Secure group communication for CoAP

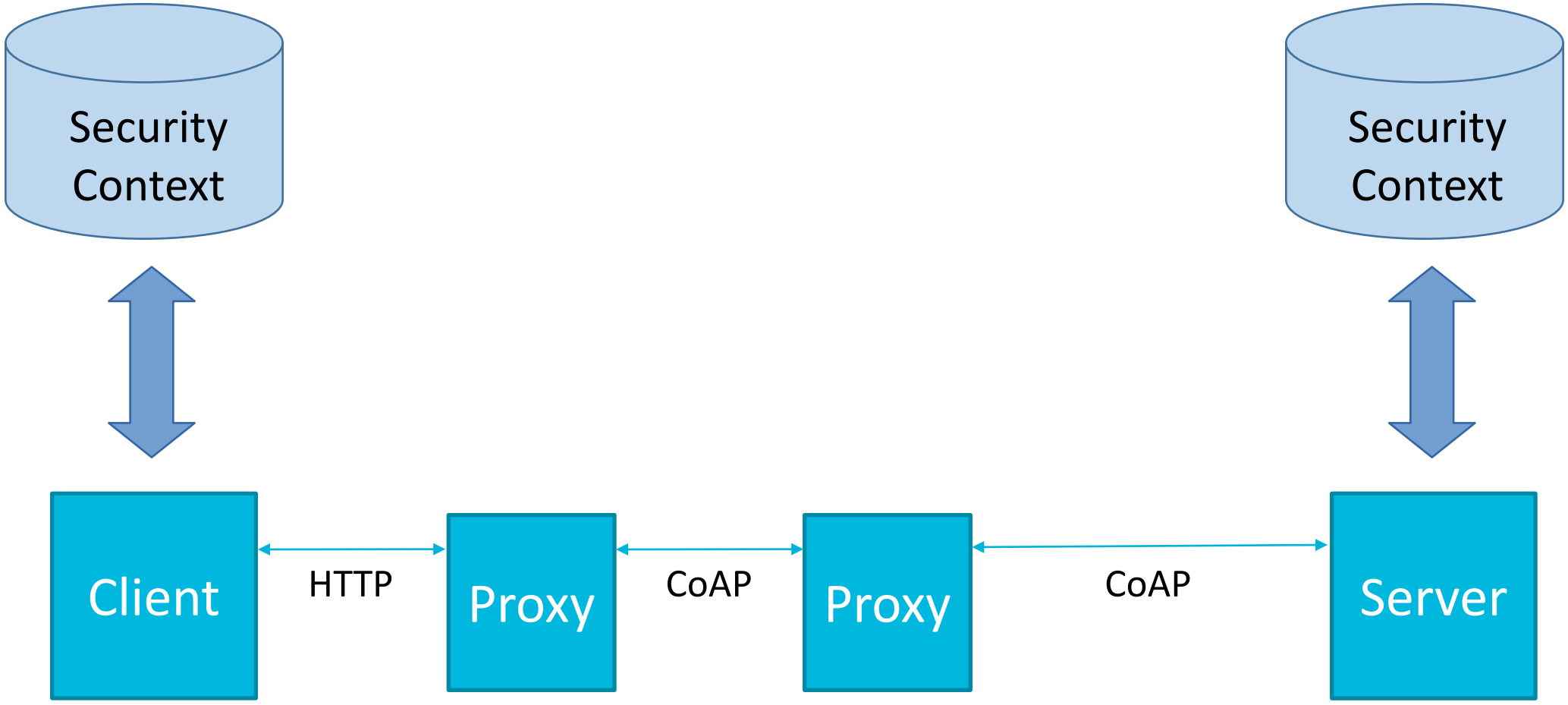
OCF & T2TRG

2018, September 27th

OSCORE

- Object Security for Constrained RESTful Environments [1]
  - End-to-end security at the application layer
  - Encryption, integrity and replay protection
  - 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
- Included in OMA DM LwM2M v1.1 as application-layer security solution

OSCORE – The mechanics



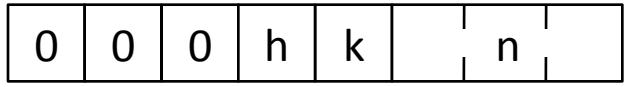
OSCORE

- How OSCORE works
  - Signaled through the “OSCORE” CoAP option
  - Wraps a CoAP message into a COSE object (RFC 8152)
  - Delivers the result as a secured CoAP message

