# OSCORE

## End-to-end secure communication for CoAP

Marco Tiloca
Ph.D., Senior Researcher
Cybersecurity Unit – RISE

RI.SE

# Just a quick look at …

- **CoAP**
- **OSCORE**
- **Related activities**

# CoAP

# CoAP

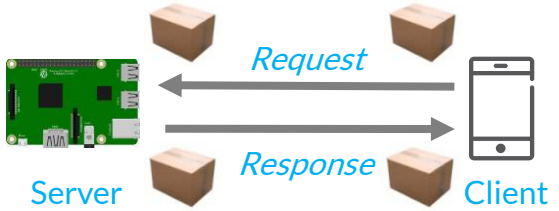
- **Constrained Application Protocol – RFC 7252**
  - Web-transfer protocol, at the application layer
  - Think of it as a “lightweight alternative to HTTP” for the Internet of Things

- **Supported transports**
  - Most used is UDP over IP
  - TCP or WebSockets (RFC8323)
  - SMS, CIoT, Bluetooth, LoRaWAN, IEEE 802.15.4, …

- **RESTful protocol**
  - What matters is the resource status on the server and its transfer
  - Supported methods: GET, POST, PUT, DELETE
  - More recent ones: FETCH and PATCH (RFC 8132)

# CoAP

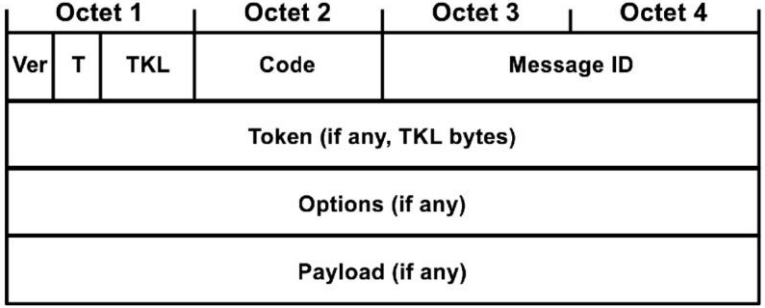
- **Message exchange based on the Client-Server model**
  - Request-Response exchanges
  - Some more message types for possible reliability




- **Token (variable length): set by the client sending a request**
  - A Response has the same Token value of the Request
  - I.e., a response matches a request by Token

# CoAP message format

- **Very compact size**




- **Options (variable length)**
  - Signaling extra protocol/message features
  - Encoding: Number/Length/Data

- **Payload (variable length)**
  - If present, it starts with a 0xFF byte as marker
  - The payload ends when the datagram ends

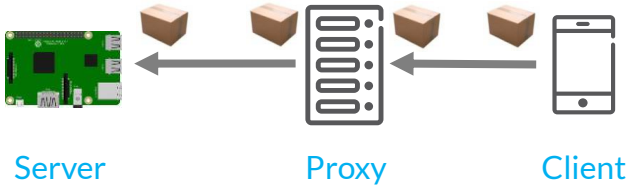
# Notable extensions

- **Observe (RFC 7641)**
  - The client sends a Request and registers as an Observer
  - The server sends a Response with the current resource representation
  - When the resource content changes, the server sends a Notification response
  - The client can unregister or just silently forget about an observation

- **Blockwise (RFC 7959)**
  - A whole content to deliver (body) can be too big to transfer
  - A sender can split the body into chunks (blocks), each sent in one CoAP message
  - The recipient will re-build the whole body when received all the blocks

# Proxies

- **Intermediary nodes between client and server**
  - Forward-proxy or reverse-proxy
  - HTTP-to-CoAP or CoAP-to-HTTP cross-proxy

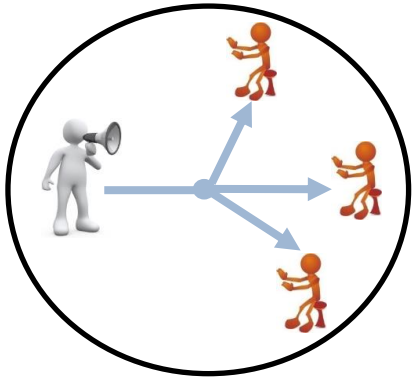



- **Forward proxy**
  - The client sends a request to the proxy, indicating the origin server to target
  - The proxy forwards the request to the origin server
  - The server sends a response to the proxy, thinking of it as a client
  - The proxy forwards the response to the client

