(s).

*Registrar: possesses IDevID CA certificate of pledge vendors / manufacturers and an own LDevID(Reg) EE credential.

*MASA: possesses own credentials (voucher signing key, TLS server certificate) as well as IDevID CA certificate of pledge vendor / manufacturer and site-specific LDevID CA certificate.

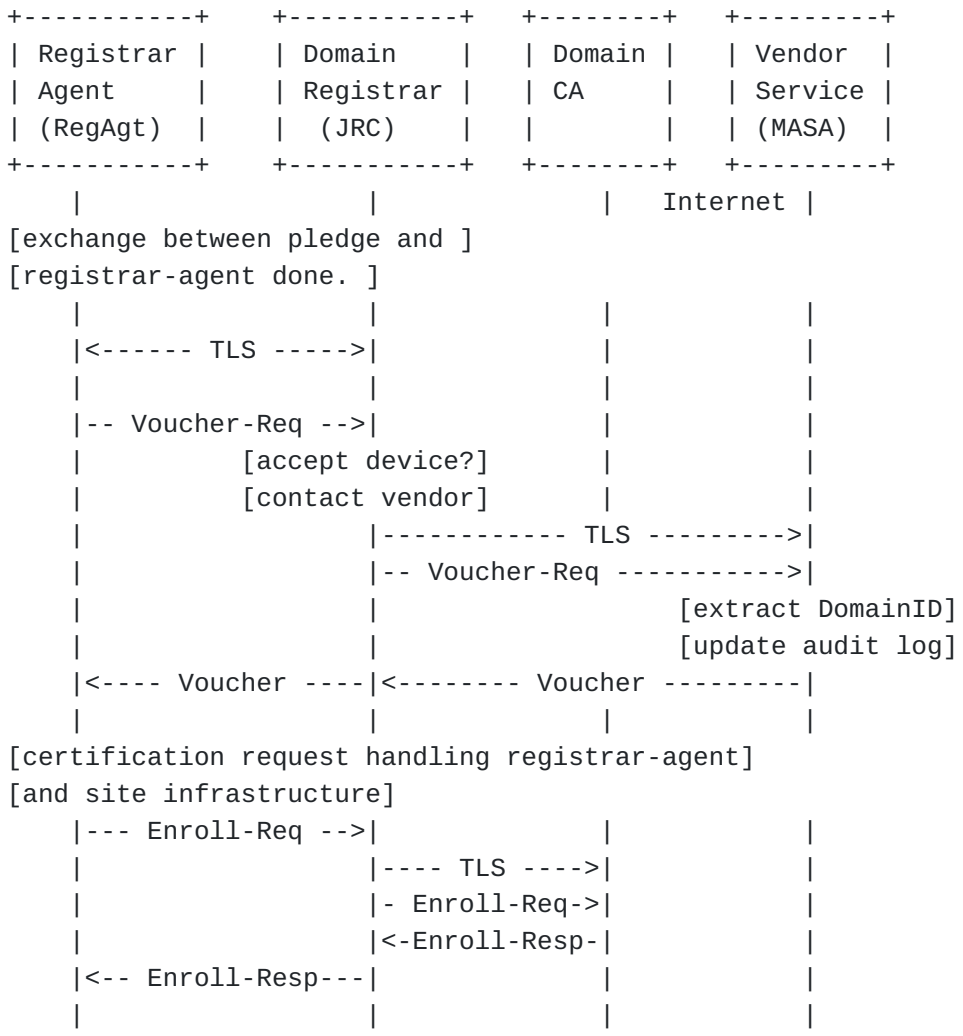


Figure 7: Request processing between registrar-agent and infrastructure bootstrapping services

The registrar-agent establishes a TLS connection with the registrar. As already stated in [RFC8995], the use of TLS 1.3 (or newer) is encouraged. TLS 1.2 or newer is REQUIRED on the registrar-agent side. TLS 1.3 (or newer) SHOULD be available on the registrar, but TLS 1.2 MAY be used. TLS 1.3 (or newer) SHOULD be available on the MASA, but TLS 1.2 MAY be used.

In contrast to [RFC8995] client authentication is achieved by using the LDevID(RegAgt) of the registrar-agent instead of the IDevID of

the pledge. This allows the registrar to distinguish between pledge-initiator-mode and pledge-responder-mode. In pledge-responder-mode the registrar has no direct connection to the pledge but via the registrar-agent. The registrar can receive request objects in different forms as defined in [RFC8995]. Specifically, the registrar will receive JOSE objects from the pledge for voucher-request and enrollment-request (instead of the objects for voucher-request (CMS-signed JSON) and enrollment-request (PKCS#10).

The registrar-agent sends the pledge-voucher-request to the registrar with an HTTP-over-TLS POST to the endpoint "/.well-known/brski/requestvoucher".

The pledge-voucher-request Content-Type used in the pledge-responder-mode is defined in [I-D.ietf-anima-jws-voucher] as:

application/voucher-jws+json (see Figure 5 for the content definition).

