ACE Working Group
Internet-Draft
Intended status: Standards Track
Expires: October 6, 2016

G. Selander
J. Mattsson
F. Palombini
Ericsson AB

April 04, 2016

Ephemeral Diffie-Hellman Over COSE (EDHOC)
draft-selander-ace-cose-ecdhe-01

Abstract

This document specifies Diffie-Hellman key exchange with ephemeral keys embedded in messages encoded with the CBOR Encoded Message Syntax.

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 October 6, 2016.

Table of Contents

1. Introduction

Security at the application layer provides an attractive option for protecting Internet of Things (IoT) deployments, for example where transport layer security is not sufficient [I-D.hartke-core-e2e-security-reqs]. IoT devices may be constrained in various ways, including memory, storage, processing capacity, and energy [RFC7228]. A method for protecting individual messages at application layer, suitable for constrained devices, is provided by the CBOR Encoded Message Syntax (COSE, [I-D.ietf-cose-msg]).

In order for a communication session to provide forward secrecy, the communicating parties could run a Diffie-Hellman (DH) key exchange protocol with ephemeral keys, from which session keys are derived. This document specifies two instances of DH key exchange using COSE messages to transport the ephemeral public keys. The DH key exchange messages are authenticated using pre-established keys, either a

secret key (Section 3) or raw public keys (Section 4). The keys may be pre-established to client and server from a trusted third party, such as an Authorization Server [I-D.ietf-ace-oauth-authz]. Successful verification of the protocol messages, defined in this document, provides a method for proof-of-possession of the corresponding secret or private key [I-D.ietf-oauth-pop-key-distribution].

This document also specifies derivation of traffic keys, from the shared secret established through the DH key exchange with ephemeral keys. The key derivation is identical to TLS 1.3 [I-D.ietf-tls-tls13].

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]. These words may also appear in this document in lowercase, absent their normative meanings.

The key exchange messages are called "message_1" and "message_2", and the parties exchanging the messages are called "client" and "server", see Figure 1. The messages are encoded using the CBOR Encoded Message Syntax (COSE, [I-D.ietf-cose-msg]), and include an ephemeral public key ("g^x"/"g^y"). The shared secret "g^(xy)" is used to derive a key called "traffic_secret_0" using the terminology of TLS 1.3 [I-D.ietf-tls-tls13].

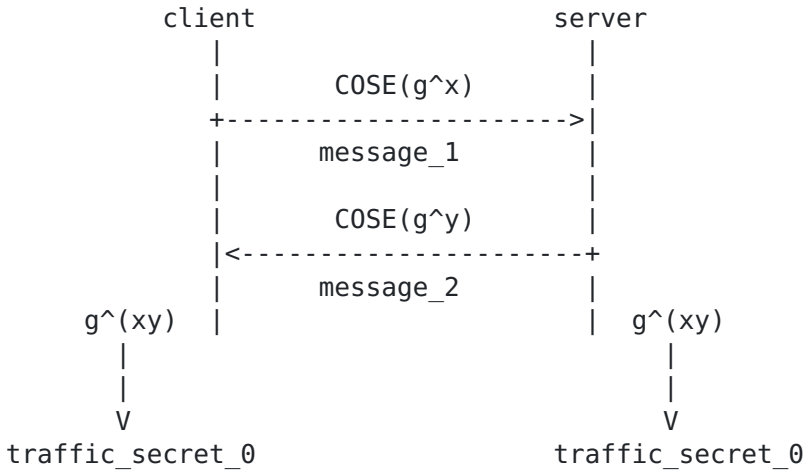

Figure 1: Diffie-Hellman key exchange and key derivation

Most keys used in this document have an associated identifier. The identifiers used in the document are placeholders for values of the identifiers. The following key identifiers/value representations are used in the draft:

- kid_x and kid_y represent the values of the key identifiers of the ECDH ephemeral public keys of the client and server, respectively.
  - kid_x is a sequence number used for replay protection of message_1.
  - kid_y is used to identify a resulting traffic key, as a means for the server to ensure that different clients establishing traffic keys using this method have different identifiers.
- kid_0 represents the value of the key identifier of the pre-shared key between client and server (Section 3).
- kid_c and kid_s represent the values of the key identifiers of the static public keys of the client and server, respectively (Section 4).
  - kid_c and kid_s are used to identify client and server, respectively.

