IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

OSCORE OBJECT SECURITY FOR COAP

Ricardo Andreasen

These slides + test Python script:
https://github.com/randreasen/oscoreslides

CoAP : Constrained Application Protocol

What is CoAP?

- Web transfer protocol for **constrained nodes and networks**,
- Want to connect these to the existing web….

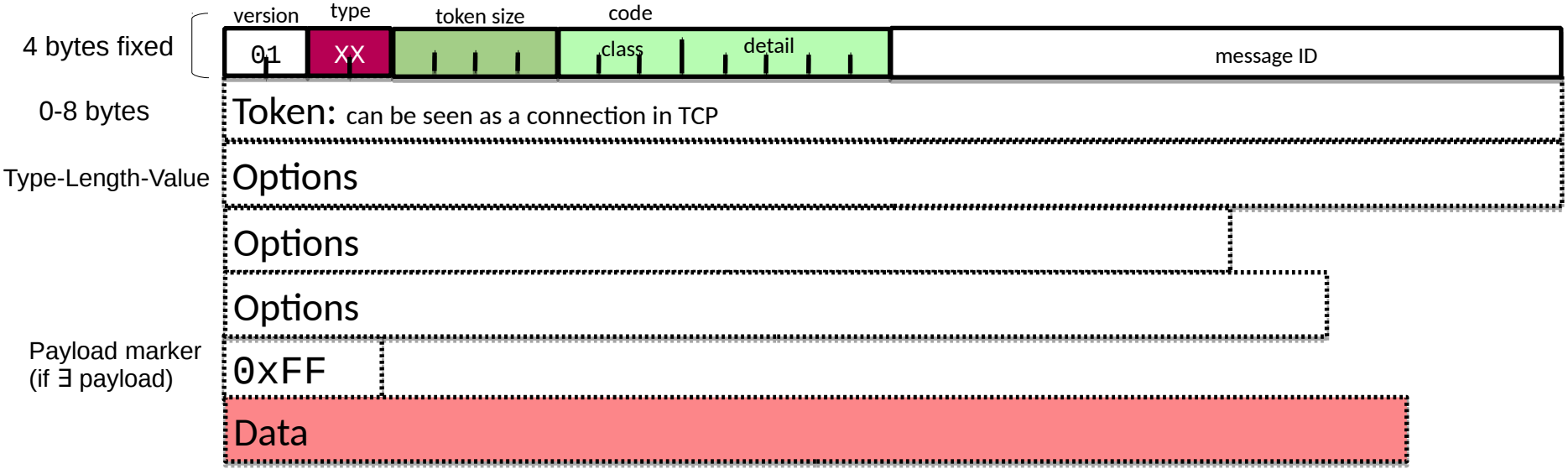
→ Can be thought of as a re-envisioning of HTTP, but tailored to M2M requirements in constrained environments:

  - Low memory
  - 8-bit processors
  - Low power
  - Lossy NW
  - ...

CoAP : Key Features

- Request/Response interaction model much like HTTP's,
- Envisioned for datagram-oriented transport (typically UDP),
- Nodes typically act as both client and server interchangeably,
- Asynchronous message exchanges (no real "connection"),
- Simple proxy and caching capabilities,
- Fixed 4 byte header followed by compact options and payload,
- URI and content-type support,
- Easy mapping to HTTP to connect with the existing web,
- **Security binding to DTLS.**

Anatomy of a CoAP message

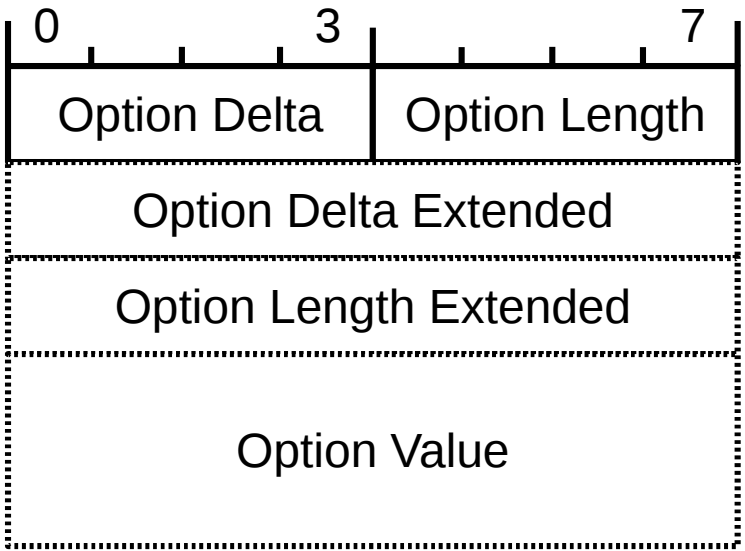


CoAP: Compact Options

3 things characterize an option

- Option Number (identifier)
- Option Length
- Option Value (can be thought as “option payload”)

For compactness, Option Number given incrementally with delta encoding:



The Problem

For security CoAP defines a binding to DTLS, but **CoAP and HTTP proxies require (D)TLS to be terminated at the proxy!**

→ Underlying reason is that the proxy needs access to (*part of*) the header and options to know how to treat the packet, but as a side effect it can:

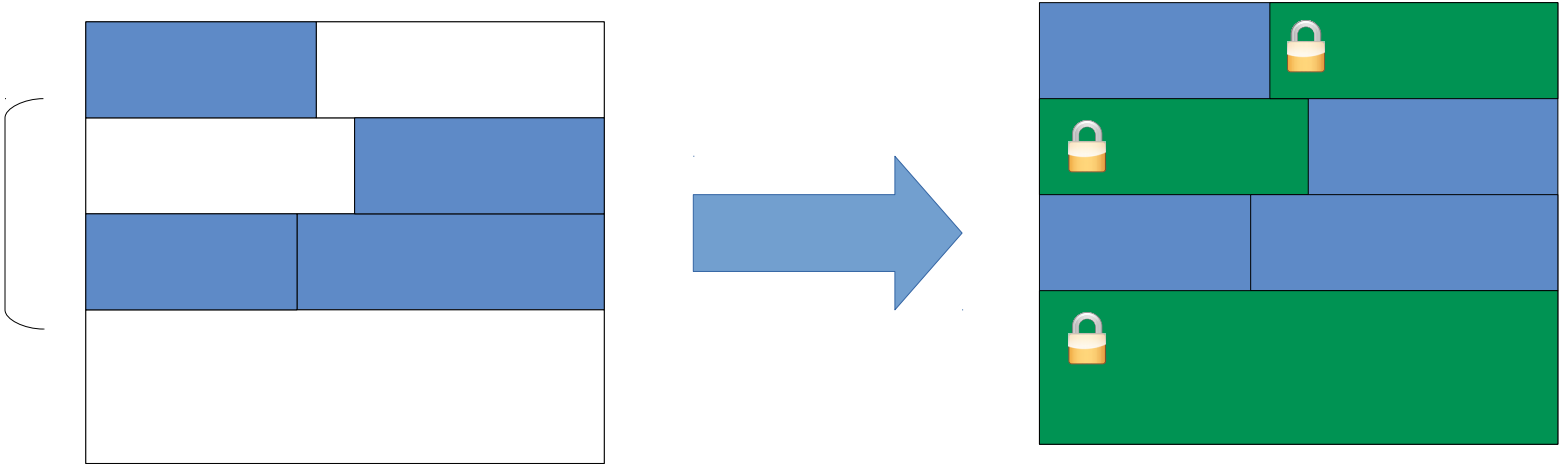
- Eavesdrop on or manipulate payload and metadata,
- Inject, delete or reorder packets.

This is where OSCORE comes in.

