

ACE
Internet-Draft
Intended status: Standards Track
Expires: December 14, 2017

S. Kumar
Philips Lighting Research
P. van der Stok
Consultant
P. Kampanakis
Cisco Systems
M. Furuhez
Nexus Group
S. Raza
RISE SICS
June 12, 2017

EST over secure CoAP (EST-coaps)
draft-vanderstok-ace-coap-est-02

Abstract

Low-resource devices in a Low-power and Lossy Network (LLN) can operate in a mesh network using the IPv6 over Low-power Wireless Personal Area Networks (6LoWPAN) and IEEE 802.15.4 link-layer standards. Provisioning these devices in a secure manner with keys (often called secure bootstrapping) used to encrypt and authenticate messages, is the subject of Bootstrapping of Remote Secure Key Infrastructures (BRSKI) [I-D.ietf-anima-bootstrapping-keyinfra] and 6tisch Secure Join [I-D.ietf-6tisch-dtsecurity-secure-join]. Enrollment over Secure Transport (EST) [RFC7030], based on TLS and HTTP, is used in BRSKI. Low-resource devices often use the lightweight Constrained Application Protocol (CoAP) [RFC7252] for message exchanges. This document defines how low-resource devices are expected to use EST over secure CoAP (EST-coaps) for secure bootstrapping and certificate enrollment. 6LoWPAN fragmentation management and extensions to CoAP registries are needed to enable EST-coaps.

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 December 14, 2017.

Copyright Notice

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

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

Table of Contents

1. Introduction	3
1.1. Terminology	4
2. EST operational differences	5
3. Conformance to RFC7925 profiles	5
4. Protocol Design and Layering	6
4.1. Discovery and URI	7
4.2. Payload format	9
4.3. Message Bindings	9
4.4. CoAP response codes	9
4.5. Message fragmentation	10
5. Transport Protocol	11
5.1. DTLS	11
5.2. 6tisch approach	12
6. Proxying	13
7. Parameters	14
8. IANA Considerations	14
8.1. Content-Format registry	15
8.2. Resource Type registry	18
9. Security Considerations	18
9.1. proxy considerations	18
9.2. EST server considerations	18
10. Acknowledgements	19
11. Change Log	20
12. References	20
12.1. Normative References	20
12.2. Informative References	22

Appendix A. EST messages to EST-coaps	24
A.1. cacerts	24
A.2. csrattrs	26
A.3. enroll / reenroll	27
A.4. serverkeygen	30
A.5. enrollstatus	33
A.6. voucher_status	33
A.7. requestvoucher	33
Appendix B. EST-coaps Block message examples	33
Authors' Addresses	36

1. Introduction

IPv6 over Low-power Wireless Personal Area Networks (6LoWPANs) [RFC4944] on IEEE 802.15.4 [ieee802.15.4] wireless networks is becoming common in many industry application domains such as lighting controls. However, commissioning of such networks suffers from a lack of standardized secure bootstrapping mechanisms for these networks.

Although IEEE 802.15.4 defines how security can be enabled between nodes within a single mesh network, it does not specify the provisioning and management of the keys. Therefore, securing a 6LoWPAN network with devices from multiple manufacturers with different provisioning techniques is often tedious and time consuming.

Bootstrapping of Remote Secure Infrastructures (BRSKI) [I-D.ietf-anima-bootstrapping-keyinfra] addresses the issue of bootstrapping networked devices in the context of Autonomic Networking Integrated Model and Approach (ANIMA). [I-D.ietf-6tisch-minimal-security] and [I-D.ietf-6tisch-dtsecurity-secure-join] also address secure bootstrapping in the 6tisch context targeted to low-resource devices. BRSKI has not been developed specifically for low-resource devices in constrained networks. Constrained networks use DTLS [RFC6347], CoAP [RFC7252], and UDP instead of TLS [RFC5246], HTTP [RFC7230] and TCP.

BRSKI relies on Enrollment over Secure Transport (EST) [RFC7030] for the provisioning of the operational domain certificates. EST-coaps provides a subset of EST functionality and extends EST with BRSKI functions. EST-coaps replaces the invocations of TLS and HTTP by DTLS and CoAP invocations thus enabling EST and BRSKI for CoAP-based low-resource devices.

Although EST-coaps paves the way for the utilization of EST for constrained devices on constrained networks, some devices will not have enough resources to handle the large payloads that come with

EST-coaps. The specification of EST-coaps is intended to ensure that bootstrapping works for less constrained devices that choose to limit their communications stack to UDP/CoAP. It is up to the network designer to decide which devices execute the EST protocol and which not.

EST-coaps is designed for use in professional control networks such as Building Control. The autonomic bootstrapping is interesting because it reduces the manual intervention during the commissioning of the network. Typing in passwords is contrary to this wish. Therefore, the HTTP Basic authentication of EST is not supported in EST-coaps.

In the constrained devices context, it is very unlikely that full PKI request messages will be used. Therefore, full PKI request messages are not supported by EST-coaps.

Because the relatively large EST messages cannot be readily transported over constrained (6LoWPAN, LLN) wireless networks, this document specifies the use of CoAP Block-Wise Transfer ("Block") [RFC7959] to fragment EST messages at the application layer.

Support for Observe CoAP options [RFC7641] with BRSKI is not supported in the current BRSKI/EST message flows and is thus out-of-scope for this discussion. Observe options could be used by the server to notify clients about a change in the cacerts or csr attributes (resources) and might be an area of future work.

1.1. 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 [RFC2119].

Many of the concepts in this document are taken over from [RFC7030]. Consequently, much text is directly traceable to [RFC7030]. The same document structure is followed to point out the differences and commonalities between EST and EST-coaps.

The following terms are defined in the BRSKI protocol [I-D.ietf-anima-bootstrapping-keyinfra]: pledge, Join proxy (or Circuit Proxy?), Join Registrar, and Manufacturer Authorized Signing Authorities (MASA).

2. EST operational differences

Only the differences to EST with respect to operational scenarios are described in this section. EST-coaps server differs from EST server as follows:

- o Replacement of TLS by DTLS and HTTP by CoAP, resulting in:
 - * DTLS-secured CoAP sessions between EST-coaps client and EST-coaps server.
- o Only certificate-based client authentication is supported, which results in:
 - * The EST-coaps client does not support HTTP Basic authentication (as described in Section 3.2.3 of [RFC7030]).
 - * The EST-coaps client does not support authentication at the application layer (as described in Section 3.2.3 of [RFC7030]).
- o EST-coaps does not support full PKI request messages[RFC5272].
 - * Consequently, the fullcmc request of section 4.3 of [RFC7030] and response MUST NOT be supported by EST-coaps].
- o EST-coaps specifies the BRSKI extensions over CoAP as specified in sections 3.2, 3.4, 3.5, and 3.8.4 of [I-D.ietf-anima-bootstrapping-keyinfra].

3. Conformance to RFC7925 profiles

This section shows how EST-coaps fits into the profiles of low-resource devices as described in [RFC7925]. Within the bootstrap context a Public Key Infrastructure (PKI) is used, where the client is called "pledge", the Registration Authority (RA) is called Join Registrar, which acts at the front-end for the Certificate Authority (CA) and receives voucher feedback from as many Manufacturer Authorized Signing Authorities (MASA) as there are manufacturers. A Join Proxy (Circuit Proxy?) is placed between client and RA to receive join requests over a 1-hop unsecured channel and transmitted over the secure network to the EST-server. The EST-server of EST-coaps is placed between Join-Proxy (Circuit Proxy) and RA or is part of RA.

EST-coaps can transport certificates and private keys. Private keys can be transported as response to a request to a server-side key generation as described in section 4.4 of [RFC7030]. In the bootstrapping context, EST-coaps transport is limited to the EST certificate transport conformant to section 4.4 of [RFC7925]. For

BRSKI, outside the profiles of [RFC7925], EST-coaps transports vouchers, which are YANG files specified in [I-D.ietf-anima-voucher].

The mandatory cipher suite for DTLS is TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8 defined in [RFC7251] which is the mandatory-to-implement cipher suite in CoAP. Additionally, the curve secp256r1 MUST be supported [RFC4492]; this curve is equivalent to the NIST P-256 curve. The hash algorithm is SHA-256. DTLS implementations MUST use the Supported Elliptic Curves and Supported Point Formats Extensions [RFC4492]; the uncompressed point format MUST be supported; [RFC6090] can be used as an implementation method.

