Ephemeral Diffie-Hellman Over COSE (EDHOC)
draft-ietf-lake-edhoc-00

Göran Selander, John Mattson, Francesca Palombini
IETF 108, LAKE WG, July 31, 2020
Introduction

— Lightweight authenticated key exchange protocol
— Matching the LAKE requirements [1]
  — Performance benchmarks
  — Reusing building blocks used by OSCORE [2]

— CBOR [3] for encoding
— COSE [4] for CBOR-encoded crypto, identification of algorithms, curves, credentials, etc.
— CoAP [5] for transport (default), and also handling message loss/re-ordering/de-duplication/fragmentation and certain DoS mitigation

NOTE: Separate presentation of security claims

[1] draft-ietf-lake-reqs
[2] RFC 8613
[3] RFC 7049
[4] RFC 8152
[5] RFC 7252
Applications

— General: Establish good quality keys for protecting communication between sensor/actuator and cloud/edge over constrained links

— Specific: Constrained device joins network
— Variations
  — Network conditions
    — Multi-hop constrained network
    — Network formation with large number of devices
  — Concurrent security application
    — Third party assisted authorization
    — Enrolment of operational device certificate
— Credential management
  — Pre-provisioned credentials
  — References to credentials
Protocol Messages

- 3 messages
- Additional error message
- Each message is a sequence of CBOR encoded data (RFC 8742)

Initiator

METHOD_CORR, SUITES_I, G_X, C_I, AD_1

message_1

C_I, G_Y, C_R, Enc(K_2e; ID_CRED_R, Signature_or_MAC_2, AD_2)

message_2

C_R, AEAD(K_3ae; ID_CRED_I, Signature_or_MAC_3, AD_3)

message_3

Responder

G_. = Public DH key
C_. = Connection id.
K_. = Symmetric key
AD_. = Auxiliary data
Diffie-Hellman, Encryption, Signature_or_MAC

- Fresh ephemeral public keys $G_X, G_Y$
- Message 2 has Enc (XOR), message 3 has AEAD
- Signature_or_MAC is either MAC-then-sign or MAC depending on authentication method
## Authentication Methods

- 4 methods in the initial scope

<table>
<thead>
<tr>
<th>Value</th>
<th>Initiator</th>
<th>Responder</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Signature Key</td>
<td>Signature Key</td>
<td>[[this document]]</td>
</tr>
<tr>
<td>1</td>
<td>Signature Key</td>
<td>Static DH Key</td>
<td>[[this document]]</td>
</tr>
<tr>
<td>2</td>
<td>Static DH Key</td>
<td>Signature Key</td>
<td>[[this document]]</td>
</tr>
<tr>
<td>3</td>
<td>Static DH Key</td>
<td>Static DH Key</td>
<td>[[this document]]</td>
</tr>
</tbody>
</table>

- Targeting IoT use cases, for example:
  - IEEE 802.1AR device certificates, device certificate enrolment (signature based)
  - Pre-provisioned RPK, minimal overhead (static DH based)
  - Mix of RPK and certificate (mixed static DH and signature)
Methods, Message Correlation, Connection Identifiers

- METHOD_CORR (int) = 4 * method + corr
  - method = 0, 1, 2, 3
  - corr = 0, 1, 2, 3 encodes correlation between messages provided by transport
- C_I, C_R (bstr/int): Explicit connection identifiers, may be omitted in case of message correlation
Cipher suites 1(3)

— Cipher suite = ordered set of COSE code points
  — For EDHOC: AEAD, hash, ECDH curve, signature algorithm, signature algorithm curve
  — For the application (OSCORE): AEAD and hash algorithm
  — Method defines how cipher suite is used

0. (10, -16, 4, -8, 6, 10, -16)
   (AES-CCM-16-64-128, SHA-256, X25519, EdDSA, Ed25519,
    AES-CCM-16-64-128, SHA-256)

1. (30, -16, 4, -8, 6, 10, -16)
   (AES-CCM-16-128-128, SHA-256, X25519, EdDSA, Ed25519,
    AES-CCM-16-64-128, SHA-256)

2. (10, -16, 1, -7, 1, 10, -16)
   (AES-CCM-16-64-128, SHA-256, P-256, ES256, P-256,
    AES-CCM-16-64-128, SHA-256)

3. (30, -16, 1, -7, 1, 10, -16)
   (AES-CCM-16-128-128, SHA-256, P-256, ES256, P-256,
    AES-CCM-16-64-128, SHA-256)
Cipher suites 2(3)

- suite = int
- SUITES_I : [ selected : suite, supported : 2* suite ] / suite
  - supported = cipher suites which the Initiator supports in order of preference
  - can be truncated at the end
  - selected = selected cipher suite by the Initiator

\[ \text{message}_1 \]
\[ C_I, G_Y, C_R, \text{Enc}(K_{2e}; ID_{CRED_R}, \text{Signature}_2, \text{AD}_2) \]
\[ \text{message}_2 \]
\[ C_R, \text{AEAD}(K_{3ae}; ID_{CRED_I}, \text{Signature}_3, \text{AD}_3) \]

