ACE
Internet-Draft
Intended status: Standards Track
Expires: September 10, 2017

S. Kumar
Philips Lighting Research
P. van der Stok
Consultant
P. Kampanakis
Cisco Systems
M. Furuhed
Nexus Group
S. Raza
RISE SICS
March 9, 2017

EST over secure CoAP (EST-coaps)
draft-vanderstok-ace-coap-est-01

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 minor extensions to CoAP are needed to enable EST-coaps.

Status of This Memo

Copyright Notice

Table of Contents

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. For that reason, full PKI messages are not supported in 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, 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:

- Replacement of TLS by DTLS and HTTP by CoAP, resulting in:

   - DTLS-secured CoAP sessions between EST-coaps client and EST-coaps server.

- 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]).

- EST-coaps does not support full PKI request messages[RFC5272].

- EST-coaps specifies the BRSKI extensions over CoAP as specified in section 5 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 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 proxy and RA or is part of RA.

EST-coaps transports Public keys and certificates. 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

- 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;

- 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.

```
+------------------------------------------------+
|    EST request/response messages               |
+------------------------------------------------+
|    CoAP for message transfer and signaling     |
+------------------------------------------------+
|    DTLS for transport security                 |
+------------------------------------------------+
|    UDP for transport                           |
+------------------------------------------------+
```

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:

- Simple enroll and reenroll.

- CA certificate retrieval.

- CSR Attributes request messages.

- Server-side key generation messages.

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 "core.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, but it is recommended that the value "/est" is used, where possible. The example below shows the discovery of the presence and location of management data.

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

RES: 2.05 Content </est>; rt="core.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	/crts
	/simpleenroll	/sen
	/simplereenroll	/sren
	/csrattrs	/att
	/serverkeygen	/skg
/requestvoucher		/rv
/voucherstatus		/vs
/enrollstatus		/es

Table 1

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

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 CBOR byte string 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 CBOR byte string (h’xxxx’ in Diagnostic JSON) for all EST-coaps CoAP payloads.

4.3. Message Bindings

This section describes BRSKI to CoAP message mappings.

All /crts, /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.

Appendix A includes some practical examples of HTTP response codes from EST translated to CoAP.

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". In order 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 with each DTLS datagram. 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].

EST-coaps does not support full PKI Requests. Consequently, the fullcmc request of section 4.3 of [RFC7030] and response MUST NOT be supported by EST-coaps.

Channel-binding information for linking proof-of-identity with message-based proof-of-possession is optional for EST-coaps. Given that CoAP and DTLS can provide proof of identity for EST-coaps clients and server, simple PKI messages can be used 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].

When proof-of-possession is desired, a set of actions are required regarding the use of tls-connect, described in section 3.5 in [RFC7030] -- Linking Identity and POP Information. The tls-unique information translates to the contents of the first "Finished" message in the TLS handshake between server and client. The client is then supposed to add this "Finished" message as a ChallengePassword to the PKCS#10 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 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].

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

[EDNOTE: This section to be populated. It will address how proxying can take place by an entity that resides at the edge of the CoAP network, such as the Registrar, and can reach the BRSKI server residing in a traditional "TCP setting". It makes sense to mention the properties that the proxy has to fulfill.]

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 then 2s, but in this case they should send a CoAP ACK every 2s while processing.]

8. IANA Considerations

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: CBOR byte string

    * 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: CBOR byte string
- 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: CBOR byte string
- 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: CBOR byte string
- Security considerations: As defined in this specification
- Published specification: [RFC5967]
- Applications that use this media type: ANIMA bootstrap (BRSKI) and EST
-
  - application/pkcs12
  - Type name: application
  - Subtype name: pkcs12
  - ID: TBD5
  - Required parameters: None
  - Optional parameters: None
  - Encoding considerations: CBOR byte string
  - Security considerations: As defined in this specification
  - Published specification: IETF
  - Applications that use this media type: ANIMA bootstrap (BRSKI) and EST
-

   - application/auditnonce
   - Type name: application
   - Subtype name: auditnonce
   - ID: TBD6
   - Required parameters: None
   - Optional parameters: None
   - Encoding considerations: CBOR byte string
   - Security considerations: As defined in this specification
   - Published specification: BRSKI??
   - Applications that use this media type: ANIMA bootstrap (BRSKI)

-
   - application/authorizationvoucher
   - Type name: application
   - Subtype name: authorizationvoucher
   - ID: TBD7
   - Required parameters: None
   - Optional parameters: None
   - Encoding considerations: CBOR byte string
   - Security considerations: As defined in this specification
   - Published specification: BRSKI??
   - Applications that use this media type: ANIMA bootstrap (BRSKI)

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

o rt="core.est" needs registration with IANA.

