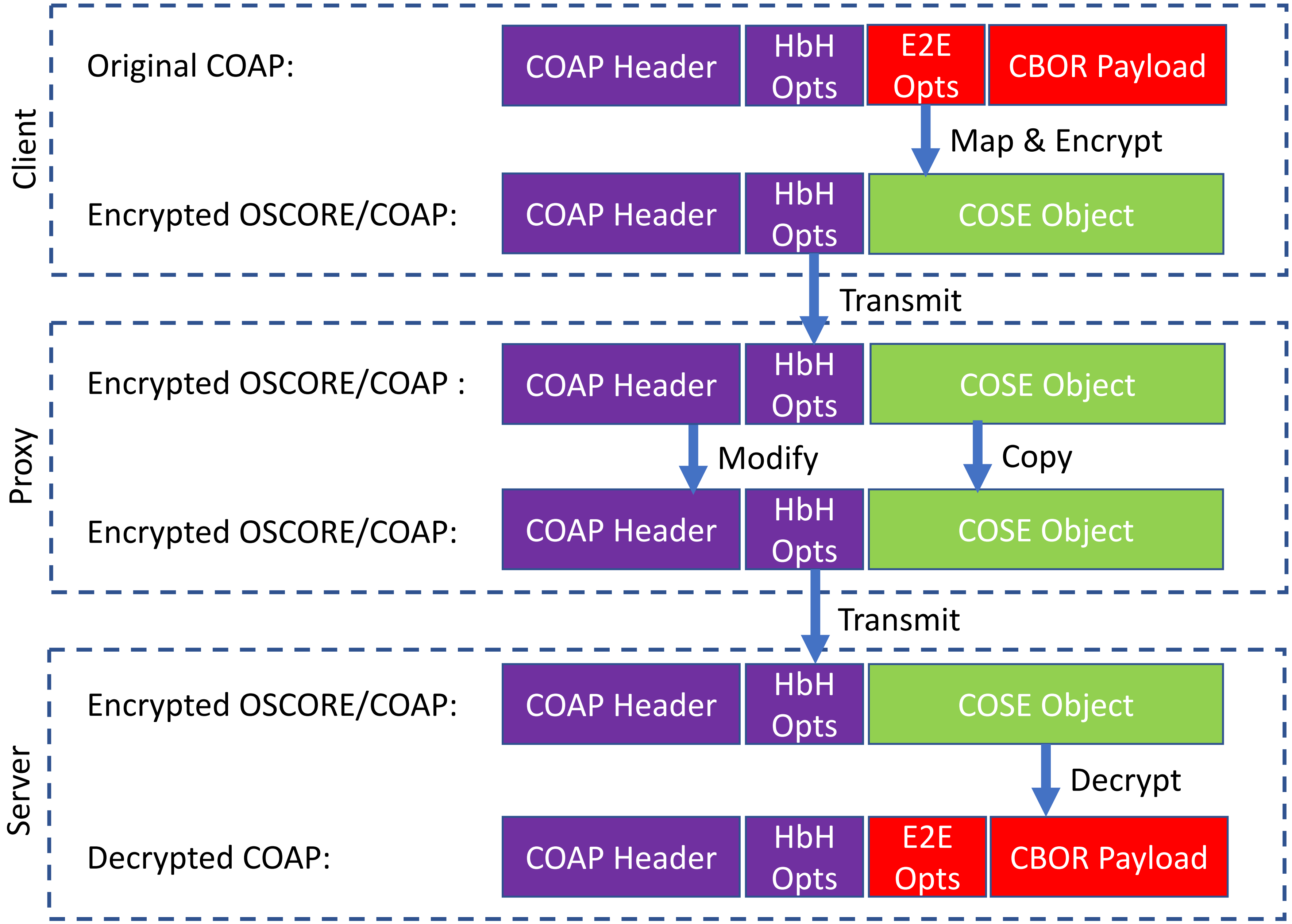
End-to-End Security with ~~OSCOAP~~ OSCORE

Dave Thaler

Object Security of COAP (OSCOAP)