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

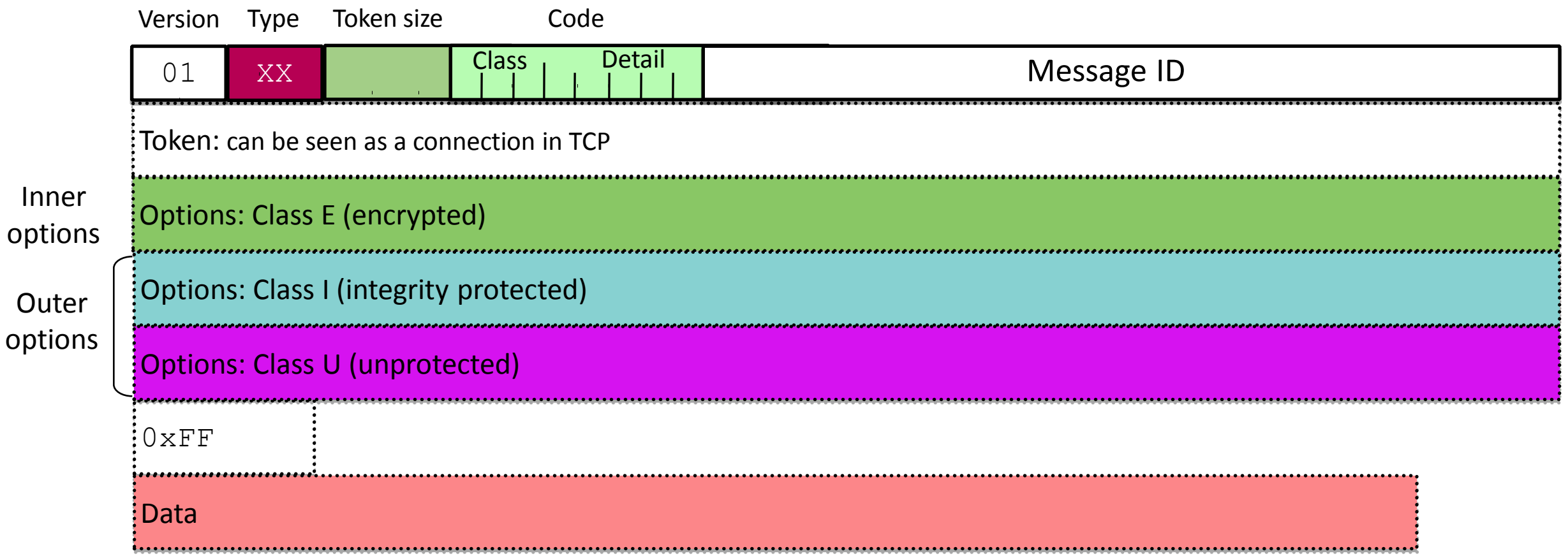
- “OSCORE” CoAP option (Class U)
  - 1 Flag byte
  - “Partial IV” set to message Sequence Number
  - “kid” set to Sender/Recipient ID, mandatory for requests
  - “kid context” optionally used to disambiguate security contexts