The EST-coaps client MUST be configured with an explicit TA database or at least an implicit TA database from its manufacturer. The authentication of the EST-coaps server by the EST-coaps client is based on Certificate authentication in the DTLS handshake.

The authentication of the EST-coaps client is based on client certificate in the DTLS handshake. This can either be

- o DTLS with a previously issued client certificate (e.g., an existing certificate issued by the EST CA); this could be a common case for simple re-enrollment of clients;
- o DTLS with a previously installed certificate (e.g., manufacturer-installed certificate or a certificate issued by some other party);

4. Protocol Design and Layering

EST-coaps uses CoAP to transfer EST messages, aided by Block-Wise Transfer [RFC7959] to transport CoAP messages in blocks thus avoiding (excessive) 6LoWPAN fragmentation of UDP datagrams. The use of "Block" for the transfer of larger EST messages is specified in Section 4.5. The Figure 1 below shows the layered EST-coaps architecture.

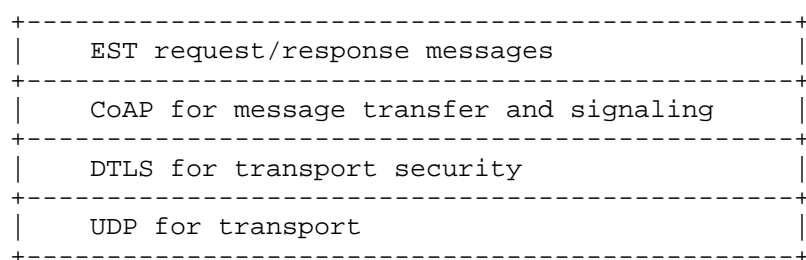


Figure 1: EST-coaps protocol layers

The EST-coaps protocol design follows closely the EST design, excluding some aspects that are not relevant for automatic bootstrapping of constrained devices within a professional context. The parts supported by EST-coaps are identified by their message types:

- o Simple enroll and reenroll, for CA to sign public client-identity key.
- o CA certificate retrieval, needed to receive the complete set of CA certificates.
- o CSR Attributes request messages, informs the pledge of the fields to include in generated CSR.
- o Server-side key generation messages, to provide a private client-identity key when the client is too restricted or because of lack of an entropy source. [Encrypting these keys is important. RFC7030 specifies how the private key can be encrypted with CMS using symmetric or asymmetric keys.]

4.1. Discovery and URI

EST-coaps is targeted to low-resource networks with small packets. Saving header space is important and the EST-coaps URI is shorter than the EST URI.

The presence and location of (path to) the management data are discovered by sending a GET request to `"/.well-known/core"` including a resource type (RT) parameter with the value `"ace.est"` [RFC6690]. Upon success, the return payload will contain the root resource of the EST 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 management data.

```
REQ: GET /.well-known/core?rt=ace.est

RES: 2.05 Content
</est>; rt="ace.est"
```

The EST-coaps server URIs differ from the EST URI by replacing the scheme `https` by `coaps` and by specifying shorter resource path names:

```
coaps://www.example.com/est/short-name
```

Figure 5 in section 3.2.2 of [RFC7030] enumerates the operations and corresponding paths which are supported by EST. Table 1 provides the mapping from the EST and BRSKI URI path to the EST-coaps URI path.

BRSKI	EST	EST-coaps
	/cacerts	/crt
	/simpleenroll	/sen
	/simplereenroll	/sren
	/csrattrs	/att
	/serverkeygen	/skg
/requestvoucher		/rv
/voucher_status		/vs
/enrollstatus		/es

Table 1

/requestvoucher and /enrollstatus are needed between pledge and Registrar.

When discovering the root path for the EST resources, the server MAY return the full resource paths and the used content types. This is useful when multiple content types are specified for EST-coaps server. For example, the following more complete response is possible.

REQ: GET /.well-known/core?rt=ace.est

```

RES: 2.05 Content
</est>; rt="ace.est"
</est/crt>; rt="ace.est";ct=TBD1
</est/sen>; rt="ace.est";ct=TBD1 TBD4
</est/sren>; rt="ace.est";ct=TBD1 TBD4
</est/att>; rt="ace.est";ct=TBD4
</est/skg>; rt="ace.est";ct=TBD1 TBD4 TBD2
</est/rv>; rt="ace.est";ct=TBD5 TBD6
</est/vs>; rt="ace.est";ct=50
</est/es>; rt="ace.est";ct=50

```

ct=50 stands for the Content-Format "application/json"

The return of the content-types allows the client to choose the most appropriate one from multiple content types.

4.2. Payload format

The content-format (media type equivalent) of the CoAP message determines which EST message is transported in the CoAP payload. The media types specified in the HTTP Content-Type header (see section 3.2.2 of [RFC7030]) are in EST-coaps specified by the Content-Format Option (12) of CoAP. The combination of URI path-suffix and content-format used for coap MUST map to an allowed combination of path-suffix and media type as defined for EST. The required content-formats for these request and response messages are defined in Section 8. The CoAP response codes are defined in Section 4.4.

EST-coaps is designed for use between low-resource devices using CoAP and hence does not need to send base64-encoded data. Simple binary is more efficient (30% less payload compared to base64) and well supported by CoAP. Therefore, the content formats specification in Section 8 requires the use of binary for all EST-coaps Content-Formats.

4.3. Message Bindings

This section describes BRSKI to CoAP message mappings.

All /crt, /sen, /sren, /att, /skg, /rv, /vs, and /es EST-coaps messages expect a response, so they are all CoAP CON messages.

The Ver, TKL, Token, and Message ID values of the CoAP header are not influenced by EST.

CoAP options are used to convey Uri-Host, Uri-Path, Uri-Port, Content-Format and more in CoAP. The CoAP Options are used to communicate the HTTP fields specified in the BRSKI REST messages.

BRSKI URLs are HTTPS based (https://), in CoAP these will be assumed to be transformed to coaps (coaps://)

Appendix A includes some practical examples of EST messages translated to CoAP.

4.4. CoAP response codes

Section 5.9 of [RFC7252] specifies the mapping of HTTP response codes to CoAP response codes. Every time the HTTP response code 200 is specified in [RFC7030] in response to a GET request, in EST-coaps the equivalent CoAP response code 2.05 MUST be used. Response code HTTP 202 in EST is mapped to CoAP 2.06 as specified in [I-D.hartke-core-pending]. All other HTTP 2xx response codes are not used by EST. For the following HTTP 4xx error codes that may occur:

400, 401, 403, 404, 405, 406, 412, 413, 415; the equivalent CoAP response code for EST-coaps is 4.xx. For the HTTP 5xx error codes: 500, 501, 502, 503, 504 the equivalent CoAP response code is 5.xx.

4.5. Message fragmentation

DTLS defines fragmentation only for the handshake part and not for secure data exchange (DTLS records). [RFC6347] states "Each DTLS record MUST fit within a single datagram". To avoid using IP fragmentation, which is not supported by 6LoWPAN, invokers of the DTLS record layer MUST size DTLS records so that they fit within any Path MTU estimates obtained from the record layer. In addition, invokers residing on a 6LoWPAN over IEEE 802.15.4 network SHOULD attempt to size CoAP messages such that each DTLS record will fit within one or two IEEE 802.15.4 frames.

That is not always possible. Even though ECC certificates are small in size, they can vary greatly based on signature algorithms, key sizes, and OID fields used. For 256-bit curves, common ECDSA cert sizes are 500-1000 bytes which could fluctuate further based on the algorithms, OIDs, SANs and cert fields. For 384-bit curves, ECDSA certs increase in size and can sometimes reach 1.5KB. Additionally, there are times when the EST cacerts response from the server can include multiple certs that amount to large payloads. CoAP [RFC7252]'s section 4.6 describes the possible payload sizes: "if nothing is known about the size of the headers, good upper bounds are 1152 bytes for the message size and 1024 bytes for the payload size". Also "If IPv4 support on unusual networks is a consideration, implementations may want to limit themselves to more conservative IPv4 datagram sizes such as 576 bytes; per [RFC0791], the absolute minimum value of the IP MTU for IPv4 is as low as 68 bytes, which would leave only 40 bytes minus security overhead for a UDP payload". Thus, even with ECC certs, EST-coaps messages can still exceed sizes in MTU of 1280 for IPv6 or 60-80 bytes for 6LoWPAN [RFC4919] as explained in section 2 of [RFC7959]. EST-coaps needs to be able to fragment EST messages into multiple DTLS datagrams. Fine-grained fragmentation of EST messages is essential.