- As of draft-ietf-core-object-security-04 (July) and earlier
- Summary:
    - COAP options categorized as end-to-end or hop-by-hop
    - End-to-end options signed and optionally encrypted in a CBOR payload (COSE object)
    - Intermediaries (no changes needed) pass payload unmodified like any other payload
- Issues:
    - Assumed all hops in the end-to-end path use COAP(S)
        - HTTP(S) was not supported, but OCF specs mention HTTP(S) as a transport
    - Also assumed all hops use the same version of COAP
        - Because COAP version was an end-to-end protected value

Example



Client

Original COAP: COAP Header | HbH Opts | E2E Opts | CBOR Payload

Map & Encrypt

Encrypted OSCORE/COAP: COAP Header | HbH Opts | COSE Object

Transmit

Proxy

Encrypted OSCORE/COAP : COAP Header | HbH Opts | COSE Object

Modify

Copy

Encrypted OSCORE/COAP: COAP Header | HbH Opts | COSE Object

Transmit

Server

Encrypted OSCORE/COAP: COAP Header | HbH Opts | COSE Object

Decrypt

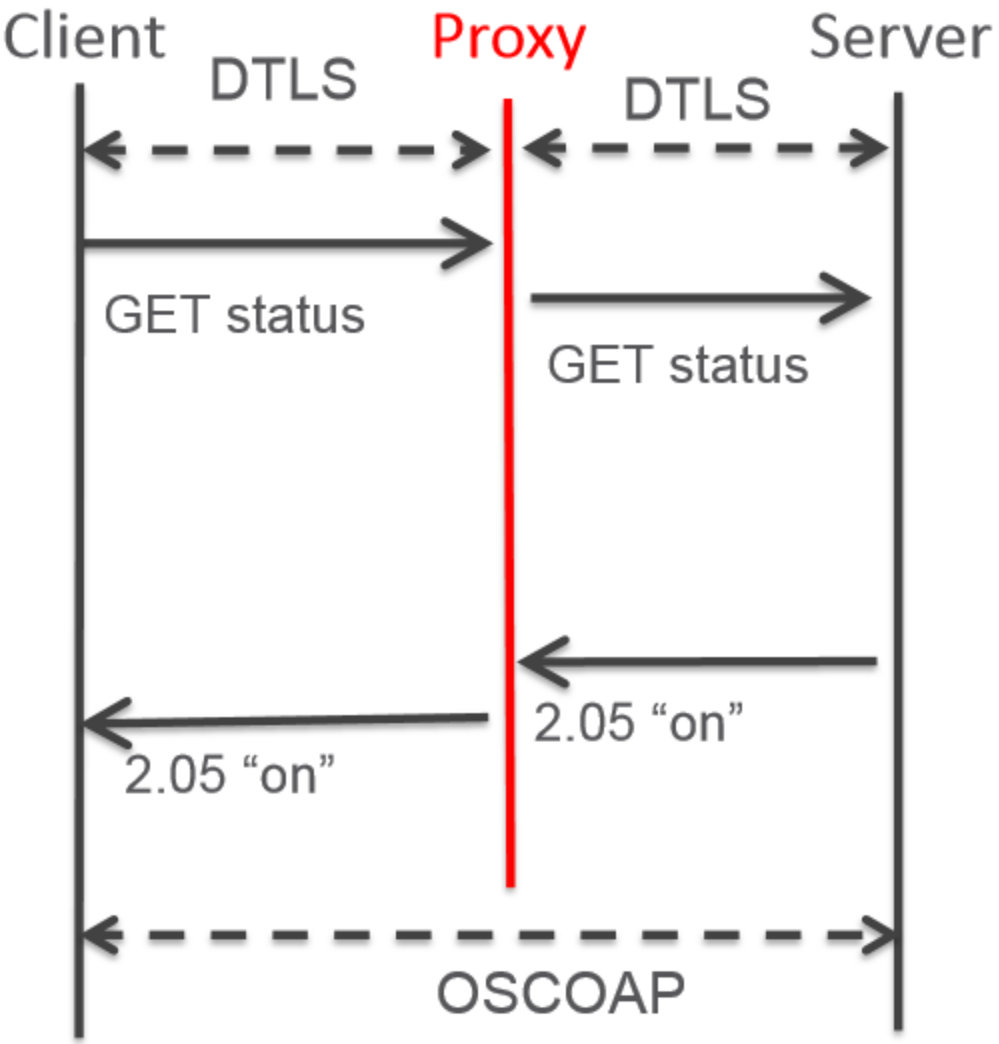
Decrypted COAP: COAP Header | HbH Opts | E2E Opts | CBOR Payload