Key Identifier	Key	Use
kid_x	g^x	ECDH ephemeral public key of the client
kid_y	g^y	ECDH ephemeral public key of the server
kid_0	PSK	Pre-shared key (Section 3)
kid_c	PKc	Static client public key (Section 4)
kid_s	PKs	Static server public key (Section 4)

Figure 2: Notation of keys and key identifiers.

2. ECDH Public Keys

This section defines the formatting of the ephemeral public keys g^x and g^y.

2.1. COSE_Key Formatting

The ECDH ephemeral public key SHALL be formatted as a COSE_Key with the following fields and values:

- o kty: The value SHALL be 2 (Elliptic Curve Keys)
- o kid:
- o crv: The value 1 SHALL be supported by the server (NIST P-256 a.k.a. secp256r1 [RFC4492])
- o x:
- o y: The value SHOULD be boolean.

[TODO: Consider replacing P-256 with Curve25519]

2.2. Example: ECDH Public Key

An example of COSE_Key structure, representing an ECDH public key, is given in Figure 3, using CBOR's diagnostic notation. In this example, the ephemeral key is identified by a 4 bytes 'kid'.

```
/ ephemeral / -1:{
            / kty / 1:2,
            / kid / 2:h'78f67901',
            / crv / -1:1,
            / x / -2:h'98f50a4ff6c05861c8860d13a638ea56c3f5ad7590b
            bfbf054e1c7b4d91d6280',
            / y / -3:true
          }
```

Figure 3: Example of an ECDH public key formatted as a COSE_Key

The equivalent CBOR encoding is: h'a120a50102024478f67901200121582098 f50a4ff6c05861c8860d13a638ea56c3f5ad7590bbfbf054e1c7b4d91d628022f5', which has a size of 50 bytes.

3. Authentication with Pre-Shared Keys

This section defines the DH key exchange protocol messages, when the exchange is authenticated with Message Authentication Codes (MACs) calculated with a pre-shared key.

The client and server are assumed to have a pre-shared key, PSK, the value of its identifier is represented by kid_0.

3.1. Message 1 with PSK

message_1 contains the client's ephemeral public key, g^x, and a MAC over g^x, calculated with the pre-shared key.

Before sending message_1, the client SHALL generate a fresh ephemeral ECDH key pair. The ephemeral public key, g^x, SHALL be formatted as in Section 2. The 'kid' value kid_x of the ephemeral public key g^x SHALL be a sequence number increased by 1 for each message_1 associated to kid_0. Note that each ephemeral ECDH key pair SHALL NOT be re-used with any other node than the one it was initially generated for.

The key identifier kid_0 SHALL be unique for client and server.

message_1 SHALL have the COSE_Mac0_Tagged structure [I-D.ietf-cose-msg] with the following fields and values:

- Header
  - Protected
    - Alg: 4 (HMAC 256/64)
    - Kid: kid_0
  - Unprotected: Empty, except for the case specified in Appendix B
- Payload: g^x (with 'kid' = kid_x, which is the sequence number)
- Tag: As in section 6.3 of [I-D.ietf-cose-msg]

[TODO: Error handling]

3.2. Example: Message 1 with PSK

An example of COSE encoding for message_1 is given in Figure 4 using CBOR's diagnostic notation. In this example, kid_0, the identifier of PSK is 4 bytes, and kid_x is one byte.

```
996(
  [
    / protected / h'a201040444e19648b5' / {
        / alg / 1:4, / HMAC 256//64 /
        / kid / 4:h'e19648b5' / kid_0
      } / ,
    / unprotected / {},
    / payload / h'a120a501020241032001215820 98f50a4ff6c05861c8860d13
    a638ea56c3f5ad7590bbfbf054e1c7b4d91d628022f5' / COSE_Key g^x / {
       / ephemeral / -1:{
         / kty / 1:2,
         / kid / 2:h'03', / kid_x
         / crv / -1:1,
         / x / -2:h'98f50a4ff6c05861c8860d13a638ea56c3f5ad7590bbfb
         f054e1c7b4d91d6280',
         / y / -3:true
       }
     } / ,
    / tag / h'e77fe81c66c3b5c0'
  ]
)
```

Figure 4: Example of message_1 authenticated with PSK

The equivalent CBOR encoding is: h'd903e48449a201040444e19648b5a0582c a120a501020241032001215820 98f50a4ff6c05861c8860d13a638ea56c3f5ad7590b bfbf054e1c7b4d91d628022f548e77fe81c66c3b5c0', which has a size of 73 bytes.