To perform fragmentation in CoAP, [RFC7959] specifies the "Block1" option for fragmentation of the request payload and the "Block2" option for fragmentation of the return payload of a CoAP flow.

The BLOCK draft defines SZX in the Block1 and Block2 option fields. These are used to convey the size of the blocks in the requests or responses.

The CoAP client MAY specify the Block1 size and MAY also specify the Block2 size. The CoAP server MAY specify the Block2 size, but not

the Block1 size. As explained in Section 1 of [RFC7959]), blockwise transfers SHOULD be used in Confirmable CoAP messages to avoid the exacerbation of lost blocks.

The Size1 response MAY be parsed by the client as a size indication of the Block2 resource in the server response or by the server as a request for a size estimate by the client. Similarly, Size2 option defined in BLOCK should be parsed by the server as an indication of the size of the resource carried in Block1 options and by the client as a maximum size expected in the 4.13 (Request Entity Too Large) response to a request.

Examples of fragmented messages are shown in Appendix B.

5. Transport Protocol

EST-coaps depends on a secure transport mechanism over UDP that can secure (confidentiality, authenticity) the CoAP messages exchanged.

5.1. DTLS

DTLS is one such secure protocol. Within BRSKI and EST when "TLS" is referred to, it is understood that in EST-coaps, security is provided using DTLS instead. No other changes are necessary (all provisional modes etc. are the same as for TLS).

CoAP was designed to avoid fragmentation. DTLS is used to secure CoAP messages. However, fragmentation is still possible at the DTLS layer during the DTLS handshake when using ECC ciphersuites. If fragmentation is necessary, "DTLS provides a mechanism for fragmenting a handshake message over a number of records, each of which can be transmitted separately, thus avoiding IP fragmentation" [RFC6347].

CoAP and DTLS can provide proof of identity for EST-coaps clients and server with simple PKI messages conformant to section 3.1 of [RFC5272]. EST-coaps supports the certificate types and Trust Anchors (TA) that are specified for EST in section 3 of [RFC7030].

Channel-binding information for linking proof-of-identity with connection-based proof-of-possession is optional for EST-coaps. When proof-of-possession is desired, a set of actions are required regarding the use of tls-unique, described in section 3.5 in [RFC7030]. The tls-unique information translates to the contents of the first "Finished" message in the TLS handshake between server and client [RFC5929]. The client is then supposed to add this "Finished" message as a ChallengePassword in the attributes section of the PKCS#10 Request Info to prove that the client is indeed in control of

the private key at the time of the TLS session when performing a /simpleenroll, for example. In the case of EST-coaps, the same operations can be performed during the DTLS handshake. In the event of handshake message fragmentation, the Hash of the handshake messages used in the MAC calculation of the Finished message

```
PRF(master_secret, finished_label, Hash(handshake_messages))  
[0..verify_data_length-1];
```

MUST be computed as if each handshake message had been sent as a single fragment [RFC6347].

In a constrained CoAP environment, endpoints can't afford to establish a DTLS connection for every EST transaction. Authenticating and negotiating DTLS keys requires resources on low-end endpoints and consumes valuable bandwidth. The DTLS connection SHOULD remain open for persistent EST connections. For example, an EST cacerts request that is followed by a simpleenroll request can use the same authenticated DTLS connection. Given that after a successful enrollment, it is more likely that a new EST transaction will take place after a significant amount of time, the DTLS connections SHOULD only be kept alive for EST messages that are relatively close to each other.

5.2. 6tisch approach

The 6tisch bootstrapping is targeted to the "imprinting" of the "pledge" with layer 2 keys. The content formats for the transport are being defined and may be expressed in a YANG module.

Instead of using transport security, the 6tisch approach relies on application security provided by OSCOAP [I-D.ietf-core-object-security] and EDHOC [I-D.selander-ace-cose-ecdhe]. [I-D.selander-ace-eals] uses OSCOAP to securely enroll certificates by using Certificate Management over CMS (CMC) (EST is profile of CMC).

It is suggested that the EST-coaps communication between pledge and registrar, specified in this document, can be freely exchanged with the same communication specified in [I-D.ietf-6tisch-dtsecurity-secure-join] and [I-D.ietf-6tisch-minimal-security].

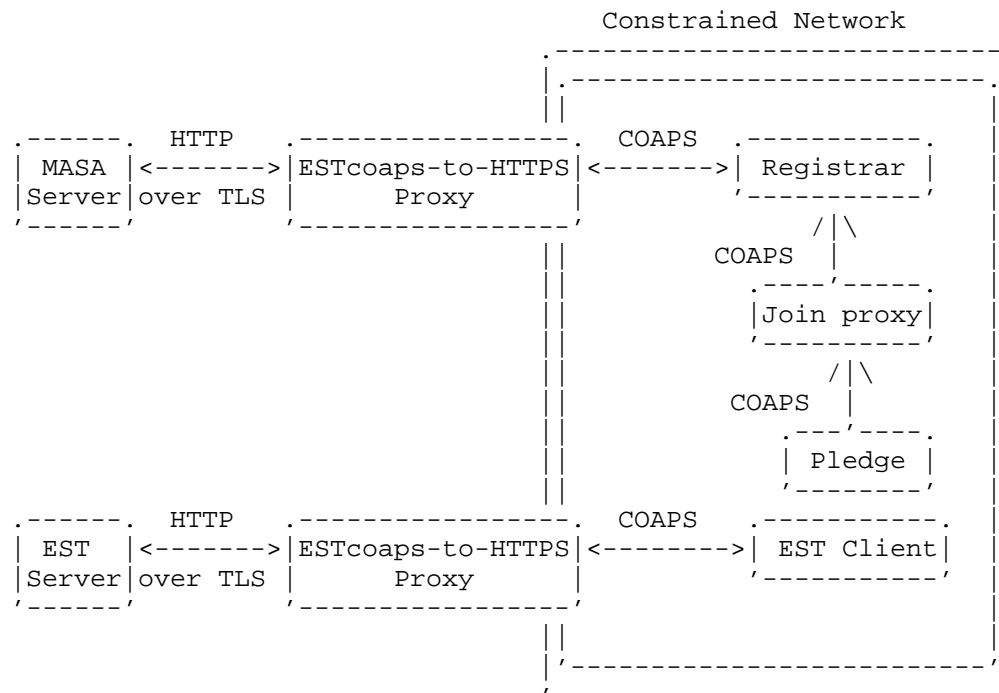
[EDNOTE: The evolution of this section depends on the directions taken by 6tisch and anima and the possible commonality that will be provided.]

6. Proxying

In real-world deployments, entities like the EST server, CA or MASA will not always reside within the COAP boundary. The MASA or a CA can exist outside the constrained network in a non-constrained network that supports TLS/HTTP. In such environments EST-coaps is used by the pledge within the COAP boundary and TLS is used to transport the EST/BRSKI messages outside the CoAP boundary. A proxy entity at the edge is required to operate between the COAP environment and the external HTTP network. The ESTcoaps-to-HTTPS proxy SHOULD terminate EST-coaps downstream and initiate EST/BRSKI connections over TLS upstream.

Two separate use-cases, shown in one figure below, are expected to be deployed in practice:

- o A proxy between any EST-client and EST-server independent of BRSKI
- o A proxy between Registrar and MASA



ESTcoaps-to-HTTPS proxy at the COAP boundary.

Table 1 contains the mapping between the EST-coaps and EST/BRSKI URIs the proxy SHOULD adhere to. Section 7 of [RFC8075] and Section 4.4 define the mapping between EST-coaps and HTTP response codes, that determines how a proxy translates COAP response codes from/to HTTP status codes. The mapping from Content-Type to media type is defined in Section 8. The conversion from binary to BSD64 needs to be done in the proxy. Conversion is possible because a TLS link exists between EST-coaps-to-HTTP proxy and HTTP MASA or EST server and a corresponding DTLS link exists between EST-coaps-to-HTTP proxy and EST client or Registrar.

Due to fragmentation of large messages into blocks, an EST-coaps-to-HTTP proxy SHOULD reassemble the BLOCKs before translating the binary content to BSD64, and consecutively relay the message upstream into the HTTP environment.