The registrar-agent SHOULD include the "Accept" header field indicating the pledge acceptable Content-Type for the voucher-response. The voucher-response Content-Type "application/voucher-jws+json" is defined in [I-D.ietf-anima-jws-voucher].

Upon reception of the pledge-voucher-request, the registrar SHALL perform the verification of the voucher-request parameter as defined in section 5.3 of [RFC8995]. In addition, the registrar shall verify the following parameters from the pledge-voucher-request:

- agent-provided-proximity-registrar-cert: MUST contain the own LDevID(Reg) EE certificate to ensure the registrar in proximity is the target registrar for the request.

- agent-signed-data: The registrar MUST verify that the data has been signed with the LDevID(RegAgt) credential indicated in the "kid" JOSE header parameter. If the certificate is not contained in the agent-sign-cert component of the pledge-voucher-request, it must fetch the certificate from a repository.

- agent-sign-cert: May contain the base64-encoded LDevID(RegAgt) certificate. If contained the registrar MUST verify that the connected credential used to sign the data was valid at signature creation time and that the corresponding registrar-agent was authorized to be involved in the bootstrapping.

If validation fails the registrar SHOULD respond with the HTTP 404 error code to the registrar-agent. If the pledge-voucher-request is in an unknown format, then an HTTP 406 error code is more appropriate.

If validation succeeds, the registrar will accept the pledge request to join the domain as defined in section 5.3 of [RFC8995]. The registrar then establishes a TLS connection with the MASA as described in section 5.4 of [RFC8995] to obtain a voucher for the pledge.

The registrar SHALL construct the body of the registrar-voucher-request object as defined in [RFC8995]. The encoding SHALL be done as JOSE object as defined in [I-D.ietf-anima-jws-voucher].

The header of the registrar-voucher-request SHALL contain the following parameter as defined in [RFC7515]:

*alg: algorithm used for creating the object signature.

*x5c: contains the base64-encoded registrar LDevID certificate.

The body of the registrar-voucher-request object MUST contain the following parameter as part of the voucher as defined in [RFC8995]:

*created-on: contains the current date and time in yang:date-and-time format for the registrar-voucher-request creation time.

*nonce: copied form the pledge-voucher-request

*serial-number: contains the base64-encoded product-serial-number. The registrar MUST verify that the product-serial-number contained in the IDevID certificate of the pledge matches the serial-number field in the pledge-voucher-request. In addition, it MUST be equal to the serial-number field contained in the agent-signed data of pledge-voucher-request.

*assertion: contains the voucher assertion requested the pledge (agent-proximity). The registrar provides this information to assure successful verification of agent proximity based on the agent-signed-data.

The voucher can be optionally enhanced with the following additional parameter as defined in Section 6:

*agent-sign-cert: Contain the base64-encoded LDevID(RegAgt) EE certificate if MASA verification of agent-proximity is required to provide the assertion "agent-proximity".

The object is signed using the registrar LDevID(Reg) credential, which corresponds to the certificate signaled in the JOSE header.

```
{
   "alg": "ES256",
   "x5c": ["MIIB2jCC...dA=="]
}
{
  "ietf-voucher-request-prm:voucher": {
   "created-on": "2021-04-16T02:37:39.235Z",
   "nonce": "eDs++/FuDHGUnRxN3E14CQ==",
   "serial-number": "callee4711",
   "assertion": "agent-proximity",
   "prior-signed-voucher-request": "base64encodedvalue==",
   "agent-sign-cert": "base64encodedvalue=="
  }
}
{
    SIGNATURE
}
```

Figure 8: Example of registrar-voucher-request

The registrar sends the registrar-voucher-request to the MASA with an HTTP-over-TLS POST at the endpoint "/.well-known/brski/requestvoucher".

The registrar-voucher-request Content-Type is defined in [I-D.ietf-anima-jws-voucher] as:

application/voucher-jws+json

The registrar SHOULD include an "Accept" header field indicating the acceptable media type for the voucher-response. The media type "application/voucher-jws+json" is defined in [I-D.ietf-anima-jws-voucher].

Once the MASA receives the registrar-voucher-request it SHALL perform the verification of the contained components as described in section 5.5 in [RFC8995]. In addition, the following additional processing SHALL be done for components contained in the prior-signed-voucher-request:

- *agent-provided-proximity-registrar-cert: The MASA MAY verify that this field contains the LDevID(Reg) certificate. If so, it MUST be consistent with the certificate used to sign the registrar-voucher-request.