3.3. Message 2 with PSK

message_2 contains the server's ephemeral public key, g^y, and a MAC over g^y and message_1, calculated with the pre-shared key.

Before sending message_2, the server SHALL verify message_1 using the pre-shared key, PSK, and that the kid_x is greater than previously verified message_1 associated to kid_0. The server SHALL generate a fresh ephemeral ECDH key pair. The ephemeral public key, g^y, SHALL be formatted as in Section 2, its identifier (kid_y) SHALL be unique among key identifiers used for traffic keys by the server.

At reception, the client SHALL verify message_2 using the pre-shared key PSK and the sent message_1.

message_2 SHALL have the COSE_Mac0_Tagged structure [I-D.ietf-cose-msg] with the following fields and values:

o Header

  * Protected

     + Alg: 4 (HMAC 256/64)

     + Kid: kid_0

  * Unprotected: empty

- Payload: g^y (with 'kid' = kid_y)

- external_aad: message_1

- Tag: as in [I-D.ietf-cose-msg], including the external_aad in the MAC_structure.

[TODO: Error handling]

3.4. Example: Message 2 with PSK

An example of COSE encoding for message_2 is given in Figure 5 using CBOR's diagnostic notation. In this example, kid_0, the identifier of PSK, and kid_y, the identifier of the server's ephemeral public key, is 4 bytes.

```
 996(
   [
     / protected / h'a201040444e19648b5' / {
         / alg / 1:4, / HMAC 256//64 /
         / kid / 4:h'e19648b5' / kid_0
       } / ,
     / unprotected / {},
     / payload / h'a120a5010202442edb61f92001215820acbee6672a28340a
     ffce41c721901ebd7868231bd1d86e41888a07822214050022f5'
     / COSE_Key g^y / {
        / ephemeral / -1:{
          / kty / 1:2,
          / kid / 2:h'2edb61f9', / kid_y
          / crv / -1:1,
          / x / -2:h'acbee6672a28340affce41c721901ebd7868231bd1d
          86e41888a078222140500',
          / y / -3:true
        }
      } / ,
     / tag / h'6113268ad246f2c9'
   ]
 )
```

Figure 5: Example of message_2 authenticated with PSK

The equivalent CBOR encoding is: h'd903e48449a201040444e19648b5a05832 a120a4010202481e6f0c642001215820acbee6672a28340affce41c721901ebd78682 31bd1d86e41888a07822214050022f5486113268ad246f2c9', which has a size of 76 bytes.

3.5. Key Derivation

The client and server SHALL derive "traffic_secret_0" from the information available through the key exchange, as described in this section. The key derivation is identical to Section 7 of [I-D.ietf-tls-tls13], using the PSK + ECDHE operational mode and HKDF [RFC5869] with SHA-256:

- The Static Secret (SS) SHALL be the pre-shared key
- The Ephemeral Secret (ES) SHALL be the ECDH shared secret, generated from the ephemeral keys, as specified in section 7.3.3. of [I-D.ietf-tls-tls13]
- The generic string "TLS 1.3, " in HkdfLabel (Section 7.1) SHALL be replaced by "EDHOC, "
- The handshake_hash is replaced by the exchange_hash = SHA-256(message_1 + message_2), where '+' denotes concatenation of octet strings

The procedure for deriving "traffic_secret_0" in Section 7 in [I-D.ietf-tls-tls13] SHALL be followed. The "traffic_secret_0" SHALL be identified by the identifier of the server's ephemeral public key (kid_y).

Appendix C provides an example of how to derive a security context from "traffic_secret_0".

4. Authentication with Raw Public Keys

This section defines the DH key exchange protocol messages, when the exchange is authenticated with signatures calculated with Pre-Established Raw Public Keys (RPK).

- The client's static public key, denoted PKc, is pre-established to the server, and SHALL be uniquely identified at the server by kid_c.
- The server's static public key, denoted PKs, is pre-established to the client, and SHALL be uniquely identified at the client by kid_s.

4.1. Message 1 with RPK

message_1 contains the client's ephemeral public key, g^x, and a signature over g^x, computed with the client's static private key.

