



IMT Atlantique

Bretagne-Pays de la Loire

École Mines-Télécom

OSCORE OBJECT SECURITY FOR COAP

Ricardo Andreassen

What is CoAP?

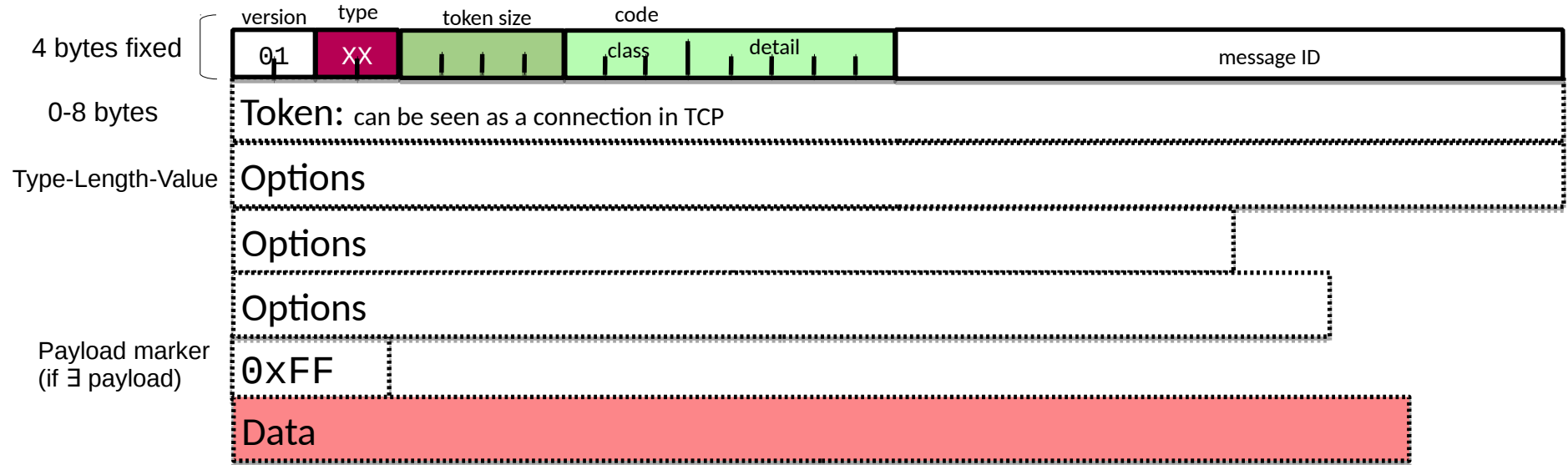
- Web transfer protocol for **constrained nodes and networks**,
- Want to connect these to the existing web....

→ 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
- ...

- 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

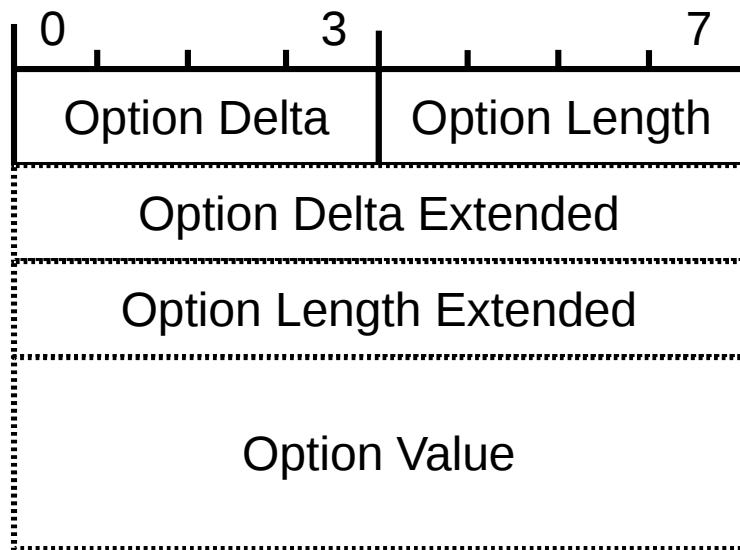


CoAP: Compact Options

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:

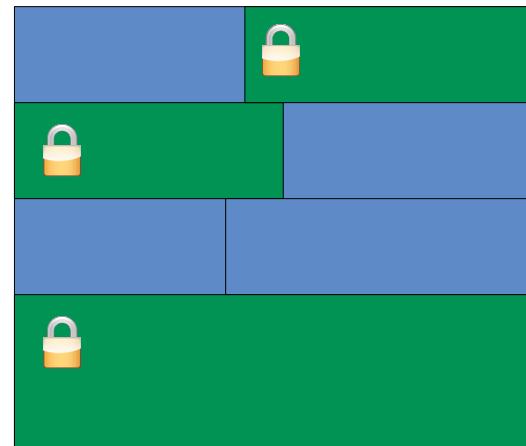
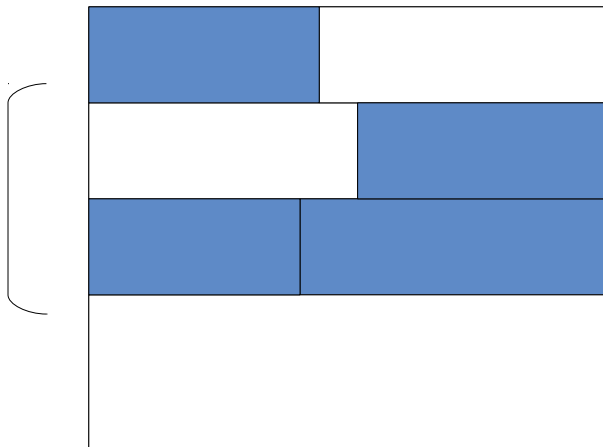
- Eavesdrop on or manipulate payload and metadata,
- Inject, delete or reorder packets.

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



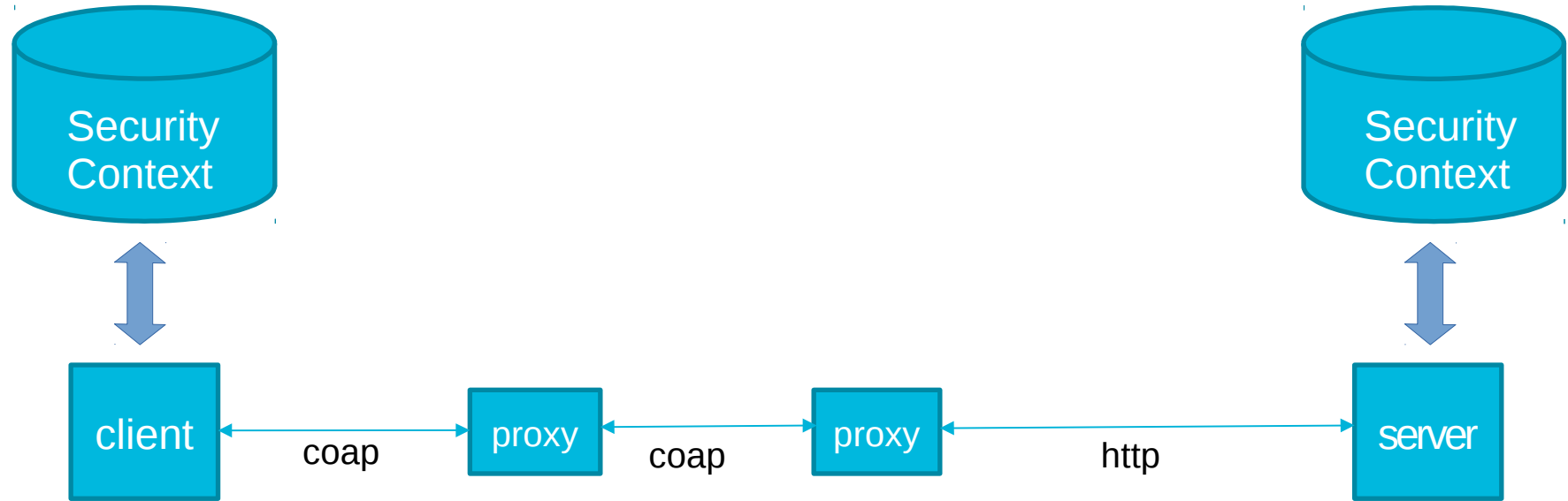
OSCORE is an application-layer protection of CoAP using COSE (CoAP Object Signing and Encryption). This provides:

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

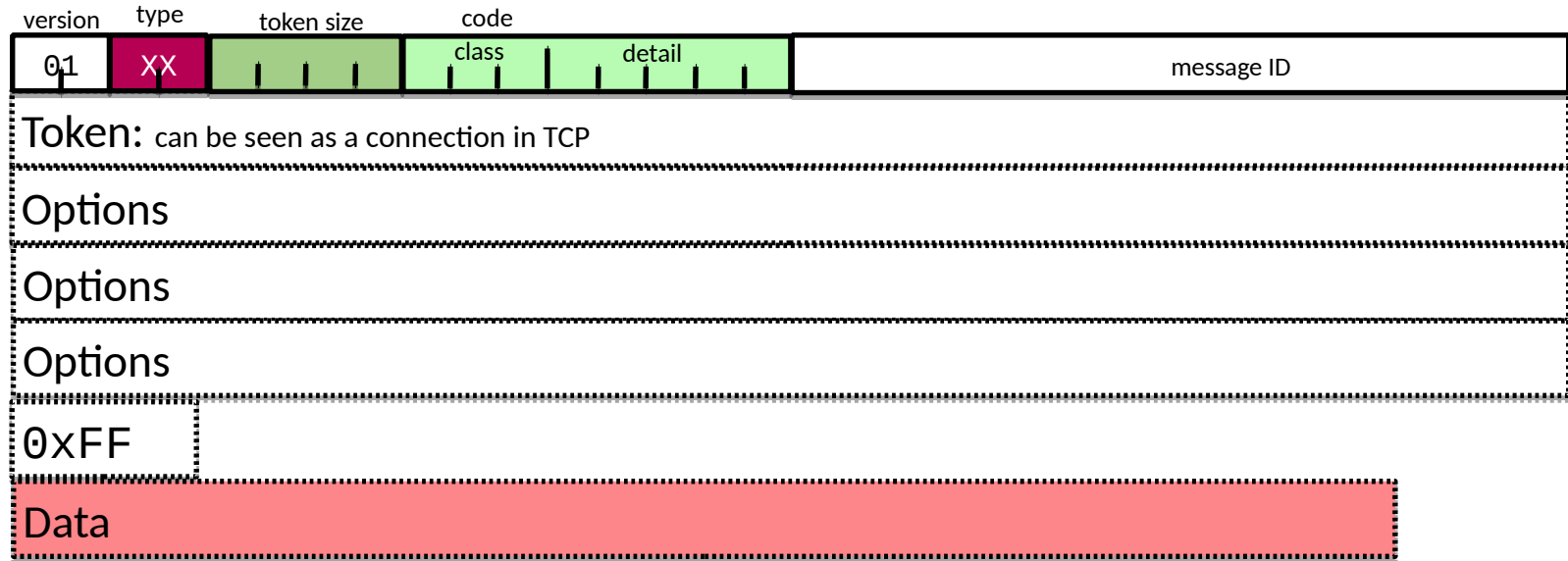
It allows to **selectively encrypt or authenticate parts of the CoAP message**. Each field is made to belong to one of three classes:

- Class E: encrypted via AEAD algorithm, 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).