Flag byte



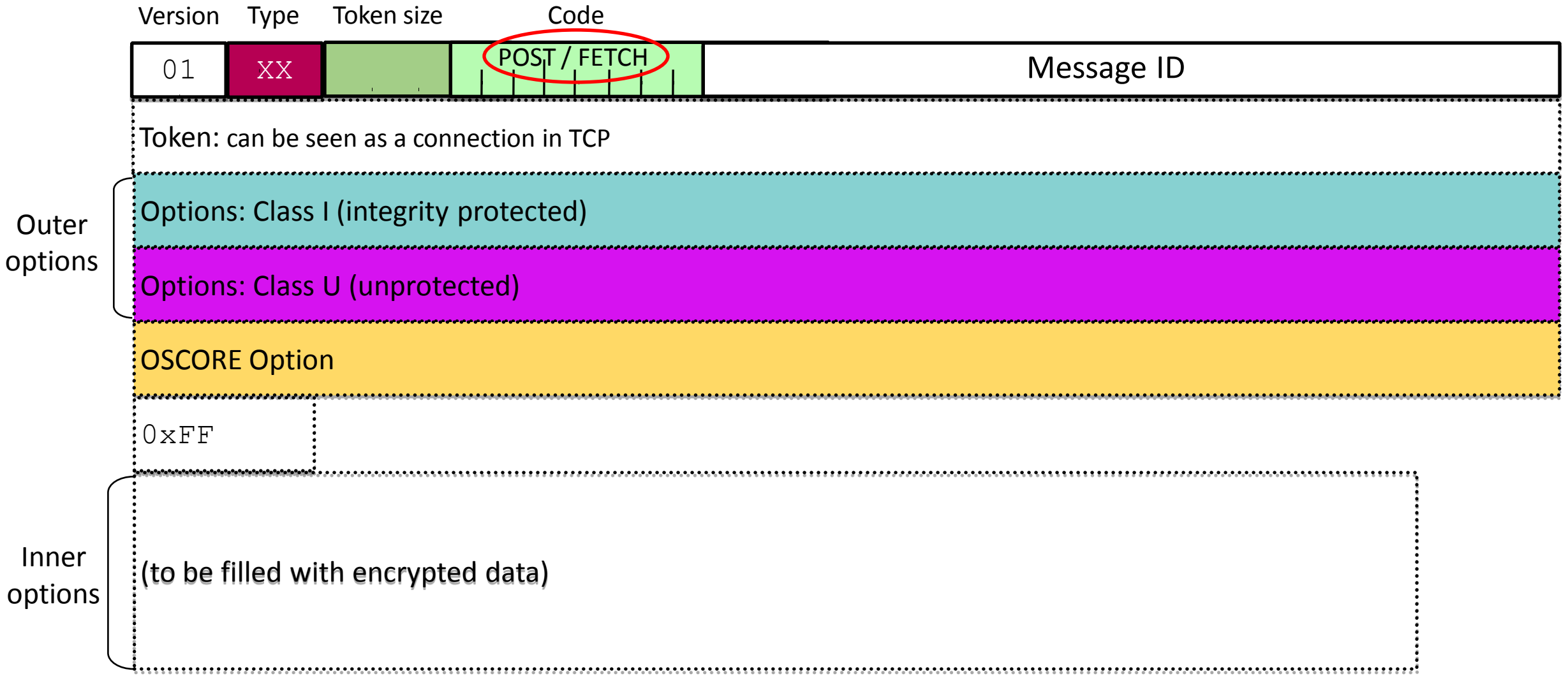
h = 1 if “kid context” is present
k = 1 if “kid” is present
n = length of Partial IV in octets

