Using The Delegated CoAP Authentication and Authorization Framework (DCAF) With CBOR Encoded Message Syntax

draft-bergmann-ace-dcaf-cose

Olaf Bergmann, Stefanie Gerdes
{gerdes | bergmann}@tzi.org

Presented by Carsten Bormann
cabo@tzi.org

IETF-94, ACE Meeting, 2015-11-02
Motivation

- **draft-gerdes-ace-dcaf-authorize**
  - Secure exchange of authorization information.
  - Establish DTLS channel between constrained nodes.
  - Support of class-1 devices (RFC 7228).
  - Support cross-domain, multi-owner scenarios.

- **draft-bergmann-ace-dcaf-cose**
  - Re-use light-weight DCAF messaging
  - Support for application-level security using COSE objects
  - Enable *piggybacked protected content* use-case
Observation: DCAF Messages are not tied to DTLS
Nor is the Ticket Data

Access Request

CAM

SAM

Face:
[server authorization info]
nonce
[lifetime]

Client Information:
verifier (session key)
[client authorization info, nonce]
[lifetime]
DCAF-Messages can be protected with COSE

Example: Access Request

POST /client-authorize
Content-Format: application/cose+cbor
[ h'a1031862',      # protected { content_type: application/dcaf+cbor }  
   { alg: AES-CCM-16-64-128         # unprotected
     nonce: h'd6150b90e6f0eb5be42164062c', # nonce
   },
   h'{encrypted payload w/ tag}',   # encrypted DCAF payload
   # recipients:
   [ [ h'',                         # protected (absent for AE algorithm)
       { alg: A128KW,               # unprotected
         kid: h'383261622e6161733432' # context identifier: "82ab.aas42"
       },
       h'52ff9ed52d...'             # encrypted session key
   ] ]
]
DCAF-Messages can be protected with COSE

Example: Access Request

POST /client-authorize
Content-Format: application/cose+cbor

[ h'a1031862', # protected { content_type: application/dcaf+cbor }  
  { alg: AES-CCM-16-64-128 # unprotected  
    nonce: h'd6150b90e6f0eb5be42164062c', # nonce  
  } ,  
  h'\{encrypted payload w/ tag\}', # encrypted DCAF payload
# recipients:  
[ [ h'', # protected (absent for AE algorithm)
    { alg: A128KW,
    # unprotected
    kid: h'383261622e6161733432' # context identifier: "82ab.aas42"
    },
    h'52ff9ed52d...' # encrypted session key
  ] ]
]
Key Derivation

Security Association

transfer Ticket Face during setup

c
use session key from Verifier (direct or derived)

derive session key from Ticket Face and \( K_{S,SAM} \)

s
Convey Ticket Face from C to S

POST /authorize
Content-Format: application/cose+cbor
[ h'a1031862',       # protected { content_type: application/dcaf+cbor }
  { alg: HMAC 256/256 },          # unprotected
  h'{ SAI: ["/s/tempC" ...]}',    # DCAF payload wrapped in CBOR binary
  h'...',                      # tag: HMAC(options+protected+payload, secret)
  [ [ h'', {}, h'' ] ]          # recipients
]
Convey Ticket Face from C to S

POST /authorize
Content-Format: application/cose+cbor
[ h'a1031862',       # protected { content_type: application/dcaf+cbor }
  { alg: HMAC 256/256 },   # unprotected
  h'{ SAI: [ "/s/tempC" ... }', # DCAF payload wrapped in CBOR binary
  h'....',               # tag: HMAC(options+protected+payload, secret
  [ [ h'', {}, h'' ] ]    # recipients
]
Creation of Security Context

POST /authorize
Content-Format: application/cose+cbor
[ h'a1031862',  # protected { content_type: application/dcaf+cbor }
  { alg: HMAC 256/256 },  # unprotected
  h'{ SAI: [ "/s/tempC" ... }',  # DCAF payload wrapped in CBOR binary
  h'....',  # tag: HMAC(options+protected+payload, secret
  [ [ h'', {}, h'' ] ]  # recipients
]