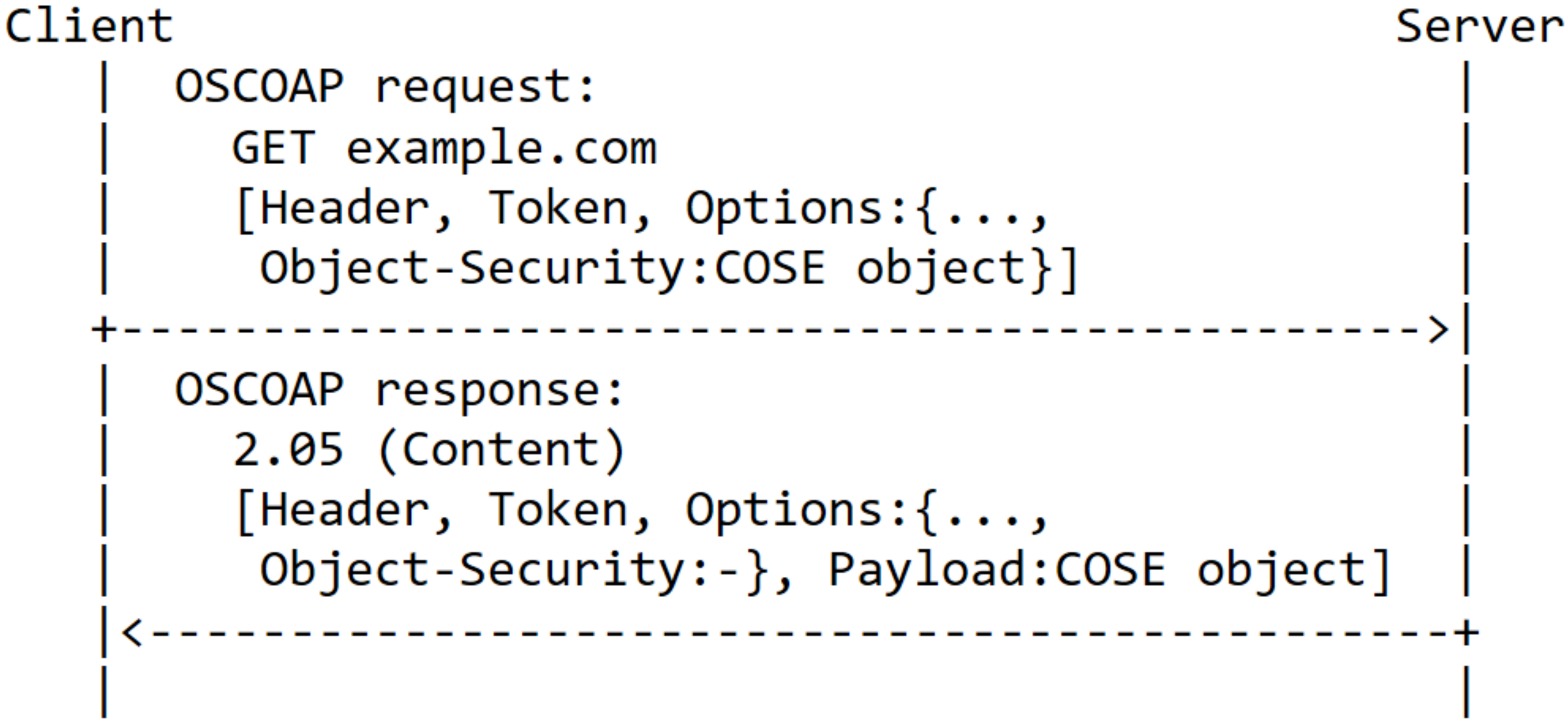
- OSCOAP defines a method for in-layer security of CoAP message exchanges using the COSE format.