OSCORE: Object Security for Constrained RESTful Environments

Idea: only show the part of the message that is essential for proxy operation; hide all we can

Proxy only really needs to know about these fields



OSCORE: Object Security for Constrained RESTful Environments

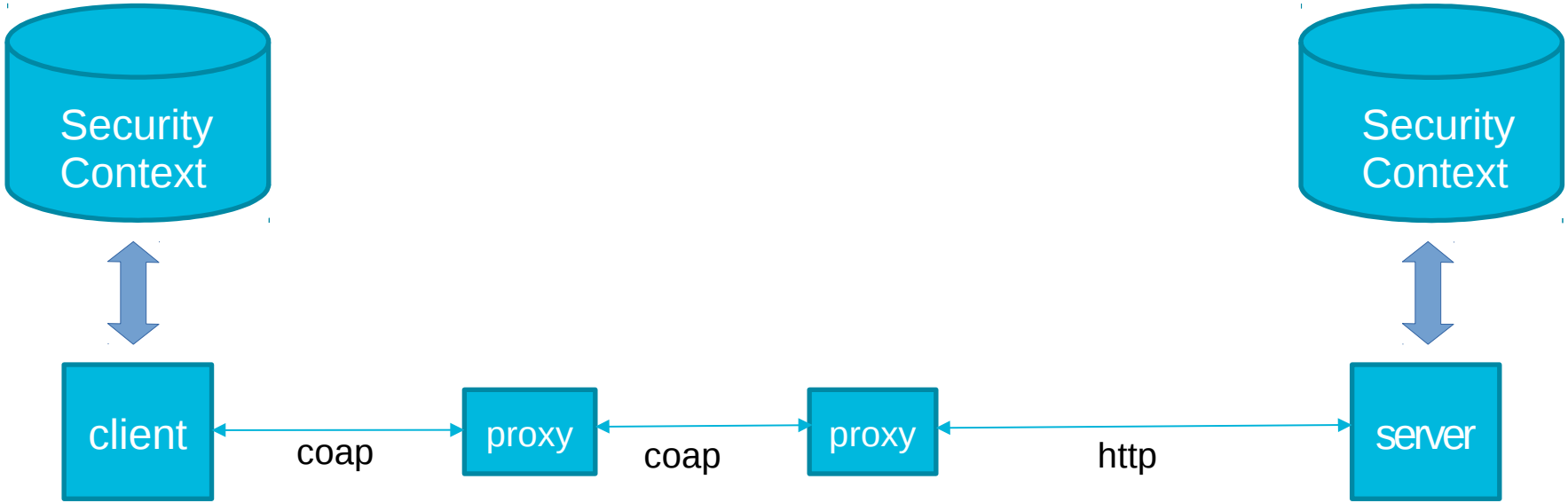
OSCORE is an application-layer protection of CoAP using COSE (CoAP Object Signing and Encryption). This provides:

- End-to-end encryption
- Integrity
- Replay protection

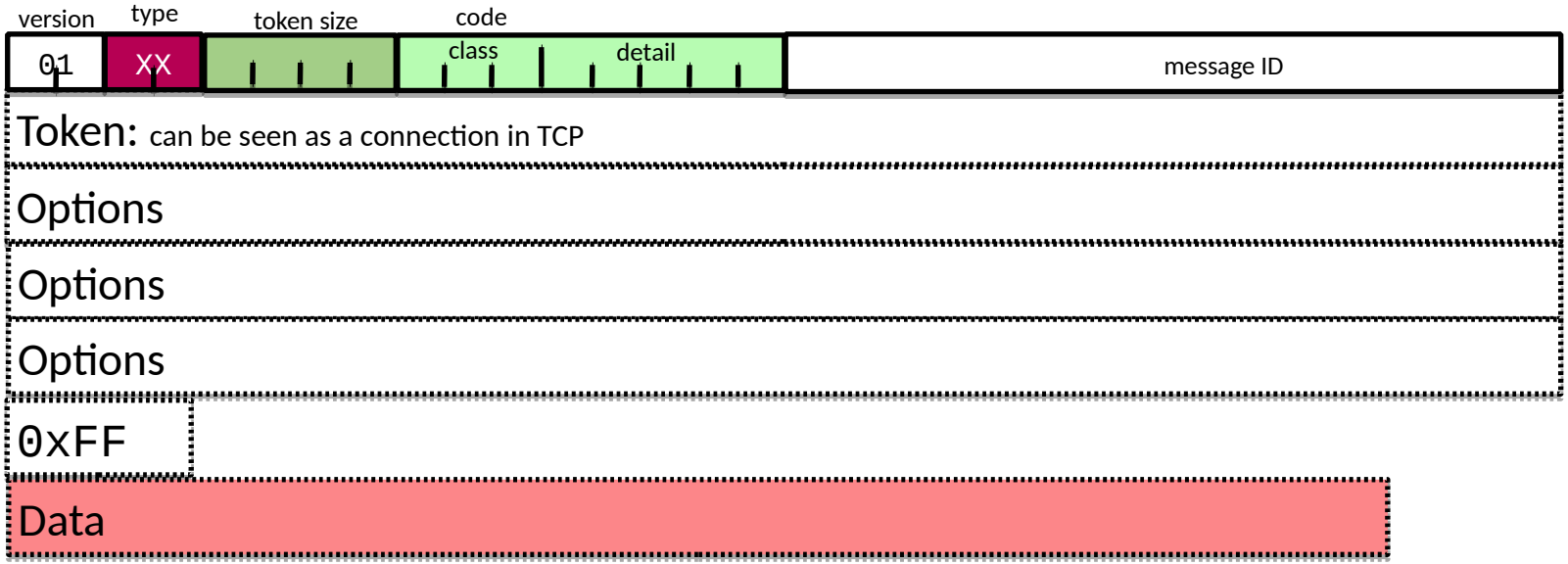
It allows to **selectively encrypt or authenticate parts of the CoAP message.** Each field is made to belong to one of three classes:

- Class E: encrypted via AEAD algorigthm, hidden inside OSCORE Payload,
- Class I: integrity protected as part of the AAD and visible from outside (outer options),
- Class U: unprotected and visible from the outside (outer options).

