ANIMA WG S. Fries
Internet-Draft H. Brockhaus
Intended status: Standards Track Siemens
Expires: December 16, 2021 E. Lear
Cisco Systems
T. Werner
Siemens
June 14, 2021

Support of asynchronous Enrollment in BRSKI (BRSKI-AE)
draft-ietf-anima-brski-async-enroll-02

Abstract

This document describes enhancements of bootstrapping a remote secure key infrastructure (BRSKI, [RFC8995]) to also operate in domains featuring no or only timely limited connectivity between involved components. Further enhancements are provided to perform the BRSKI approach in environments, in which the role of the pledge changes from a client to a server . This changes the interaction model from a pledge-initiator-mode to a pledge-responder-mode. To support both use cases, BRSKI-AE relies on the exchange of authenticated self-contained objects (signature-wrapped objects) also for requesting and distributing of domain specific device certificates. The defined approach is agnostic regarding the utilized enrollment protocol allowing the application of existing and potentially new certificate management protocols.

Status of This Memo

Copyright Notice

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 authenticated self-contained (signed) objects, which may be provided online (synchronous) or offline (asynchronous) via the domain registrar to the pledge and originate from a Manufacturer's Authorized Signing Authority (MASA).

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) or local registration authority (LRA). The binding to TLS is used to protect the exchange of a certification request (for a LDevID EE certificate) and to provide data origin authentication (client identity information), to support the authorization decision for processing the certification request. The TLS connection is mutually authenticated and the client-side authentication utilizes the pledge's manufacturer issued device certificate (IDevID certificate). This approach requires an on-site availability of a local asset or inventory management system performing the authorization decision based on tuple of the certification request and the pledge authentication using the IDevID certificate, to issue a domain specific certificate to the pledge. The EST server (the domain registrar) terminates the security association with the pledge and thus the binding between the certification request and the authentication of the pledge via TLS.

This type of enrollment utilizing an online connection to the PKI is considered as synchronous enrollment.

For certain use cases on-site support of a RA/CA component and/or an asset management is not available and rather provided by an operator's backend and may be provided timely limited or completely through offline interactions. This may be due to higher security requirements for operating the certification authority or for optimization of operation for smaller deployments to avoid the always on-site operation. The authorization of a certification request based on an asset management in this case will not / can not be performed on-site at enrollment time. Enrollment, which cannot be performed in a (timely) consistent fashion is considered as asynchronous enrollment in this document. It requires the support of a store and forward functionality of certification request together with the requester authentication (and identity) information. This enables processing of the request at a later point in time. A similar situation may occur through network segmentation, which is utilized in industrial systems to separate domains with different security needs. Here, a similar requirement arises if the communication channel carrying the requester authentication is terminated before the RA/CA authorization handling of the certification request. If a second communication channel is opened to forward the certification request to the issuing RA/ CA, the requester authentication information needs to be retained and ideally bound to the certification request. This uses case is independent from timely limitations of the first use case. For both cases, it is assumed that the requester authentication information is utilized in the process of authorization of a certification request. There are different options to perform store and forward of certification requests including the requester authentication information:

o Providing a trusted component (e.g., an LRA) in the target domain, which stores the certification request combined with the requester authentication information (based on the IDevID) and potentially the information about a successful proof of possession (of the corresponding private key) in a way prohibiting changes to the combined information. Note that the assumption is that the information elements may not be cryptographically bound together. Once connectivity to the backend is available, the trusted component forwards the certification request together with the requester information (authentication and proof of possession) to the off-site PKI for further processing. It is assumed that the off-site PKI in this case relies on the local pledge authentication result and thus performs the authorization and issues the requested certificate. In BRSKI the trusted component may be the EST server residing co-located with the registrar in the target domain.

o Utilization of authenticated self-contained objects for the enrollment, binding the certification request and the requester authentication in a cryptographic way. This approach reduces the necessary trust in a domain component to storage and delivery. Unauthorized modifications of the requester information (request and authentication) can be detected during the verification of the authenticated self-contained object.

Focus of this document the support of handling authenticated self-contained objects for bootstrapping. As it is intended to enhance BRSKI it is named BRSKI-AE, where AE stands for asynchronous enrollment. As BRSKI, BRSKI-AE results in the pledge storing an X.509 domain certificate and sufficient information for verifying the domain registrar / proxy identity (LDevID CA Certificate) as well as domain specific X.509 device certificates (LDevID EE certificate).

Based on the proposed approach, a second set of scenarios can be addressed, in which the pledge has either no direct communication path to the domain registrar, e.g., due to missing network connectivity or a different technology stack. In such scenarios the pledge is expected to act as a server rather than a client. The pledge will be triggered to generate request objects to be onboarded 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 could 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 applicable to the additional use cases. This is addressed by

o enhancing the well-known URI approach with an additional path for the utilized enrollment protocol.

o defining a certificate waiting indication and handling, if the certifying component is (temporarily) not available.

o 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.

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

Note that in contrast to BRSKI, BRSKI-AE assumes support of multiple enrollment protocols on the infrastructure side, allowing the pledge manufacturer to select the most appropriate. Thus, BRSKI-AE can be applied for both, asynchronous and synchronous enrollment.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD 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.

LRA: Local registration authority, an optional RA system component with proximity to end entities.