- *agent-signed-data: The MASA MAY verify this field to be able to provide an assertion "agent-proximity". If so, the agent-signed-data MUST contain the product-serial-number of the pledge contained in the serial-number component of the prior-signed-voucher and also in serial-number component of the registrar-

voucher-request. The LDevID(RegAgt) used to generate provide the signature is identified by the "kid" parameter of the JOSE header (agent-signed-data). If the assertion "agent-proximity" is requested, the registrar-voucher-request MUST contain the corresponding LDevID(RegAgt) EE certificate in the agent-sign-cert, which can be verified by the MASA as issued by the same domain CA as the LDevID(Reg) EE certificate. If the agent-sign-cert is not provided, the MASA MAY provide a lower level assertion "logged" or "verified"

If validation fails, the MASA SHOULD respond with an HTTP error code to the registrar. The error codes are kept as defined in section 5.6 of [RFC8995]. and comprise the response codes 403, 404, 406, and 415.

The voucher response format is as indicated in the submitted Accept header fields or based on the MASA's prior understanding of proper format for this pledge. Specifically for the pledge-responder-mode the "application/voucher-jws+json" as defined in [I-D.ietf-anima-jws-voucher] is applied. The syntactic details of vouchers are described in detail in [RFC8366]. Figure 9 shows an example of the contents of a voucher.

```
{
    "alg": "ES256",
    "x5c": ["MIIBkzCCAT...dA=="]
}
{
  "ietf-voucher:voucher": {
    "assertion": "agent-proximity",
    "serial-number": "callee4711",
    "nonce": "eDs++/FuDHGUnRxN3E14CQ==",
    "created-on": "2021-04-17T00:00:02.000Z",
    "pinned-domain-cert": "MIIBpDCCA...w=="
  }
}
{
    SIGNATURE
}
```

Figure 9: Example of MASA issued voucher

The MASA sends the voucher in the indicated form to the registrar. After receiving the voucher the registrar may evaluate the voucher for transparency and logging purposes as outlined in section 5.6 of [RFC8995]. The registrar forwards the voucher without changes to the registrar-agent.

After receiving the voucher, the registrar-agent sends the pledge's enrollment-request to the registrar. Deviating from BRSKI the enrollment-request is not a raw PKCS#10 request. As the registrar-agent is involved in the exchange, the PKCS#10 is contained in the JOSE object. The signature is created using the pledge's IDevID to provide proof-of-identity as outlined in Figure 6.

When using EST, the registrar-agent sends the enrollment request to the registrar with an HTTP-over-TLS POST at the endpoint "/.well-known/est/simpleenroll".

The enrollment-request Content-Type is:

application/jose

If validation of the wrapping signature fails, the registrar SHOULD respond with the HTTP 404 error code. If the voucher-request is in an unknown format, then an HTTP 406 error code is more appropriate. A situation that could be resolved with administrative action (such as adding a vendor/manufacturer IDevID CA as trusted party) MAY be responded with an 403 HTTP error code.

This results in a deviation from the content types used in [RFC7030] and results in additional processing at the domain registrar as EST server as following. Note that the registrar is already aware that the bootstrapping is performed in a pledge-responder-mode due to the use of the LDevID(RegAgt) certificate in the TLS establishment and the provided pledge-voucher-request in JOSE object.

* If registrar receives the enrollment-request with the Content Type application/jose, it MUST verify the signature using the certificate indicated in the JOSE header.

* The domain registrar verifies that the serial-number contained in the pledge's IDevID certificate contained in the JOSE header as being accepted to join the domain, based on the verification of the pledge-voucher-request.

* If both succeed, the registrar utilizes the PKCS#10 request contained in the JOSE body as "P10" parameter of "ietf-sztp-csr:csr" for further processing of the enrollment request with the domain CA.

[RFC Editor: please delete] /*

Open Issues:

* The domain registrar may either enhance the PKCS#10 request or generate a structure containing the attributes to be included by the CA and sends both (the original PKCS#10 request and the

enhancements) to the domain CA. As enhancing the PKCS#10 request destroys the initial proof of possession of the corresponding private key, the CA would need to accept RA-verified requests. */

A successful interaction with the domain CA will result in the pledge LDevID EE certificate, which is then forwarded by the registrar to the registrar-agent using the content type "application/pkcs7-mime".

[RFC Editor: please delete] /*

Open Issue: the enrollment response object may also be an application/jose object with a signature of the domain registrar. Note: Communicaion between domain CA and registrar is of content type "application/pkcs7-mime" Communicaion between registrar, registrar-agent and furter to the pledge should be of content type "application/jose" . */

The registrar-agent has now finished the exchanges with the domain registrar and can supply the voucher-response (from MASA via Registrar) and the enrollment-response (LDevID EE certificate) to the pledge. It can close the TLS connection to the domain registrar and provide the objects to the pledge(s). The content of the response objects is defined through the voucher [RFC8366] and the certificate [RFC5280].

5.1.3.3. Response Object Supply by Registrar-Agent to Pledge

The following description assumes that the registrar-agent has obtained the response objects from the domain registrar. It will re-start the interaction with the pledge. To contact the pledge, it may either discover the pledge as described in Section 5.1.2.2 or use stored information from the first contact with the pledge.

Preconditions in addition to Section 5.1.3.2:

*Registrar-agent: possesses voucher and LDevID certificate.