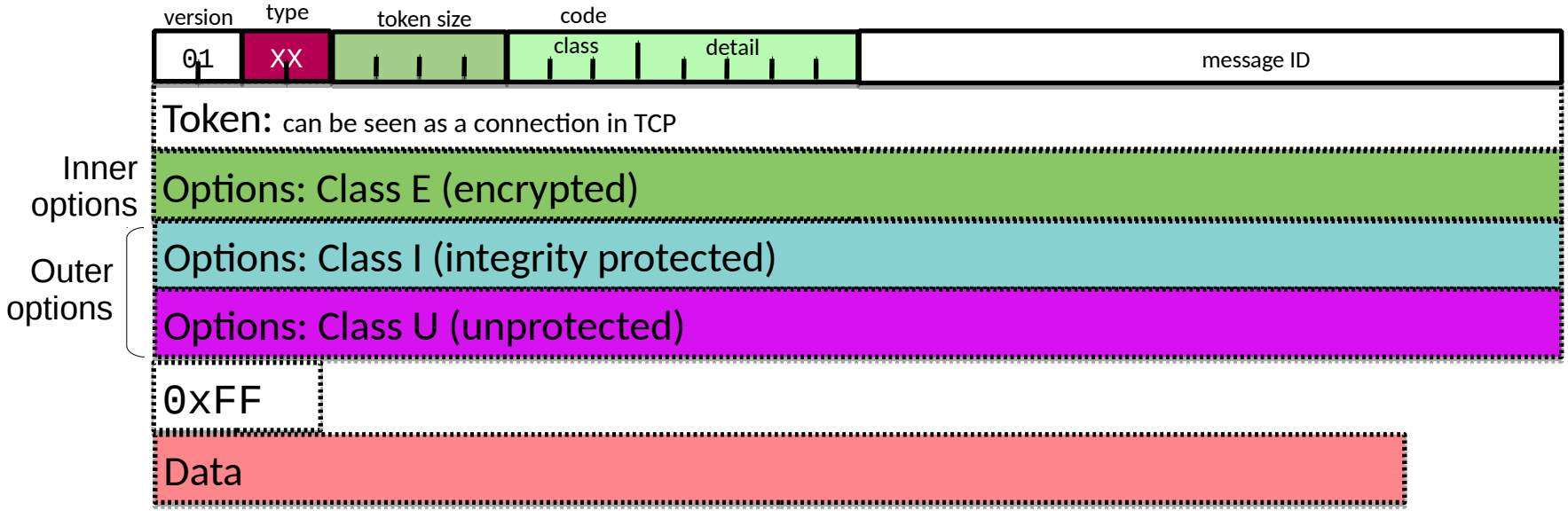
OSCORE: The Mechanics



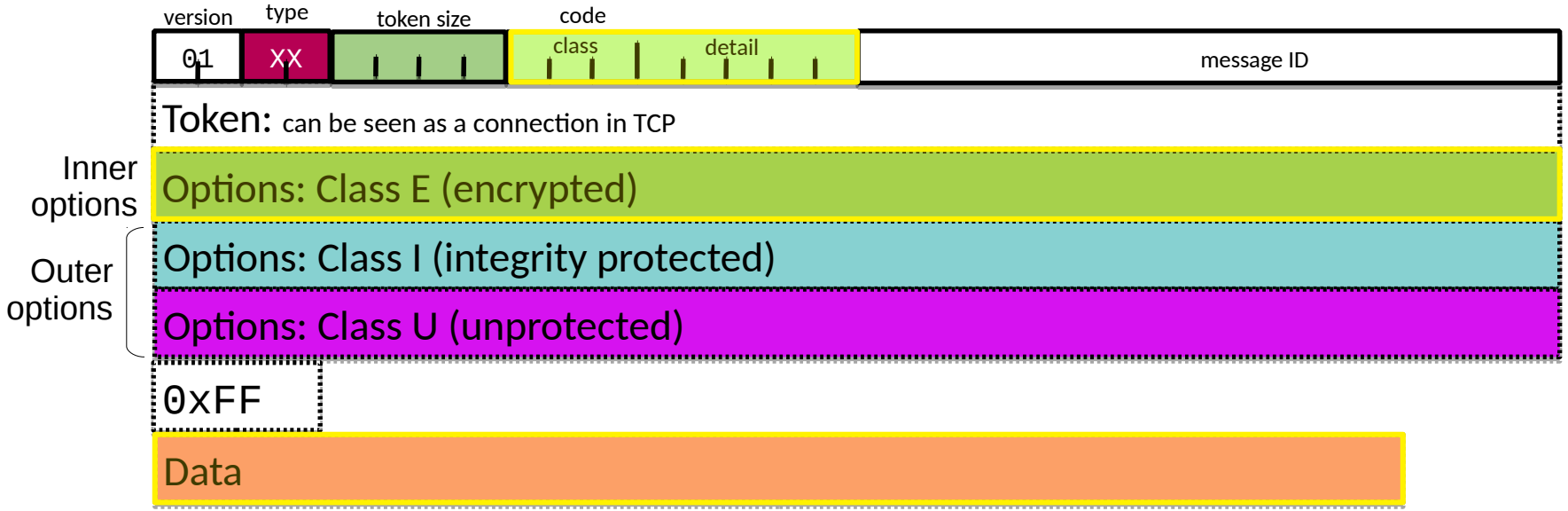
CoAP field classification



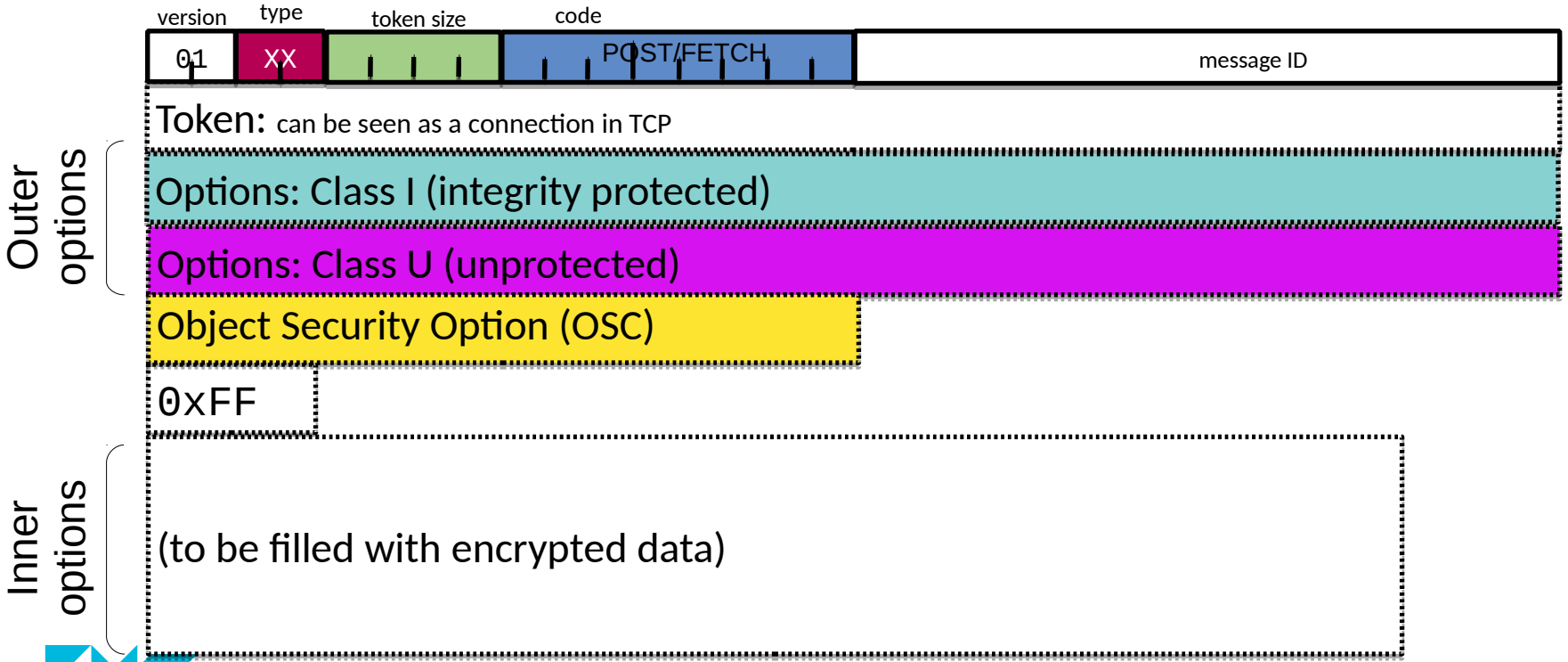
CoAP field classification



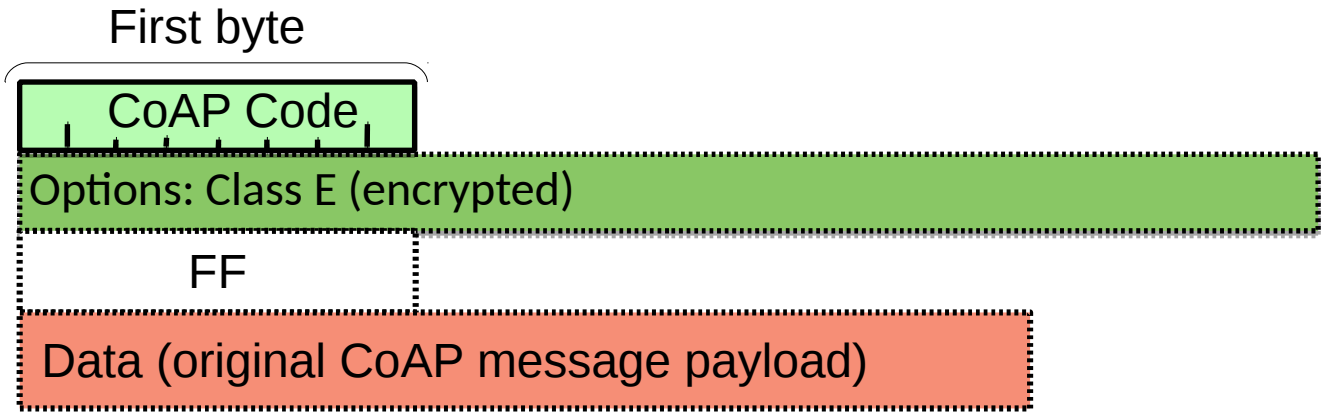
CoAP field classification



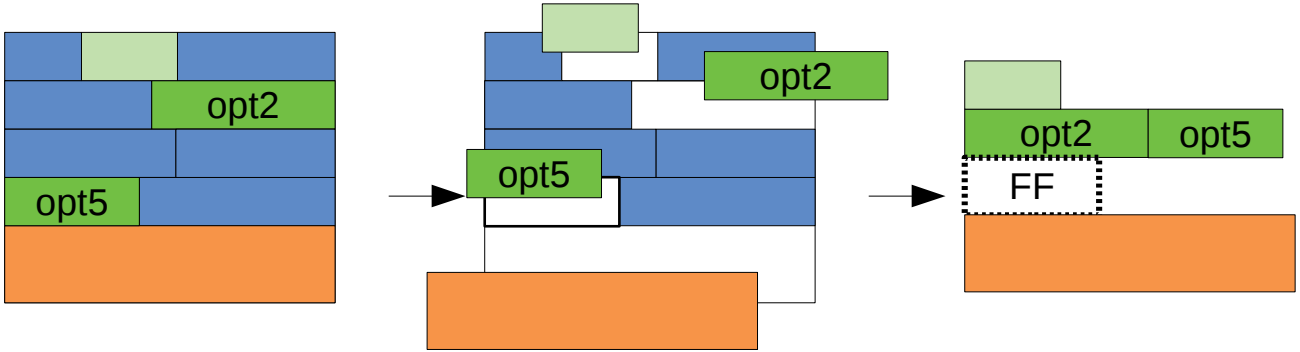
Prepare target OSCORE message



OSCORE Plaintext

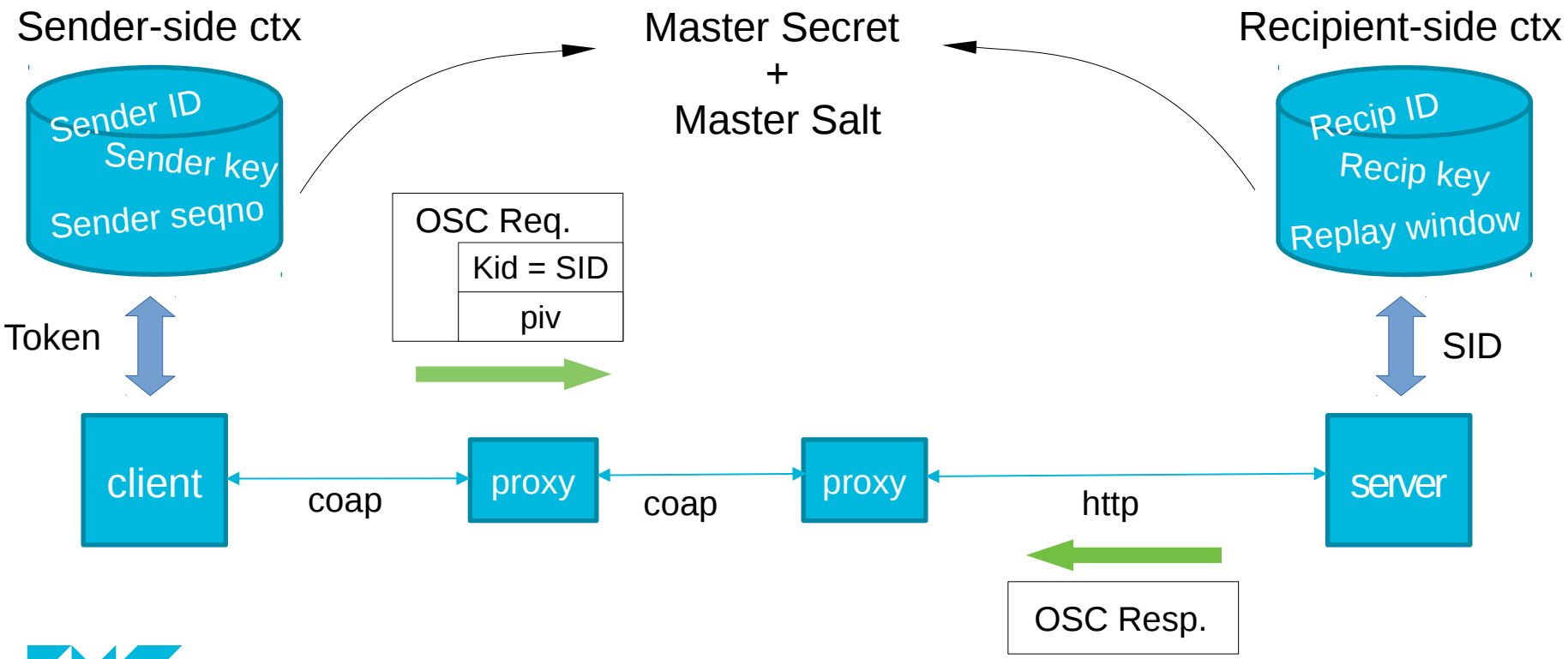


Options are reordered and re-compressed with delta encoding as per CoAP



IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Security Context