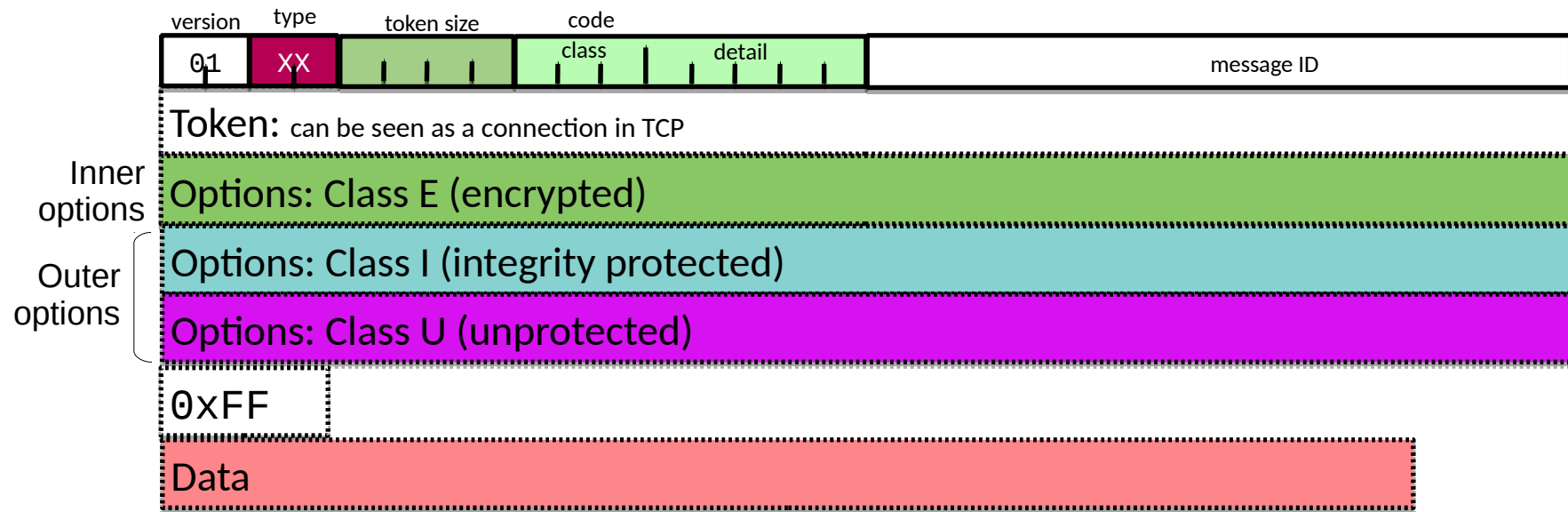
OSCORE: The Mechanics



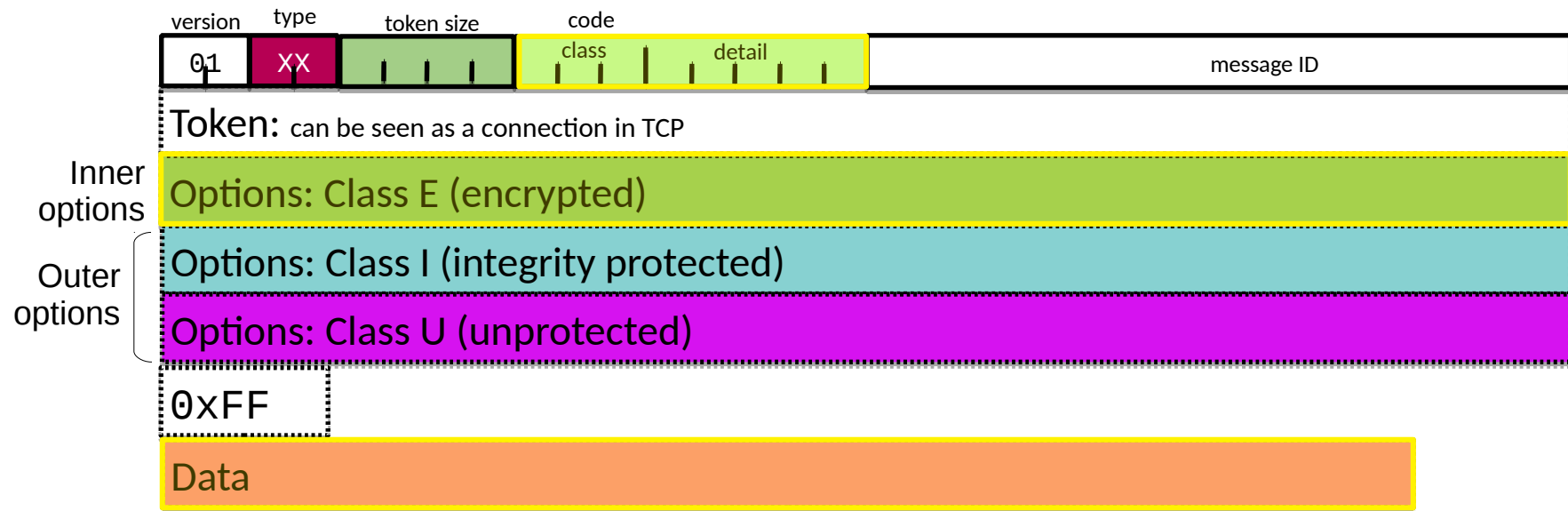
CoAP field classification



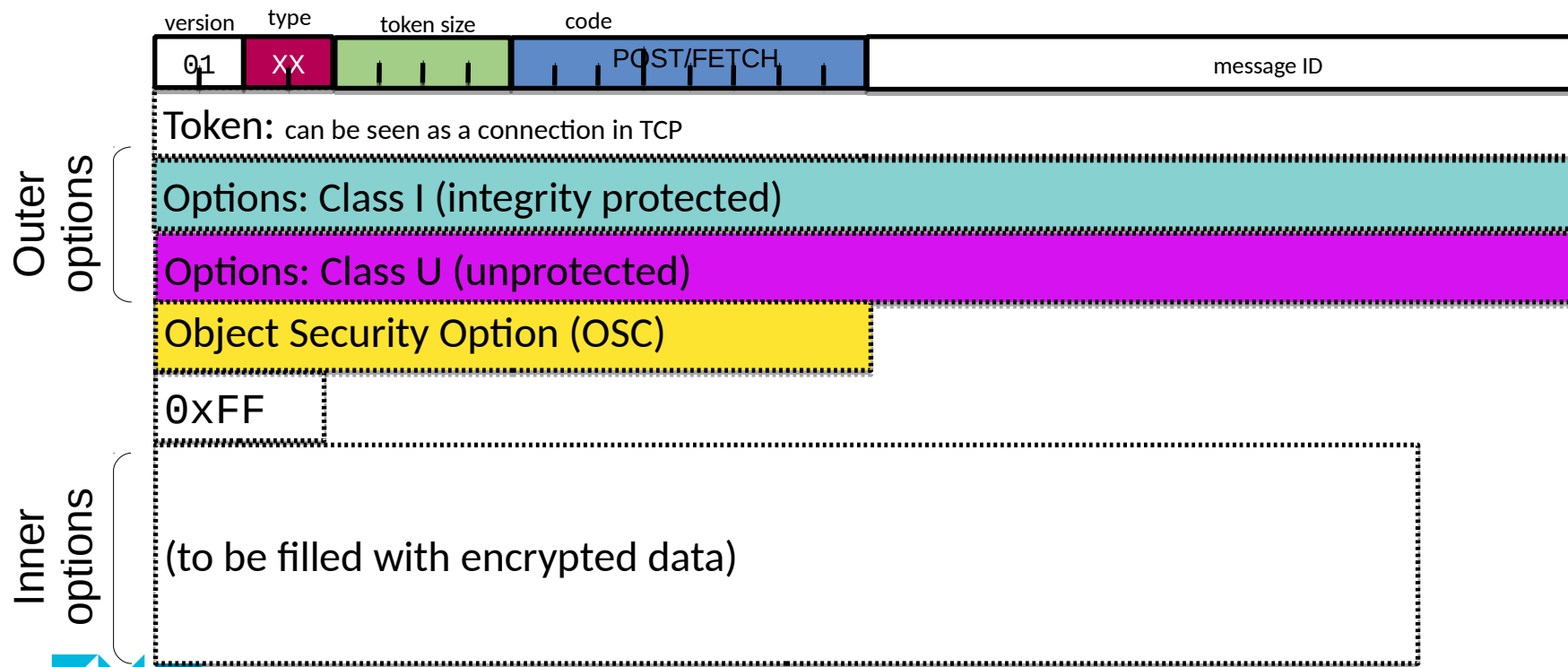