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

This solution is intended to be used in domains with limited support of on-site PKI services and comprises use cases in which:

o there is no registration authority available in the target domain. The connectivity to an off-site RA in an operator's network may only be available temporarily. A local store and forward device is used for the communication with the off-site services.

o authoritative actions of a LRA are limited and may not comprise authorization of certification requests of pledges. Final authorization is done at the RA residing in the operator domain.

o the target deployment domain already has an established certificate management approach that shall be reused to (e.g., in brownfield installations).

In addition, 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 asynchronous operation as intended with BRSKI-AE.

3.2.1. Rolling stock

Rolling stock or railroad cars contain a variety of sensors, actuators, and controllers, which communicate within the railroad car but also exchange information between railroad cars building a train, or with a backend. These devices are typically unaware of backend connectivity. Managing certificates may be done during maintenance cycles of the railroad car, but can already be prepared during operation. The preparation may comprise the generation of

certification requests by the components which are collected and forwarded for processing, once the railroad car is connected to the operator backend. The authorization of the certification request is then done based on the operator's asset/inventory information in the backend.

3.2.2. 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.

3.2.3. Substation automation

In electrical substation automation a control center typically hosts PKI services to issue certificates for Intelligent Electronic Devices (IED)s operated in a substation. Communication between the substation and control center is done through a proxy/gateway/DMZ, which terminates protocol flows. Note that [NERC-CIP-005-5] requires inspection of protocols at the boundary of a security perimeter (the substation in this case). In addition, security management in substation automation assumes central support of different enrollment protocols to facilitate the capabilities of IEDs from different vendors. The IEC standard IEC62351-9 [IEC-62351-9] specifies the mandatory support of two enrollment protocols, SCEP [RFC8894] and EST [RFC7030] for the infrastructure side, while the IED must only support one of the two.

3.2.4. Electric vehicle charging infrastructure

For the electric vehicle charging infrastructure protocols have been defined for the interaction between the electric vehicle (EV) and the charging point (e.g., ISO 15118-2 [ISO-IEC-15118-2]) as well as between the charging point and the charging point operator (e.g. OCPP [OCPP]). Depending on the authentication model, unilateral or mutual authentication is required. In both cases the charging point uses an X.509 certificate to authenticate itself in the context of a TLS connection between the EV and the charging point. The management of this certificate depends (beyond others) on the selected backend connectivity protocol. Specifically, in case of OCPP it is intended as single communication protocol between the charging point and the backend carrying all information to control the charging operations and maintain the charging point itself. This means that the certificate management is intended to be handled in-band of OCPP. This requires to be able to encapsulate the certificate management exchanges in a transport independent way. Authenticated self-containment will ease this by allowing the transport without a separate enrollment protocol. This provides a binding of the exchanges to the identity of the communicating endpoints.

3.2.5. 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 external PKI resources will be allowed in carefully controlled short periods of time, for example when a batch of new devices are deployed, but impossible at other times.

3.2.6. 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. Requirement discussion and mapping to solution elements

For the requirements discussion it is assumed that the domain registrar receiving a certification request as authenticated self-contained object is not the authorization point for this certification request. If the domain registrar is the authorization point and the pledge has a direct connection to the registrar, BRSKI can be used directly. Note that BRSKI-AE could also be used in this case.

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 certification request and further information to support asynchronous operation.

At least the following properties are required:

- o 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.

- o 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:

- o 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 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.

  * CRMF [RFC4211]: Defines a structure for the certification request message. The structure supports integrity protection and proof of possession, through a signature generated over parts of the structure by using the private key corresponding to the contained public key. CRMF also supports further proof-of-possession methods for key pairs not capable to be used for signing.

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.

o Proof of Identity options: The certification request should be bound to an existing credential (here IDevID) to enable a proof of identity and based on it an authorization of the certification request. The binding may be realized through security options in an underlying transport protocol if the authorization of the certification request is done at the next communication hop. Alternatively, this binding can be done by a wrapping signature employing an existing credential (initial: IDevID, renewal: LDevID). This requirement is addressed by existing enrollment protocols in different ways, for instance:

  * EST [RFC7030]: Utilizes PKCS#10 to encode the certification request. The Certificate Signing Request (CSR) may contain a binding to the underlying TLS by including the tls-unique value in the self-signed CSR structure. The tls-unique value is one result of the TLS handshake. As the TLS handshake is performed mutually authenticated and the pledge utilized its IDevID for it, the proof of identity can be provided by the binding to the TLS session. This is supported in EST using the simpleenroll endpoint. To avoid the binding to the underlying authentication in the transport layer, EST offers the support of a wrapping the CSR with an existing certificate by using Full PKI Request messages.

  * SCEP [RFC8894]: Provides the option to utilize either an existing secret (password) or an existing certificate to protect the CSR based on SCEP Secure Message Objects using CMS wrapping ([RFC5652]). Note that the wrapping using an existing IDevID credential in SCEP is referred to as renewal. SCEP therefore does not rely on the security of an underlying transport.

  * CMP [RFC4210] Provides the option to utilize either an existing secret (password) or an existing certificate to protect the PKIMessage containing the certification request. The certification request is encoded utilizing CRMF. PKCS#10 is

optionally supported. The proof of identity of the PKIMessage containing the certification request can be achieved by using IDevID credentials to a PKIProtection carrying the actual signature value. CMP therefore does not rely on the security of an underlying transport protocol.