For the discovery of the EST server by the EST client in the coap environment, the EST-coaps-to-HTTP proxy MUST announce itself according to the rules of Section 4.1. The available functions of the proxies MUST be announced with as many resource paths. The discovery of MASA and EST server in the http environment follow the rules specified in [I-D.ietf-anima-bootstrapping-keyinfra].

[EDNOTE: PoP will be addressed here.]

A proxy SHOULD authenticate the client downstream and it should be authenticated by the EST or BRSKI server or CA upstream. A trust relationship needs to be pre-established between the proxy and the TCP entities (EST, BRSKI servers) to be able to proxy these connections on behalf of various clients.

[EDNOTE: To add more details about trust relations in this section.]

7. Parameters

[EDNOTE: This section to be populated. It will address transmission parameters for BRSKI described in sections 4.7 and 4.8 of the CoAP draft. BRSKI does not impose any unique parameters that affect the CoAP parameters in Table 2 and 3 in the CoAP draft but the ones in CoAP could be affecting BRSKI. For example, the processing delay of CAs could be less than 2s, but in this case they should send a CoAP ACK every 2s while processing.]

8. IANA Considerations

8.1. Content-Format registry

Additions to the sub-registry "CoAP Content-Formats", within the "CoRE Parameters" registry are needed for the below media types. These can be registered either in the Expert Review range (0-255) or IETF Review range (256-9999).

1.

- * application/pkcs7-mime
- * Type name: application
- * Subtype name: pkcs7-mime
- * smime-type: certs-only
- * ID: TBD1
- * Required parameters: None
- * Optional parameters: None
- * Encoding considerations: binary
- * Security considerations: As defined in this specification
- * Published specification: [RFC5751]
- * Applications that use this media type: ANIMA Bootstrap (BRSKI) and EST

2.

- * application/pkcs8
- * Type name: application
- * Subtype name: pkcs8
- * ID: TBD2
- * Required parameters: None
- * Optional parameters: None
- * Encoding considerations: binary

- * Security considerations: As defined in this specification
- * Published specification: [RFC5958]
- * Applications that use this media type: ANIMA Bootstrap (BRSKI) and EST

3.

- * application/csrattrs
- * Type name: application
- * Subtype name: csrattrs
- * ID: TBD3
- * Required parameters: None
- * Optional parameters: None
- * Encoding considerations: binary
- * Security considerations: As defined in this specification
- * Published specification: [RFC7030]
- * Applications that use this media type: ANIMA Bootstrap (BRSKI) and EST

4.

- * application/pkcs10
- * Type name: application
- * Subtype name: pkcs10
- * ID: TBD4
- * Required parameters: None
- * Optional parameters: None
- * Encoding considerations: binary
- * Security considerations: As defined in this specification

- * Published specification: [RFC5967]
- * Applications that use this media type: ANIMA bootstrap (BRSKI) and EST
- *
 - + application/voucherrequest
 - + Type name: application
 - + Subtype name: voucherrequest
 - + ID: TBD5
 - + Required parameters: None
 - + Optional parameters: None
 - + Encoding considerations: binary
 - + Security considerations: As defined in this specification
 - + Published specification: BRSKI??
 - + Applications that use this media type: ANIMA bootstrap (BRSKI)
- *
 - + application/voucher+cms
 - + Type name: application
 - + Subtype name: voucher+cms
 - + ID: TBD6
 - + Required parameters: None
 - + Optional parameters: None
 - + Encoding considerations: binary
 - + Security considerations: As defined in this specification
 - + Published specification: BRSKI??

- + Applications that use this media type: ANIMA bootstrap (BRSKI)

8.2. Resource Type registry

Additions to the sub-registry "CoAP Resource Type", within the "CoRE Parameters" registry are needed for a new resource type.

- o rt="ace.est" needs registration with IANA.

9. Security Considerations

9.1. proxy considerations

In the BRSKI bootstrap protocol, there is a direct TLS connection from pledge to EST-server. With the EST-coaps specification a direct DTLS connection from pledge to EST-server is possible thus avoiding the placement of a https/coaps proxy between pledge and http EST-server. Such a https/coaps proxy presents a security issue because the proxy needs to make a TLS connection with the EST-server and a DTLS connection with the pledge.

In [I-D.ietf-anima-bootstrapping-keyinfra] the EST-server and Registrar are co-located on the same host, thus avoiding security connections between Registrar and EST-server. It is RECOMMENDED that the links "Registrar/MASA" and "Registrar/CA" use a http TLS connection, identical to BRSKI protocol. The consequence is that the Registrar host provides both a coaps and a https stack.

The proxies proposed in Section 6 must be deployed with great care, and only when the recommended connections are impossible.

9.2. EST server considerations

The security considerations of section 6 of [RFC7030] are only partially valid for the purposes of this document. As HTTP Basic Authentication is not supported, the considerations expressed for using passwords do not apply.

Given that the client has only limited resources and may not be able to generate sufficiently random keys to encrypt its identity, it is possible that the client uses server generated private/public keys to encrypt its certificate. The transport of these keys is inherently risky. A full probability analysis MUST be done to establish whether server side key generation enhances or decreases the probability of identity stealing.

When a client uses the Implicit TA database for certificate validation, the client cannot verify that the implicit data base can act as an RA. It is RECOMMENDED that such clients include "Linking Identity and POP Information" Section 5.1 in requests (to prevent such requests from being forwarded to a real EST server by a man in the middle). It is RECOMMENDED that the Implicit Trust Anchor database used for EST server authentication be carefully managed to reduce the chance of a third-party CA with poor certification practices from being trusted. Disabling the Implicit Trust Anchor database after successfully receiving the Distribution of CA certificates response (Section 4.1.3 of [RFC7030]) limits any vulnerability to the first DTLS exchange.

In accordance with [RFC7030], TLS cipher suites that include "_EXPORT_" and "_DES_" in their names MUST NOT be used. More information about recommendations of TLS and DTLS are included in [RFC7525].

As described in CMC, Section 6.7 of [RFC5272], "For keys that can be used as signature keys, signing the certification request with the private key serves as a POP on that key pair". The inclusion of `tls-unique` in the certification request links the proof-of-possession to the TLS proof-of-identity. This implies but does not prove that the authenticated client currently has access to the private key.

Regarding the CSR attributes that the CA may list for inclusion in an enrollment request, an adversary could exclude attributes that a server may want, include attributes that a server may not want, and render meaningless other attributes that a server may want. The CA is expected to be able to enforce policies to recover from improper CSR requests.

Interpreters of ASN.1 structures should be aware of the use of invalid ASN.1 length fields and should take appropriate measures to guard against buffer overflows, stack overruns in particular, and malicious content in general.

10. Acknowledgements

The authors are very grateful to Klaus Hartke for his detailed explanations on the use of Block with DTLS. The authors would like to thank Esko Dijk and Michael Verschoor for the valuable discussions that helped in shaping the solution. They would also like to thank Peter Panburana from Cisco for his feedback on technical details of the solution. Constructive comments were received from Eliot Lear and Julien Vermillard.

11. Change Log

-02:

binary instead of CBOR binary in mime types.

supported content types are discoverable.

DTLS POP text improved.

First version of Security considerations section written.

First version of Proxying section written.

Various text improvements.

-01:

Merging of draft-vanderstok-ace-coap-est-00 and draft-pritikin-coap-bootstrap-01

URI and discovery are modified

More text about 6tisch bootstrap including EDHOC and OSCOAP

mapping to DICE IoT profiles

adapted to BRSKI progress

12. References

12.1. Normative References

[I-D.hartke-core-pending]

Stok, P. and K. Hartke, "The 'Pending' Response Code for the Constrained Application Protocol (CoAP)", draft-hartke-core-pending-00 (work in progress), February 2017.

[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-06 (work in progress), May 2017.

[I-D.selander-ace-eals]

Selander, G., Raza, S., Vucinic, M., Furuheid, M., and M. Richardson, "Enrollment with Application Layer Security", draft-selander-ace-eals-00 (work in progress), March 2017.