- OSCOAP protects CoAP end-to-end and can be used instead of DTLS
  - Allows legitimate proxy operations
  - Detects illegitimate proxy operations

- Independent of how CoAP is transported (UDP, TCP, Bluetooth, 802.15.4, foo…)

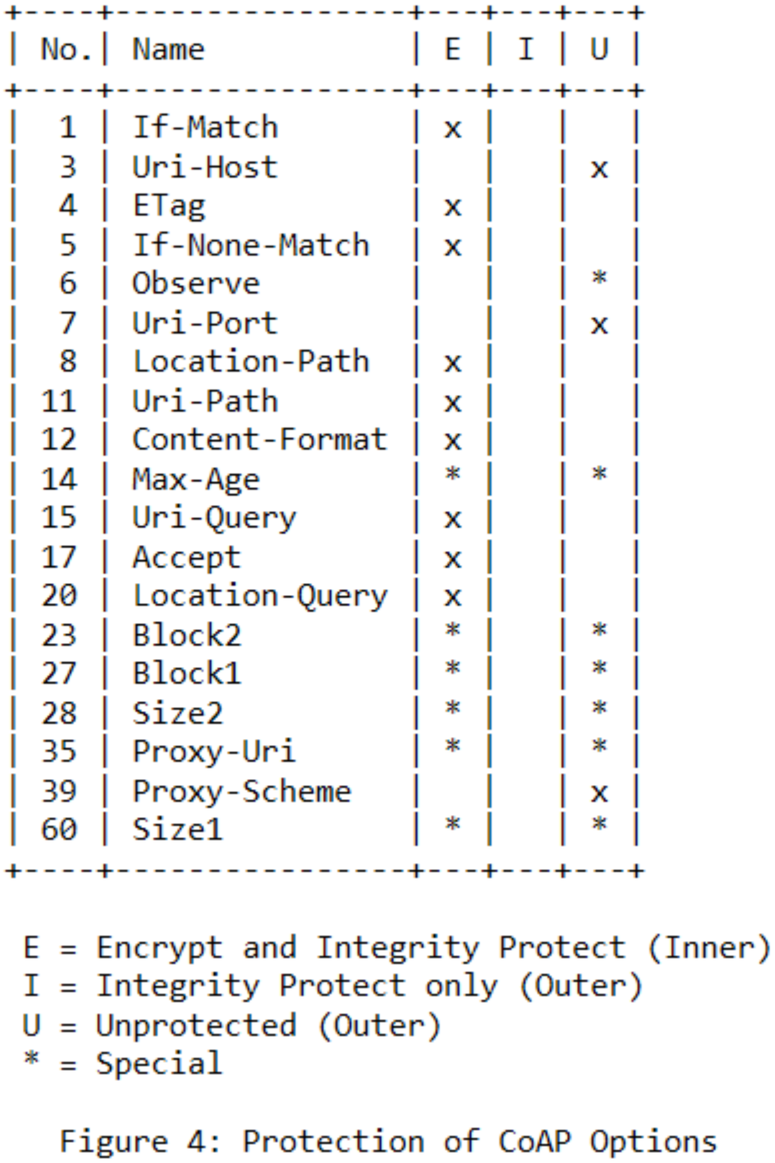
- Requirements: draft-hartke-core-e2e-security-reqs





```
Client                                          Server
   |  OSCOAP request:                             |
   |    GET example.com                           |
   |    [Header, Token, Options:{...,             |
   |     Object-Security:COSE object}]            |
   +--------------------------------------------->|
   |  OSCOAP response:                            |
   |    2.05 (Content)                            |
   |    [Header, Token, Options:{...,             |
   |     Object-Security:-}, Payload:COSE object] |
   |<---------------------------------------------+
   |                                              |
```