* CMC [RFC5272] Provides the option to utilize either an existing secret (password) or an existing certificate to protect the certification request (either in CRMF or PKCS#10) based on CMS [RFC5652]). Here a FullCMCRequest can be used, which allows signing with an existing IDevID credential to provide a proof of identity. CMC therefore does not rely on the security of an underlying transport.

Note that besides the already existing enrollment protocols there is ongoing work in the ACE WG to define an encapsulation of EST messages in OSCORE to result in a TLS independent way of protecting EST. This approach [I-D.selander-ace-coap-est-oscore] may be considered as further variant.

5. Architectural Overview and Communication Exchanges

To support asynchronous enrollment, the base system architecture defined in BRSKI [RFC8995] is enhanced to facilitate the two target use cases.

o Use case 1 (Pledge-initiator-mode): the pledge requests certificates from a PKI operated off-site via the domain registrar. The communication model follows the BRSKI model in which the pledge initiates the communication.

o Use case 2 (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.

Both use cases are described in the next subsections. They utilize the existing BRSKI architecture elements as much as possible. 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-AE relies on the defined message wrapping mechanisms of the enrollment protocols stated in Section 4 above.

5.1. Use Case 1 (pledge-initiator-mode): Support of off-site PKI service

One assumption of BRSKI-AE is that the authorization of a certification request is performed based on an authenticated self-contained object, binding the certification request to the authentication using the IDevID. This supports interaction with off-site or off-line PKI (RA/CA) components. In addition, the authorization of the certification request may not be done by the domain registrar but by a PKI residing in the backend of the domain operator (off-site) as described in Section 3.1. Also, the certification request may be piggybacked by another protocol. This leads to changes in the placement or enhancements of the logical elements as shown in Figure 1.

```
                                              +------------------------+
      +--------------Drop Ship--------------->| Vendor Service         |
      |                                       +------------------------+
      |                                       | M anufacturer|         |
      |                                       | A uthorized  |Ownership|
      |                                       | S igning     |Tracker  |
      |                                       | A uthority   |         |
      |                                       +--------------+---------+
      |                                                      ^
      |                                                      |
      V                                                      |
   +--------+     .........................................  |
   |        |     .                                       .  | BRSKI-
   |        |     .  +------------+       +------------+  .  | MASA
   | Pledge |     .  |   Join     |       | Domain     <-----+
   |        |     .  |   Proxy    |       | Registrar/ |  .
   |        <-------->............<-------> Enrollment |  .
   |        |     .  |        BRSKI-AE    | Proxy      |  .
   | IDevID |     .  |            |       +------^-----+  .
   |        |     .  +------------+              |        .
   |        |     .                              |        .
   +--------+     ...............................|.........
                   "on-site domain" components   |
                                                 |e.g., RFC 7030,
                                                 |      RFC 4210, ...
    .............................................|.....................
    . +---------------------------+     +--------v------------------+ .
    . | Public Key Infrastructure |<----+ PKI RA                    | .
    . | PKI CA                    |---->+                           | .
    . +---------------------------+     +---------------------------+ .
    ...................................................................
            "off-site domain" components
```

Figure 1: Architecture overview using off-site PKI components

The architecture overview in Figure 1 utilizes the same logical elements as BRSKI but with a different placement in the deployment architecture for some of the elements. The main difference is the placement of the PKI RA/CA component, which is performing the authorization decision for the certification request message. It is placed in the off-site domain of the operator (not the deployment site directly), which may have no or only temporary connectivity to the deployment or on-site domain of the pledge. This is to underline the authorization decision for the certification request in the backend rather than on-site. The following list describes the components in the target domain:

o Join Proxy: same functionality as described in BRSKI.

o Domain Registrar / Enrollment Proxy: In general the domain registrar proxy has a similar functionality regarding the imprinting of the pledge in the deployment domain to facilitate the communication of the pledge with the MASA and the PKI. Different is the authorization of the certification request. BRSKI-AE allows to perform this in the operator's backend (off-site), and not directly at the domain registrar.

   * Voucher exchange: The voucher exchange with the MASA via the domain registrar is performed as described in BRSKI [RFC8995] .

   * Certificate enrollment: For the pledge enrollment the domain registrar in the deployment domain supports the adoption of the pledge in the domain based on the voucher request. Nevertheless, it may not have sufficient information for authorizing the certification request. If the authorization of the certification request is done in the off-site domain, the domain registrar forwards the certification request to the RA to perform the authorization. Note that this requires, that the certification request object is enhanced with a proof-of-identity to allow the authorization based on the bound identity information of the pledge. As stated above, this can be done by an additional signature using the IDevID. The domain registrar here acts as an enrollment proxy or local registration authority. It is also able to handle the case having no connection temporarily to an off-site PKI, by storing the authenticated certification request and forwarding it to the RA upon reestablished connectivity. As authenticated self-contained objects are used, it requires an enhancement of the domain registrar. This is done by supporting alternative enrollment approaches (protocol options, protocols, encoding) by enhancing the addressing scheme to communicate with the domain registrar (see Section 5.1.5).

The following list describes the vendor related components/service outside the deployment domain:

o MASA: general functionality as described in [RFC8995]. Assumption is that the interaction with the MASA may be synchronous (voucher request with nonce) or asynchronous (voucher request without nonce).

o Ownership tracker: as defined in [RFC8995].