CoAP field classification



CoAP field classification

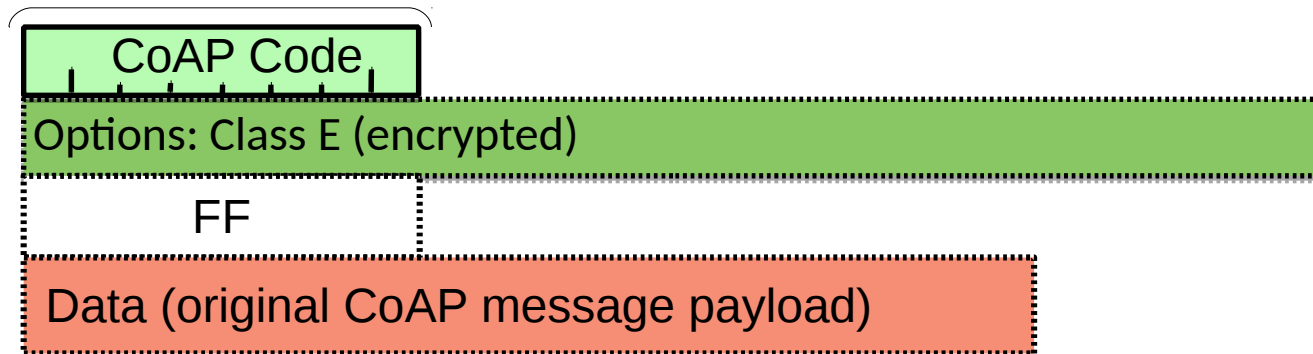


Prepare target OSCORE message

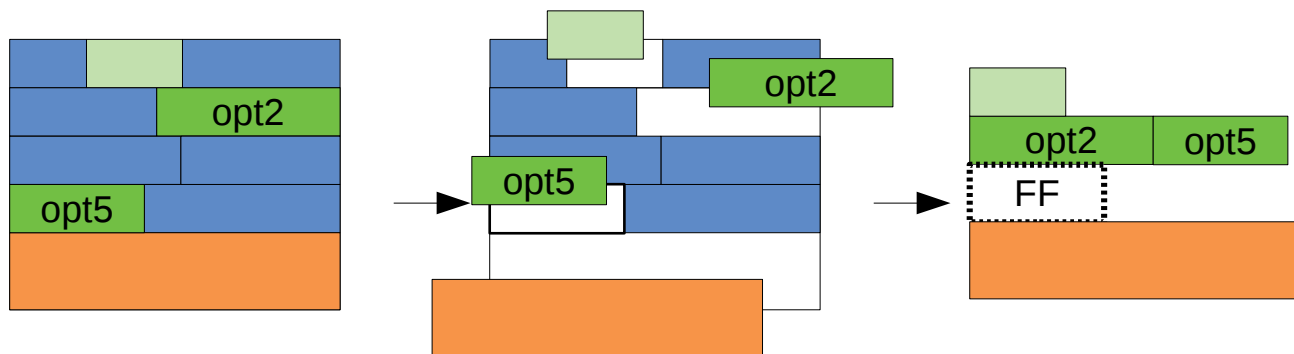


OSCORE Plaintext