OSCORE – CoAP field classification

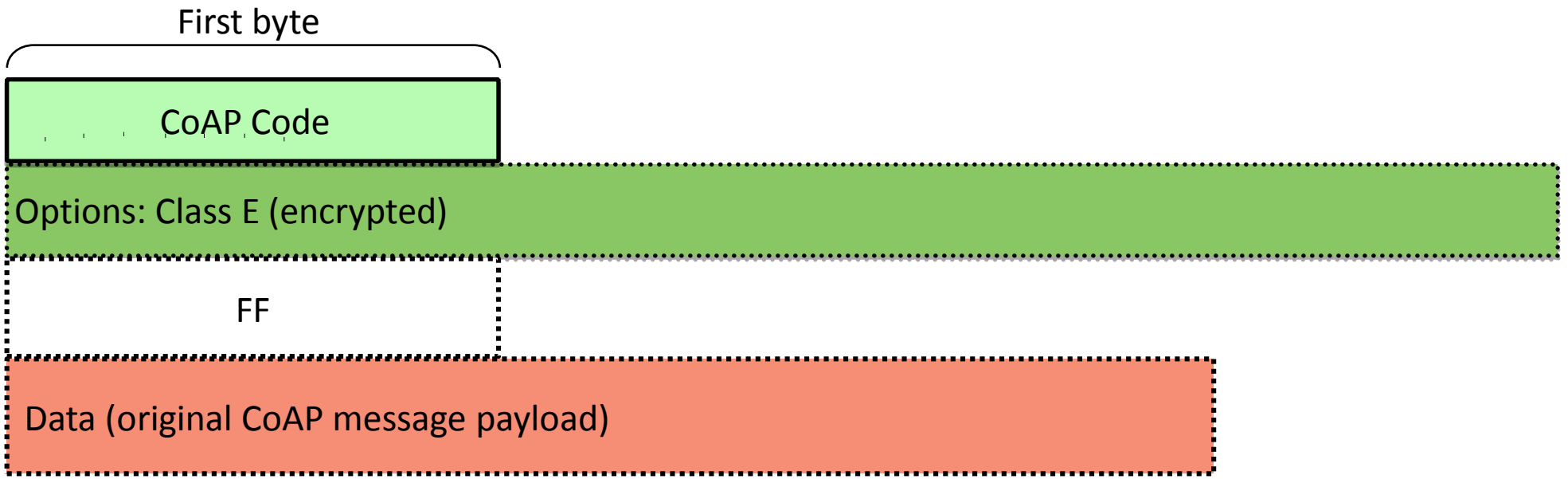


The representation of Options above is purely symbolic

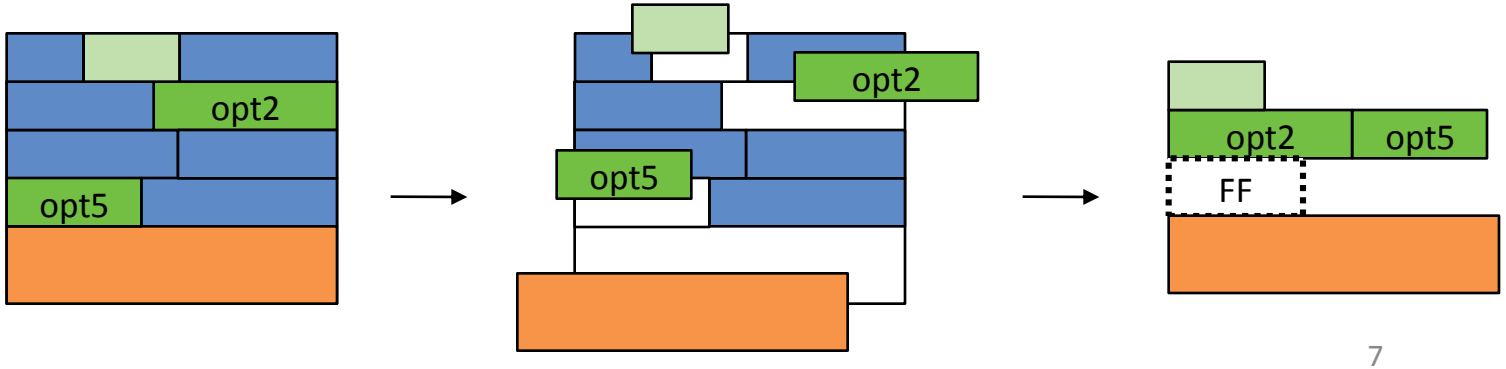
Prepare target OSCORE message



OSCORE Plaintext



Options are reordered with delta encoding as per CoAP



Security parameters

Pre-established parameters:

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

'?' indicates an optional parameter

Default value is assumed if not specified

Derivation of Key & Common IV:

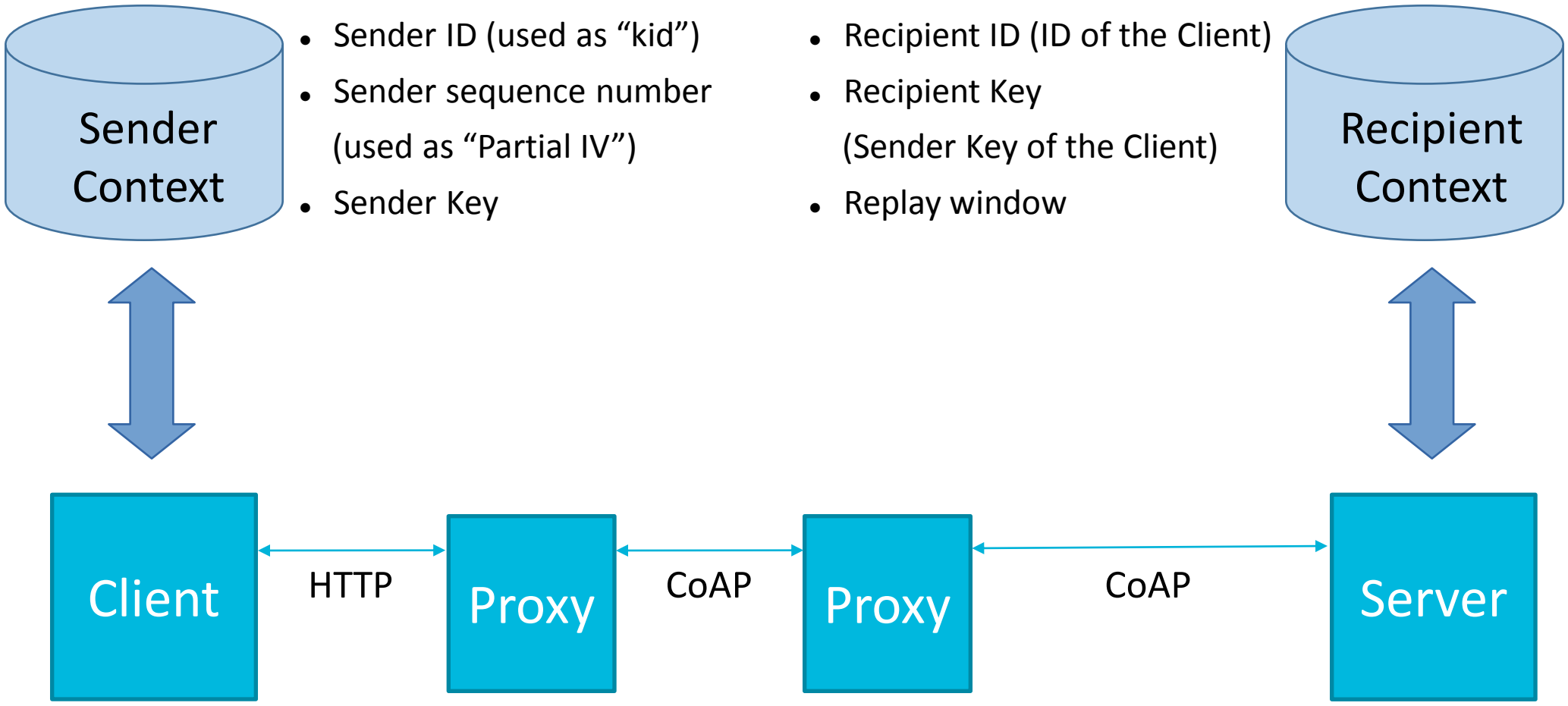
key/IV = HKDF(salt, IKM, info, L)