[EDNOTE: This section will be expanded to include types needed that do not exist in CoAP.]

9. Security Considerations

[EDNOTE: This section to be populated. This document describes an existing protocol moved to CoAP and there should not be additional security concerns added beyond the protocol's or CoAP's specifics security considerations. The security considerations mentioned in EST applies also to EST-coaps. Specifically for server-side key generation, it introduces implications for the endpoints and their private keys, which will be covered here.]

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.

11. Change Log

-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-04 (work in progress), October 2016.

[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>.

[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>.

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., and K. Pister, "Minimal Security Framework for 6TiSCH", draft-ietf-6tisch-minimal-security-01 (work in progress), February 2017.

[I-D.ietf-anima-voucher]
Watsen, K., Richardson, M., Pritikin, M., and T. Eckert, "Voucher and Voucher Revocation Profiles for Bootstrapping Protocols", draft-ietf-anima-voucher-00 (work in progress), January 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-01 (work in progress), December 2016.

[I-D.selander-ace-cose-ecdhe]
Selander, G., Mattsson, J., and F. Palombini, "Ephemeral Diffie-Hellman Over COSE (EDHOC)", draft-selander-ace-cose-ecdhe-04 (work in progress), October 2016.

[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>.

[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>.

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

[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>.

[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

[EDNOTE: This section to be expanded to ensure it covers all BRSKI edge conditions.]

A.1. cacerts

In EST, an HTTPS cacerts message can be

```
GET /.well-known/est/cacerts HTTP/1.1
     User-Agent: curl/7.22.0 (i686-pc-linux-gnu) libcurl/7.22.0
                 OpenSSL/1.0.1 zlib/1.2.3.4 libidn/1.23 librtmp/2.3
     Host: 192.0.2.1:8085
     Accept: */*
```

The corresponding secure CoAP request is

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

with CoAP 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 = 0x9
     Option Value = /est/crts
  Payload = [Empty]
```

A 200 OK response with a cert in EST will then be

```
     200 OK
      Status: 200 OK
      Content-Type: application/pkcs7-mime
      Content-Transfer-Encoding: base64
      Content-Length: 4246 [EDNOTE: this example overflows and would
                            need fragmentation. Choose a better example.
                            Regardless we might need an CoAP option for
                            the content-length ie the CoAP payload?)

      MIIMOQYJKoZIhvcNAQcCoIIMKjCCDCYCAQExADALBgkqhkiG9w0BBwGgggwMMIIC
      +zCCAeOgAwIBAgIJAJpY3nUZO3qcMA0GCSqGSIb3DQEBBQUAMBsxGTAXBgNVBAMT
      ...
```

The corresponding CoAP response is

```
   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 note)

     Payload = h'123456789ABCDEF...'
```

A.2. enroll / reenroll

[EDNOTE: username/password authentication can be described here but is not a primary focus for BRSKI. It is important for generic EST exchanges but would an endpoint device with sufficient user interface to allow username/password input from an end user be required to use CoAP instead of a full HTTPS exchange?]

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

[EDNOTE: Include CoAP message examples.]

A.3. csrattr

[EDNOTE: Include CoAP message examples.]

A.4. enrollstatus

[EDNOTE: Include CoAP message examples.]

A.5. voucher_status

[EDNOTE: Include CoAP message examples.]

A.6. requestvoucher

[EDNOTE: Include CoAP message examples.]

A.7. requestlog

[EDNOTE: Include CoAP message examples.]

[EDNOTE: More examples can be added, for server-side key generation in CMS envelopes.]

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 3185 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 50 packets with a payload of 64 bytes each. The client sends an IPv6 packet containing the UDP datagram with the DTLS record that encapsulates the CoAP Request 50 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     -->
              <--   (2:0/1/64) 2.05 Content
    GET URI (2:1/1/64)                           -->
              <--   (2:1/1/64) 2.05 Content
                      |
                      |
                      |
     GET URI (2:49/1/64)                         -->
              <--   (2:49/0/64) 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:

[EDNOTE: The contents of the payload do not need to be written as they are encoded with DTLS into something unreadable.]

```
     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 = h'123456789ABCDEF...' (512 bytes)
```

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 = 0x1D (block number = 1, M=1, SZX=2)
     Payload = = h'123456789ABCDEF...' (512 bytes)
```

The 49th 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 = 0x312 (block number = 49, M=0, SZX=2)
     Payload = = h'123456789ABCDEF...' (512 bytes)
```

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