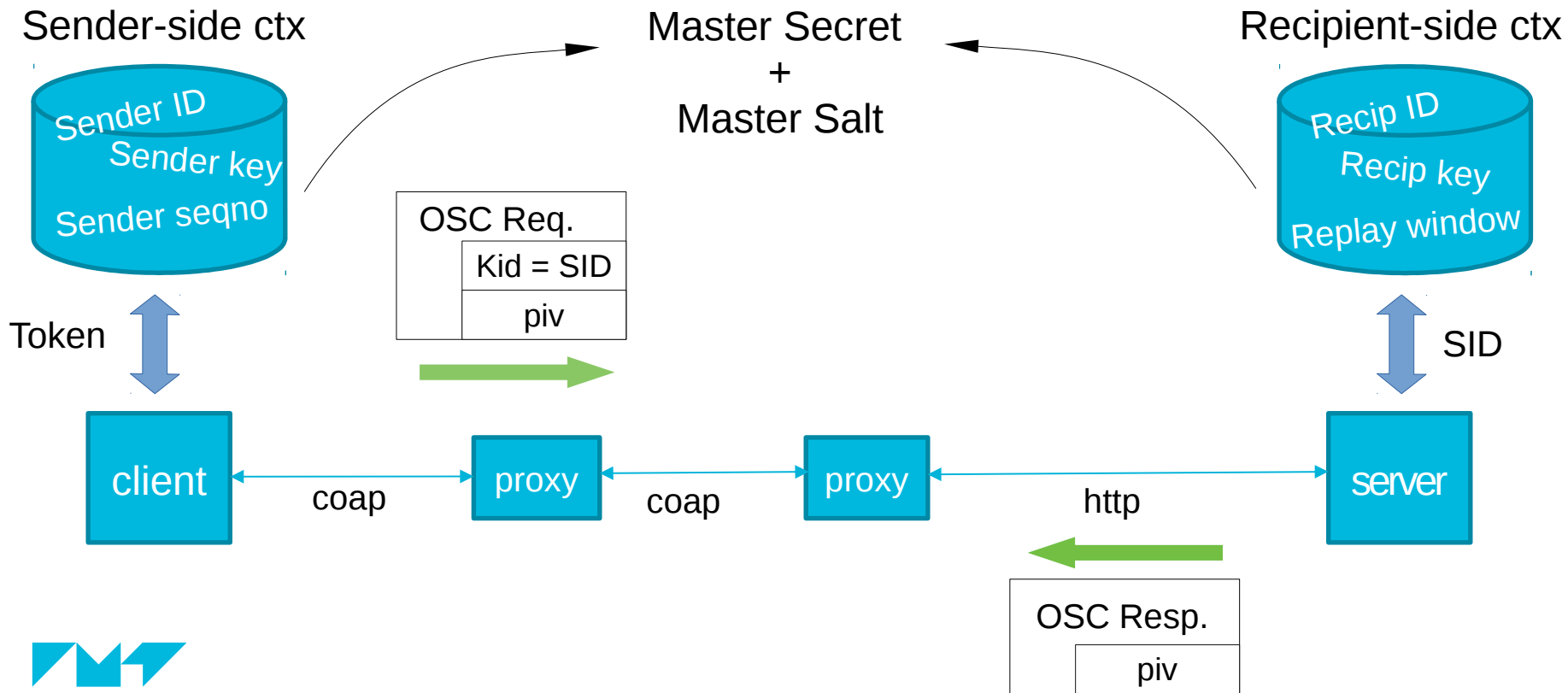
First byte



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



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)