Master Salt

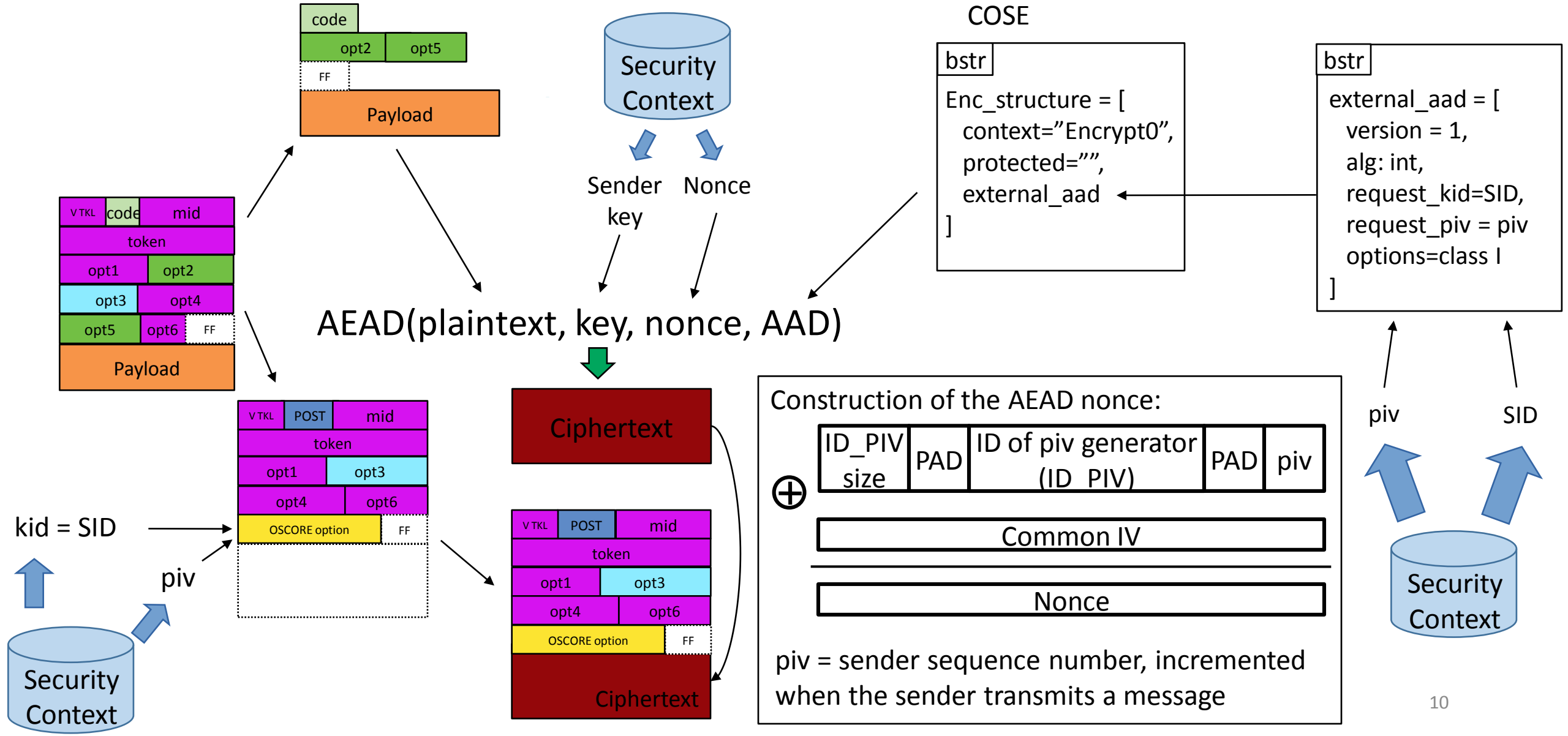
Master Secret

[
id = SID / RID / nil,
kid_context = bstr / nil
alg_aead = int /tstr
type = "key" / "iv",
L = size of key in octets
]

OSCORE – The mechanics

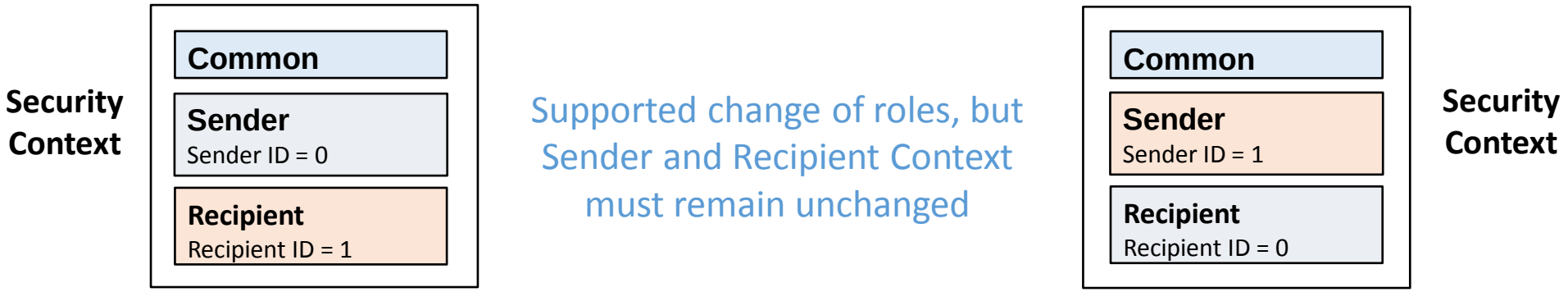


OSCORE - Encryption process



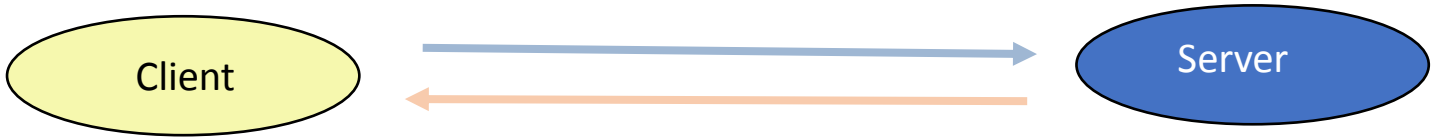
OSCORE Security Context

- The Security Context consists of Common, Sender and Recipient parts
  - Contexts are derived from a small set of input parameters using HKDF (RFC 5869)
  - Master Secret, Sender ID and Recipient ID must be pre-established