Before sending message_1, the client SHALL generate a fresh ephemeral ECDH key pair. The client's ephemeral public key, g^x, SHALL be formatted as in Section 2. The 'kid' value kid_x of the ephemeral public key g^x SHALL be a sequence number increased by 1 for each message_1 associated to kid_c. Note that each ephemeral ECDH key pair SHALL NOT be re-used with any other node than the one it was initially generated for.

message_1 SHALL have the COSE_Sign_Tagged structure [I-D.ietf-cose-msg], with the following fields and values:

- Header
  - Protected: empty
  - Unprotected: empty, except in the specified in Appendix B
- Payload: g^x (with 'kid' = kid_x, which is the sequence number)
- Signatures
  - Protected
    - Alg: -7 (ECDSA 256)
    - Kid: kid_c
  - Unprotected: empty
  - Signature: as in {{I-D.ietf-cose-msg}
- external_aad: kid_s

[TODO: Error handling]

4.2. Example: Message 1 with RPK

An example of COSE encoding for message_1 is given in Figure 6, using CBOR's diagnostic notation. In this example, the size of the identifiers of the static public keys kid_c and kid_s are 4 bytes, and the identifier of the client's ephemeral key, kid_x is 1 byte.

```
991(
  [
    / protected / h'',
    / unprotected / {},
    / payload / h'a120a501020241122001215820 98f50a4ff6c05861c8860d13
    a638ea56c3f5ad7590bbfbf054e1c7b4d91d628022f5' / COSE_Key g^x / {
       / ephemeral / -1:{
         / kty / 1:2,
         / kid / 2:h'12', / kid_x
         / crv / -1:1,
         / x / -2:h'98f50a4ff6c05861c8860d13a638ea56c3f5ad7590bbfb
         f054e1c7b4d91d6280',
         / y / -3:true
       }
     } / ,
    / signatures / [
      [
        / protected / h'a201260444c150d41c' / {
            / alg / 1:-7, / ECDSA 256 /
            / kid / 4:h'c150d41c', / kid_c /
          } / ,
        / unprotected / {},
        / signature / h'eae868ecc1276883766c5dc5ba5b8dca25dab3c2e56a
51ce5705b793914348e14eea4aee6e0c9f09db4ef3ddeca8f3506cd1a98a8fb64327
be470355c9657ce0'
      ]
    ]
  ]
)
```

Figure 6: Example of message_1 authenticated by the client

The equivalent CBOR encoding is: d903df8440a0582fa120a501020241122001
21582098f50a4ff6c05861c8860d13a638ea56c3f5ad7590bbfbf054e1c7b4d91d628
022f5818349a201260444c150d41ca05840eae868ecc1276883766c5dc5ba5b8dca25
dab3c2e56a51ce5705b793914348e14eea4aee6e0c9f09db4ef3ddeca8f3506cd1a98
a8fb64327be470355c9657ce0, which has a size of 134 bytes.

4.3. Message 2 with RPK

message_2 contains the server's ephemeral public key, g^y, and a signature over g^y and message_1, computed with the server's static private key.

Before sending message_2, the server SHALL verify message_1, and that the kid_x is greater than previously verified with public key corresponding to kid = kid_c. The server SHALL generate a fresh ephemeral ECDH key pair, formatted as in Section 2, the value of the

key identifier (kid_y) SHALL be unique among key identifiers used for traffic keys by the server.

At reception, the client SHALL verify message 2 using the pre-shared key PSK and the sent message_1.

message_2 SHALL have the COSE_Sign_Tagged structure [I-D.ietf-cose-msg] with the following fields and values:

- Header
  - Protected: empty
  - Unprotected: empty
- Payload: g^y (with 'kid' = kid_y)
- Signatures
  - Protected
    - Alg: -7 (ECDSA 256)
    - Kid: kid_s
  - Unprotected: empty
  - Signature: as in {{I-D.ietf-cose-msg}
- external_aad: message_1

[TODO: Error handling]

4.4. Example: Message 2 with RPK

An example of COSE encoding for Message 2 is given in Figure 7, using CBOR's diagnostic notation. In this example, the size of the identifiers of the public keys: kid_x, kid_y, and kid_s are 4 bytes.

```
991(
  [
    / protected / h'',
    / unprotected / {},
    / payload / h'a120a5010202442edb61f9200121582098f50a4ff6c0
    5861c8860d13a638ea56c3f5ad7590bbfbf054e1c7b4d91d628022f5'
    / COSE_Key g^y / {
       / ephemeral / -1:{
         / kty / 1:2,
         / kid / 2:h'2edb61f9', / kid_y
         / crv / -1:1,
         / x / -2:h'acbee6672a28340affce41c721901ebd7868231bd1d
         86e41888a078222140500',
         / y / -3:true
       }
     } / ,
    / signatures / [
      [
        / protected / h'a2012604447a2af164' / {
            / alg / 1:-7, / ECDSA 256 /
            / kid / 4:h'7a2af164', / kid_s /
          } / ,
        / unprotected / {},
        / signature / h'2374e27a3d9eeb4f66c5dc5ba5b8dca25dab3c2e56a5
51ce5705b793914348e14eea4aee6e0c9f09db4ef3ddeca8f3506cd1a98a8fb64327
be470355c9657ce0'
      ]
    ]
  ]
)
```