Figure 1: Sketch of OSCOAP



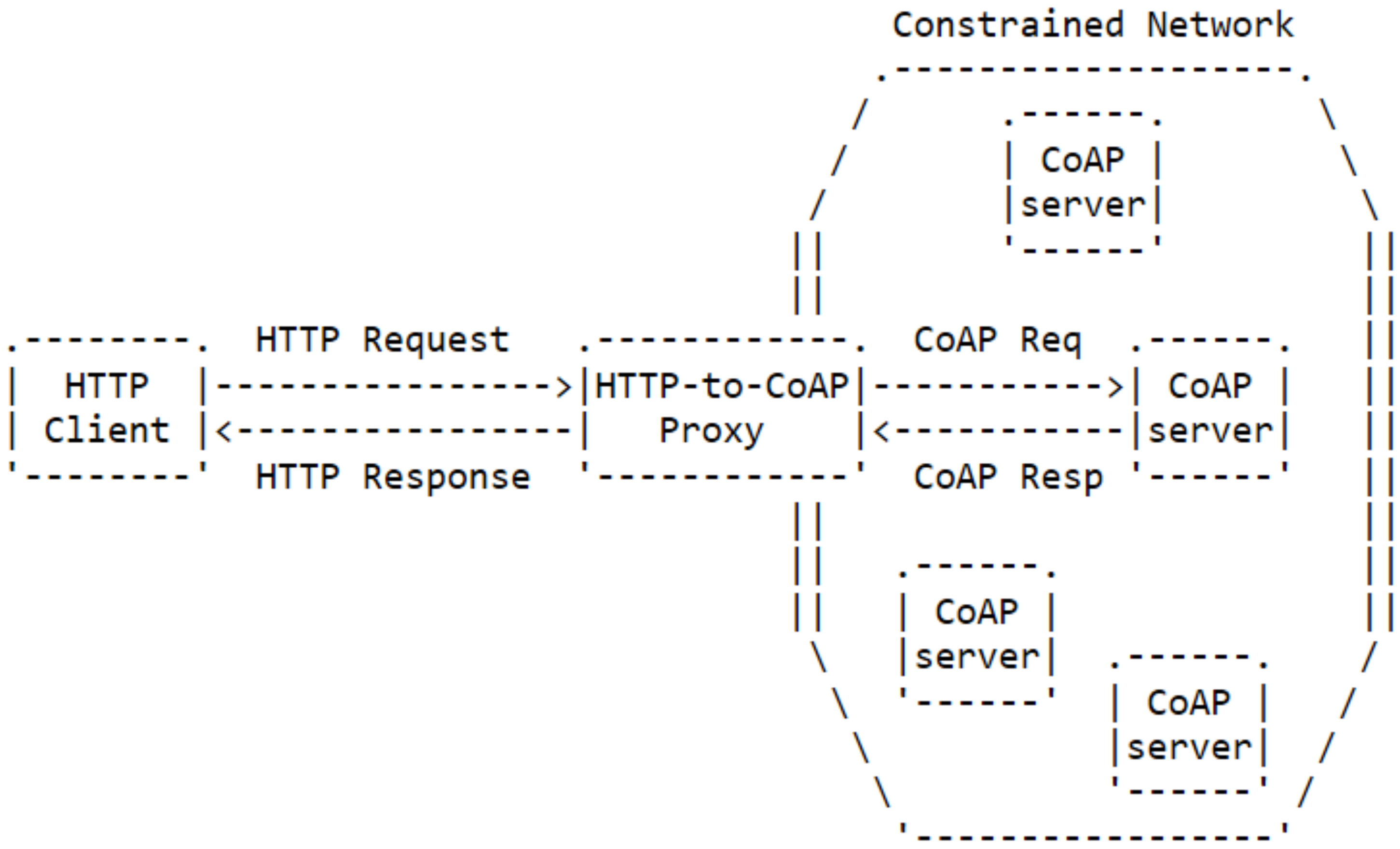
No.	Name	E	I	U
1	If-Match	x		
3	Uri-Host			x
4	ETag	x		
5	If-None-Match	x		
6	Observe			*
7	Uri-Port			x
8	Location-Path	x		
11	Uri-Path	x		
12	Content-Format	x		
14	Max-Age	*		*
15	Uri-Query	x		
17	Accept	x		
20	Location-Query	x		
23	Block2	*		*
27	Block1	*		*
28	Size2	*		*
35	Proxy-Uri	*		*
39	Proxy-Scheme			x
60	Size1	*		*