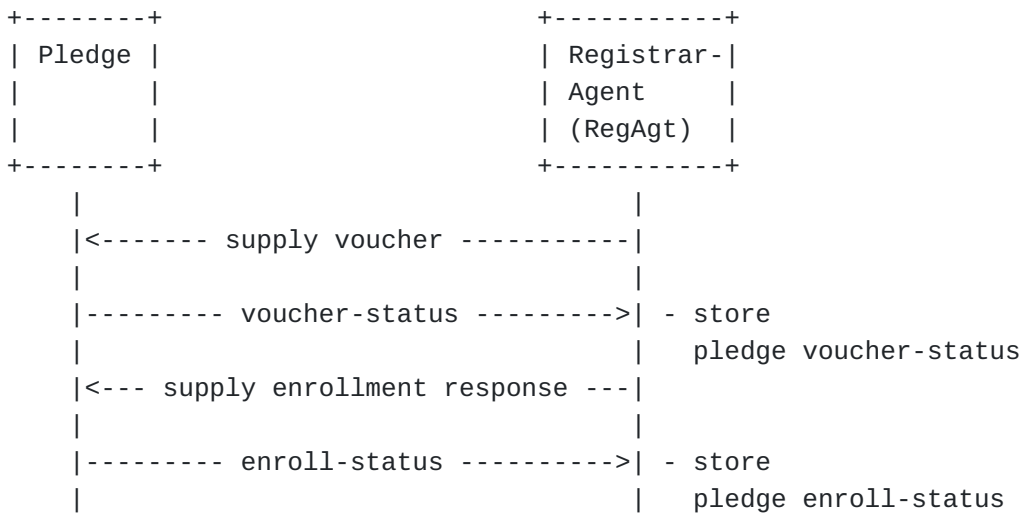


Figure 10: Response and status handling between pledge and registrar-agent

The registrar-agent provides the information via two distinct endpoints to the pledge as following.

The voucher response is provided with a HTTP POST using the operation path value of "/.well-known/brski/pledge-voucher".

The registrar-agent voucher-response Content-Type header is "application/voucher-jws+json and contains the voucher as provided by the MASA. An example if given in Figure 9.

The pledge verifies the voucher as described in section 5.6.1 in [RFC8995].

After successful verification the pledge MUST reply with a status telemetry message as defined in section 5.7 of [RFC8995]. As for the other objects, the defined object is provided with an additional signature using JOSE. The pledge generates the voucher-status-object and provides it in the response message to the registrar-agent.

The response has the Content-Type "application/jose", signed using the IDevID of the pledge as shown in Figure 11. As the reason field is optional (see [RFC8995]), it MAY be omitted in case of success.

```
{
    "alg": "ES256",
    "x5c": ["MIIB2jCC...dA=="]
{
    "version": 1,
    "status":true,
    "reason":"Informative human readable message",
    "reason-context": { "additional" : "JSON" }
}
{
    SIGNATURE
}
```

Figure 11: Example of pledge voucher-status telemetry

The enrollment response is provided with a HTTP POST using the operation path value of "/.well-known/brski/pledge-enrollment".

The registrar-agent enroll-response Content-Type header when using EST [RFC7030] as enrollment protocol, from the registrar-agent to the infrastructure is:

application/pkcs7-mime: note that it only contains the LDevID certificate for the pledge, not the certificate chain.

[RFC Editor: please delete] /*

Open Issue: the enrollment response object may also be an application/jose object with a signature of the domain registrar. This may be used either to transport additional data which is bound to the LDevID or it may be considered for enrollment status to ensure that in an error case the registrar providing the certificate can be identified. */

After successful verification the pledge MUST reply with a status telemetry message as defined in section 5.9.4 of [RFC8995]. As for the other objects, the defined object is provided with an additional signature using the JOSE. The pledge generates the enrollment status and provides it in the response message to the registrar-agent.

The response has the Content-Type "application/jose", signed using the LDevID of the pledge as shown in Figure 12. As the reason field is optional, it MAY be omitted in case of success.

```
{
  "alg": "ES256",
  "x5c": ["MIIB56uz...dA=="]
{
  "version": 1,
  "status":true,
  "reason":"Informative human readable message",
  "reason-context": { "additional" : "JSON" }
}
{
  SIGNATURE
}
```

Figure 12: Example of pledge enroll-status telemetry

Once the registrar-agent has collected the information, it can connect to the registrar agent to provide the status responses to the registrar.

5.1.3.4. Telemetry status handling (registrar-agent - domain registrar)

The following description assumes that the registrar-agent has collected the status objects from the pledge. It will provide the status objects to the registrar for further processing and audit log information of voucher-status for MASA.

Preconditions in addition to Section 5.1.3.2:

*Registrar-agent: possesses voucher-status and enroll-status objects from pledge.