Figure 7: Example of message_2 authenticated by server

The equivalent CBOR encoding is: h'd903df8440a05832a120a5010202442edb
61f9200121582098f50a4ff6c05861c8860d13a638ea56c3f5ad7590bbfbf054e1c7b
4d91d628022f5818349a2012604447a2af164a058402374e27a3d9eeb4f66c5dc5ba5
b8dca25dab3c2e56a551ce5705b793914348e14eea4aee6e0c9f09db4ef3ddeca8f35
06cd1a98a8fb64327be470355c9657ce0', which has a size of 137 bytes.

4.5. Key Derivation

The client and server SHALL derive "traffic_secret_0" from the information available through the key exchange, as described in this section. The key derivation is identical to Section 7 of [I-D.ietf-tls-tls13], using the ECDHE operational mode and HKDF [RFC5869] with SHA-256:

- The Static Secret (SS) and the Ephemeral Secret (ES) SHALL be the ECDH shared secret, generated from the ephemeral keys, as specified in section 7.3.3. of [I-D.ietf-tls-tls13]

- The generic string "TLS 1.3, " in HkdfLabel (Section 7.1) SHALL be replaced by "EDHOC, "

- The handshake_hash is replaced by the exchange_hash = SHA-256(message_1 + message_2), where '+' denotes concatenation of octet strings

The procedure for deriving "traffic_secret_0" in Section 7 in [I-D.ietf-tls-tls13] SHALL be followed. The "traffic_secret_0" SHALL be identified with the value of the 'kid' field of the server's ephemeral public key (kid_y).

Appendix C provides an example of how to derive a security context from "traffic_secret_0".

5. Security Considerations

After the key derivation is completed, the intermediate computed values should be securely deleted, along with any ephemeral ECDH secrets.

The choice of key length used in the different algorithms needs to be harmonized, so that right security level is maintained throughout the calculations.

The identifier of the ephemeral key of the client is used for replay protection or client requests.

The verification of client and server identity is important, the static public key identifiers serve the role to identify the entity holding the private key. In case of pre-shared key, the key identifier serves the role to identify the "other" party.

With the current protocol, key confirmation of the Diffie-Hellman shared secret/traffic keys is performed when the keys are successfully used. The addition of key confirmation to the protocol is for further study.

A two-pass authenticated key exchange protocol can at most provide a weak form of forward secrecy in the following sense; when agents' long-term keys are compromised, the secrecy of previously established session-keys is guaranteed, but only for sessions in which the adversary did not actively interfere.

The algorithm for generating the shared secret has important security consequences. In this part we follow closely TLS 1.3 and may inherit any vulnerabilities in that construction.

[TODO: Expand on the security considerations in a future version of the draft]

6. Privacy Considerations

[TODO:]

7. IANA Considerations

8. Acknowledgments

The authors are grateful to Ilari Liusvaara for pointing out vulnerabilities in version -00. The authors wish to thank Ludwig Seitz for timely review and helpful comments.

9. References

9.1. Normative References

[I-D.ietf-cose-msg]
Schaad, J., "CBOR Encoded Message Syntax", draft-ietf-cose-msg-11 (work in progress), March 2016.

[I-D.ietf-tls-tls13]
Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", draft-ietf-tls-tls13-12 (work in progress), March 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>.

9.2. Informative References

[I-D.hartke-core-e2e-security-reqs]
Selander, G., Palombini, F., Hartke, K., and L. Seitz, "Requirements for CoAP End-To-End Security", draft-hartke-core-e2e-security-reqs-00 (work in progress), March 2016.

[I-D.ietf-ace-oauth-authz]
Seitz, L., Selander, G., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "Authorization for the Internet of Things using OAuth 2.0", draft-ietf-ace-oauth-authz-01 (work in progress), February 2016.

[I-D.ietf-oauth-pop-key-distribution]
Bradley, J., Hunt, P., Jones, M., and H. Tschofenig, "OAuth 2.0 Proof-of-Possession: Authorization Server to Client Key Distribution", draft-ietf-oauth-pop-key-distribution-02 (work in progress), October 2015.

