anima Working Group   M. Richardson
Internet-Draft   Sandelman Software Works
Intended status: Standards Track   P. van der Stok
Expires: September 11, 2019   vanderstok consultancy
                                                        P. Kampanakis
                                                        Cisco Systems
                                                       March 10, 2019

Constrained Join Proxy for Bootstrapping Protocols
draft-vanderstok-anima-constrained-join-proxy-01

Abstract

   This document defines a protocol to securely assign a pledge to an
   owner, using an intermediary node between pledge and owner. This
   intermediary node is known as a "constrained-join-proxy".

   This document extends the work of [ietf-anima-bootstrapping-keyinfra]
   by replacing the Circuit-proxy by a stateless constrained join-proxy,
   that transports routing information.

Status of This Memo

   This Internet-Draft is submitted in full conformance with the
   provisions of BCP 78 and BCP 79.

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

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

   This Internet-Draft will expire on September 11, 2019.

Table of Contents

1. Introduction

   Enrolment of new nodes into constrained networks with constrained
   nodes present is described in [I-D.ietf-anima-bootstrapping-keyinfra]
   and makes use of Enrolment over Secure Transport (EST) [RFC7030].
   The specified solutions use https and may be too large in terms of
   code space or bandwidth required. Constrained devices in constrained
   networks [RFC7228] typically implement the IPv6 over Low-Power
   Wireless personal Area Networks (6LoWPAN) [RFC4944] and Constrained
   Application Protocol (CoAP) [RFC7252].

CoAP has chosen Datagram Transport Layer Security (DTLS) [RFC6347] as
the preferred security protocol for authenticity and confidentiality
of the messages. A constrained version of EST, using Coap and DTLS,
is described in [I-D.ietf-ace-coap-est].

DTLS is a client-server protocol relying on the underlying IP layer
to perform the routing between the DTLS Client and the DTLS Server.
However, the new "joining" device will not be IP routable until it is
authenticated to the network. A new "joining" device can only
initially use a link-local IPv6 address to communicate with a
neighbour node using neighbour discovery [RFC6775] until it receives
the necessary network configuration parameters. However, before the
device can receive these configuration parameters, it needs to
authenticate itself to the network to which it connects. In
[I-D.ietf-anima-bootstrapping-keyinfra] Enrolment over Secure
Transport (EST) [RFC7030] is used to authenticate the joining device.
However, IPv6 routing is necessary to establish a connection between
joining device and the EST server.

This document specifies a Join-proxy and protocol to act as
intermediary between joining device and EST server to establish a
connection between joining device and EST server.

This document is very much inspired by text published earlier in
[I-D.kumar-dice-dtls-relay].

2. Terminology

The following terms are defined in [RFC8366], and are used
identically as in that document: artifact, imprint, domain, Join
Registrar/Coordinator (JRC), Manufacturer Authorized Signing
Authority (MASA), pledge, Trust of First Use (TOFU), and Voucher.

3. Requirements Language

In this document, the key words "MUST", "MUST NOT", "REQUIRED",
"SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY",
and "OPTIONAL" are to be interpreted as described in BCP 14, RFC 2119
[RFC2119] and indicate requirement levels for compliant STuPiD
implementations.

4. Join Proxy functionality

As depicted in the Figure 1, the joining Device, or pledge (P), is
more than one hop away from the EST server (E) and not yet
authenticated into the network. At this stage, it can only
communicate one-hop to its nearest neighbour, the Join proxy (J)
using their link-local IPv6 addresses. However, the Pledge (P) needs

to communicate with end-to-end security with a Registrar hosting the
EST server (E) to authenticate and get the relevant system/network
parameters. If the Pledge (P) initiates a DTLS connection to the EST
server whose IP address has been pre-configured, then the packets are
dropped at the Join Proxy (J) since the Pledge (P) is not yet
admitted to the network or there is no IP routability to Pledge (P)
for any returned messages.

```
         ++++
         |E |----         +--+          +--+
         |  |    \        |J |........|P |
         ++++     \-----|  |          |  |
      EST server        +--+          +--+
      REgistrar       Join Proxy   PLedge
                                   "Joining" Device
```

                Figure 1: multi-hop enrolment.

Furthermore, the Pledge (P) may wish to establish a secure connection
to the EST server (E) in the network assuming appropriate credentials
are exchanged out-of-band, e.g. a hash of the Pledge (P)'s raw public
key could be provided to the EST server (E). However, the Pledge (P)
is unaware of the IP address of the EST-server (E) to initiate a DTLS
connection and perform authentication with.