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC5272] Schaad, J. and M. Myers, "Certificate Management over CMS (CMC)", RFC 5272, DOI 10.17487/RFC5272, June 2008, <<http://www.rfc-editor.org/info/rfc5272>>.
- [RFC5751] Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification", RFC 5751, DOI 10.17487/RFC5751, January 2010, <<http://www.rfc-editor.org/info/rfc5751>>.
- [RFC5967] Turner, S., "The application/pkcs10 Media Type", RFC 5967, DOI 10.17487/RFC5967, August 2010, <<http://www.rfc-editor.org/info/rfc5967>>.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <<http://www.rfc-editor.org/info/rfc6347>>.
- [RFC6690] Shelby, Z., "Constrained RESTful Environments (CoRE) Link Format", RFC 6690, DOI 10.17487/RFC6690, August 2012, <<http://www.rfc-editor.org/info/rfc6690>>.
- [RFC7030] Pritikin, M., Ed., Yee, P., Ed., and D. Harkins, Ed., "Enrollment over Secure Transport", RFC 7030, DOI 10.17487/RFC7030, October 2013, <<http://www.rfc-editor.org/info/rfc7030>>.
- [RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <<http://www.rfc-editor.org/info/rfc7252>>.
- [RFC7959] Bormann, C. and Z. Shelby, Ed., "Block-Wise Transfers in the Constrained Application Protocol (CoAP)", RFC 7959, DOI 10.17487/RFC7959, August 2016, <<http://www.rfc-editor.org/info/rfc7959>>.
- [RFC8075] Castellani, A., Loreto, S., Rahman, A., Fossati, T., and E. Dijk, "Guidelines for Mapping Implementations: HTTP to the Constrained Application Protocol (CoAP)", RFC 8075, DOI 10.17487/RFC8075, February 2017, <<http://www.rfc-editor.org/info/rfc8075>>.

12.2. Informative References

- [I-D.ietf-6tisch-dtsecurity-secure-join]
Richardson, M., "6tisch Secure Join protocol", draft-ietf-6tisch-dtsecurity-secure-join-01 (work in progress), February 2017.
- [I-D.ietf-6tisch-minimal-security]
Vucinic, M., Simon, J., Pister, K., and M. Richardson, "Minimal Security Framework for 6TiSCH", draft-ietf-6tisch-minimal-security-02 (work in progress), March 2017.
- [I-D.ietf-anima-voucher]
Watsen, K., Richardson, M., Pritikin, M., and T. Eckert, "Voucher Profile for Bootstrapping Protocols", draft-ietf-anima-voucher-03 (work in progress), June 2017.
- [I-D.ietf-core-object-security]
Selander, G., Mattsson, J., Palombini, F., and L. Seitz, "Object Security of CoAP (OSCOAP)", draft-ietf-core-object-security-03 (work in progress), May 2017.
- [I-D.selander-ace-cose-ecdhe]
Selander, G., Mattsson, J., and F. Palombini, "Ephemeral Diffie-Hellman Over COSE (EDHOC)", draft-selander-ace-cose-ecdhe-06 (work in progress), April 2017.
- [ieee802.15.4]
Institute of Electrical and Electronics Engineers, "IEEE Standard 802.15.4-2006", 2006.
- [RFC4492] Blake-Wilson, S., Bolyard, N., Gupta, V., Hawk, C., and B. Moeller, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)", RFC 4492, DOI 10.17487/RFC4492, May 2006, <<http://www.rfc-editor.org/info/rfc4492>>.
- [RFC4919] Kushalnagar, N., Montenegro, G., and C. Schumacher, "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals", RFC 4919, DOI 10.17487/RFC4919, August 2007, <<http://www.rfc-editor.org/info/rfc4919>>.
- [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, <<http://www.rfc-editor.org/info/rfc4944>>.

- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <<http://www.rfc-editor.org/info/rfc5246>>.
- [RFC5929] Altman, J., Williams, N., and L. Zhu, "Channel Bindings for TLS", RFC 5929, DOI 10.17487/RFC5929, July 2010, <<http://www.rfc-editor.org/info/rfc5929>>.
- [RFC5958] Turner, S., "Asymmetric Key Packages", RFC 5958, DOI 10.17487/RFC5958, August 2010, <<http://www.rfc-editor.org/info/rfc5958>>.
- [RFC6090] McGrew, D., Igoe, K., and M. Salter, "Fundamental Elliptic Curve Cryptography Algorithms", RFC 6090, DOI 10.17487/RFC6090, February 2011, <<http://www.rfc-editor.org/info/rfc6090>>.
- [RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <<http://www.rfc-editor.org/info/rfc7230>>.
- [RFC7251] McGrew, D., Bailey, D., Campagna, M., and R. Dugal, "AES-CCM Elliptic Curve Cryptography (ECC) Cipher Suites for TLS", RFC 7251, DOI 10.17487/RFC7251, June 2014, <<http://www.rfc-editor.org/info/rfc7251>>.
- [RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <<http://www.rfc-editor.org/info/rfc7525>>.
- [RFC7641] Hartke, K., "Observing Resources in the Constrained Application Protocol (CoAP)", RFC 7641, DOI 10.17487/RFC7641, September 2015, <<http://www.rfc-editor.org/info/rfc7641>>.
- [RFC7925] Tschofenig, H., Ed. and T. Fossati, "Transport Layer Security (TLS) / Datagram Transport Layer Security (DTLS) Profiles for the Internet of Things", RFC 7925, DOI 10.17487/RFC7925, July 2016, <<http://www.rfc-editor.org/info/rfc7925>>.

Appendix A. EST messages to EST-coaps

This section takes all examples from Appendix A of [RFC7030], changes the payload from Base64 to binary and replaces the http headers by their CoAP equivalents.

A.1. cacerts

In EST-coaps, a coaps cacerts message can be:

```
GET coaps://[192.0.2.1:8085]/est/crts
```

The corresponding CoAP header fields are shown below. The use of block and DTLS are worked out in Appendix B.

```
Ver = 1
T = 0 (CON)
Code = 0x01 (0.01 is GET)
Options
  Option1 (Uri-Host)
    Option Delta = 0x3 (option nr = 3)
    Option Length = 0x9
    Option Value = 192.0.2.1
  Option2 (Uri-Port)
    Option Delta = 0x4 (option nr = 4+3=7)
    Option Length = 0x4
    Option Value = 8085
  Option3 (Uri-Path)
    Option Delta = 0x4 (option nr = 7+4= 11)
    Option Length = 0x9
    Option Value = /est/crts
Payload = [Empty]
```

A 2.05 Content response with a cert in EST-coaps will then be:

```
2.05 Content (Content-Format: application/pkcs7-mime)
  {payload}
```

with CoAP fields

```
Ver = 1
T = 2 (ACK)
Code = 0x45 (2.05 Content)
Options
  Option1 (Content-Format)
    Option Delta = 0xC (option nr = 12)
    Option Length = 0x2
    Option Value = TBD1 (defined in this document)
```