Security Parameters

Pre-established parameters:

- Master Secret
- ? Master Salt
- Sender ID
- Recipient ID
- ? AEAD Algorithm
- ? kdf
- ? Replay Window type & size

* the '?' indicates optional param. Default value is assumed if absent

Key & Common IV derivation:

key/IV = HKDF(salt, IKM, info, L)

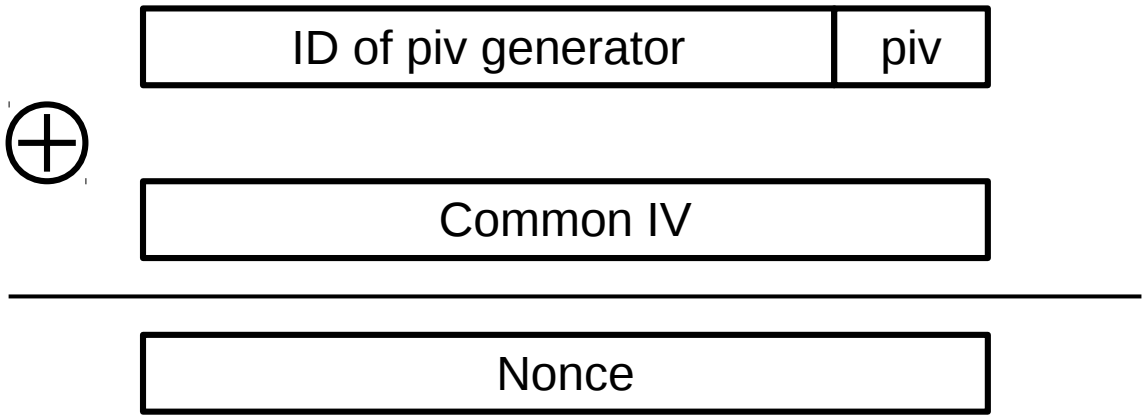
Master Salt

Master Secret

[
id = SID / RID / nil,
type = "key" / "iv",
L = size of key in octets
]

Security Parameters (cont.)

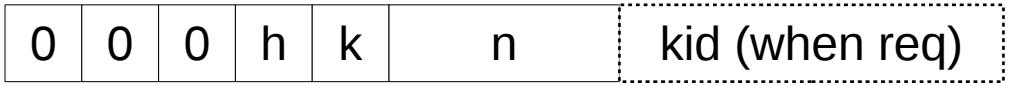
Nonce:



piv = sender sequence number, incremented each time we send a message

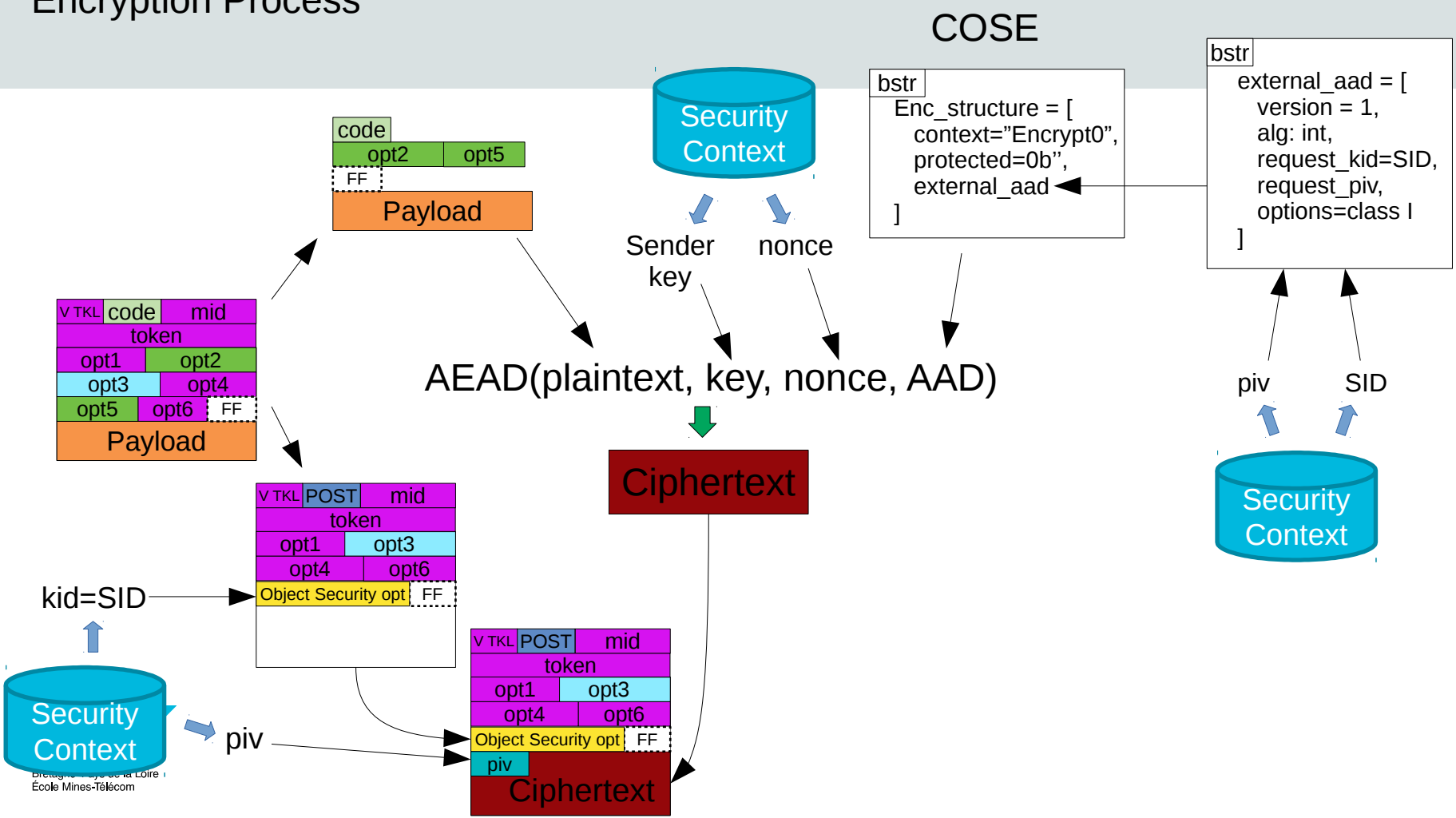
Security Parameters (cont.)

Object Security Option:



- h = 1 if there is a context hint in payload,
- k = 1 if option carries kid (e.g. on common request),
- n = length of piv in octets,
- kid = kid

Encryption Process



Putting it to the test: aiocoap

Aiocoap is a Python implementation of CoAP with asynchrounous I/O which implements OSCORE.

Repo:
https://github.com/chrysn/aiocoap
Documentation:
http://aiocoap.readthedocs.io/en/latest/guidedtour.html

Quick setup for OSCORE:
$ git clone https://github.com/chrysn/aiocoap
$ cp -r contrib/oscore-plugtest/* .

These slides + test Python script:
https://github.com/randreasen/oscoreslides

OSCORE + SCHC examples
GET Request

Original message:
==========
0x4101000182396c6f63616c686f73748b74656d7065726174757265

Header:
0x4101
01 Ver
 00 CON
  0001 tkl
   00000001 Request Code 1 "GET"

0x0001 = mid
0x82 = token

Options:
0x396c6f63616c686f73748b74656d7065726174757265
Option 3: URI_HOST
Value = localhost
Option 11: URI_PATH
Value = temperature

OSCORE + SCHC examples
GET Request (protected and with inner compression)

Protected message:
==========
0x4102000182396c6f63616c686f7374d70509636c69656e74ff00598a724e09a6842d9a

Header:
0x4102
01 Ver
  00 CON
    0001 tkl
      00000010 Request Code 2 "POST"

0x0001 = mid
0x82 = token

Options:
0x396c6f63616c686f7374d70509636c69656e74
Option 3: URI_HOST
Value = localhost
Option 21: OBJECT_SECURITY
Value = b'\tclient'

0xFF Payload marker
Payload:
0x00598a724e09a6842d9a

OSCORE + SCHC examples
GET Request (protected and with inner + outer compression)

Compressed message (protected):
==========
0x001400b314e49c134d085b34
0x00 = Rule ID

Compression residue:
0b0001010 (0.875 bytes)

Payload
0x00598a724e09a6842d9a

Original msg length: 27
Protected msg length: 35
Compressed msg length: 12

vs.

Compressed message (no OSCORE):
==========
0x0114
0x01 = Rule ID

Compression residue:
0b00010100 (1 bytes)

Original msg length: 27
Compressed msg length: 2

So cost of security was 10

OSCORE + SCHC examples
CONTENT Response

Original message:
==========
0x61450001823b74656d7065726174757265ff32332043

Header:
0x6145
01 Ver
 10 ACK
  0001 tkl
   01000101 Successful Response Code 69 "2.05 Content"

0x0001 = mid
0x82 = token

Options:
0x3b74656d7065726174757265
Option 3: URI_HOST
Value = temperature

0xFF Payload marker
Payload:
0x32332043

OSCORE + SCHC examples
CONTENT Response (protected with inner compression)

Protected message:
==========
0x6144000182d008fff96f4e5c0a64b9fd132ab764b413

Header:
0x6144
01 Ver
  10 ACK
    0001 tkl
      01000100 Successful Response Code 68 "2.04 Changed"

0x0001 = mid
0x82 = token

Options:
0xd008
Option 21: OBJECT_SECURITY
Value = b''

0xFF Payload marker
Payload:
0xf96f4e5c0a64b9fd132ab764b413

OSCORE + SCHC examples
CONTENT Response (protected with inner + outer compression)

Compressed message (protected):
==========
0x0015f2de9cb814c973fa26556ec96826
0x00 = Rule ID

Compression residue:
0b0001010 (0.875 bytes)

Payload
0xf96f4e5c0a64b9fd132ab764b413

Original msg length: 22
Protected msg length: 22
Compressed msg length: 16

vs.

Compressed message (no OSCORE):
==========
0x010a32332043
0x01 = Rule ID

Compression residue:
0b00001010 (1.0 bytes)

Payload
0x32332043

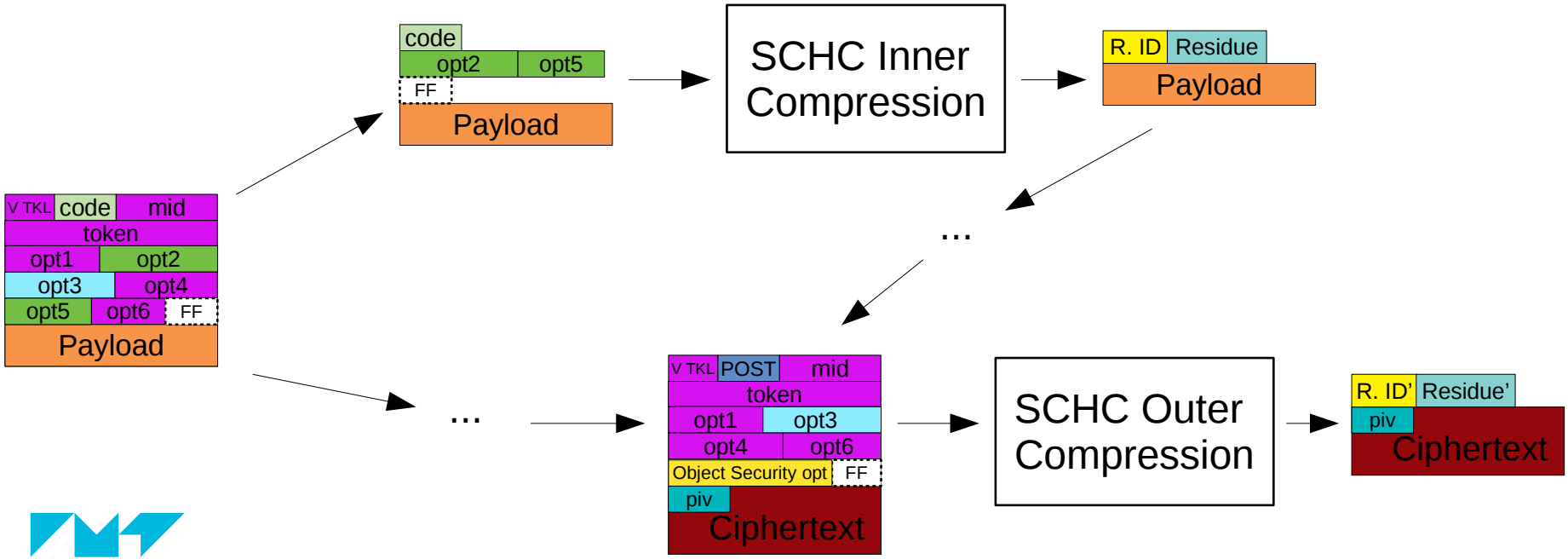
Original msg length: 22
Compressed msg length: 6