G_. = Public DH key
C_. = Connection id.
K_. = Symmetric key
AD_. = Auxiliary data

IETF 108 | LAKE WG | 2020-07-31 | Page 9
Error Message, Cipher suites 3(3)

- error = (?, C_., ERR_MSG, ? SUITES_R)
  - ERR_MSG = tstr
  - SUITES_R : [ supported : 2* suite ] / suite
    - supported = cipher suites in SUITES_I supported by the Responder
  - Example of error message: Figure below shows Responder supporting cipher suite 6, but not 5

---

G_. = Public DH key
C_. = Connection id.
K_. = Symmetric key
AD_. = Auxiliary data

---

![Diagram of error message and cipher suites](image-url)
Credentials and Identifiers

— ID_CRED_I and ID_CRED_R are COSE header maps (CBOR maps using IANA "COSE Header Parameters")
— used to enable retrieval of credential (CRED_I and CRED_R, resp.)
— RPK may be identified with a ‘kid’ (RFC 8152)
— Certificates may be identified with ‘x5t’, ‘x5u’ (draft-ietf-cose-x509)
— CRED_I and CRED_R covered by Signature_or_MAC

<table>
<thead>
<tr>
<th>Initiator</th>
<th>Responder</th>
</tr>
</thead>
<tbody>
<tr>
<td>METHOD_CORR, SUITES_I, G_X, C_I, AD_1</td>
<td></td>
</tr>
<tr>
<td>message_1</td>
<td></td>
</tr>
<tr>
<td>C_I, G_Y, C_R, Enc(K_2e; ID_CRED_R, Signature_or_MAC_2, AD_2)</td>
<td></td>
</tr>
<tr>
<td>message_2</td>
<td></td>
</tr>
<tr>
<td>C_R, AEAD(K_3ae; ID_CRED_I, Signature_or_MAC_3, AD_3)</td>
<td></td>
</tr>
<tr>
<td>message_3</td>
<td></td>
</tr>
</tbody>
</table>
Keys

method \( (m) = 0, 1, 2, 3 \)

**DH-keys**

**Responder:**
- Ephemeral \( G_Y \)
- Static \( G_R \) \( (m = 1, 3) \)

**Initiator:**
- Ephemeral \( G_X \)
- Static \( G_I \) \( (m = 2, 3) \)

- **DH-shared secret**
- **Pseudo-random key**
- **Encryption/MAC key**

\( \text{TH}_2 = H(\text{message}_1, \text{data}_2), \text{where} \ \text{message}_2 = (\text{data}_2, \text{CIPHERTEXT}_2) \)

\( \text{Enc} \) (XOR) in message_2

\( \text{MAC}_2 \) (signed if \( m = 0, 2 \))

AEAD in message_3

\( \text{MAC}_3 \) (signed if \( m = 0, 1 \))

\( \text{EDHOC-Exporter}() \)

\( \text{TH}_4 = H(\text{TH}_3, \text{CIPHERTEXT}_3) \)
Auxiliary Data

— AD_1, AD_2, AD_3
— Integration points for EDHOC in lightweight security applications, such as
  — Third party authorization, e.g., draft-selander-ace-ake-authz (figure below)
  — Enrolment of device certificate
CoAP Transport and OSCORE Context

- EDHOC may be mapped to CoAP POST / 2.04 (Changed)

- Initiator may be CoAP Client or CoAP Server
  - The former assumed here

- OSCORE request/response may be included in second roundtrip, see draft-palombini-core-oscore-edhoc

<table>
<thead>
<tr>
<th>Client</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>+-------+ Header: POST (Code=0.02)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>POST</td>
</tr>
<tr>
<td></td>
<td>Content-Format: application/edhoc</td>
</tr>
<tr>
<td></td>
<td>Payload: EDHOC message_1</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;-------+ Header: 2.04 Changed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2.04</td>
</tr>
<tr>
<td></td>
<td>Payload: EDHOC message_2</td>
</tr>
<tr>
<td></td>
<td>+-------+ Header: POST (Code=0.02)</td>
</tr>
<tr>
<td></td>
<td>POST</td>
</tr>
<tr>
<td></td>
<td>Content-Format: application/edhoc</td>
</tr>
<tr>
<td></td>
<td>Payload: EDHOC message_3</td>
</tr>
<tr>
<td></td>
<td>&lt;-------+ Header: 2.04 Changed</td>
</tr>
<tr>
<td></td>
<td>2.04</td>
</tr>
</tbody>
</table>

- OSCORE security context:
  - Client SenderID = C_R
  - Server SenderID = C_I
  - Master Secret/Salt derived with EDHOC-Exporter()
  - Algs. according to cipher suite
Code

- Several implementations announced/in progress, including:
  - Univ. of Fribourg/Martin Disch
    - https://github.com/martindisch/oscore
  - Fraunhofer AISEC/Stefan Hristozov
  - Univ. of Murcia/Eduardo Ingles

- Code to generate test vectors
  - https://github.com/EricssonResearch/EDHOC/tree/master/Test%20Vectors