E = Encrypt and Integrity Protect (Inner)
I = Integrity Protect only (Outer)
U = Unprotected (Outer)
* = Special

Figure 4: Protection of CoAP Options

Object Security for Constrained RESTful Environments (OSCORE)

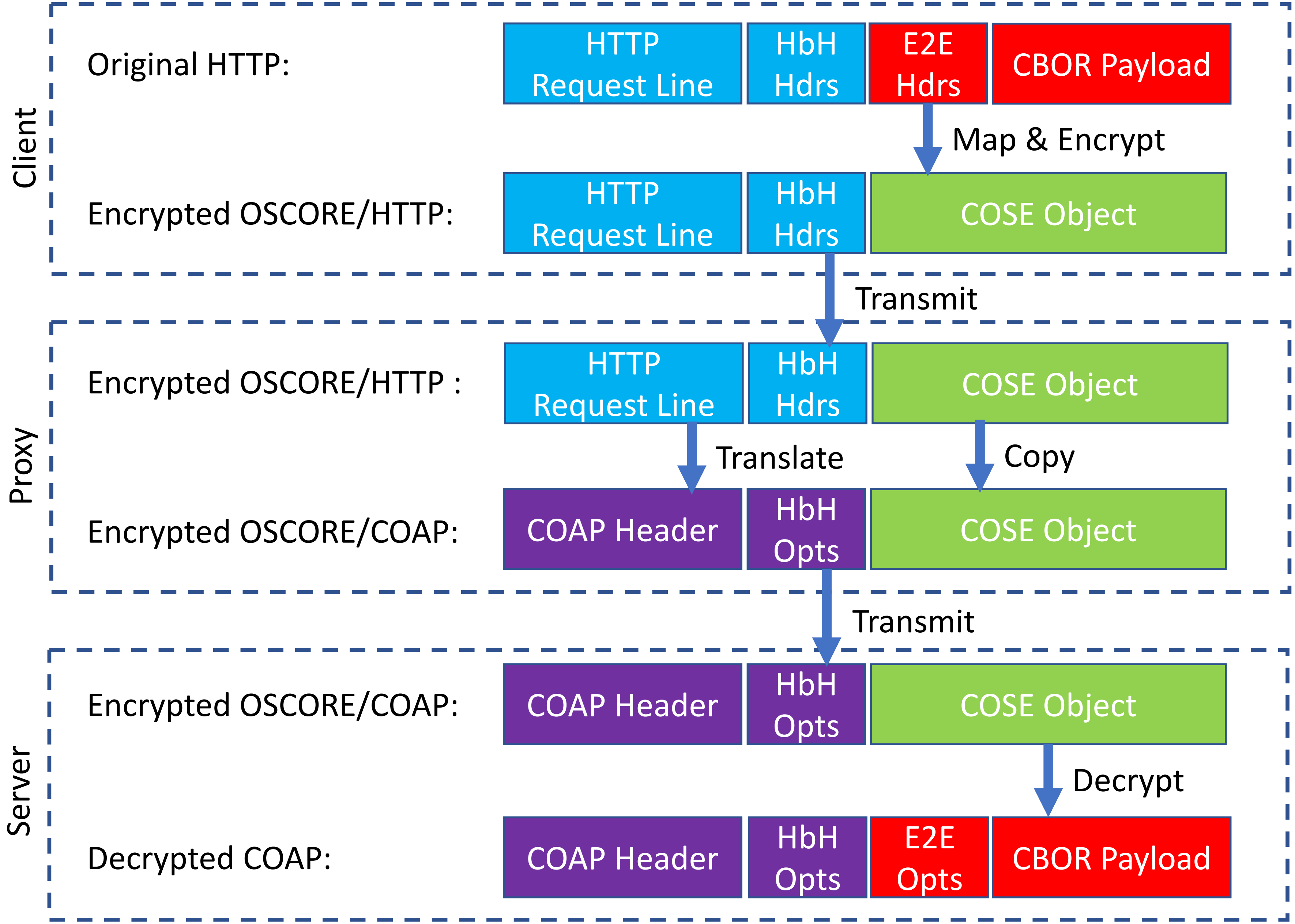
- As of draft-ietf-core-object-security-05 (September) and later
- Summary of changes:
    - COAP protocol version is not an end-to-end option (OSCORE payload version is important, but transport protocol version is not)
    - HTTP is also supported
        - Existing RFCs (7252, 8075) cover HTTP <-> COAP translation
        - HTTP headers (via their mapped COAP options) automatically categorized as end-to-end vs hop-by-hop
        - OSCORE/HTTP implementation conceptually converts HTTP->COAP->OSCORE
            - Implementation can just do HTTP->OSCORE directly
            - (analogy is that OCF Core specifies JSON but puts CBOR on wire and IoTivity just does CBOR)