A DTLS connection is required between Pledge and EST server. To
overcome the problems with non-routability of DTLS packets and/ or
discovery of the destination address of the EST Server to contact,
the Join Proxy is introduced. This Join-Proxy functionality is
configured into all authenticated devices in the network which may
act as the Join Proxy for newly joining nodes. The Join Proxy allows
for routing of the packets from the Pledge using IP routing to the
intended EST Server.

5. Join Proxy specification

The Join Proxy can operate in two modes:

o Statefull mode

o Stateless mode

In the statefull mode two configuration are envisaged:

o Join Proxy knows EST Server address

o Pledge knows EST Server address

5.1. Statefull Join Proxy

In stateful mode, the joining node forwards the DTLS messages to the
EST Server.

Assume the Pledge knows the adddress of the EST server. The message
is transmitted to the EST Server as if it originated from the joining
node, by replacing the IP address and port of the Pledge to the DTLS
IP address of the proxy and a randomly chosen port. The DTLS message
itself is not modified. Consequently, the Join Proxy must track the
ongoing DTLS connections based on the following 4-tuple stored
locally:

o Pledge link-local IP address (IP_C)

o Pledge source port (p_C)

o EST Server IP address (IP_S)

o EST Server source port (p_R)

The EST Server communicates with the Join Proxy as if it were
communicating with the Pledge, without any modification required to
the DTLS messages. On receiving a DTLS message from the EST Server,
the Join Proxy looks up its locally stored 4-tuple array to identify
to which Pledge (if multiple exist) the message belongs. The DTLS
message's destination address and port are replaced with the link-
local address and port of the corresponding Pledge and the DTLS
message is then forwarded to the Pledge. The Join Proxy does not
modify the DTLS packets and therefore the normal processing and
security of DTLS is unaffected.

In Figure 2 the various steps of the process are shown where the EST
Server address in known to the Pledge:

```
+------------+-----------+------------+-------------------------+
| EST Client | Join-Proxy |  EST Server |          Message        |
|    (P)     |     (J)    |      (E)    | Src_IP:port | Dst_IP:port|
+------------+-----------+------------+------------+------------+
|      --ClientHello-->               |   IP_C:p_C  | IP_S:5684  |
|                --ClientHello-->     |   IP_R:p_R  | IP_S:5684  |
|                                     |             |            |
|                <--ServerHello--     |   IP_S:5684 | IP_R:p_R   |
|                       :             |             |            |
|      <--ServerHello--       :       |   IP_S:5684 | IP_C:p_C   |
|             :               :       |             |            |
```

```
|            :            :      |      :    |    :       |
|            :            :      |      :    |    :       |
|      --Finished-->             | IP_C:p_C  | IP_S:5684  |
|                 --Finished-->  | IP_R:p_R  | IP_S:5684  |
|                                |           |            |
|                 <--Finished--  | IP_S:5684 | IP_R:p_R   |
|      <--Finished---            | IP_S:5684 | IP_C:p_C   |
|            :            :      |      :    |    :       |
+-------------------------------+-----------+------------+
IP_C:p_C = Link-local IP address and port of EST Client
IP_S:5684 = IP address and coaps port of EST Server
IP_R:p_R = IP address and port of Join Proxy
```