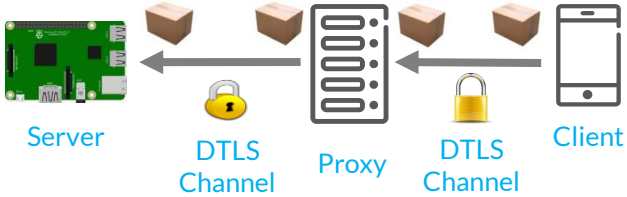
- **Performance benefit**
  - The proxy can cache responses to GET/FETCH requests
  - After the first client, next clients will get the response from the proxy

# Some more features

- **Support for one-to-many communication**
  - Over UDP and IP multicast
  - Over any transport supporting one-to-many delivery



- **Message security**
  - Nothing natively
  - Original suggestion for DTLS 1.2 (RFC 6347)
    - Security at the transport layer
    - Hop-by-hop – i.e. not end-to-end
  - With DTLS 1.2
    - Security terminates at a proxy, that starts a second secure session with the server
    - No support for CoAP over multicast



# OSCORE

# OSCORE

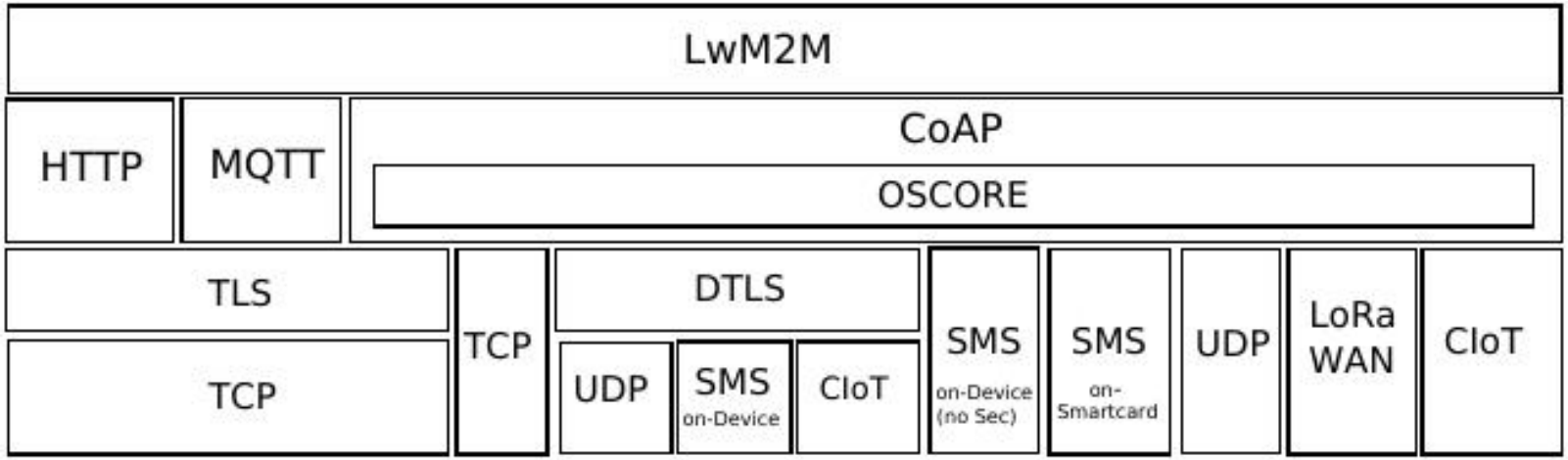
- **Object Security for Constrained RESTful Environments – RFC 8613**
  - End-to-end security at the application layer
  - Encryption, integrity and replay protection
  - Selective protection of CoAP messages
  - Use COSE (CBOR Object Signing and Encryption) – RFC 8152

- **Faithful to CoAP**
  - Works wherever CoAP works (UDP, TCP, SMS, CIoT, IEEE 802.15.4, …)
  - Supports options and proxy forwarding (RFC 7252)
  - Supports Observe (RFC 7641) and Blockwise (RFC7959)
  - Supports PATCH & FETCH (RFC 8132) and CoAP over TCP (RFC 8323)