1. Protect request with Sender Context

4. Verify response with Recipient Context



2. Verify request with Recipient Context

3. Protect response with Sender Context

Group OSCORE

- Motivation – From Group CoAP (RFC7390)
  - To further mitigate the threats, security enhancements need to be developed at the IETF for group communication.

- It is natural to extend OSCORE to secure group communication
  - Devices are organized into groups and communicate (also) over Multicast IP
  - Lighting control, building control , SW/FW update, emergency multicast, …

- Design choices must ensure:
  - Fulfillment of the same OSCORE security requirements
  - Flexible accommodation of different group configurations

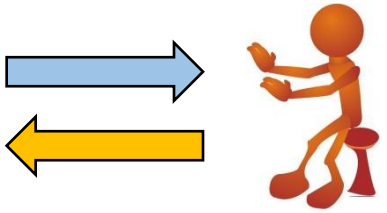
- This led to the design of Group OSCORE [5]

Group OSCORE

- Group Manager
  - Coordinate the join process and group rekeying
  - Ensure uniqueness of Group IDs in its pool of groups
  - Ensures uniqueness of Endpoint IDs in a same group
  - Recommended repository for public keys of group members

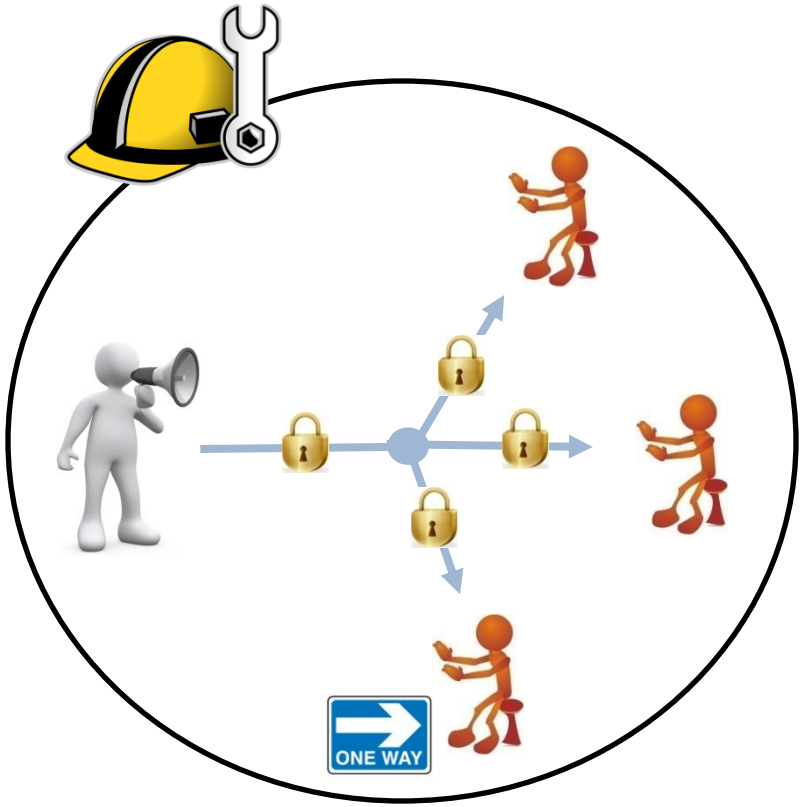
- Clients can send also CoAP requests over Multicast IP

- Servers may reply to (multicast) CoAP requests
  - CoAP responses are optional and always over Unicast IP