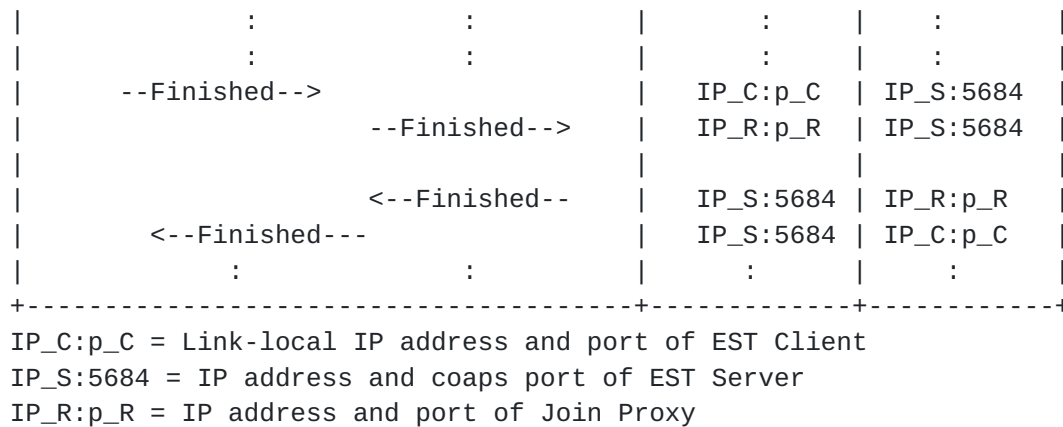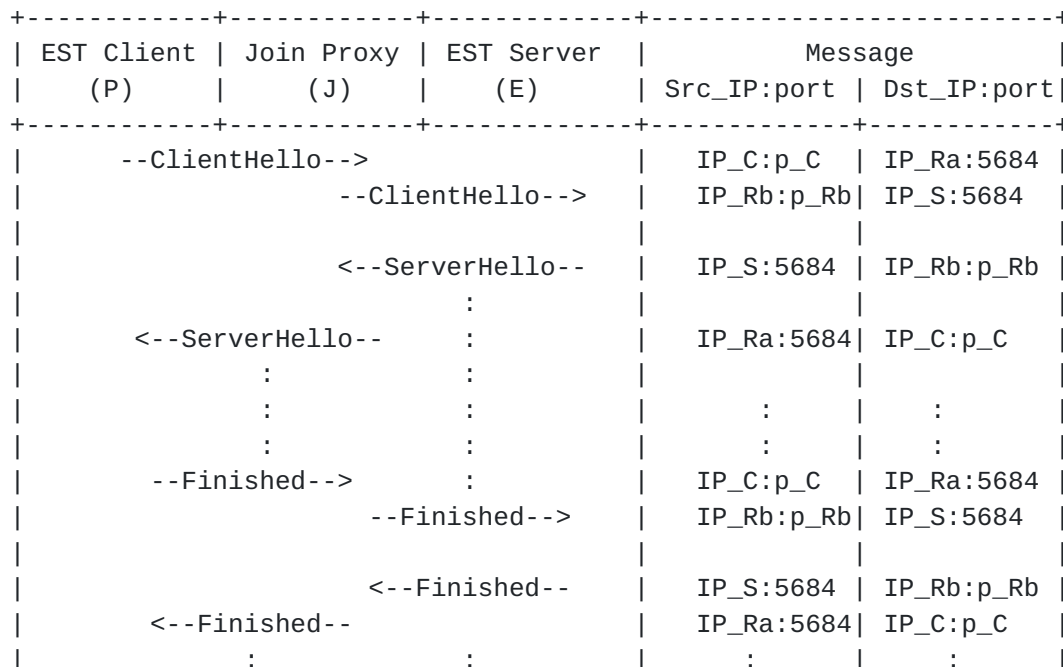   Figure 2: constrained statefull joining message flow with EST server
                             address known to Join Proxy.

   Assume that the pledge does not know the IP address of the EST Server
   it needs to contact. In that situation, the Join Proxy can be
   configured with the IP address of a default EST Server that an EST
   client needs to contact. The EST client initiates its request as if
   the Join Proxy is the intended EST Server. The Join Proxy changes
   the IP packet (without modifying the DTLS message) as in the previous
   case by modifying both the source and destination addresses to
   forward the message to the intended EST Server. The Join Proxy keeps
   a similar 4-tuple array to enable translation of the DTLS messages
   received from the EST Server and forwards it to the EST Client. In
   Figure 3 the various steps of the message flow are shown:

```
+-----------+-----------+------------+-------------------------+
| EST Client | Join Proxy | EST Server |          Message        |
|    (P)     |    (J)     |    (E)     | Src_IP:port | Dst_IP:port|
+-----------+-----------+------------+------------+-----------+
|      --ClientHello-->             |  IP_C:p_C  | IP_Ra:5684 |
|                 --ClientHello-->  | IP_Rb:p_Rb | IP_S:5684  |
|                                   |            |            |
|                 <--ServerHello--  | IP_S:5684  | IP_Rb:p_Rb |
|                      :            |            |            |
|      <--ServerHello--      :      | IP_Ra:5684 | IP_C:p_C   |
|            :            :          |            |            |
|            :            :          |      :     |    :       |
|            :            :          |      :     |    :       |
|      --Finished-->       :         |  IP_C:p_C  | IP_Ra:5684 |
|                 --Finished-->      | IP_Rb:p_Rb | IP_S:5684  |
|                                    |            |            |
|                 <--Finished--      | IP_S:5684  | IP_Rb:p_Rb |
|      <--Finished--                 | IP_Ra:5684 | IP_C:p_C   |
|            :            :          |      :     |    :       |
```

```
+---------------------------------------+------------+-----------+
```

IP_C:p_C = Link-local IP address and port of DTLS Client
IP_S:5684 = IP address and coaps port of DTLS Server
IP_Ra:5684 = Link-local IP address and coaps port of DTLS Relay
IP_Rb:p_Rb = IP address (can be same as IP_Ra) and port of DTLS Relay

Figure 3: constrained statefull joining message flow with EST server
                  address known to Join Proxy.