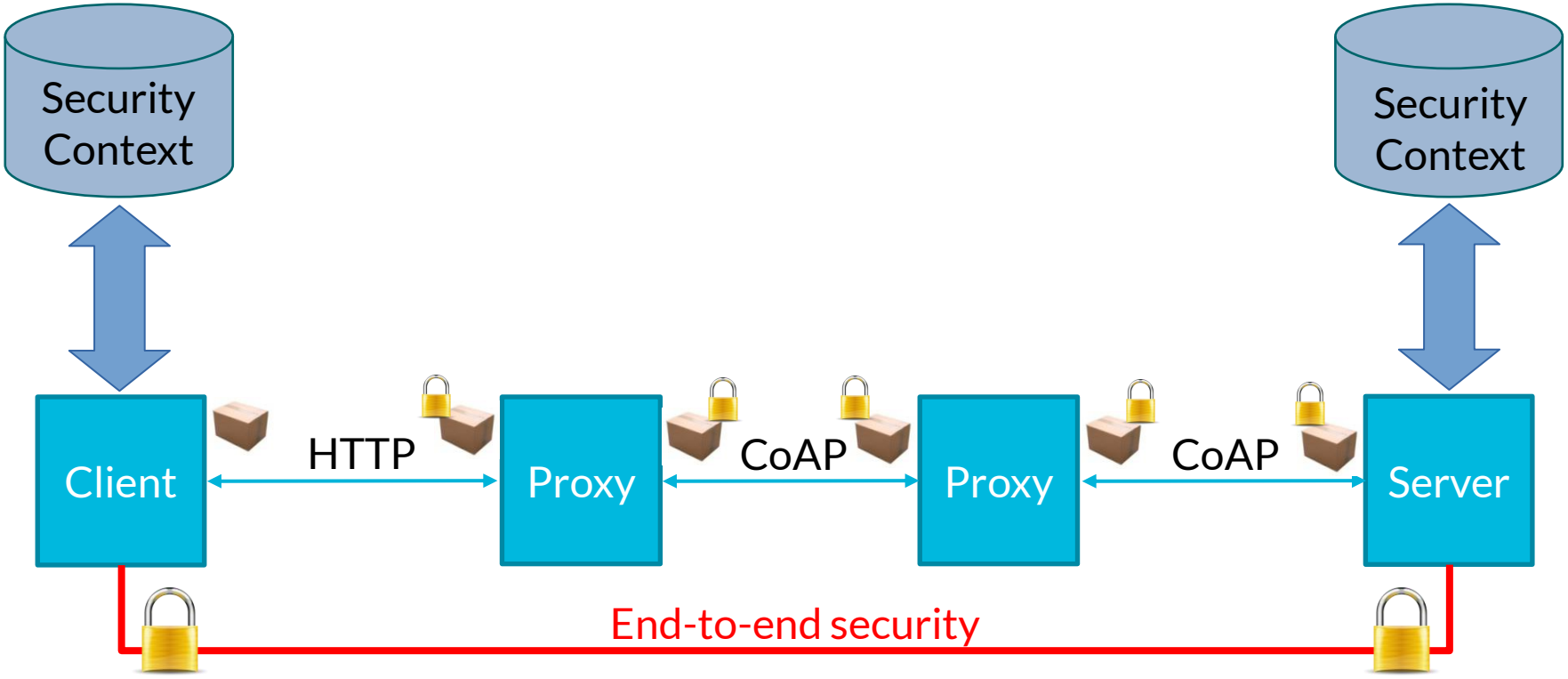
- **Can be used with HTTP and when translating between CoAP and HTTP**