The following list describes the operator related components/service operated in the backend:

- o PKI RA: Performs certificate management functions (validation of certification requests, interaction with inventory/asset management for authorization of certification requests, etc.) for issuing, updating, and revoking certificates for a domain as a centralized infrastructure for the domain operator. The inventory (asset) management may be a separate component or integrated into the RA directly.

- o PKI CA: Performs certificate generation by signing the certificate structure provided in the certification request.

Based on BRSKI and the architectural changes the original protocol flow is divided into three phases showing commonalities and differences to the original approach as depicted in the following.

- o Discovery phase (same as BRSKI)

- o Voucher exchange with deployment domain registrar (same as BRSKI).

- o Enrollment phase (changed to support the application of authenticated self-contained objects).

5.1.1. Behavior of a pledge

The behavior of a pledge as described in [RFC8995] is kept with one exception. After finishing the imprinting phase (4) the enrollment phase (5) is performed with a method supporting authenticated self-contained objects. Using EST with simple-enroll cannot be applied here, as it binds the pledge authentication with the existing IDevID to the transport channel (TLS) rather than to the certification request object directly. This authentication in the transport layer is not visible / verifiable at the authorization point in the off-site domain. Section 7 discusses potential enrollment protocols and options applicable.

5.1.2. Pledge - Registrar discovery and voucher exchange

The discovery phase is applied as specified in [RFC8995].

5.1.3. Registrar - MASA voucher exchange

The voucher exchange is performed as specified in [RFC8995].

5.1.4. Pledge - Registrar - RA/CA certificate enrollment

As stated in Section 4 the enrollment shall be performed using an authenticated self-contained object providing proof of possession and proof of identity.

```
+--------+         +---------+    +------------+     +------------+
| Pledge |         | Circuit |    | Domain     |     | Operator   |
|        |         | Join    |    | Registrar  |     | RA/CA      |
|        |         | Proxy   |    |  (JRC)     |     | (OPKI)     |
+--------+         +---------+    +------------+     +------------+
  /-->                                      |                     |
[Request of CA Certificates]                |                     |
  |---------- CA Certs Request ------------>|                     |
  |              [if connection to operator domain is available] |
  |                                         |-Request CA Certs ->|
  |                                         |<- CA Certs Response|
  |<-------- CA Certs Response--------------|                     |
  /-->                                      |                     |
[Request of Certificate Attributes to be included]                |
  |---------- Attribute Request ----------->|                     |
  |              [if connection to operator domain is available] |
  |                                         |Attribute Request ->|
  |                                         |<-Attribute Response|
  |<--------- Attribute Response -----------|                     |
  /-->                                      |                     |
[Certification request]                     |                     |
  |-------------- Cert Request ------------>|                     |
  |              [if connection to operator domain is available] |
  |                                         |--- Cert Request -->|
  |                                         |                     |
[Optional Certificate waiting indication]   |                     |
  /-->                                      |                     |
  |<----- Cert Response (with Waiting) -----|                     |
  |-- Cert Polling (with orig request ID) ->|                     |
  |                                         |                     |
  /-->                                      |                     |
  |                                         |<-- Cert Response --|
  |                                         |                     |
  |<-- Cert Response (with Certificate) ----|                     |
  /-->                                      |                     |
[Certificate confirmation]                  |                     |
  |-------------- Cert Confirm ------------>|                     |
  |                                        /-->                   |
  |                                         |[optional]           |
  |                                         |--- Cert Confirm -->|
  |                                         |<-- PKI Confirm ----|
  |<------------- PKI/Registrar Confirm ----|                     |
```

Figure 2: Certificate enrollment

The following list provides an abstract description of the flow depicted in Figure 2.

- CA Cert Request: The pledge SHOULD request the full distribution of CA Certificates. This ensures that the pledge has the complete set of current CA certificates beyond the pinned-domain-cert (which may be the domain registrar certificate contained in the voucher).

- CA Cert Response: Contains at least one CA certificate of the issuing CA.

- Attribute Request: Typically, the automated bootstrapping occurs without local administrative configuration of the pledge. Nevertheless, there are cases, in which the pledge may also include additional attributes specific to the deployment domain into the certification request. To get these attributes in advance, the attribute request SHOULD be used.

- Attribute Response: Contains the attributes to be included in the certification request message.

- Cert Request: Depending on the utilized enrollment protocol, this certification request contains the authenticated self-contained object ensuring both, proof-of-possession of the corresponding private key and proof-of-identity of the requester.

- Cert Response: certification response message containing the requested certificate and potentially further information like certificates of intermediary CAs on the certification path.

- Cert Waiting: waiting indication for the pledge to retry after a given time. For this a request identifier is necessary. This request identifier may be either part of the enrollment protocol or build based on the certification request.

- Cert Polling: querying the registrar, if the certificate request was already processed; can be answered either with another Cert Waiting, or a Cert Response.

- Cert Confirm: confirmation message from pledge after receiving and verifying the certificate.

- PKI/Registrar Confirm: confirmation message from PKI/registrar about reception of the pledge's certificate confirmation.

The generic messages described above can implemented using various protocols implementing authenticated self-contained objects, as described in Section 4. Examples are available in Section 7.