5.2. Stateless Join Proxy

The Join-proxy is stateless to minimize the requirements on the
constrained Join-proxy device.

When a joining device as a client attempts a DTLS connection to the
EST server, it uses its link-local IP address as its IP source
address. This message is transmitted one-hop to a neighbour node.
Under normal circumstances, this message would be dropped at the
neighbour node since the joining device is not yet IP routable or it
is not yet authenticated to send messages through the network.
However, if the neighbour device has the Join Proxy functionality
enabled, it routes the DTLS message to a specific EST Server.
Additional security mechanisms need to exist to prevent this routing
functionality being used by rogue nodes to bypass any network
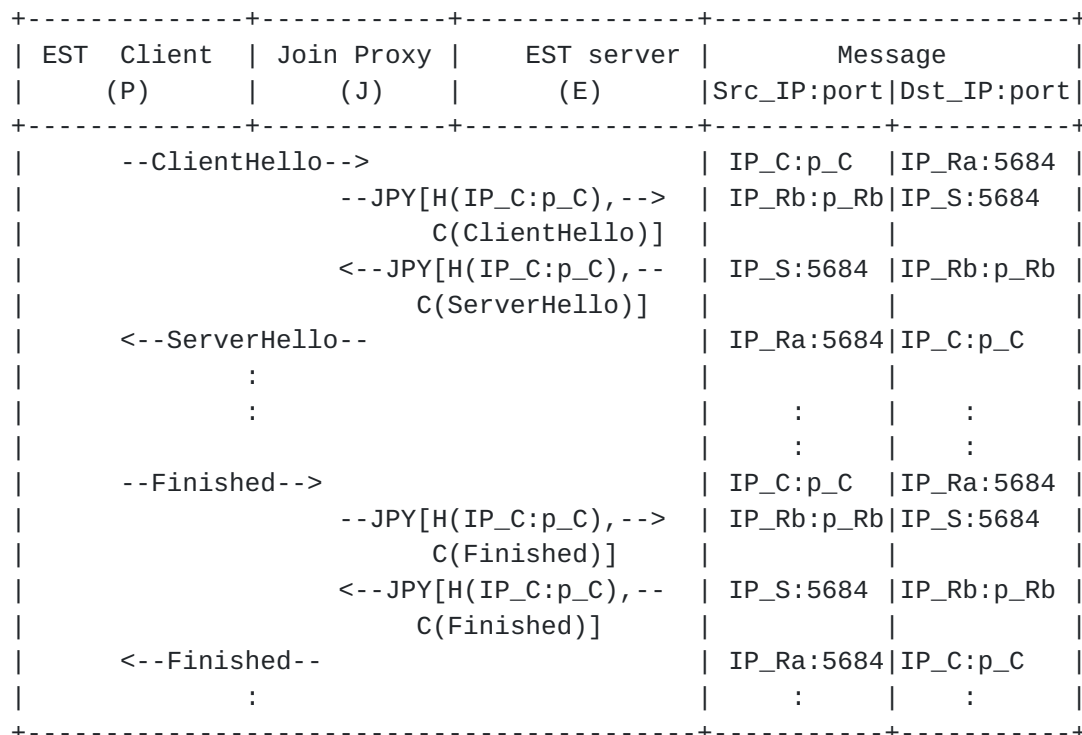authentication procedures.

If an untrusted DTLS Client that can only use link-local addressing
wants to contact a trusted end-point EST Server, it sends the DTLS
message to the Join Proxy. The Join Proxy extends this message into
a new type of message called Join ProxY (JPY) message and sends it on
to the EST server. The JPY message payload consists of two parts:

o Header (H) field: consisting of the source link-local address and
   port of the Pledge (P), and

o Contents (C) field: containing the original DTLS message.

On receiving the JPY message, the EST Server retrieves the two parts.
The EST Server transiently stores the Header field information. The
EST server uses the Contents field to execute the EST server
functionality. However, when the EST Server replies, it also extends
its DTLS message with the header field in a JPY message and sends it
back to the Join Proxy. The Header contains the original source
link-local address and port of the DTLS Client from the transient
state stored earlier (which can now be discarded) and the Contents
field contains the DTLS message.