# OSCORE

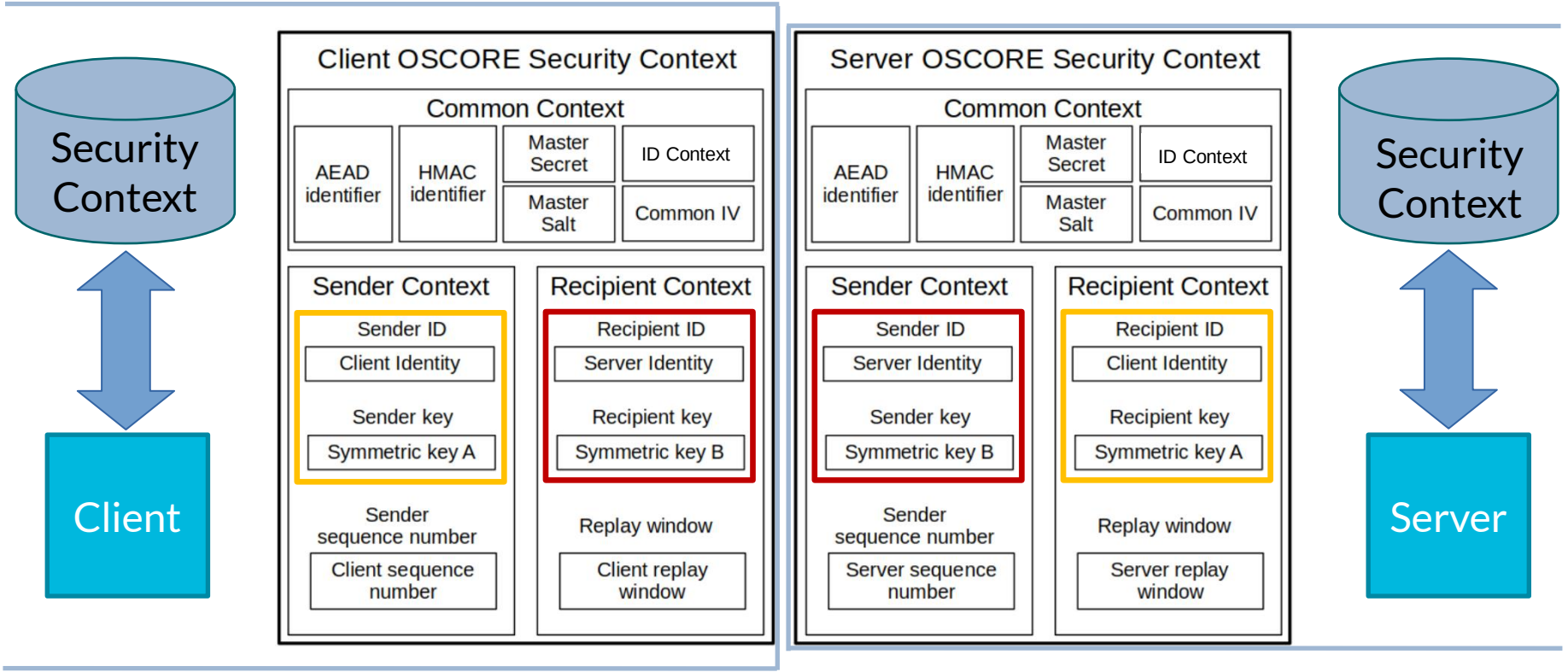
- **Included in OMA LwM2M as application-layer security solution**



# OSCORE – End-to-End message protection

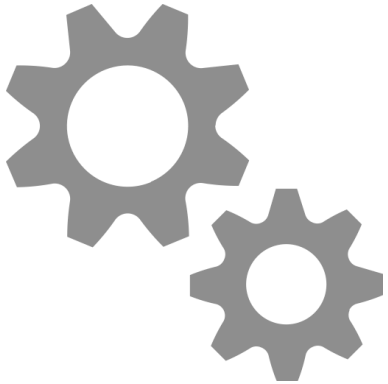


# OSCORE – Security context



# OSCORE – The mechanics

- **How it works**
  - OSCORE Security Context shared between the two peers
  - Use signaled through the “OSCORE” CoAP option
  - It wraps a CoAP message into a COSE object
  - It delivers the result as a secured CoAP message