5.1.5. Addressing Scheme Enhancements

BRSKI-AE provides enhancements to the addressing scheme defined in [RFC8995] to accommodate the additional handling of authenticated self-contained objects for the certification request. As this is supported by different enrollment protocols, they can be directly employed (see also Section 7).

The addressing scheme in BRSKI for client certificate request and CA certificate distribution function during the enrollment uses the definition from EST [RFC7030], here on the example on simple enroll: "/.well-known/est/simpleenroll" This approach is generalized to the following notation: "/.well-known/enrollment-protocol/request" in which enrollment-protocol may be an already existing protocol or a newly defined approach. Note that enrollment is considered here as a sequence of at least a certification request and a certification response. In case of existing enrollment protocols the following notation is used proving compatibility to BRSKI:

o enrollment-protocol: references either EST [RFC7030] as in BRSKI or CMP, CMC, SCEP, or newly defined approaches as alternatives. Note: additional endpoints (well-known URI) at the registrar may need to be defined by the utilized enrollment protocol.

o request: depending on the utilized enrollment protocol, the request describes the required operation at the registrar side. Enrollment protocols are expected to define the request endpoints as done by existing protocols (see also Section 7).

5.2. Use Case 2 (pledge-responder-mode): Registrar-agent communication with Pledges

To support mutual trust establishment of pledges, not directly connected to the domain registrar. It 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-AE). This approach has also been applied also for the use case 1. 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 3. 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.

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

Figure 3: Architecture overview using registrar-agent

The architecture overview in Figure 3 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-AE.

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) as in use case 1 Section 5.1.

- 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.

o 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.

o 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.

"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.

5.2.1. Behavior of a pledge in pledge-responder-mode

In contrast to use case 1 Section 5.1 the pledge acts as a server component if data 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 5. 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:

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

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

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

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

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

5.2.2. Behavior of a 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 9.

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-AE, 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-AE 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:

o LDevID(RegAgt): own operational key pair.

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

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

5.2.2.1. Registrar discovery by the 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.

5.2.2.2. Pledge discovery by the 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.

5.2.3. 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 4 provides an overview of the exchanges detailed in the following sub sections.

```
+--------+  +-----------+    +-----------+   +--------+   +---------+
| Pledge |  | Registrar |    | Domain    |   | Domain |   | Vendor  |
|        |  | Agent     |    | Registrar |   | CA     |   | Service |
|        |  | (RegAgt)  |    |  (JRC)    |   |        |   | (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 4: 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.

5.2.3.1. Request objects acquisition (registrar-agent - pledge)

The following description assumes that the registrar-agent already discovered the pledge. This may be done as described in Section 5.2.2.2 based on mDNS.

The focus is on the exchange of signature-wrapped objects using endpoints defined for the pledge in Section 5.2.1.

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.

```
+--------+                             +-----------+
| Pledge |                             | Registrar |
|        |                             | Agent     |
|        |                             | (RegAgt)  |
+--------+                             +-----------+
    |                                        |-create
    |                                        | agent-signed-data
    |<--- trigger pledge-voucher-request ----|
    |-agent-provided-proximity-registrar-cert|
    |-agent-signed-data                      |
    |-agent-sign-cert (optional)             |
    |                                        |
    |----- pledge-voucher-request ---------->|-store
    |                                        | pledge-voucher-request
    |<----- trigger enrollment request ------|
    |       (empty)                          |
    |                                        |
    |------ pledge-enrollment-request ------>|-store
    |                                        | pledge-enrollment-req.
```

Figure 5: Request collection (registrar-agent - pledge)

Triggering the pledge to create the pledge-voucher-request is done using HTTPS 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:

- o agent-provided-proximity-registrar-cert: base64-encoded LDevID(Reg) TLS EE certificate.
- o agent-sign-cert: base64-encoded LDevID(RegAgt) signing certificate (optional).
- o 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 JOSE object and contains the following information:

The header of the agent-signed-data contains:

- o alg: algorithm used for creating the object signature.
- o kid: contains the base64-encoded SubjectKeyIdentifier of the LDevID(RegAgt) certificate.

The body of the agent-signed-data contains an ietf-voucher-request:agent-signed-data element:

[RFC Editor: please delete] /*

Open Issue regarding YANG Definition. Is either definition of ietf-voucher-request:agent-signed-data as new leaf in the existing or ietf-voucher-request-trigger:agent-signed-data as new module or or would it be sufficient to just keep the product-serial-number and the date?*/

- o created-on: MUST contain the creation date and time in yang:date-and-time format.
- o 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-trigger:agent-signed-data": {
    "created-on": "2021-04-16T00:00:01.000Z",
    "serial-number": "callee4711"
  }
}
{
    SIGNATURE
}
```


Figure 6: 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.richardson-anima-jose-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: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:voucher is enhanced with additional parameters:

- o 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).

- o agent-signed-data: MUST contain the base64-encoded agent-signed-data (as defined in Figure 6) and provided as trigger parameter.

- o 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: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 7: Example of pledge-voucher-request

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

application/voucher-jose+json

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

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 HTTPS GET 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.

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 for the CSR as defined in [I-D.ietf-netconf-sztp-csr].

Depending on the capability of the pledge, it MAY construct the enrollment request as plain PKCS#10. Note that the focus here 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 certificate request objects from CMP or CMC.

The pledge SHOULD construct the pledge-enrollment-request as PKCS#10 object and sign it additionally with its IDevID credential. The pledge-enrollment-request should be encoded as JOSE object.