On receiving the JPY message, the Join Proxy retrieves the two parts.
It uses the Header field to route the DTLS message retrieved from the
Contents field to the Pledge.

The Figure 4 depicts the message flow diagram when the EST Server
end-point address is known only to the Join Proxy:

```
+--------------+-----------+--------------+----------------------+
| EST  Client  | Join Proxy |   EST server |        Message       |
|      (P)     |     (J)    |      (E)     |Src_IP:port|Dst_IP:port|
+--------------+-----------+--------------+----------+-----------+
|      --ClientHello-->                   | IP_C:p_C  |IP_Ra:5684 |
|                   --JPY[H(IP_C:p_C),--> | IP_Rb:p_Rb|IP_S:5684  |
|                        C(ClientHello)]  |           |           |
|                   <--JPY[H(IP_C:p_C),-- | IP_S:5684 |IP_Rb:p_Rb |
|                        C(ServerHello)]  |           |           |
|      <--ServerHello--                   | IP_Ra:5684|IP_C:p_C   |
|              :                          |           |           |
|              :                          |     :     |     :     |
|                                         |     :     |     :     |
|      --Finished-->                      | IP_C:p_C  |IP_Ra:5684 |
|                   --JPY[H(IP_C:p_C),--> | IP_Rb:p_Rb|IP_S:5684  |
|                        C(Finished)]     |           |           |
|                   <--JPY[H(IP_C:p_C),-- | IP_S:5684 |IP_Rb:p_Rb |
|                        C(Finished)]     |           |           |
|      <--Finished--                      | IP_Ra:5684|IP_C:p_C   |
|              :                          |     :     |     :     |
+-----------------------------------------+----------+-----------+
```
IP_C:p_C = Link-local IP address and port of the Pledge
IP_S:5684 = IP address and coaps port of EST Server
IP_Ra:5684 = Link-local IP address and coaps port of Join Proxy
IP_Rb:p_Rb = IP address(can be same as IP_Ra) and port of Join Proxy

JPY[H(),C()] = Join ProxY message with header H and content C

         Figure 4: constrained stateless joining message flow.

5.3. Stateless Message structure

The JPY message is constructed as a payload with media-type
application/multipart-core specified in [I-D.ietf-core-multipart-ct].
Header and Contents fields use different media formats:

1. header field: application/CBOR containing a CBOR array [RFC7049]
    with the pledge IPv6 Link Local address as a 16-byte binary
    value, the pledge's UDP port number, if different from 5684, as a
    CBOR integer, and the proxy's ifindex or other identifier for the

physical port on which the pledge is connected. Header is not
DTLS encrypted.

2. Content field: Any of the media types specified in
    [I-D.ietf-ace-coap-est] and [I-D.ietf-anima-constrained-voucher]
    dependent on the function that is requested:

* application/pkcs7-mime; smime-type=server-generated-key
* application/pkcs7-mime; smime-type=certs-only
* application/voucher-cms+cbor
* application/voucher-cose+cbor
* application/pkcs8
* application/csrattrs
* application/pkcs10
* application/pkix-cert

Examples are shown in Appendix A. The content fields are DTLS
encrypted.

6. Comparison of stateless and statefull modes

The stateful and stateless mode of operation for the Join Proxy have
their advantages and disadvantages. This section should enable to
make a choice between the two modes based on the available device
resources and network bandwidth.

```
+-------------+---------------------------+-----------------------+
| Properties  |        Stateful mode      |     Stateless mode    |
+-------------+---------------------------+-----------------------+
| State       |The Proxy needs additional | No information is      |
| Information |storage to maintain mapping | maintained by the Join |
|             |of the Pledge's address    | Proxy                 |
|             | with the port number      |                       |
|             |being used to communicate  |                       |
|             |with the Server.           |                       |
+-------------+---------------------------+-----------------------+
|Packet size  |The size of the forwarded  |Size of the forwarded  |
|             |message is the same as the |message is bigger than |
|             | original message.         |the original,it includes|
|             |                           |additional source and  |
|             |                           |destination addresses. |
+-------------+---------------------------+-----------------------+
|Specification|The additional functionality|New JPY message to     |
|complexity   |the Proxy to maintain state |encapsulate DTLS message|
|             |information, and modify    |The Server and the proxy|
|             |the source and destination |have to understand the |
|             |addresses of the DTLS      |JPY message in order   |
|             |handshake messages         |to process it.         |
```

```
   +-------------+--------------------------+-----------------------+
```

             Figure 5: Comparison between stateful and stateless mode

7. Discovery

   It is assumed that Join-Proxy seamlessly provides a coaps connection
   between Pledge and coaps EST-server. An additional Registrar is
   needed to connect the Pledge to an http EST server, see section 8 of
   [I-D.ietf-ace-coap-est].