Payload =

```
30233906092a6206734107028c2a3023260201013100300b06092a6206734107018
c0c3020bb302063c20102020900a61e75193b7acc0d06092a620673410105050030
1b31193017060355040313106573744578616d706c654341204f774f301e170d313
3303530393033353333315a170d3134303530393033353333315a301b3119301706
0355040313106573744578616d706c654341204f774f302062300d06092a6206734
10101050003204f0030204a022041003a923a2968bae4aae136ca4e2512c5200680
358482ac39d6f640e4574e654ea35f48b1e054c5da3372872f7a1e429f4edf39584
32efb2106591d3eb783c1034709f251fc86566bda2d541c792389eac4ec9e181f4b
9f596e5ef2679cc321542b11337f90a44df3c85f1516561fa968a1914f265bc0b82
76ebe3106a790d97d34c8c37c74felc30b396424664ac426284a9f6022e02693843
6880adfdc95c98caldfc2e6d75319b85d0458de28a9d13fb16d620ffff7541f6a25d
7daf004355020301000130b040300f0603551d130101f10530030101fc1d0603551
d0e04160414084d321ca0135e77217a486b686b334b00e0603551d0f0101f104030
20106300d06092a62067341010505000320410023703b965746a0c2c978666d787a
94f89b495a11f0d369b28936ec2475c0f0855c8e83f823f2b871a1d92282f323c45
904ba008579216cf5223b8b1bc425a0677262047f7700240631c17f3035d1c3780b
2385241cba1f4a6e98e6be6820306b3a786de5a557795d1893822347b5f825d34a7
ad2876f8feba4d525b31066f6505796f71530003431a3e6bbfe788b4565029a7e20
a51107677552586152d051e8eebf383e92288983421d5c5652a4870c3af74b9bdbe
d6b462e2263d30f6d3020c330206bc20102020101300d06092a6206734101050500
301b31193017060355040313106573744578616d706c654341204f774f301e170d3
133303530393033353333325a170d3134303530393033353333325a301b31193017
060355040313106573744578616d706c654341204e774f302062300d06092a62067
3410101050003204f0030204a02204100ef6b677a3247c1fc03d2b9baf113e5e7e1
1f49e0421120e6b8384160f2bf02630ef544d5fd0d5623b35713c79a7229283a790
8751a634aa420a3e2a4b1f10519d046f02f5a5dd6d760c2a842356e067b7bd94338
d1faa3b3ddd4813060a207b0a097067007e45b052b60fdbae4656e11562c4f5abb7
b0cf87a79d221f1127313c53371ce1245d63db45a1203a23340ba08042c768d03b8
076a028d3a51d87d2ef107bbd6f2305ce5e67668724002fb726df9c14476c37de0f
55033f192a5ad21f9a2a71c20301000134b050300e0603551d0f0101f104030204c
1d0603551d0e04160414112966e304761732fbfe6a2c823c301f0603551d2304183
0165084d321ca0135e77217a486b686b334b00d06092a6206734101050500032041
00b382ba3355a50e287bae15758b3beff63d34d3e357b90031495d018868e49589b
9faf46a4ad49b1d35b06ef380106677440934663c2cc111c183655f4dc411c0b3401
123d35387389db91f1elb4131b16c291d35730b3f9b33c7475124851555fe5fc647
e8fd029605367c7e01281bf6617110021b0d10847dce0e9f0ca6c764b6334784055
172c3983d1e3a3a82301a54fcc9b0670c543a1c747164619101ff23b240b2a26394
clf7d38d0e2f4747928ece5c34627a075a8b3122011e9d9158055c28f020c330206
bc20102020102300d06092a6206734101050500301b311930170603550403131065
73744578616d706c654341204e774e301e170d3133303530393033353333325a170
d3134303530393033353333325a301b31193017060355040313106573744578616d
706c654341204f774e302062300d06092a620673410101050003204f0030204a022
041003a923a2968bae4aae136ca4e2512c5200680358482ac39d6f640e4574e654e
a35f48b1e054c5da3372872f7a1e429f4edf3958432efb2106591d3eb783c103470
9f251fc86566bda2d541c792389eac4ec9e181f4b9f596e5ef2679cc321542b1133
7f90a44df3c85f1516561fa968a1914f265bc0b8276ebe3106a790d97d34c8c37c7
```

```
4felc30b396424664ac426284a9f6022e026938436880adfc95c98caldfc2e6d75
319b85d0458de28a9d13fb16d620ffff7541f6a25d7daf004355020301000134b050
300e0603551d0f0101f104030204c1d0603551d0e04160414084d321ca0135e7721
7a486b686b334b01f0603551d230418301653112966e304761732fbfe6a2c823c30
0d06092a6206734101050500032041002e106933a443070acf5594a3a584d08af7e
06c295059370a06639eff9bd418d13bc25a298223164a6cf1856b11a81617282e4a
410d82ef086839c6e235690322763065455351e4c596acc7c016b225dec094706c2
a10608f403b10821984c7c152343b18a768c2ad30238dc45dd653ee6092b0d5cd4c
2f7d236043269357f76d13f95fb5f00d0e19263c6833948e1ba612ce8197af650e2
5d882c12f4b6b9b67252c608ef064aca3f9bc867d71172349d510bb7651cd438837
73d927deb41c4673020bb302063c201020209009b9dda3324700d06092a62067341
01050500301b31193017060355040313106573744578616d706c654341204e774e3
01e170d31333035303930333333325a170d313430353039303333333325a301b
31193017060355040313106573744578616d706c654341204e774e302062300d060
92a620673410101050003204f0030204a02204100ef6b677a3247c1fc03d2b9baf1
13e5e7e11f49e0421120e6b8384160f2bf02630ef544d5fd0d5623b35713c79a722
9283a7908751a634aa420a3e2a4b1f10519d046f02f5a5dd6d760c2a842356e067b
7bd94338d1faa3b3ddd4813060a207b0a097067007e45b052b60fdbae4656e11562
c4f5abb7b0cf87a79d221f1127313c53371ce1245d63db45a1203a23340ba08042c
768d03b8076a028d3a51d87d2ef107bbd6f2305ce5e67668724002fb726df9c1447
6c37de0f55033f192a5ad21f9a2a71c20301000130b040300f0603551d130101f10
530030101fc1d0603551d0e04160414112966e304761732fbfe6a2c823c300e0603
551d0f0101f10403020106300d06092a620673410105050003204100423f06d4b76
0f4b42744a279035571696f272a0060f1325a40898509601ad14004f652db6312a1
475c4d7cd50f4b269035585d7856c5337765a66b38462d5bdaa7778aab24bbe2815
e37722cd10e7166c50e75ab75a1271324460211991e7445a2960f47351a1a629253
34119794b90e320bc730d6c1bee496e7ac125ce9aleca595a3a4c54a865e6b623c9
247bfd0a7c19b56077392555c955e233642bec643ae37c166c5e221d797aea3748f
0391c8d692a5cf9bb71f6d0e37984d6fa673a30d0c006343116f58403100
```

A.2. csrattrs

In the following valid /csrattrs exchange, the EST-coaps client authenticates itself with a certificate issued by the connected CA.

The initial DTLS handshake is identical to the enrollment example.
The CoAP GET request looks like:

```
GET coaps://[192.0.2.1:8085]/est/att
```

with CoAP header fields

```

Ver = 1
T = 0 (CON)
Code = 0x01 (0.01 is GET)
Options
  Option1 (Uri-Host)
    Option Delta = 0x3 (option nr = 3)
    Option Length = 0x9
    Option Value = 192.0.2.1
  Option2 (Uri-Port)
    Option Delta = 0x4 (option nr = 4+3=7)
    Option Length = 0x4
    Option Value = 8085
  Option3 (Uri-Path)
    Option Delta = 0x4 (option nr = 7+4= 11)
    Option Length = 0x8
    Option Value = /est/att
Payload = [Empty]

```

A 2.05 Content response contains attributes which are relevant for the authenticated client. In this example, the EST-coaps server two attributes that the client can ignore when they are unknown to him.:

```

2.05 Content (Content-Format: application/crsattrs)
  {payload}

```

with CoAP fields

```

Ver = 1
T = 2 (ACK)
Code = 0x45 (2.05 Content)
Options
  Option1 (Content-Format)
    Option Delta = 0xC (option nr = 12)
    Option Length = 0x2
    Option Value = TBD3 (defined in this document)

  Payload =
307c06072b060101010116302206038dc1311b131950617273652053455420617
320322e3939392e31206461746106092a620673410907302c06038dc231250603
8dc306038dc4131950617273652053455420617320322e3939392e32206461746
106092b240303020801010b06096062016503040202

```

A.3. enroll / reenroll

[EDNOTE: We might need a new Option for the Retry-After response message. We might need a new Option for the WWW-Authenticate response.]

During the Enroll/Reenroll exchange, the EST-coaps client uses a CSR (PKCS#10) request in the POST request payload.