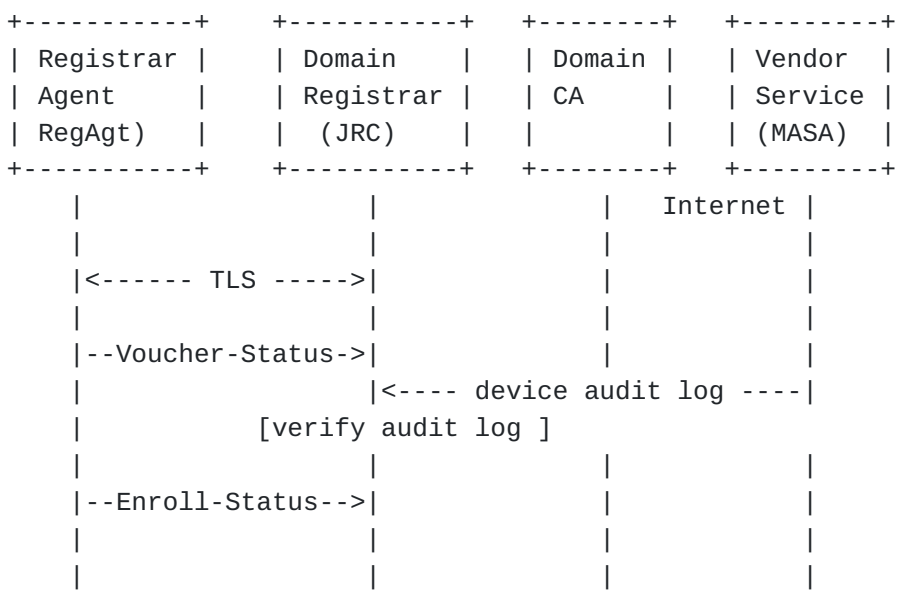


Figure 13: Bootstrapping status handling

The registrar-agent MUST provide the collected pledge voucher-status to the registrar. This status indicates the pledge could process the voucher successfully or not.

If the TLS connection to the registrar was closed, the registrar-agent establishes a TLS connection with the registrar as stated in Section 5.1.3.2.

The registrar-agent sends the pledge voucher-status object without modification to the registrar with an HTTP-over-TLS POST using the operation path value of "/.well-known/brski/voucher_status". The Content-Type header is kept as "application/jose" as described in Figure 10 and depicted in the example in Figure 11.

The registrar SHALL verify the signature of the pledge voucher-status and validate that it belongs to an accepted device in his domain based on the contained "serial-number" in the IDevID certificate referenced in the header of the voucher-status object.

According to [RFC8995] section 5.7, the registrar SHOULD respond with an HTTP 200 but MAY simply fail with an HTTP 404 error. The registrar-agent may use the response to signal success / failure to the service technician operating the registrar agent. Within the server logs the server SHOULD capture this telemetry information.

The registrar SHOULD proceed with the collecting and logging the status information by requesting the MASA audit-log from the MASA service as described in section 5.8 of [RFC8995].

The registrar-agent MUST provide the enroll-status object to the registrar. The status indicates the pledge could process the enroll-response object and holds the corresponding private key.

The registrar-agent sends the pledge enroll-status object without modification to the registrar with an HTTP-over-TLS POST using the operation path value of "/.well-known/brski/enrollstatus". The Content-Type header is kept as "application/jose" as described in Figure 10 and depicted in the example in Figure 12.

The registrar SHALL verify the signature of the pledge enroll-status object and validate that it belongs to an accepted device in his domain based on the contained product-serial-number in the LDevID EE certificate referenced in the header of the enroll-status object. Note that the verification of a signature of the object is a deviation form the described handling in section 5.9.4 of [RFC8995].

According to [RFC8995] section 5.9.4, the registrar SHOULD respond with an HTTP 200 but MAY simply fail with an HTTP 404 error. The

registrar-agent may use the response to signal success / failure to the service technician operating the registrar agent. Within the server log the registrar SHOULD capture this telemetry information.

6. Voucher Request Artifact

The following enhancement extends the voucher-request as defined in [RFC8995] to include additional fields necessary for handling bootstrapping in the pledge-responder-mode.

6.1. Tree Diagram

The following tree diagram is mostly a duplicate of the contents of [RFC8995], with the addition of the fields agent-signed-data, the registrar-proximity-certificate, and agent-signing certificate. The tree diagram is described in [RFC8340]. The enhanced fields are described in Section Each node in the diagram is fully described by the YANG module in Section Section 6.2. Please review the YANG module for a detailed description of the voucher-request format.