   The Discovery of the coaps EST server by the Join Proxy follows
   section 6 of [I-D.ietf-ace-coap-est]. The discovery of the Join-
   Proxy by the Pledge is an extension to the discovery described in
   section 4 of [I-D.ietf-anima-bootstrapping-keyinfra]. In particular
   this section replaces section 4.2 of
   [I-D.ietf-anima-bootstrapping-keyinfra]. Three discovery cases are
   discussed: coap discovery, 6tisch discovery and GRASP discovery.

7.1. GRASP discovery

   In the context of autonomous networks, discovery takes place via the
   GRASP protocol as described in
   [I-D.ietf-anima-bootstrapping-keyinfra]. The port number is.

   EDNote: to be specified further

7.2. 6tisch discovery

   The discovery of EST server by the pledge uses the enhanced beacons
   as discussed in [I-D.ietf-6tisch-enrollment-enhanced-beacon].

7.3. Coaps discovery

   In the context of a coap network without Autonomous Network support,
   discovery follows the standard coap policy. The Pledge can discover
   a Join-Proxy by sending a link-local multicast message to ALL CoAP
   Nodes with address FF02::FD. Multiple or no nodes may respond. The
   handling of multiple responses and the absence of responses follow
   section 4 of [I-D.ietf-anima-bootstrapping-keyinfra].

   The presence and location of (path to) the join-proxy resource are
   discovered by sending a GET request to "/.well-known/core" including
   a resource type (rt) parameter with the value "brski-proxy"
   [RFC6690]. Upon success, the return payload will contain the root
   resource of the Join-Proxy resources. It is up to the implementation
   to choose its root resource; throughout this document the example
   root resource /est is used. The example below shows the discovery of
   the presence and location of join-proxy resources.

     REQ: GET coap://[FF02::FD]/.well-known/core?rt=brski-proxy

     RES: 2.05 Content
     </est>; rt="brski-proxy";ct=62

   Port numbers, not returned in the example, are assumed to be the
   default numbers 5683 and 5684 for coap and coaps respectively
   (sections 12.6 and 12.7 of [RFC7252]. Discoverable port numbers MAY
   be returned in the <href> of the payload.

8. Security Considerations

   It should be noted here that the contents of the CBOR map are not
   protected, but that the communication is between the Proxy and a
   known registrar (a connected UDP socket), and that messages from
   other origins are ignored.

9. IANA Considerations

   This document needs to create a registry for key indices in the CBOR
   map. It should be given a name, and the amending formula should be
   IETF Specification.

9.1. Resource Type registry

   This specification registers a new Resource Type (rt=) Link Target
   Attributes in the "Resource Type (rt=) Link Target Attribute Values"
   subregistry under the "Constrained RESTful Environments (CoRE)
   Parameters" registry.

   rt="brski-proxy". This EST resource is used to query and return
   the supported EST resource of a join-proxy placed between Pledge
   and EST server.

10. Acknowledgements

Many thanks for the comments by Brian Carpenter.

11. Contributors

Sandeep Kumar, Sye loong Keoh, and Oscar Garcia-Morchon are the co-
authors of the draft-kumar-dice-dtls-relay-02. Their draft has
served as a basis for this document. Much text from their draft is
copied over to this draft.

12. Changelog

12.1. 00 to 01

o Added Contributors section

o Adapted content-formats to est-coaps formats

o Aligned examples with est-coaps examples

o Added statefull Proxy to stateless proxy

12.2. 00 to 00

o added payload examples in appendix

o discovery for three cases: AN, 6tisch and coaps

13. References

13.1. Normative References

[I-D.ietf-6tisch-enrollment-enhanced-beacon]
          Dujovne, D. and M. Richardson, "IEEE802.15.4 Informational
          Element encapsulation of 6tisch Join and Enrollment
          Information", draft-ietf-6tisch-enrollment-enhanced-
          beacon-00 (work in progress), July 2018.

[I-D.ietf-ace-coap-est]
          Stok, P., Kampanakis, P., Kumar, S., Richardson, M.,
          Furuhed, M., and S. Raza, "EST over secure CoAP (EST-
          coaps)", draft-ietf-ace-coap-est-00 (work in progress),
          February 2018.