POST coaps://[192.0.2.1:8085]/est/sen
(Content-Format: application/pkcs10)

with CoAP header fields

```
Ver = 1
T = 0 (CON)
Code = 0x02 (0.02 is POST)
Options
  Option1 (Uri-Host)
    Option Delta = 0x3 (option nr = 3)
    Option Length = 0x9
    Option Value = 192.0.2.1
  Option2 (Uri-Port)
    Option Delta = 0x4 (option nr = 4+3=7)
    Option Length = 0x4
    Option Value = 8085
  Option3 (Uri-Path)
    Option Delta = 0x4 (option nr = 7+4= 11)
    Option Length = 0x8
    Option Value = /est/sen
  Option4 (Content-Format)
    Option Delta = 0x1 (option nr = 11+1 = 12)
    Option Length = 0x2
    Option Value = TBD5 (defined in this document)
```

```
Payload =
[EDNOTE: If POP is used, make sure tls-unique in the CSR is a
 valid HMAC output. ]
30208530206d020100301f311d301b0603550403131464656d6f7374657034203
1333638313431333532302062300d06092a620673410101050003204f0030204a
022041005d9f4dffd3c5949f646a9584367778560950b355c35b8e34726dd3764
54231734795b4c09b9c6d75d408311307a81f7adef7f5d241f7d5be85620c5d44
38bbb4242cf215c167f2ccf36c364ea2618a62f0536576369d6304e6a96877224
7d86824f079faac7a6f694cfda5b84c42087dc062d462190c525813f210a036a7
37b4f30d8891f4b75559fb72752453146332d51c937557716ccec624f5125c3a4
447ad3115020048113fef54ad554ee88af09a2583aac9024075113db4990b1786
b871691e0f02030100018701f06092a620673410907311213102b72724369722f
372b45597535305434300d06092a620673410105050003204100441b40177a3a6
5501487735a8ad5d3827a4eaa867013920e2afcd87aa81733c7c0353be47e1bf
a7cda5176e7ccc6be22ae03498588d5f2de3b143f2b1a6175ec544e8e7625af6b
836fd4416894c2e55ea99c6606f69075d6d53475d410729aa6d806afb9986caf
7b844b5b3e4545f19071865ada007060cad6db26a592d4a7bda7d586b68110962
17071103407553155cddc75481e272b5ed553a8593fb7e25100a6f7605085dab4
fc7e0731f0e7fe305703791362d5157e92e6b5c2e3edbcadb40
```

After verification of the certificate by the server, a 2.05 Content response with the issued certificate will be:

```
2.05 Content (Content-Format: application/pkcs7-mime)
  {payload}
```

with CoAP fields

```

Ver = 1
T = 2 (ACK)
Code = 0x45 (2.05 Content)
Options
  Option1 (Content-Format)
    Option Delta = 0xC (option nr = 12)
    Option Length = 0x2
    Option Value = TBD1 (defined in this document)

```

```

Payload =
3020f806092a62067341070283293020e50201013100300b06092a62067341070
1830b3020c730206fc20102020115300d06092a6206734101050500301b311930
17060355040313106573744578616d706c654341204e774e301e170d313330353
0393233313535335a170d3134303530393233313535335a301f311d301b060355
0403131464656d6f73746570342031333638313431333532302062300d06092a6
20673410101050003204f0030204a022041005d9f4dffd3c5949f646a95843677
78560950b355c35b8e34726dd376454231734795b4c09b9c6d75d408311307a81
f7adef7f5d241f7d5be85620c5d4438bbb4242cf215c167f2ccf36c364ea2618a
62f0536576369d6304e6a968772247d86824f079faac7a6f694cfda5b84c42087
dc062d462190c525813f210a036a737b4f30d8891f4b75559fb72752453146332
d51c937557716ccec624f5125c3a4447ad3115020048113fef54ad554ee88af09
a2583aac9024075113db4990b1786b871691e0f020301000134b050300e060355
1d0f0101f104030204c1d0603551d0e04160414e81d0788aa2710304c5ecd4d1e
065701f0603551d230418301653112966e304761732fbfe6a2c823c300d06092a
6206734101050500032041002910d86f2ffeeb914c046816871de601567d291b4
3fabee0f0e8ff81cea27302a7133e20e9d04029866a8963c7d14e26fbe8a0ab1b
77fbb1214bbcdc906fbc381137ec1de685f79406c3e416b8d82f97174bc691637
5a4e1c4bf744c7572b4b2c6bade9fb35da786392ee0d95e3970542565f3886ad6
7746dlb12484bb02616e63302dc371dc6006e431fb7c457598dd204b367b0b3d3
258760a303f1102db26327f929b7c5a60173e1799491b69150248756026b80553
171e4733ad3d13c0103100

```

A.4. serverkeygen

During this valid /serverkeygen exchange, the EST-coaps client authenticates itself using the certificate provided by the connected CA.

The initial DTLS handshake is identical to the enrollment example. The CoAP GET request looks like:

```
POST coaps://[192.0.2.1:8085]/est/skg
```

with CoAP header fields

```
Ver = 1
T = 0 (CON)
Code = 0x02 (0.02 is POST)
Options
  Option1 (Uri-Host)
    Option Delta = 0x3 (option nr = 3)
    Option Length = 0x9
    Option Value = 192.0.2.1
  Option2 (Uri-Port)
    Option Delta = 0x4 (option nr = 4+3=7)
    Option Length = 0x4
    Option Value = 8085
  Option3 (Uri-Path)
    Option Delta = 0x4 (option nr = 7+4= 11)
    Option Length = 0x8
    Option Value = /est/skg
  Option4 (Content-Format)
[EDNOTE: the client includes a CSR with a public key that the
server should ignore, so we need a content-format here. ]
    Option Delta = 0x1 (option nr = 12)
    Option Length = 0x2
    Option Value = TBD5 (defined in this document)
Payload =
[EDNote: If POP is used, make sure tls-unique in the CSR
is a valid HMAC output. ]
302081302069020100305b313e303c060355040313357365727665724b6579476
56e2072657120627920636c69656e7420696e2064656d6f207374657020313220
3133363831343139353531193017060355040513105049443a576964676574205
34e3a3130302062300d06092a620673410101050003204f0030204a02204100f4
dfa6c03f7f2766b23776c333d2c0f9d1a7a6ee36d01499bbe6f075d1e38a57e98
ecc197f51b75228454b7f19652332de5e52e4a974c6ae34e1df80b33f15f47d3b
cbf76116bb0e4d3e04a9651218a476a13fc186c2a255e4065ff7c271cff104e47
31fad53c22b21a1e5138bf9ad0187314ac39445949a48805392390e78c7659621
6d3e61327a534f5ea7721d2b1343c7362b37da502717cfc2475653c7a3860c5f4
0612a5db6d33794d755264b6327e3a3263b149628585b85e57e42f6b3277591b0
2030100018701f06092a6206734109073112131064467341586d4a6e6a6f6b427
4447672300d06092a6206734101050003204100472d11007e5a2b2c2023d47a
6d71d046c307701d8ebc9e47272713378390b4ee321462a3dbe54579f5a514f6f
4050af497f428189b63655d03a194ef729f101743e5d03fbc6ae1e84486d1300a
f9288724381909188c851fa9a5059802eb64449f2a3c9e441353d136768da27ff
4f277651d676a6a7e51931b08f56135a2230891fd184960e1313e7a1a9139ed19
28196867079a456cd2266cb754a45151b7b1b939e381be333fea61580fe5d25bf
4823dbd2d6a98445b46305c10637e202856611
```

Without the DecryptKeyIdentifier attribute, the response has no additional encryption beyond the DTLS one. The EST-coaps server response is:

```
2.05 Content (Content-Format: application/pkcs8)
    {payload}
```