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

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

6.2. YANG Module

The following YANG module extends the [RFC8995] Voucher Request to include a signed artifact from the registrar-agent (agent-signed-data) as well as the registrar-proximity-certificate and the agent-signing certificate.

```
<CODE BEGINS> file "ietf-voucher-request-prm@2021-10-15.yang"


module ietf-voucher-request-prm {
  yang-version 1.1;

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

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

  import ietf-voucher-request {
    prefix vcr;
    description
      "This module defines the format for a voucher request,
          which is produced by a pledge as part of the RFC8995
          onboarding process.";
    reference
      "RFC 8995: Bootstrapping Remote Secure Key Infrastructure";
  }

  organization
   "IETF ANIMA Working Group";

  contact
   "WG Web:   <http://tools.ietf.org/wg/anima/>
    WG List:  <mailto:anima@ietf.org>
    Author:   Steffen Fries
              <mailto:steffen.fries@siemens.com>
    Author:   Eliot Lear
              <mailto: lear@cisco.com>
    Author:   Thomas Werner
              <mailto: thomas-werner@siemens.com>
    Author:   Michael Richardson
              <mailto: mcr+ietf@sandelman.ca>";

  description
   "This module defines an extension of the RFC8995 voucher
    request to permit a registrar-agent to convey the adjacency
    relationship from the registrar-agent to the registrar.

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

```
  and 'OPTIONAL' in the module text are to be interpreted as
  described in RFC 2119.";
revision 2021-08-19 {
  description
   "Initial version";
  reference
   "RFC XXXX: VBRSKI for Pledge in Responder Mode";
}
rc:yang-data voucher-request-prm-artifact {
  // YANG data template for a voucher-request.
  uses voucher-request-prm-grouping;
}
// Grouping defined for future usage
grouping voucher-request-prm-grouping {
  description
    "Grouping to allow reuse/extensions in future work.";
  uses vcr:voucher-request-grouping {

    augment voucher {
      description "Base the voucher-request-prm upon the
        regular one";
      leaf agent-signed-data {
        type binary;
        description
          "The agent-signed-data field contains a JOSE [RFC7515]
           object provided by the Registrar-Agent to the Pledge.

           This artifact is signed by the Registrar-Agent
           and contains a copy of the pledge's serial-number.";
      }

      leaf agent-provided-proximity-registrar-cert {
        type binary;
        description
          "An X.509 v3 certificate structure, as specified by
           RFC 5280, Section 4, encoded using the ASN.1
           distinguished encoding rules (DER), as specified
           in ITU X.690.
           The first certificate in the registrar TLS server
           certificate_list sequence (the end-entity TLS
           certificate; see RFC 8446) presented by the
           registrar to the registrar-agent and provided to
           the pledge.
           This MUST be populated in a pledge's voucher-request
           when an agent-proximity assertion is requested.";
        reference
          "ITU X.690: Information Technology - ASN.1 encoding
           rules: Specification of Basic Encoding Rules (BER),
           Canonical Encoding Rules (CER) and Distinguished
```

```
            Encoding Rules (DER)
            RFC 5280: Internet X.509 Public Key Infrastructure
            Certificate and Certificate Revocation List (CRL)
            Profile
            RFC 8446: The Transport Layer Security (TLS)
            Protocol Version 1.3";
        }

        leaf agent-sign-cert {
          type binary;
          description
            "An X.509 v3 certificate structure, as specified by
             RFC 5280, Section 4, encoded using the ASN.1
             distinguished encoding rules (DER), as specified
             in ITU X.690.
             This certificate can be used by the pledge,
             the registrar, and the MASA to verify the signature
             of agent-signed-data. It is an optional component
             for the pledge-voucher request.
             This MUST be populated in a registrar's
             voucher-request when an agent-proximity assertion
             is requested.";
          reference
            "ITU X.690: Information Technology - ASN.1 encoding
             rules: Specification of Basic Encoding Rules (BER),
             Canonical Encoding Rules (CER) and Distinguished
             Encoding Rules (DER)
             RFC 5280: Internet X.509 Public Key Infrastructure
             Certificate and Certificate Revocation List (CRL)
             Profile";
        }
      }
    }
  }
}