So cost of security was 10

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

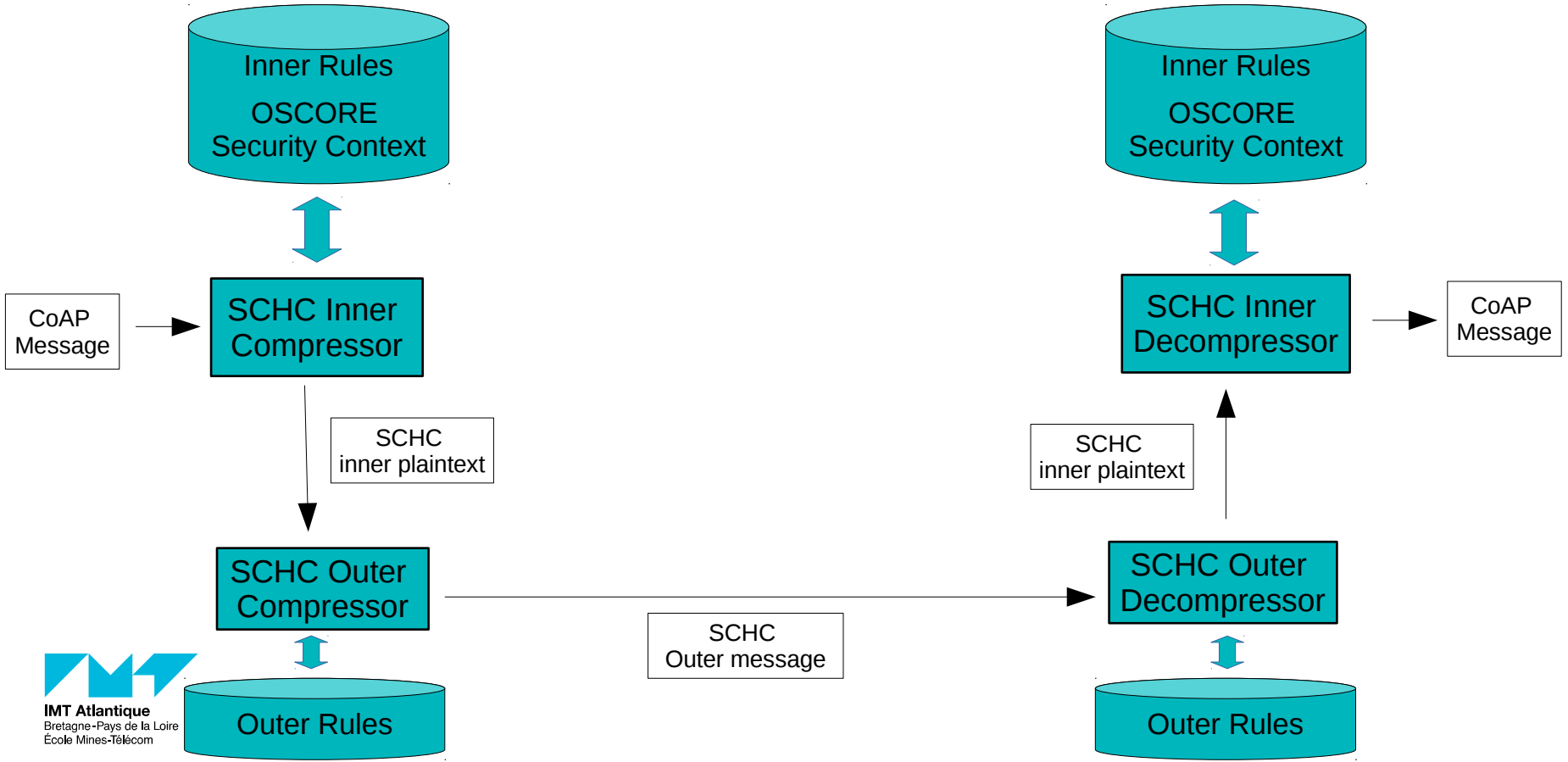
SCHC + OSCORE: How do we compress it?

Main idea: Inner + Outer compression.



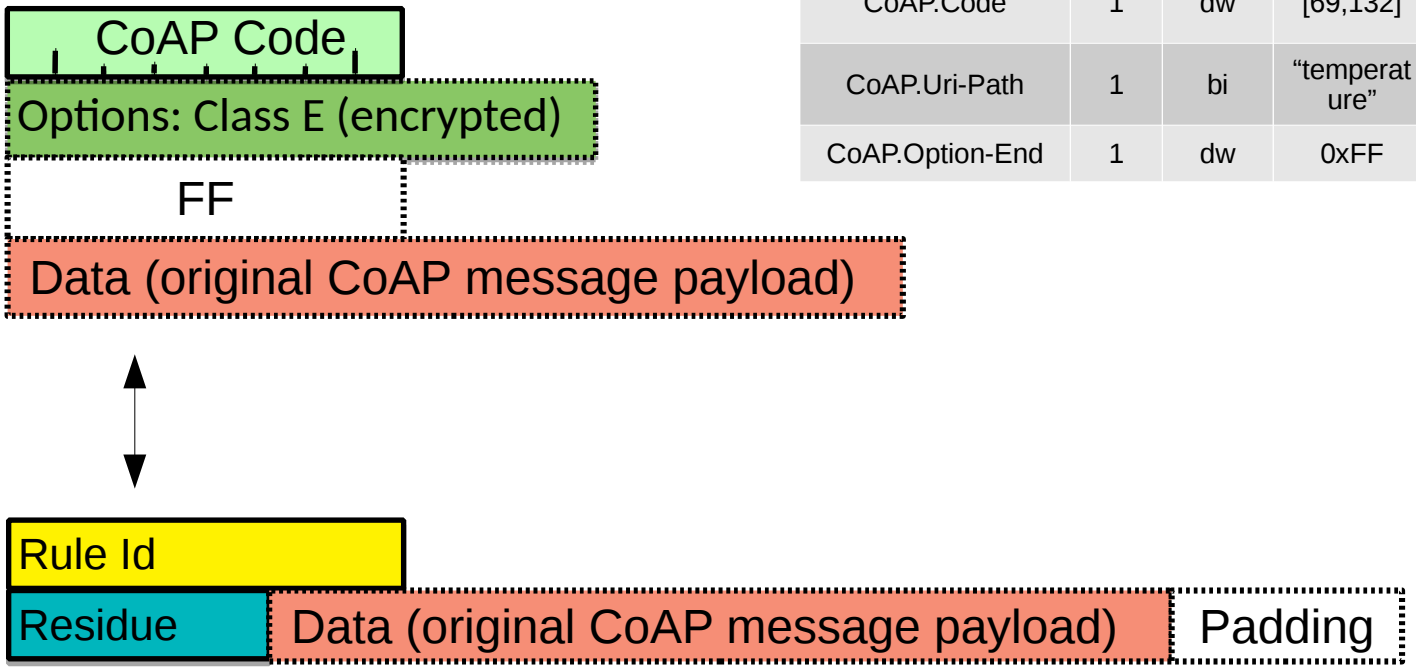
IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Inner and Outer C/D: Deployment