[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 certificate 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]:

o alg: algorithm used for creating the object signature.

o 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-sztp-csr:csr in [I-D.ietf-netconf-sztp-csr]:

o 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-sztp-csr:csr": {
    "p10": "base64encodedvalue=="
  }
}
{
    SIGNATURE
}
```

Figure 8: 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 */

- o 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.

5.2.3.2. 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:

- o 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).

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

- o 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.

```
+-----------+    +-----------+   +--------+   +---------+
| Registrar |    | Domain    |   | Domain |   | Vendor  |
| Agent     |    | Registrar |   | CA     |   | Service |
| (RegAgt)  |    |  (JRC)    |   |        |   | (MASA)  |
+-----------+    +-----------+   +--------+   +---------+
    |                  |              |   Internet |
[exchange between pledge and ]
[registrar-agent done. ]
    |                  |              |            |
    |<------ TLS ----->|              |            |
    |                  |              |            |
    |-- Voucher-Req -->|              |            |
    |          [accept device?]       |            |
    |          [contact vendor]       |            |
    |                  |------------ TLS --------->|
    |                  |-- Voucher-Req ----------->|
    |                  |                   [extract DomainID]
    |                  |                   [update audit log]
    |<---- Voucher ----|<--------- Voucher --------|
    |                  |              |            |
[certification request handling registrar-agent]
[and site infrastructure]
    |--- Enroll-Req -->|              |            |
    |                  |---- TLS ---->|            |
    |                  |- Enroll-Req->|            |
    |                  |<-Enroll-Resp-|            |
    |<-- Enroll-Resp---|              |            |
    |                  |              |            |
```

Figure 9: 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 HTTPS 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.richardson-anima-jose-voucher] as:

application/voucher-jose+json (see Figure 7 for the content definition).

The registrar-agent SHOULD include the "Accept" header field received during the communication with the pledge, indicating the pledge acceptable Content-Type for the voucher-response. The voucher-response Content-Type "application/voucher-jose+json" is defined in [I-D.richardson-anima-jose-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:

- o 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.

- o 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.

- o 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.richardson-anima-jose-voucher].

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

o alg: algorithm used for creating the object signature.

o 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 ietf-voucher-request:voucher as defined in [RFC8995]:

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

o nonce: copied form the pledge-voucher-request

o 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.

o 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 ietf-voucher-request:voucher can be optionally enhanced with the following additional parameter:

o 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: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 10: Example of registrar-voucher-request

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

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

application/voucher-jose+json

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

- o 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.

- o 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-jose+json" as defined in [I-D.richardson-anima-jose-voucher] is applied. The syntactic details of vouchers are described in detail in [RFC8366]. Figure 11 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 11: 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 8.

When using EST, the registrar-agent sends the enrollment request to the registrar with an HTTPS 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.

- o 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.

- o 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.

- o 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:

- o 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".

The registrar-agent has now finished the exchanges with the domain registrar. Now the registrar-agent 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].

5.2.3.3. Response object supply (registrar-agent - 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 5.2.2.2 or use stored information from the first contact with the pledge.

Preconditions in addition to Section 5.2.3.2:

- o Registrar-agent: possesses voucher and LDevID certificate.

```
+--------+                           +-----------+
| Pledge |                           | Registrar |
|        |                           | Agent     |
|        |                           | (RegAgt)  |
+--------+                           +-----------+
    |                                    |
    |<------- supply voucher -----------|
    |                                    |
    |--------- voucher-status --------->| - store
    |                                    |   pledge voucher-status
    |<--- supply enrollment response ---|
    |                                    |
    |--------- enroll-status ---------->| - store
    |                                    |   pledge enroll-status
```

Figure 12: 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-jose+json and contains the voucher as provided by the MASA. An example if given in Figure 11.

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 13. 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 13: 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 14. 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 14: 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.

5.2.3.4. 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 5.2.3.2:

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

```
+-----------+    +-----------+   +--------+   +---------+
| Registrar |    | Domain    |   | Domain |   | Vendor  |
| Agent     |    | Registrar |   | CA     |   | Service |
| RegAgt)   |    |  (JRC)    |   |        |   | (MASA)  |
+-----------+    +-----------+   +--------+   +---------+
    |                  |              |   Internet |
    |                  |              |            |
    |<------ TLS ----->|              |            |
    |                  |              |            |
    |--Voucher-Status->|              |            |
    |                  |<---- device audit log ----|
    |           [verify audit log ]
    |                  |              |            |
    |--Enroll-Status-->|              |            |
    |                  |              |            |
    |                  |              |            |
```

Figure 15: 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 5.2.3.2.

The registrar-agent sends the pledge voucher-status object without modification to the registrar with an HTTPS 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 12 and depicted in the example in Figure 13.

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 HTTPS POST using the operation path value of "/.well-known/brski/enrollstatus". The Content-Type header is kept as "application/jose" as described in Figure 12 and depicted in the example in Figure 14.

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.

5.3. Domain registrar support of different enrollment options

Well-known URIs for different endpoints on the domain registrar are already defined as part of the base BRSKI specification. In addition, alternative enrollment endpoints may be supported at the domain registrar. The pledge / registrar-agent will recognize if its supported enrollment option is supported by the domain registrar by sending a request to its preferred enrollment endpoint.

The following provides an illustrative example for a domain registrar supporting different options for EST as well as CMP to be used in BRSKI-AE. The listing contains the supported endpoints for the bootstrapping, to which the pledge may connect. This includes the voucher handling as well as the enrollment endpoints. The CMP related enrollment endpoints are defined as well-known URI in CMP Updates [I-D.ietf-lamps-cmp-updates].