<CODE ENDS>
```

Examples for the pledge-voucher-request are provided in Section 5.1.3.2.

7. IANA Considerations

This document requires the following IANA actions:

IANA is requested to enhance the Registry entitled: "BRSKI well-known URIs" with the following:

URI	document	description
pledge-voucher-request	[THISRFC]	create pledge-voucher-request
pledge-enrollment-request	[THISRFC]	create pledge-enrollment-request
pledge-voucher	[THISRFC]	supply voucher response
pledge-enrollment	[THISRFC]	supply enrollment response
pledge-CACerts	[THISRFC]	supply CA certs to pledge

8. Privacy Considerations

The credential used by the registrar-agent to sign the data for the pledge in case of the pledge-initiator-mode should not contain personal information. Therefore, it is recommended to use an LDevID certificate associated with the device instead of a potential service technician operating the device, to avoid revealing this information to the MASA.

9. Security Considerations

9.1. Exhaustion Attack on Pledge

Exhaustion attack on pledge based on DoS attack (connection establishment, etc.)

9.2. Misuse of acquired Voucher and Enrollment responses by Registrar-Agent

A Registrar-agent that uses acquired voucher and enrollment response for domain 1 in domain 2 can be detected by the pledge-voucher-request processing on the domain registrar side. This requires the domain registrar to verify the proximity-registrar-cert leaf in the pledge-voucher-request against his own LDevID. In addition, the domain registrar has to verify the association of the pledge to his domain based on the product-serial-number contained in the pledge-voucher-request.

Misbinding of pledge by a faked domain registrar is countered as described in BRSKI security considerations (section 11.4).

9.3. Misuse of Registrar-Agent Credentials

Concerns have been raised, that there may be opportunities to misuse the registrar-agent with a valid LDevID. This may be addressed by utilizing short-lived certificates (e.g., valid for a day) to authenticate the registrar-agent against the domain registrar. The LDevID certificate for the registrar-agent may be provided by a prior BRSKI execution based on an existing IDevID. Alternatively,

the LDevID may be acquired by a service technician after authentication against the issuing CA.

9.4. YANG Module Security Considerations

The enhanced voucher-request described in section Section 6 bases on [RFC8995], but uses a different encoding, based on [I-D.ietf-anima-jws-voucher]. Therefore, similar considerations as described in Section 11.7 (Security Considerations) of [RFC8995] apply. The YANG module specified in this document defines the schema for data that is subsequently encapsulated by a JOSE signed-data content type, as described [I-D.ietf-anima-jws-voucher]. As such, all of the YANG-modeled data is protected from modification. The use of YANG to define data structures, via the "yang-data" statement, is relatively new and distinct from the traditional use of YANG to define an API accessed by network management protocols such as NETCONF [RFC6241] and RESTCONF [RFC8040]. For this reason, these guidelines do not follow the template described by Section 3.7 of [RFC8407]].

10. Acknowledgments

We would like to thank the various reviewers, in particular Brian E. Carpenter, Michael Richardson, Giorgio Romanenghi, Oskar Camenzind, for their input and discussion on use cases and call flows.

11. References

11.1. Normative References

[I-D.ietf-anima-jws-voucher] Richardson, M. and T. Werner, "JWS signed Voucher Artifacts for Bootstrapping Protocols", Work in Progress, Internet-Draft, draft-ietf-anima-jws-voucher-00, 25 July 2021, <https://www.ietf.org/archive/id/draft-ietf-anima-jws-voucher-00.txt>.

[I-D.ietf-netconf-sztp-csr] Watsen, K., Housley, R., and S. Turner, "Conveying a Certificate Signing Request (CSR) in a Secure Zero Touch Provisioning (SZTP) Bootstrapping Request", Work in Progress, Internet-Draft, draft-ietf-netconf-sztp-csr-08, 24 August 2021, <https://www.ietf.org/archive/id/draft-ietf-netconf-sztp-csr-08.txt>.

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

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol

(NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

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

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

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

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <https://www.rfc-editor.org/info/rfc7515>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

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

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

[RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <https://www.rfc-editor.org/info/rfc8407>.

[RFC8995] Pritikin, M., Richardson, M., Eckert, T., Behringer, M., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructure (BRSKI)", RFC 8995, DOI 10.17487/RFC8995, May 2021, <https://www.rfc-editor.org/info/rfc8995>.

11.2. Informative References

[IEEE-802.1AR] Institute of Electrical and Electronics Engineers, "IEEE 802.1AR Secure Device Identifier", IEEE 802.1AR, June 2018.

[RFC2986] Nystrom, M. and B. Kaliski, "PKCS #10: Certification Request Syntax Specification Version 1.7", RFC 2986, DOI 10.17487/RFC2986, November 2000, <https://www.rfc-editor.org/info/rfc2986>.

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

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

Appendix A. History of Changes [RFC Editor: please delete]

From IETF draft-ietf-anima-brski-async-enroll-03 -> IETF anima-brski-prm-internal-00:

* Moved UC2 related parts defining the pledge in responder mode from draft-ietf-anima-brski-async-enroll-03 to this document This required changes and adaptations in several sections to remove the description and references to UC1.

* Addressed feedback for voucher-request enhancements from YANG doctor early review in Section 6 as well as in the security considerations (formerly named ietf-async-voucher-request).

* Renamed ietf-async-voucher-request to IETF-voucher-request-prm to to allow better listing of voucher related extensions; aligned with constraint voucher (#20)

* Utilized ietf-voucher-request-async instead of ietf-voucher-request in voucher exchanges to utilize the enhanced voucher-request.