[I-D.ietf-anima-bootstrapping-keyinfra]
          Pritikin, M., Richardson, M., Behringer, M., Bjarnason,
          S., and K. Watsen, "Bootstrapping Remote Secure Key

Infrastructures (BRSKI)", draft-ietf-anima-bootstrapping-keyinfra-15 (work in progress), April 2018.

[I-D.ietf-anima-constrained-voucher]
          Richardson, M., Stok, P., and P. Kampanakis, "Constrained
          Voucher Artifacts for Bootstrapping Protocols", draft-
          ietf-anima-constrained-voucher-02 (work in progress),
          September 2018.

[I-D.ietf-core-multipart-ct]
          Fossati, T., Hartke, K., and C. Bormann, "Multipart
          Content-Format for CoAP", draft-ietf-core-multipart-ct-02
          (work in progress), August 2018.

[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>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer
          Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347,
          January 2012, <https://www.rfc-editor.org/info/rfc6347>.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object
          Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049,
          October 2013, <https://www.rfc-editor.org/info/rfc7049>.

[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>.

13.2. Informative References

[I-D.kumar-dice-dtls-relay]
          Kumar, S., Keoh, S., and O. Garcia-Morchon, "DTLS Relay
          for Constrained Environments", draft-kumar-dice-dtls-
          relay-02 (work in progress), October 2014.

[RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler,
          "Transmission of IPv6 Packets over IEEE 802.15.4
          Networks", RFC 4944, DOI 10.17487/RFC4944, September 2007,
          <https://www.rfc-editor.org/info/rfc4944>.

[RFC6690] Shelby, Z., "Constrained RESTful Environments (CoRE) Link
          Format", RFC 6690, DOI 10.17487/RFC6690, August 2012,
          <https://www.rfc-editor.org/info/rfc6690>.

   [RFC6775] Shelby, Z., Ed., Chakrabarti, S., Nordmark, E., and C.
              Bormann, "Neighbor Discovery Optimization for IPv6 over
              Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC
              6775, DOI 10.17487/RFC6775, November 2012, <https://www
              .rfc-editor.org/info/rfc6775>.

   [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>.

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

   [RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained
              Application Protocol (CoAP)", RFC 7252, DOI 10.17487/
              RFC7252, June 2014, <https://www.rfc-editor.org/info/
              rfc7252>.

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

   [pledge]   Dictionary.com, ., "Dictionary.com Unabridged", 2015,
              <http://dictionary.reference.com/browse/pledge>.

Appendix A. Stateless Proxy payload examples

   Examples are extensions of two examples shown in
   [I-D.ietf-ace-coap-est].

   EDNote:
   provisional stake holder examples to be improved and corrected.

A.1. cacerts

   The request from Join-Proxy to EST-server looks like:

```
   Get coaps://192.0.2.1/est/crts
   (Accept: 62)
   (Content-format: 62)
   payload =
   82                    # array(2)
   18 3C                 # unsigned(60)
```

```
83                      # array(3)
69                      # text(9)
    464538303A3A414238 # "FE80::AB8"
19 237D                 # unsigned(9085)
65                      # text(5)
    6964656E74          # "ident"
```

The response will then be

```
2.05 Content
(Content-format: 62)
  Payload =
83                                  # array(3)
18 3C                               # unsigned(60)
83                                  # array(3)
69                                  # text(9)
    464538303A3A414238              # "FE80::AB8"
19 237D                             # unsigned(9085)
65                                  # text(5)
    6964656E74                      # "ident"
82                                  # array(2)
19 0119                             # unsigned(281)
59 027F                             # bytes(639)
3082027b06092a864886f70d010702a082026c308202680201013100300b
06092a864886f70d010701a082024e3082024a308201f0a0030201020209
009189bcdf9c99244b300a06082a8648ce3d0403023067310b3009060355
040613025553310b300906035504080c024341310b300906035504070c02
4c4131143012060355040a0c0b4578616d706c6520496e63311630140603
55040b0c0d636572746966696361746f6e3110300e06035504030c0752
6f6f74204341301e170d313930313037313034303415a170d3339303130
323130343034315a3067310b300906035504061302555331 0b3009060355
04080c024341310b300906035504070c024c4131143012060355040a0c0b
4578616d706c6520496e633116301406035504 0b0c0d6365727469666963
6174696f6e3110300e06035504030c07526f6f742043413059301306072a
8648ce3d020106082a8648ce3d030107034200048149 94082b6e8185f3df
53f5e0bee698973335200023ddf78cd17a443ffd8ddd40908769c55652ac
2ccb75c4a50a7c7ddb7c22dae6c85cca538209fdbbf104c9a38184308181
301d0603551d0e041604142495e816ef6ffcaaf356ce4adffe33cf492abb
a8301f0603551d230418301680142495e816ef6ffcaaf356ce4adffe33cf
492abba8300f0603551d130101ff040530030101ff300e0603551d0f0101
ff040403020106301e0603551d1104173015811363657274696667940 6578
616d706c652e636f6d300a06082a8648ce3d0403020348003045022100da
e37c96f154c32ec0b4af52d46f3b7ecc9687ddf267bcec368f7b7f135327
2f022047a28ae5c7306163b3c3834bab3c103f743070594c089aaa0ac870
cd13b902caa1003100
]
```