2.01 Created
Content-Format: application/cose+cbor
Location-Path: 238dsa29
Authorization: [ h'a1031862',  # protected
  { alg: HMAC 256/256 },  # unprotected
  h'',  # empty payload
  h'....',  # tag: HMAC(options+protected, secret
  [ [ h'', {}, h'' ] ]  # recipients
]
Creation of Security Context

POST /authorize
Content-Format: application/cose+cbor
[ h'a1031862', # protected { content_type: application/dcaf+cbor }
  { alg: HMAC 256/256 }, # unprotected
  h'{ SAI: [ "/s/tempC" ... }', # DCAF payload wrapped in CBOR binary
  h'....', # tag: HMAC(options+protected+payload, secret
  [ [ h'', {}, h'' ] ] # recipients
]

2.01 Created
Content-Format: application/cose+cbor
Location-Path: 238dsa29
Authorization: [ h'a1031862', # protected
  { alg: HMAC 256/256 }, # unprotected
  h'', # empty payload
  h'....', # tag: HMAC(options+protected, secret
  [ [ h'', {}, h'' ] ] # recipients
]
Usage of Security Context

without payload

GET /r
Authorization: [ h'',
  { alg: HMAC 256/256,
    kid: h'3233386473613239'
  },
  nil,
  h'....',
  [ [ h'', {}, h'' ] ]
]
Usage of Security Context

without payload

```
GET /r
Authorization: [ h'',
  { alg: HMAC 256/256,
    kid: h'3233386473613239' },
nil,
h'....',
[ [ h'', {}, h'' ] ] ]
```

# protected (empty)
# unprotected
# context identifier: "238dsa29"

# payload (empty)
# tag: HMAC(options+protected, secret)
# recipients

as returned in Location-Path
Usage of Security Context

with payload

GET /r
Authorization: [ h'',
  { alg: HMAC 256/256,
    kid: h'3233386473613239'
  },
  nil,
  h'....',
  [ [ h'', {}, h'' ] ]
]

PUT /r
Content-Format: application/cose+cbor
[ h'a10300',
  { alg: HMAC 256/256,
    kid: h'3233386473613239'
  },
  h'48656c6c6f20576f726c6421',
  h'....',
  [ [ h'', {}, h'' ] ]
]
Usage of Security Context

with payload

GET /r
Authorization: [ h'',
  
  { alg: HMAC 256/256,
    kid: h'3233386473613239'
  },
  nil,
  h'....',
  [ [ h'', {}, h'' ] ]
]

PUT /r
Content-Format: application/cose+cbor
[ h'a10300',
  
  { alg: HMAC 256/256,
    kid: h'3233386473613239'
  },
  h'48656c6c6f20576f726c6421',
  h'....',
  [ [ h'', {}, h'' ] ]
]
Open Issues

- protection of changeable CoAP options (→ CoRE WG)
- describe how to apply key derivation methods from draft-ietf-cose-msg
Summary

- DCAF-COSE supports application-level security using COSE objects
- use plain COSE as described in draft-ietf-cose-msg
- two new CoAP options:
  - Authorization: convey authorization information
  - Authorization-Format: allow for future extensions
<table>
<thead>
<tr>
<th>Changes to COSE</th>
<th>DCAF-COSE</th>
<th>OSCOAP</th>
</tr>
</thead>
</table>
| use COSE as is (-06) no changes required | invent "Secure Message format" (COSE-profile in Appendix A) | invent "COSE Optimizations" that are not COSE-compatible (new message types, remove unprotected header, alg ...)
| Security Context | use parameter kid (identifies auth info and session key) | invent new parameter cid (identifies cipher suite, keys, alg-specific parameters, different for client and server: "typically identifies the sending party") |
| Replay protection | use parameter nonce (-> local time) | invent new parameter seq (-> sequence number, no freshness information) |
| Re-key | Server sends SAM Information Message | "out of scope" (Section 7.1) |
| Signaling | use existing payload types two new options (not critical due to usual content-format handling) | implicit, new payload type new critical option |
| Handling of unknown options | COSE extension parameter to signal required options | not supported |
| RFC 7252, 7641 options block-wise | needs more work in CoRE WG |