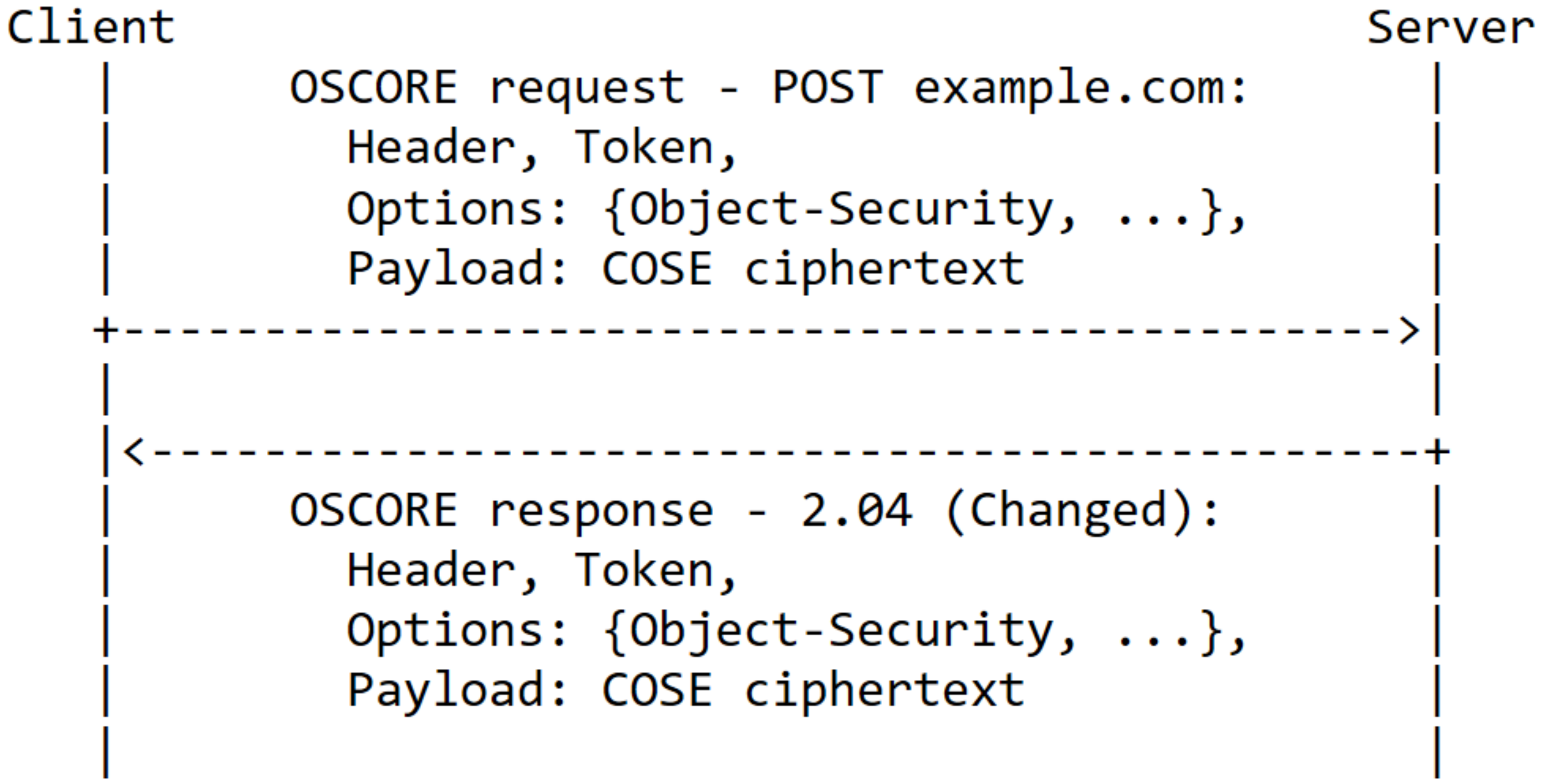
```
                                          Constrained Network
                                         .-------------------.
                                        /      .------.       \
                                       /       | CoAP |        \
                                      /        |server|         \
                                     ||        '------'         ||
                                     ||                         ||
.--------.  HTTP Request   .------------.  CoAP Req  .------.   ||
|  HTTP  |---------------->|HTTP-to-CoAP|----------->| CoAP |   ||
| Client |<----------------|   Proxy    |<-----------|server|   ||
'--------'  HTTP Response  '------------'  CoAP Resp '------'   ||
                                     ||                         ||
                                     ||   .------.              ||
                                     ||   | CoAP |              ||
                                      \   |server|  .------.    /
                                       \  '------'  | CoAP |   /
                                        \           |server|  /
                                         \          '------' /
                                          '-----------------'
```