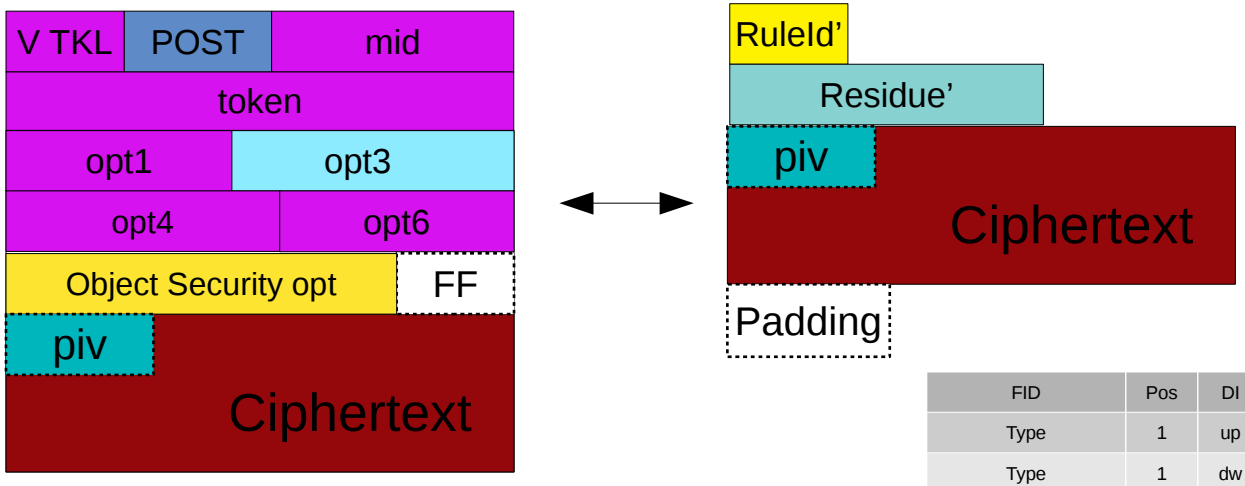


Inner Compression

FID	Pos	DI	TV	MO	CDA
CoAP.Code	1	up	1	equal	not-sent
CoAP.Code	1	dw	[69,132]	match-mapping	mapping-sent
CoAP.Uri-Path	1	bi	“temperature”	equal	not-sent
CoAP.Option-End	1	dw	0xFF	equal	not-sent

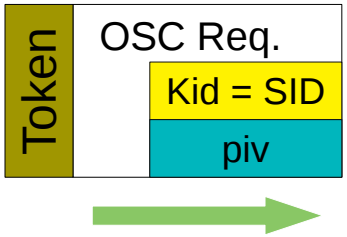


Outer Compression



FID	Pos	DI	TV	MO	CDA
Type	1	up	0	equal	not-sent
Type	1	dw	2	equal	not-sent
TKL	1	bi	1	equal	not-sent
Code	1	up	2	equal	not-sent
Code	1	dw	68	equal	not-sent
mid	1	bi	0	MSB(12)	LSB
token	1	bi	0x80	MSB(5)	LSB
Uri-Host	1	up	localhost	equal	not-sent
Object-Security	1	up	b'\tclient'	equal	not-sent
Object-Security	1	dw	b''	equal	not-sent
Option-End	1	bi	0xFF	equal	not-sent

Outer Compression: Object-Security option



Object-Security option

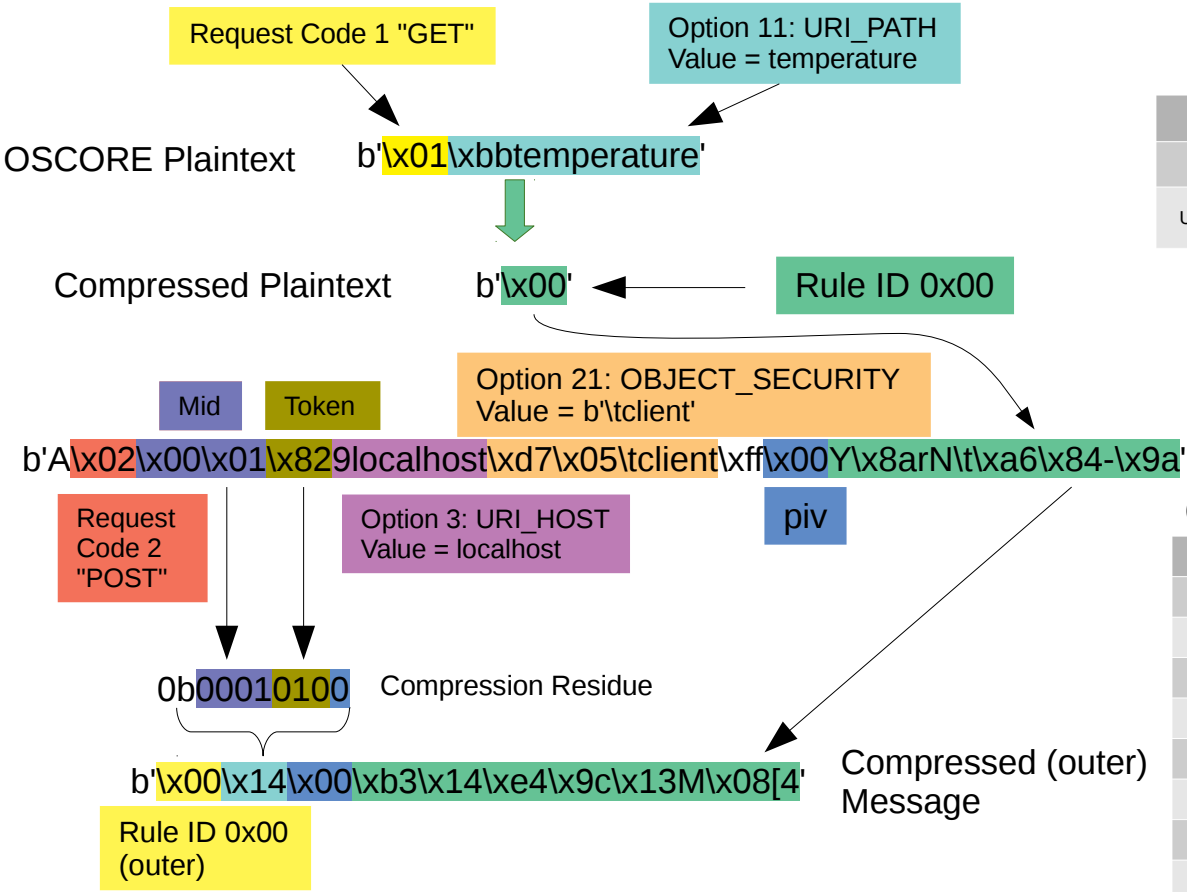
	0	0	0	h	k	n	kid
e.g.	0	0	0	0	1	1	b'client'

→ TV = b'\tclient', MO = equal, CDA= not-sent

OptionLength = 0 (sends empty O_S option)

→ TV = b'', MO = equal, CDA= not-sent.