* Included changes from draft-ietf-netconf-sztp-csr-06 regarding the YANG definition of csr-types into the enrollment request exchange.

From IETF draft 02 -> IETF draft 03:

- *Housekeeping, deleted open issue regarding YANG voucher-request in Section 5.1.3.1 as voucher-request was enhanced with additional leaf.

- *Included open issues in YANG model in Section 5.1 regarding assertion value agent-proximity and csr encapsulation using SZTP sub module).

From IETF draft 01 -> IETF draft 02:

- *Defined call flow and objects for interactions in UC2. Object format based on draft for JOSE signed voucher artifacts and aligned the remaining objects with this approach in Section 5.1.3 .

- *Terminology change: issue #2 pledge-agent -> registrar-agent to better underline agent relation.

- *Terminology change: issue #3 PULL/PUSH -> pledge-initiator-mode and pledge-responder-mode to better address the pledge operation.

- *Communication approach between pledge and registrar-agent changed by removing TLS-PSK (former section TLS establishment) and associated references to other drafts in favor of relying on higher layer exchange of signed data objects. These data objects are included also in the pledge-voucher-request and lead to an extension of the YANG module for the voucher-request (issue #12).

- *Details on trust relationship between registrar-agent and registrar (issue #4, #5, #9) included in Section 5.1.

- *Recommendation regarding short-lived certificates for registrar-agent authentication towards registrar (issue #7) in the security considerations.

- *Introduction of reference to agent signing certificate using SKID in agent signed data (issue #11).

- *Enhanced objects in exchanges between pledge and registrar-agent to allow the registrar to verify agent-proximity to the pledge (issue #1) in Section 5.1.3.

- *Details on trust relationship between registrar-agent and pledge (issue #5) included in Section 5.1.

- *Split of use case 2 call flow into sub sections in Section 5.1.3.

From IETF draft 00 -> IETF draft 01:

- Update of scope in Section 3.1 to include in which the pledge acts as a server. This is one main motivation for use case 2.

- Rework of use case 2 in Section 5.1 to consider the transport between the pledge and the pledge-agent. Addressed is the TLS channel establishment between the pledge-agent and the pledge as well as the endpoint definition on the pledge.

- First description of exchanged object types (needs more work)

- Clarification in discovery options for enrollment endpoints at the domain registrar based on well-known endpoints do not result in additional /.well-known URIs. Update of the illustrative example. Note that the change to /brski for the voucher related endpoints has been taken over in the BRSKI main document.

- Updated references.

- Included Thomas Werner as additional author for the document.

From individual version 03 -> IETF draft 00:

- Inclusion of discovery options of enrollment endpoints at the domain registrar based on well-known endpoints in new section as replacement of section 5.1.3 in the individual draft. This is intended to support both use cases in the document. An illustrative example is provided.

- Missing details provided for the description and call flow in pledge-agent use case Section 5.1, e.g. to accommodate distribution of CA certificates.

- Updated CMP example in to use lightweight CMP instead of CMP, as the draft already provides the necessary /.well-known endpoints.

- Requirements discussion moved to separate section in Section 4. Shortened description of proof of identity binding and mapping to existing protocols.

- Removal of copied call flows for voucher exchange and registrar discovery flow from [RFC8995] in UC1 to avoid doubling or text or inconsistencies.

- Reworked abstract and introduction to be more crisp regarding the targeted solution. Several structural changes in the document to have a better distinction between requirements, use case description, and solution description as separate sections. History moved to appendix.

From individual version 02 -> 03:

- Update of terminology from self-contained to authenticated self-contained object to be consistent in the wording and to underline the protection of the object with an existing credential. Note that the naming of this object may be discussed. An alternative name may be attestation object.

- Simplification of the architecture approach for the initial use case having an offsite PKI.

- Introduction of a new use case utilizing authenticated self-contain objects to onboard a pledge using a commissioning tool containing a pledge-agent. This requires additional changes in the BRSKI call flow sequence and led to changes in the introduction, the application example,and also in the related BRSKI-PRM call flow.

From individual version 01 -> 02:

- Update of introduction text to clearly relate to the usage of IDevID and LDevID.

- Update of description of architecture elements and changes to BRSKI in Section 5.

- Enhanced consideration of existing enrollment protocols in the context of mapping the requirements to existing solutions in Section 4.

From individual version 00 -> 01:

- Update of examples, specifically for building automation as well as two new application use cases in Section 3.2.

- Deletion of asynchronous interaction with MASA to not complicate the use case. Note that the voucher exchange can already be handled in an asynchronous manner and is therefore not considered further. This resulted in removal of the alternative path the MASA in Figure 1 and the associated description in Section 5.

- Enhancement of description of architecture elements and changes to BRSKI in Section 5.

- Consideration of existing enrollment protocols in the context of mapping the requirements to existing solutions in Section 4.

- New section starting with the mapping to existing enrollment protocols by collecting boundary conditions.

Authors' Addresses

Steffen Fries
Siemens AG
Otto-Hahn-Ring 6
81739 Munich
Germany

Email: steffen.fries@siemens.com
URI: https://www.siemens.com/

Thomas Werner
Siemens AG
Otto-Hahn-Ring 6
81739 Munich
Germany

Email: thomas-werner@siemens.com
URI: https://www.siemens.com/

Eliot Lear
Cisco Systems
Richtistrasse 7
CH-8304 Wallisellen
Switzerland

Phone: +41 44 878 9200
Email: lear@cisco.com

Michael C. Richardson
Sandelman Software Works

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