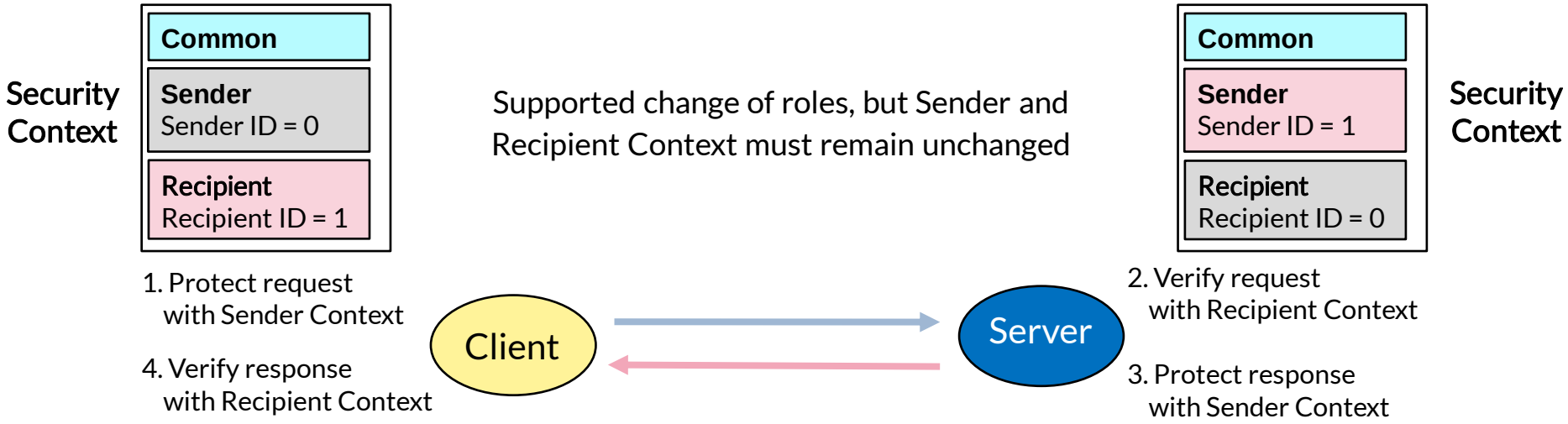


- **Selective encryption/authentication of CoAP fields**
  - Class E: encrypted via the AEAD algorithm, hidden inside the OSCORE payload
  - Class I: integrity protected and visible from the outside (outer options)
  - Class U: unprotected and visible from the outside (outer options)

- **What is protected?**
  - Encrypted and authenticated: original payload; original REST code; class E options
  - Authenticated only: class I options; other info as Additional Authenticated Data (AAD)

# OSCORE – The mechanics

- **OSCORE Security Context – Common, Sender and Recipient parts**
  - Contexts derived from a few input parameters, to be pre-established



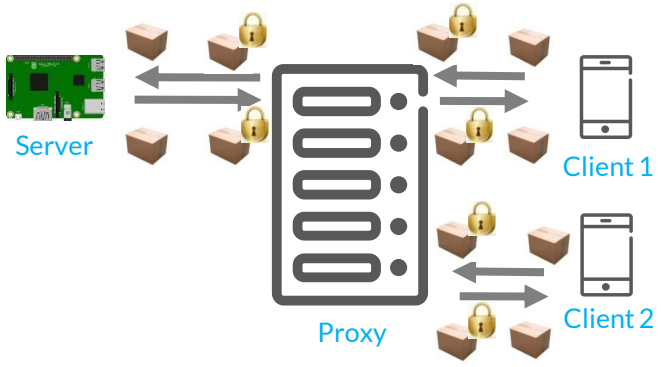
# Related activities

# Related activites (1/2)

- **Methods to establish OSCORE keys**
  1. OSCORE profile – Use the ACE Framework for Authentication and Authorization
  2. Optimized execution of the EDHOC key establishment protocol

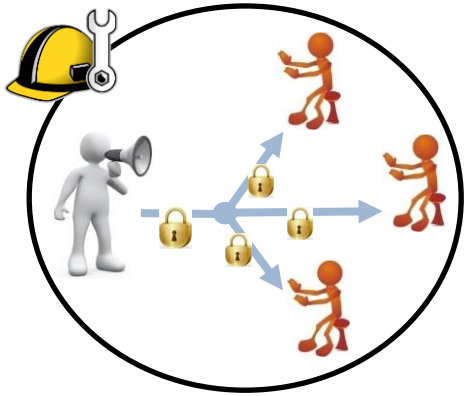
- **Lightweight method to update OSCORE keys (KUDOS)**
  - Compact, with minimal message exchanges
  - Able to preserve high-level security properties
  - Robust against node rebooting

- **Advanced use for proxies**
  - Caching of OSCORE-protected responses
  - Using of OSCORE also at proxies



# Related activites (2/2)

- **Group OSCORE**
  - End-to-end security for CoAP group communication
  - Adapted structures, constructs and mechanisms from OSCORE
  - The same security requirements of OSCORE are fulfilled
    - Source authentication is ensured
    - Secure binding between request and responses is ensured



- **Key provisioning for Group OSCORE**
  - Coordinated by a Group Manager
  - Contextual with the authorized joining of a group
  - Authorization to join enforced by using the ACE framework

# IETF standardization

- Open standards through an open process

> The mission of the IETF is to make the Internet work better by producing high quality, relevant technical documents that influence the way people design, use, and manage the Internet. [RFC 3935]

- Several IoT-related Working Groups
  - CoRE, ACE, LAKE, …

- Join and participate!

# Thanks!

*marco.tiloca@ri.se*

RI.
SE