```
  </brski/voucherrequest>,ct=voucher-cms+json
  </brski/voucher_status>,ct=json
  </brski/enrollstatus>,ct=json
  </est/cacerts>;ct=pkcs7-mime
  </est/simpleenroll>;ct=pkcs7-mime
  </est/simplereenroll>;ct=pkcs7-mime
  </est/fullcmc>;ct=pkcs7-mime
  </est/serverkeygen>;ct= pkcs7-mime
  </est/csrattrs>;ct=pkcs7-mime
  </cmp/initialization>;ct=pkixcmp
  </cmp/certification>;ct=pkixcmp
  </cmp/keyupdate>;ct=pkixcmp
  </cmp/p10>;ct=pkixcmp
  </cmp/getCAcert>;ct=pkixcmp
  </cmp/getCSRparam>;ct=pkixcmp
```

[RFC Editor: please delete] /*

Open Issues:

o In addition to the current content types, we may specify that the response provide information about different content types as multiple values. This would allow to further adopt the encoding of the objects exchanges (ASN.1, JSON, CBOR, ...). -> dependent on the utilized protocol.

*/

6. YANG Extensions to Voucher Request

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

```
module ietf-async-voucher-request {
  yang-version 1.1;

  namespace
    "urn:ietf:params:xml:ns:yang:ietf-async-voucher-request";
  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 ivr;
       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:   Hendrik Brockhaus
                 <mailto: hendrik.brockhaus@siemens.com>
       Author:   Eliot Lear
                 <mailto: lear@cisco.com>";
       Author:   Thomas Werner
                 <mailto: thomas-werner@siemens.com>";
     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 "YYYY-MM-DD" {
       description
        "Initial version";
       reference
        "RFC XXXX: Voucher Request for Asynchronous Enrollment";
     }
     rc:yang-data voucher-request-async-artifact {
       // YANG data template for a voucher.
       uses voucher-request-async-grouping;
     }
     // Grouping defined for future usage
     grouping voucher-request-async-grouping {
       description
```