[I-D.selander-ace-object-security]
Selander, G., Mattsson, J., Palombini, F., and L. Seitz, "Object Security of CoAP (OSCOAP)", draft-selander-ace-object-security-04 (work in progress), March 2016.

[I-D.wahlstroem-ace-cbor-web-token]
Wahlstroem, E., Jones, M., and H. Tschofenig, "CBOR Web Token (CWT)", draft-wahlstroem-ace-cbor-web-token-00 (work in progress), December 2015.

[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>.

[RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010, <http://www.rfc-editor.org/info/rfc5869>.

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

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

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <http://www.rfc-editor.org/info/rfc7519>.

Appendix A. Implementing EDHOC with CoAP

The DH key exchange specified in this document can be implemented as a CoAP [RFC7252] message exchange. A strawman is sketched here.

The client makes the following request:

- The request method is POST
- Content-Format is "application/cose+cbor"
- The Uri-Path is "edhoc"
- The Payload is message_1

The server performs the verifications of the COSE object as specified in [I-D.ietf-cose-msg]. If successful, then the server provides the following response:

- The response Code is 2.04 (Changed)
- The Payload is message_2

Appendix B. Integrating EDHOC with ACE

A pre-requisite for using the DH key exchange protocols in Section 3 and Section 4 of this document is that some keys are pre-established in client and server. The ACE framework [I-D.ietf-ace-oauth-authz] specifies how an authorization server (AS) supports the establishment of keys in client and (resource) server, either a shared secret key or each others' public keys, which is exactly what is required in Section 3 and Section 4, respectively.

The ACE protocol specifies a client making a 'token request' to the AS to retrieve an access token (JWT [RFC7519], or CWT [I-D.wahlstroem-ace-cbor-web-token]) containing authorization information about the client regarding a certain resource on a certain server. The client can then transfer the access token to the server in the CoAP payload of the following request:

POST /authz-info

The access token may also contain a shared secret key or the public key of the client, for use by the server.

In case of symmetric keys, the AS generates this key and protects it for the client and server, after which the protocol in Section 3 can start.

In case of asymmetric keys, the ACE framework allows the client to include its public key in the 'token request', which results in the key being included in the access token reaching the server. The server's public key can be assumed to be known to the AS, which can therefore provide also this information to the client in the response to the token request.

The transfer of the access token as defined in [I-D.ietf-ace-oauth-authz] can be combined with the execution of EDHOC, for example, by including the access token in the Unprotected of Header of message_1. A dedicated resource could be defined for this combined message exchange, for example:

POST /authz-info-edhoc

The strawman in Appendix A applies also to this case.

Appendix C. Deriving Security Context for OSCOAP

In this section we show how to establish security context for OSCOAP [I-D.selander-ace-object-security], using the method specified in this document.

We assume that "traffic_secret_0" has been established, e.g. as described in Appendix B using a DH key exchange specified in this document. OSCOAP requires traffic keying material Client/Server Write Key/IV to be established at client and server, see section 3 of [I-D.selander-ace-object-security]. The computation of keying material mimics the traffic key calculation of Section 7.3 in TLS 1.3 [I-D.ietf-tls-tls13] using HKDF with SHA-256 and the following parameters:

- Secret = traffic_secret_0
- phase = "application data key expansion"
- purpose = "client write key" / "server write key" / "client write IV" / "server write IV"
- handshake_context = message_1 + message_2, the concatenation of the exchanged messages
- key_length for key and IV is algorithm specific.

The first three bullets are identical to TLS 1.3.

With the mandatory OSCOAP algorithm AES-CCM-64-64-128 (see Section 10.2 in [I-D.ietf-cose-msg]), key_length for the keys is 128 bits and key_length for the static IVs is 56 bits.

The Context Identifier (Cid) is set to the key identifier of traffic_secret_0 (i.e. kid_y, using the terminology of Section 3 and Section 4).

Authors' Addresses

Goeran Selander
Ericsson AB
Farogatan 6
Kista SE-16480 Stockholm
Sweden

Email: goran.selander@ericsson.com

John Mattsson
Ericsson AB
Farogatan 6
Kista SE-16480 Stockholm
Sweden

Email: john.mattsson@ericsson.com

Francesca Palombini
Ericsson AB
Farogatan 6
Kista SE-16480 Stockholm
Sweden

Email: francesca.palombini@ericsson.com