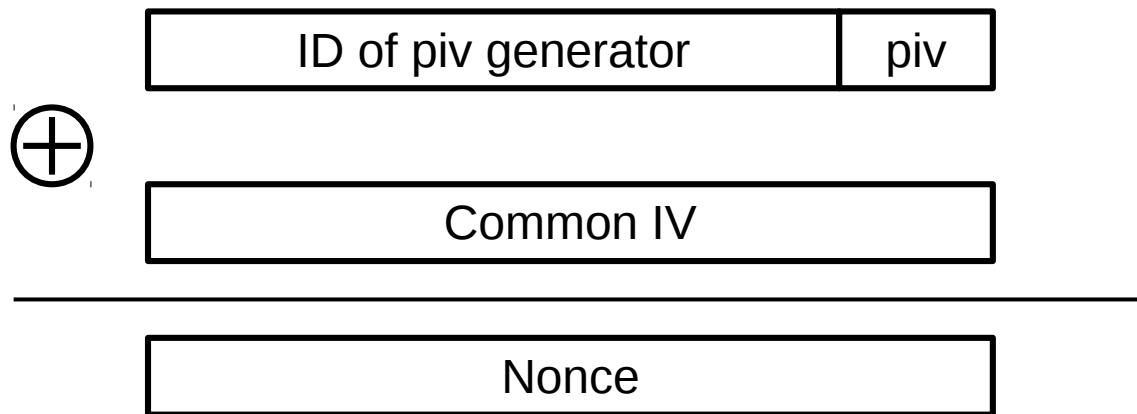
Master Salt

Master Secret

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

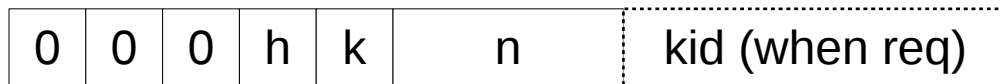
Security Parameters (cont.)