Figure 1: HTTP-To-CoAP Proxy Deployment Scenario

Example



Client
Original HTTP:
HTTP Request Line
HbH Hdrs
E2E Hdrs
CBOR Payload
Map & Encrypt
Encrypted OSCORE/HTTP:
HTTP Request Line
HbH Hdrs
COSE Object
Transmit
Proxy
Encrypted OSCORE/HTTP :
HTTP Request Line
HbH Hdrs
COSE Object
Translate
Copy
Encrypted OSCORE/COAP:
COAP Header
HbH Opts
COSE Object
Transmit
Server
Encrypted OSCORE/COAP:
COAP Header
HbH Opts
COSE Object
Decrypt
Decrypted COAP:
COAP Header
HbH Opts
E2E Opts
CBOR Payload



```
Client                                             Server
  |      OSCORE request - POST example.com:           |
  |        Header, Token,                             |
  |        Options: {Object-Security, ...},           |
  |        Payload: COSE ciphertext                   |
  +-------------------------------------------------->|
  |                                                   |
  |<--------------------------------------------------+
  |      OSCORE response - 2.04 (Changed):            |
  |        Header, Token,                             |
  |        Options: {Object-Security, ...},           |
  |        Payload: COSE ciphertext                   |
  |                                                   |
```

Figure 1: Sketch of CoAP with OSCORE

OCF feedback/asks for IETF

- Editorial: Figure 4 (options) nice, maybe add figure for hdr fields too
- “Tunneling” OSCORE to prevent traffic analysis based on Uri-Host etc.
- COAP intermediary work: sleepy nodes, caching, etc.

- Not specific to OSCORE:
    - A way (new COAP response code?) to distinguish between “Forbidden by definition” vs “Forbidden by policy”
    - COAP client-HTTP server mapping details (Opposite of RFC 8075)
        - Immediate OCF need is for response code mapping like Table 2 of RFC 8075, to enable using Swagger/RAML with COAP