A.2. **serverkeygen**

The request from Join-Proxy to EST-server looks like:

```
Get coaps://192.0.2.1/est/skg
(Accept: 62)
(Content-Format: 62)
  Payload =
83                                  # array(3)
18 3C                               # unsigned(60)
83                                  # array(3)
69                                  # text(9)
    464538303A3A414238              # "FE80::AB8"
19 237D                             # unsigned(9085)
65                                  # text(5)
    6964656E74                      # "ident"
82                                  # array(2)
19 011E                             # unsigned(286)
58 D2                               # bytes(210)
3081cf307802010030163114301206035504 0a0c0b736b67206578616d70
6c653059301306072a8648ce3d020106082a 8648ce3d030107034200041b
b8c1117896f98e4506c03d70efbe820d8e38ea97e9d65d52c8460c5852c5
1dd89a61370a2843760fc859799d78cd33f3c1846e304f1717f8123f1a28
4cc99fa000300a06082a8648ce3d040302034700304402203 87cd4e9cf62
8d4af77f92ebed4890d9d141dca86cd2757dd14cbd59cdf6961802202f24
5e828c77754378b66660a4977f113cacdaa0cc7bad7d1474a7fd155d090d
```

The response will then be

```
2.05 Content
(Content-format: 62)
  Payload =
84                                  # array(4)
18 3C                               # unsigned(60)
83                                  # array(3)
69                                  # text(9)
    464538303A3A414238              # "FE80::AB8"
19 237D                             # unsigned(9085)
65                                  # text(5)
    6964656E74                      # "ident"
82                                  # array(2)
19 011E                             # unsigned(286)
58 8A                               # bytes(138)
308187020100301306072a8648ce3d020106082a8648ce3d030107046d30
6b02010104200b9a67785b65e07360b6d28cfc1d3f3925c0755799deeca7
45372b01697bd8a6a144034200041bb8c1117896f98e4506c03d70efbe82
0d8e38ea97e9d65d52c8460c5852c51dd89a61370a2843760fc859799d78
cd33f3c1846e304f1717f8123f1a284cc99f
```

```
19 0119                                # unsigned(281)
59 01D3                                # bytes(467)
308201cf06092a864886f70d010702a08201c0308201bc0201013100300b
06092a864886f70d010701a08201a23082019e30820143a0030201020208
126de8571518524b300a06082a8648ce3d0403023016311430120603550ce
0a0c0b736b67206578616d706c65301e170d31393031303930383537303038
5a170d3339303130343038353730385a30163114301206035504a0a0c0b73
6b67206578616d706c653059301306072a8648ce3d020106082a8648ce3d
030107034200041bb8c1117896f98e4506c03d70efbe820d8e38ea97e9d6
5d52c8460c5852c51dd89a61370a2843760fc859799d78cd33f3c1846e30
4f1717f8123f1a284cc99fa37b307930090603551d1304023000302c0609
6086480186f842010d041f161d4f70656e53534c2047656e657261746564
204365727469666963617465301d0603551d0e04160414494be598dc8dbc
0dbc071c486b777460e5cce621301f0603551d23041830168014494be598
dc8dbc0dbc071c486b777460e5cce621300a06082a8648ce3d0403020349
003046022100a4b167d0f9add9202810e6bf6a290b8cfdfc9b9c9fea2cc1
c8fc3a464f79f2c202210081d31ba142751a7b4a34fd1a01fcfb08716b9e
b53bdaadc9ae60b08f52429c0fa1003100
```

Authors' Addresses

Michael Richardson
Sandelman Software Works

Email: mcr+ietf@sandelman.ca

Peter van der Stok
vanderstok consultancy

Email: consultancy@vanderstok.org

Panos Kampanakis
Cisco Systems

Email: pkampana@cisco.com