Nonce:



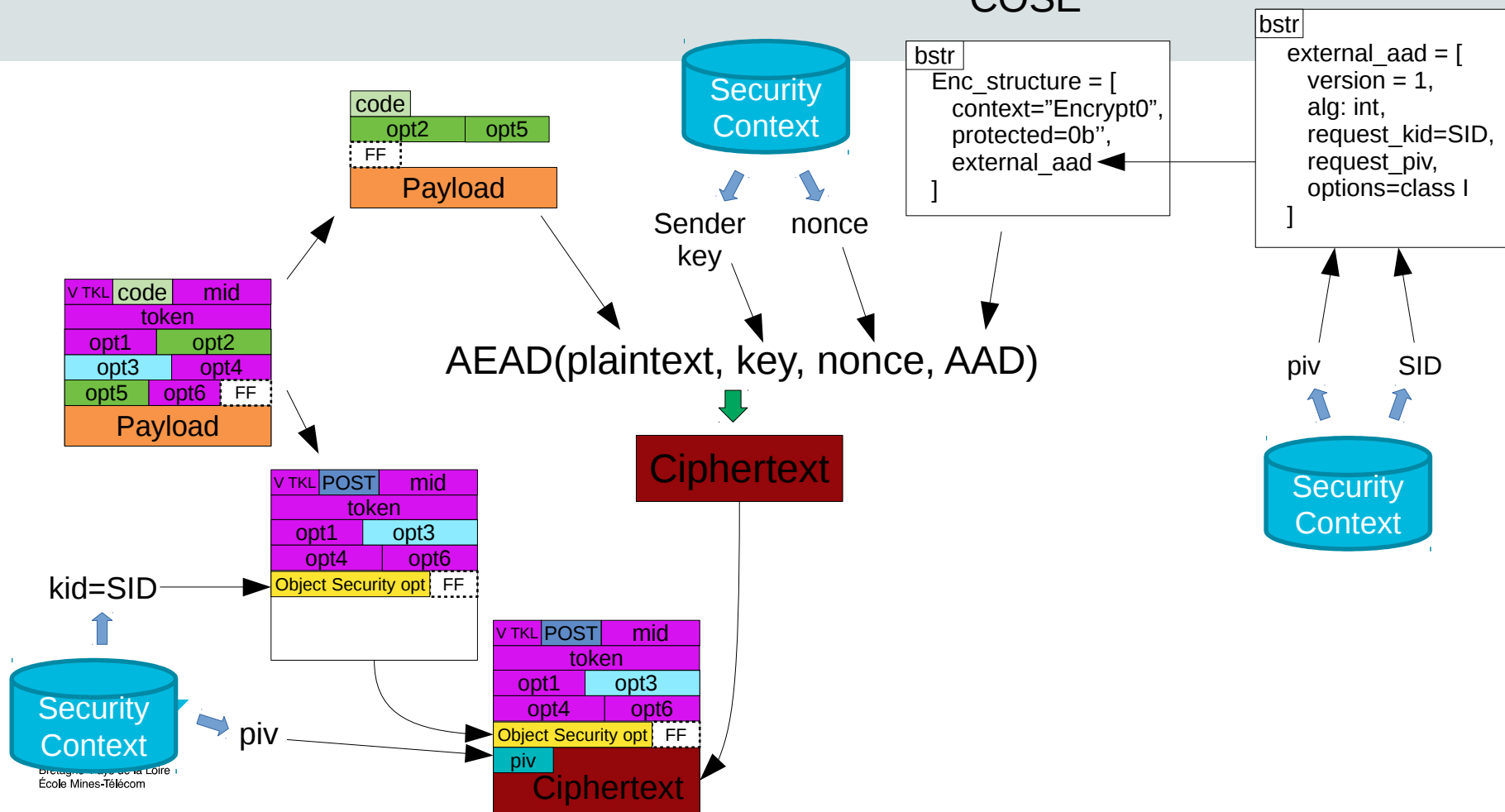
piv = sender sequence number, incremented each time we send a message

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 kid in octets,
- kid = kid

Encryption Process



Putting it to the test: aiocoap

Aiocoap is a Python implementation of CoAP with asynchronous 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>