GET Temperature:
Analysis – SCHC + OSCORE



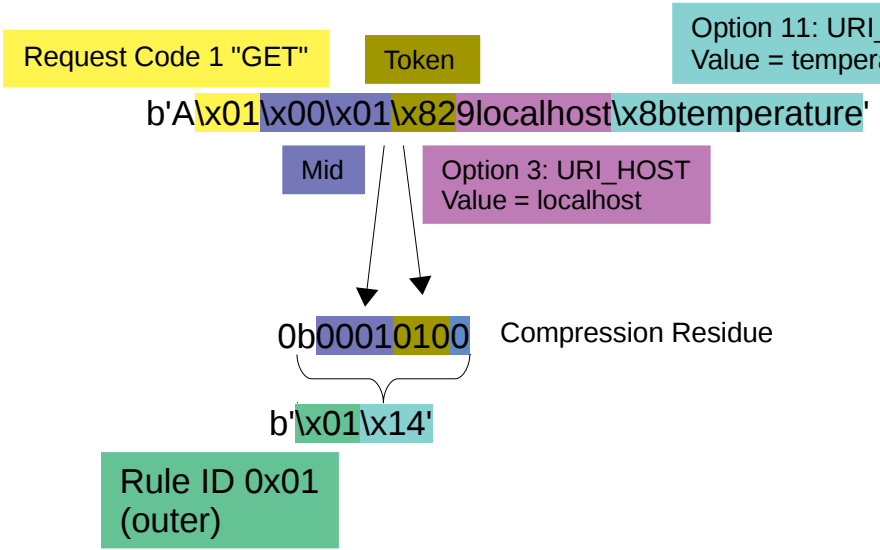
Inner Rule (Rule ID 0x00)

FID	Pos	DI	TV	MO	CDA
Code	1	up	1	equal	not-sent
Uri-Path	1	bi	temperature	equal	not-sent

Outer Rule (Rule ID 0x00)

FID	Pos	DI	TV	MO	CDA
Type	1	up	0	equal	not-sent
TKL	1	bi	1	equal	not-sent
Code	1	up	2	equal	not-sent
mid	1	bi	0	MSB(12)	LSB
token	1	bi	0x80	MSB(5)	LSB
Uri-Host	1	up	localhost	equal	not-sent
Object-Security	1	up	b'\tclient'	equal	not-sent
Option-End	1	bi	0xFF	equal	not-sent

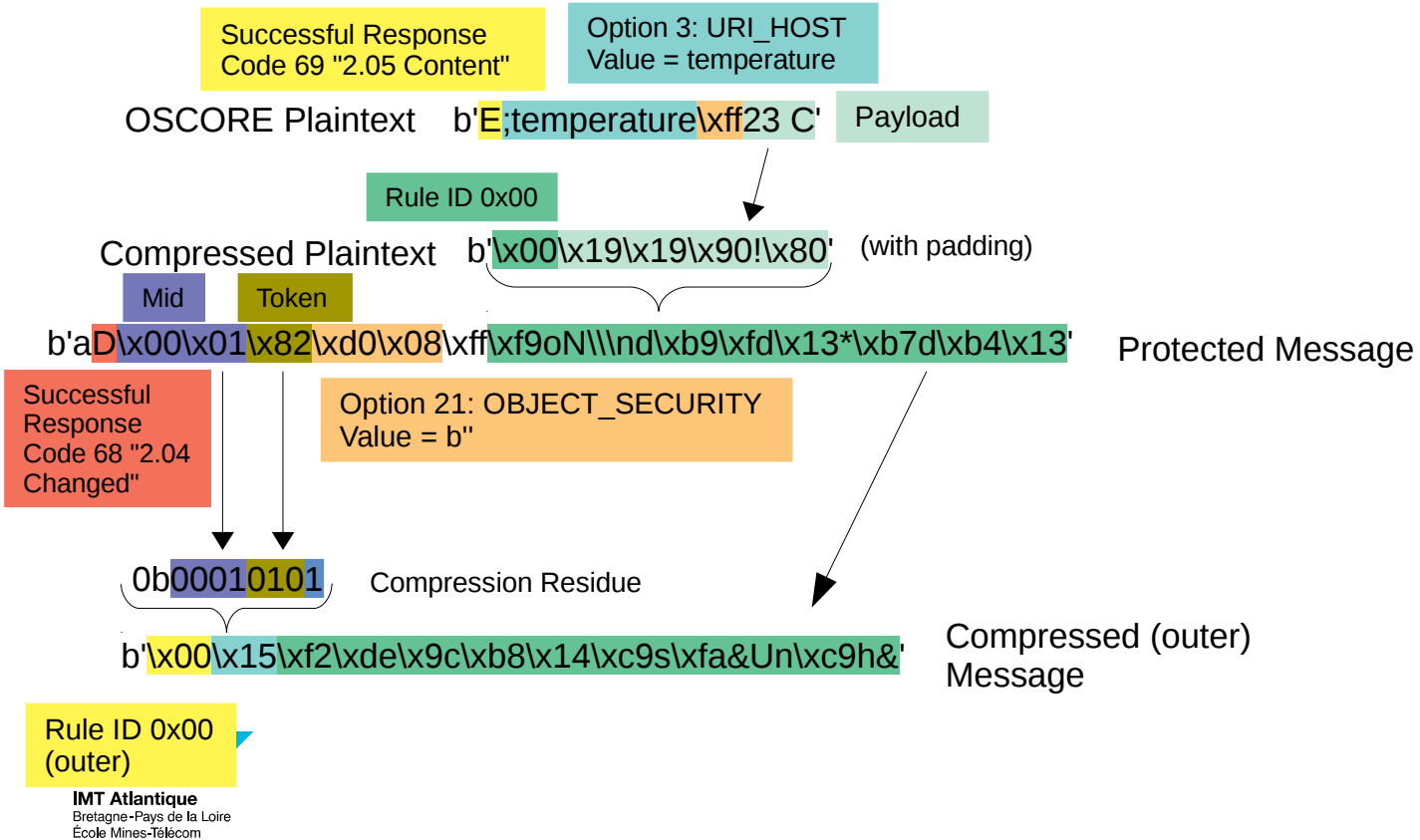
GET Temperature:
Analysis – SCHC only (no OSCORE)



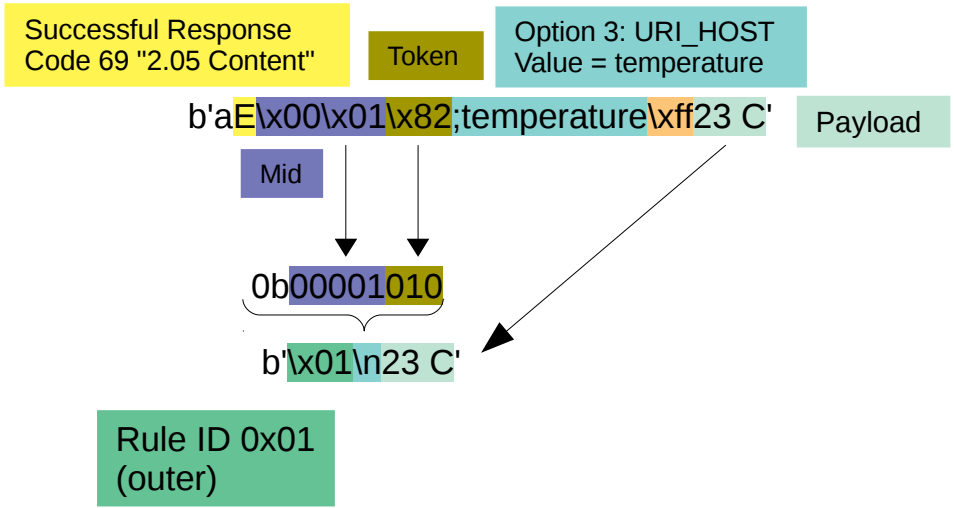
Outer Rule (Rule ID 0x01)

FID	Pos	DI	TV	MO	CDA
Type	1	up	0	equal	not-sent
TKL	1	bi	1	equal	not-sent
Code	1	up	1	equal	not-sent
mid	1	bi	0	MSB(12)	LSB
token	1	bi	0x80	MSB(5)	LSB
Uri-Host	1	up	localhost	equal	not-sent
Uri-Path	1	up	temperature	equal	not-sent

CONTENT Temperature: Analysis – SCHC + OSCORE



CONTENT Temperature:
Analysis – SCHC only (no OSCORE)



FID	Pos	DI	TV	MO	CDA
Type	1	up	0	equal	not-sent
Type	1	dw	2	equal	not-sent
TKL	1	bi	1	equal	not-sent
Code	1	up	2	equal	not-sent
Code	1	dw	68	equal	not-sent
mid	1	bi	0	MSB(12)	LSB
token	1	bi	0x80	MSB(5)	LSB
Uri-Host	1	up	localhost	equal	not-sent
Object-Security	1	up	b'\tclient'	equal	not-sent
Object-Security	1	dw	b''	equal	not-sent
Option-End	1	bi	0xFF	equal	not-sent