The response contains first a preamble that can be ignored. The EST-coaps server can use the preamble to include additional explanations, like ownership or support information

```
Ver = 1
T = 2 (ACK)
Code = 0x45 (2.05 Content)
Options
  Option1 (Content-Format)
    Option Delta = 0xC (option nr = 12)
    Option Length = 0x2
    Option Value = TBD2 (defined in this document)
```

```
    Payload =
30213e020100300d06092a6206734101010500042128302124020100022041003
c0bc2748f2003e3e8ea15f746f2a71e83f585412b92cf6f8e64de02e056153274
dd01c95dd9cfff3112aa141774ab655c3d56359c3b3df055294692ed848e7e30a1
1bf14e47e0693d93017022b4cdb3e6d40325356152b213c8b535851e681a7074c
0c6d2b60e7c32fc0336b28e743eba4e5921074d47195d3c05e43c527526e692d5
45e562578d2d4b5f2191bff89d3eef0222764a2674637a1f99257216647df6704
efec5adbf54dab24231844eb595875795000e673dd6862310a146ad7e31083010
001022041004e6b3f78b7791d6377f33117c17844531c81111fb8000282816264
915565bc7c3f3f643b537a2c69140a31c22550fa97e5132c61b74166b68626704
260620333050f510096b6570f5880e7e1c15dc0ca6ce2b5f187e2325da14ab705
ad004717f3b2f779127b5c535e0cee6a343b502722f2397a26126e0af606b5aa7
f96313511c0b7eb26354f91b82269de62757e3def807a6afdf83ddcbb0614bb7c
542e6975d6456554e7bd9988fbd1930cd44d0e01ee9182ca54539418653150254
1ad1a2a11e5021040bfce554b642c29131e7d65455e83c5406d76771912f758f5
ee3ee36af386f38ffa313c0f661880c5a2b0970485d36f528e7f77a2e55b4ad76
1242d1c2f75939c8061217d31491d305d3e07d6161c43e26f7de4477b1811de92
33dc75b426302104015bf48ac376f52887813461fc54635517bcb67293837053e
8cela33da7a35565a75a370dc14555b5316cb55742380350774d769d151ff0456
0214389a232a2258326163167504cfce44cd316f63bb8a52da53a4cb74fd87194
c0844881f791f23b0813ea0921325edd14459d41c8a1593f04316388e40b35fef
7d2a195a5930fa54774427ac821eee2c62790d2c17bd192af794c611011506557
83d4efe22185cbd83368786f2b1e68a5a27067e321066f0217b4b6d7971a3c21a
241366b7907187583b511102103369047e5cce0b65012200df5ec697b5827575c
db6821ff299d6a69574b31ddf0fbe9245ea2f74396c24b3a7565067e41366423b
5bdd2b2a78194094dbe333f493d159b8e07722f2280d48388db7f1c9f0633bb0e
173de2c3aa1f200af535411c7090210401421e2ea217e37312dcc606f453a6634
f3df4dc31a9e910614406412e70eec9247f10672a500947a64356c015a845a7d1
50e2e3911a2b3b61070a73247166da10bb45474cc97d1ec2bc392524307f35118
f917438f607f18181684376e13a39e07
--estServerExampleBoundary
Ver = 1
```



```

T = 2 (ACK)
Code = 0x45 (2.05 Content)
Options
  Option1 (Content-Format)
    Option Delta = 0xC (option nr = 12)
    Option Length = 0x2
    Option Value = TBD1 (defined in this document)
Payload=
3020c506092a62067341070283363020f20201013100300b06092a62067341070
183183020d430207cc20102020116300d06092a6206734101050500301b311930
17060355040313106573744578616d706c654341204e774e301e170d313330353
0393233323535365a170d3134303530393233323535365a302c312a3028060355
0403132173657276657273696465206b65792067656e657261746564207265737
06f6e7365302062300d06092a620673410101050003204f0030204a022041003c
0bc2748f2003e3e8ea15f746f2a71e83f585412b92cf6f8e64de02e056153274d
d01c95dd9cff3112aa141774ab655c3d56359c3b3df055294692ed848e7e30a11
bf14e47e0693d93017022b4cdb3e6d40325356152b213c8b535851e681a7074c0
c6d2b60e7c32fc0336b28e743eba4e5921074d47195d3c05e43c527526e692d54
5e562578d2d4b5f2191bff89d3eef0222764a2674637a1f99257216647df6704e
fec5adb5f54dab24231844eb595875795000e673dd6862310a146ad7e310830100
0134b050300e0603551d0f0101f104030204c1d0603551d0e04160414764b1bd5
e69935626e476b195a1a8c1f0603551d230418301653112966e304761732fbfe6
a2c823c300d06092a620673410105050003204100474e5100a9cdaaa813b30f48
40340fb17e7d6d6063064a5a7f2162301c464b5a8176623dfb1a4a484e618de1c
3c3c5927cf590f4541233ff3c251e772a9a3f2c5fc6e5ef2fe155e5e385deb846
b36eb4c3c7ef713f2d137ae8be4c022715fd033a818d55250f4e6077718180755
a4fa677130da60818175ca4ab2af1d15563624c51e13dfdcf381881b72327e2f4
9b7467e631a27b5b5c7d542bd2edaf78c0ac294f3972278996bdf673a334ff74c
84aa7d65726310252f6a4f41281ec10ca2243864e3c5743103100

```

A.5. enrollstatus

[EDNOTE: Include CoAP message examples.]

A.6. voucher_status

[EDNOTE: Include CoAP message examples.]

A.7. requestvoucher

[EDNOTE: Include CoAP message examples.]

Appendix B. EST-coaps Block message examples

This section provides a detailed example of the messages using DTLS and BLOCK option Block2. The minimum PMTU is 1280 bytes, which is the example value assumed for the DTLS datagram size. The example block length is taken as 64 which gives an SZX value of 2.

The following is an example of a valid /cacerts exchange over DTLS. The content length of the cacerts response in appendix A.1 of [RFC7030] is 4246 bytes using base64. This leads to a length of 2509 bytes in binary. The CoAP message adds around 10 bytes, the DTLS record 29 bytes. To avoid IP fragmentation, the CoAP block option is used and an MTU of 127 is assumed to stay within one IEEE 802.15.4 packet. To stay below the MTU of 127, the payload is split in 39 packets with a payload of 64 bytes each, followed by a packet of 13 bytes. The client sends an IPv6 packet containing the UDP datagram with the DTLS record that encapsulates the CoAP Request 40 times. The server returns an IPv6 packet containing the UDP datagram with the DTLS record that encapsulates the CoAP response. The CoAP request-response exchange with block option is shown below. Block option is shown in a decomposed way indicating the kind of Block option (2 in this case because used in the response) followed by a colon, and then the block number (NUM), the more bit (M = 0 means last block), and block size exponent ($2^{*(SZX+4)}$) separated by slashes. The Length 64 is used with SZX= 2 to avoid IP fragmentation. The CoAP Request is sent with confirmable (CON) option and the content format of the Response is /application/cacerts.

```

GET [192.0.2.1:8085]/est/crts      -->
      <-- (2:0/1/39) 2.05 Content
GET URI (2:1/1/39)                  -->
      <-- (2:1/1/39) 2.05 Content
      |
      |
      |
GET URI (2:65/1/39)                -->
      <-- (2:65/0/39) 2.05 Content

```

For further detailing the CoAP headers of the first two blocks are written out.

The header of the first GET looks like:

```
Ver = 1
T = 0 (CON)
Code = 0x01 (0.1 GET)
Options
  Option1 (Uri-Host)
    Option Delta = 0x3 (option nr = 3)
    Option Length = 0x9
    Option Value = 192.0.2.1
  Option2 (Uri-Port)
    Option Delta = 0x4 (option nr = 3+4=7)
    Option Length = 0x4
    Option Value = 8085
  Option3 (Uri-Path)
    Option Delta = 0x4 (option nr = 7+4=11)
    Option Length = 0x9
    Option Value = /est/crts
Payload = [Empty]
```

The header of the first response looks like:

```
Ver = 1
T = 2 (ACK)
Code = 0x45 (2.05 Content.)
Options
  Option1 (Content-Format)
    Option Delta = 0xC (option 12)
    Option Length = 0x2
    Option Value = TBD1
  Option2 (Block2)
    Option Delta = 0xB (option 23 = 12 + 11)
    Option Length = 0x1
    Option Value = 0x0A (block number = 0, M=1, SZX=2)
Payload =
30233906092a6206734107028c2a3023260201013100300b06092a6206734107018
c0c3020bb302063c20102020900a61e75193b7acc0d06092a6206734101
```

The second Block2:

```
Ver = 1
T = 2 (means ACK)
Code = 0x45 (2.05 Content.)
Options
  Option1 (Content-Format)
    Option Delta = 0xC (option 12)
    Option Length = 0x2
    Option Value = TBD1
  Option2 (Block2)
    Option Delta = 0xB (option 23 = 12 + 11)
    Option Length = 0x1
    Option Value = 0x1A (block number = 1, M=1, SZX=2)
Payload =
05050030
1b31193017060355040313106573744578616d706c654341204f774f301e170d313
3303530393033353333315a170d3134303530393033353333315a
```

The 40th and final Block2:

```
Ver = 1
T = 2 (means ACK)
Code = 0x21
Options
  Option1 (Content-Format)
    Option Delta = 0xC (option 12)
    Option Length = 0x2
    Option Value = TBD1
  Option2 (Block2)
    Option Delta = 0xB (option 23 = 12 + 11)
    Option Length = 0x2
    Option Value = 0x272 (block number = 39, M=0, SZX=2)
Payload = 73a30d0c006343116f58403100
```

Authors' Addresses

Sandeep S. Kumar
Philips Lighting Research
High Tech Campus 7
Eindhoven 5656 AE
NL

Email: ietf@sandeep.de

Peter van der Stok
Consultant

Email: consultancy@vanderstok.org

Panos Kampanakis
Cisco Systems

Email: pkampana@cisco.com

Martin Furuhed
Nexus Group

Email: martin.furuhed@nexusgroup.com

Shahid Raza
RISE SICS
Isafjordsgatan 22
Kista, Stockholm 16440
SE

Email: shahid@sics.se