```
         "Grouping to allow reuse/extensions in future work.";
       uses ivr:voucher-request-grouping {
         augment "voucher-request" {
           description "Base the constrained voucher-request 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";
           }
         }
       }
     }
   }
```

7. Example for signature-wrapping using existing enrollment protocols

This section map the requirements to support proof of possession and proof of identity to selected existing enrollment protocols. Note that that the work in the ACE WG described in [I-D.selander-ace-coap-est-oscore] may be considered here as well, as it also addresses the encapsulation of EST in a way to make it independent from the underlying TLS using OSCORE resulting in an authenticated self-contained object.

7.1. EST Handling

When using EST [RFC7030], the following constraints should be considered:

o Proof of possession is provided by using the specified PKCS#10 structure in the request.

o Proof of identity is achieved by signing the certification request object, which is only supported when Full PKI Request (the /fullcmc endpoint) is used. This contains sufficient information for the RA to make an authorization decision on the received certification request. Note: EST references CMC [RFC5272] for the definition of the Full PKI Request. For proof of identity, the signature of the SignedData of the Full PKI Request would be calculated using the IDevID credential of the pledge.

- [RFC Editor: please delete] /* TBD: in this case the binding to the underlying TLS connection is not be necessary. */

- When the RA is not available, as per [RFC7030] Section 4.2.3, a 202 return code should be returned by the Registrar. The pledge in this case would retry a simpleenroll with a PKCS#10 request. Note that if the TLS connection is teared down for the waiting time, the PKCS#10 request would need to be rebuilt if it contains the unique identifier (tls_unique) from the underlying TLS connection for the binding.

- [RFC Editor: please delete] /* TBD: clarification of retry for fullcmc is necessary as not specified in the context of EST */

7.2. CMP Handling

Instead of using CMP [RFC4210], this specification refers to the lightweight CMP profile [I-D.ietf-lamps-lightweight-cmp-profile], as it restricts the full featured CMP to the functionality needed here. For this, the following constrains should be observed:

- For proof of possession, the defined approach in Lightweight CMP Profile section 4.1.1 (based on CRMF) and 4.1.5 (based on PCKS#10) should be supported.

- Proof of identity can be provided by using the signatures to protect the certificate request message as outlined in section 3.2. of [I-D.ietf-lamps-lightweight-cmp-profile].

- When the RA/CA is not available, a waiting indication should be returned in the PKIStatus by the Registrar. The pledge in this case would retry using the PollReqContent with a request identifier certReqId provided in the initial CertRequest message as specified in section 5.2.4 of [I-D.ietf-lamps-lightweight-cmp-profile] with delayed enrollment.

8. 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

[RFC Editor: please delete] /* to be done: IANA consideration to be included for the defined namespaces in Section 5.1.5 and Section 5.3 . */

9. 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.

10. Security Considerations

10.1. Exhaustion attack on pledge

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

10.2. Misuse of acquired voucher and enrollment responses

Registrar-agent that uses acquired voucher and enrollment response for domain 1 in domain 2: can be detected in Voucher Request processing on domain registrar side. Requires domain registrar to verify the proximity-registrar-cert leaf in the pledge-voucher-request against his own as well as the association of the pledge to his domain based on the product-serial-number contained in the voucher.

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

Misuse of registrar-agent LDevID may be addressed by utilizing short-lived certificates to be used for authenticating the registrar-agent against the 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.

11. Acknowledgments

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

12. References

12.1. Normative References

[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", draft-ietf-netconf-sztp-csr-01 (work in progress), November 2020.

[I-D.richardson-anima-jose-voucher]
Richardson, M. and T. Werner, "JOSE signed Voucher Artifacts for Bootstrapping Protocols", draft-richardson-anima-jose-voucher-00 (work in progress), December 2020.

[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>.

[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>.

[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>.

[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>.

12.2. Informative References

[I-D.ietf-lamps-cmp-updates]
Brockhaus, H. and D. V. Oheimb, "Certificate Management Protocol (CMP) Updates", draft-ietf-lamps-cmp-updates-09 (work in progress), April 2021.

[I-D.ietf-lamps-lightweight-cmp-profile]
Brockhaus, H., Fries, S., and D. V. Oheimb, "Lightweight Certificate Management Protocol (CMP) Profile", draft-ietf-lamps-lightweight-cmp-profile-05 (work in progress), February 2021.

[I-D.selander-ace-coap-est-oscore]
Selander, G., Raza, S., Furuhed, M., Vucinic, M., and T. Claeys, "Protecting EST Payloads with OSCORE", draft-selander-ace-coap-est-oscore-04 (work in progress), November 2020.

[IEC-62351-9]
International Electrotechnical Commission, "IEC 62351 - Power systems management and associated information exchange - Data and communications security - Part 9: Cyber security key management for power system equipment", IEC 62351-9 , May 2017.

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

[ISO-IEC-15118-2]
International Standardization Organization / International Electrotechnical Commission, "ISO/IEC 15118-2 Road vehicles - Vehicle-to-Grid Communication Interface - Part 2: Network and application protocol requirements", ISO/IEC 15118-2 , April 2014.

[NERC-CIP-005-5]
North American Reliability Council, "Cyber Security - Electronic Security Perimeter", CIP 005-5, December 2013.

[OCPP] Open Charge Alliance, "Open Charge Point Protocol 2.0.1 (Draft)", December 2019.

[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>.

[RFC4210] Adams, C., Farrell, S., Kause, T., and T. Mononen, "Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)", RFC 4210, DOI 10.17487/RFC4210, September 2005, <https://www.rfc-editor.org/info/rfc4210>.

[RFC4211] Schaad, J., "Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF)", RFC 4211, DOI 10.17487/RFC4211, September 2005, <https://www.rfc-editor.org/info/rfc4211>.

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

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

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

[RFC8894] Gutmann, P., "Simple Certificate Enrolment Protocol", RFC 8894, DOI 10.17487/RFC8894, September 2020, <https://www.rfc-editor.org/info/rfc8894>.

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

From IETF draft 01 -> IETF 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 5.2.3
.

- 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 5.2.

- 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 5.2.3.

- Details on trust relationship between registrar-agent and pledge (issue #5) included in Section 5.2.

- Split of use case 2 call flow into sub sections in Section 5.2.3.

From IETF draft 00 -> IETF 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 5.2 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)

- o Clarification in discovery options for enrollment endpoints at the domain registrar based on well-known endpoints in Section 5.3 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.

- o Updated references.

- o Included Thomas Werner as additional author for the document.

From individual version 03 -> IETF draft 00:

- o Inclusion of discovery options of enrollment endpoints at the domain registrar based on well-known endpoints in Section 5.3 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.

- o Missing details provided for the description and call flow in pledge-agent use case Section 5.2, e.g. to accommodate distribution of CA certificates.

- o Updated CMP example in Section 7 to use lightweight CMP instead of CMP, as the draft already provides the necessary /.well-known endpoints.

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

- o Removal of copied call flows for voucher exchange and registrar discovery flow from [RFC8995] in Section 5.1 to avoid doubling or text or inconsistencies.

- o 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:

- o 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.

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

o 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-AE call flow.

o Update of provided examples of the addressing approach used in BRSKI to allow for support of multiple enrollment protocols in Section 5.1.5.

From individual version 01 -> 02:

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

o Definition of the addressing approach used in BRSKI to allow for support of multiple enrollment protocols in Section 5.1.5. This section also contains a first discussion of an optional discovery mechanism to address situations in which the registrar supports more than one enrollment approach. Discovery should avoid that the pledge performs a trial and error of enrollment protocols.

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

o Enhanced consideration of existing enrollment protocols in the context of mapping the requirements to existing solutions in Section 4 and in Section 7.

From individual version 00 -> 01:

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

o 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.

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

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

o New section starting Section 7 with the mapping to existing enrollment protocols by collecting boundary conditions.

Authors' Addresses

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

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

Hendrik Brockhaus
Siemens AG
Otto-Hahn-Ring 6
Munich, Bavaria 81739
Germany

Email: hendrik.brockhaus@siemens.com
URI: https://www.siemens.com/

Eliot Lear
Cisco Systems
Richtistrasse 7
Wallisellen CH-8304
Switzerland

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

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

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