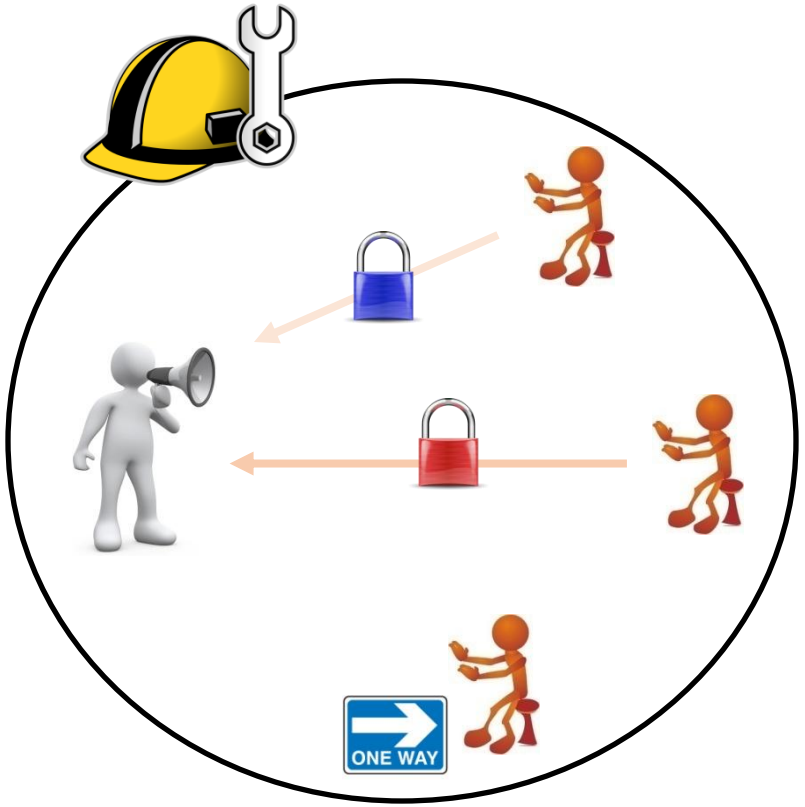


- Silent server
  - A server endpoint that never replies to received CoAP requests

Group OSCORE - Protocol overview



Secure group request



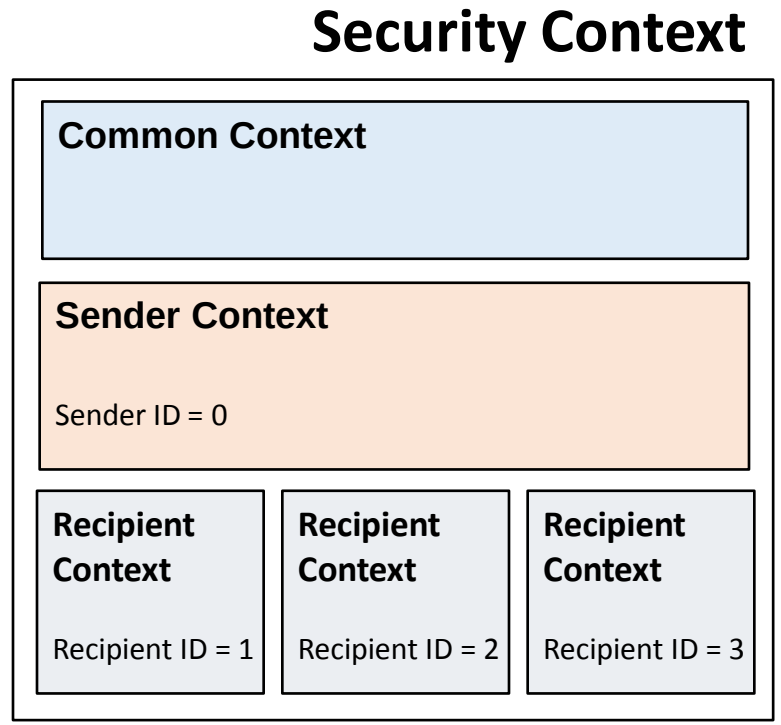
Secure individual responses

Group OSCORE - Features at a glance

- Same structures, constructs and mechanisms of OSCORE
- Same security requirements from OSCORE are fulfilled
  - Binding of multicast requests with multiple related responses
- Source authentication
  - Signature algorithm specified in the Common Security Context
  - Asymmetric-key counter signature embedded in the COSE object
- "OSCORE" CoAP option
  - The Flag byte signals also the presence of a counter signature
  - The "kid" as Sender/Recipient ID is mandatory also in Responses
  - The "kid context" is included in Requests and specifies the ID of the group
- Keying material
  - Shared keying material to locally derive individual keying material
  - Individual keying material used to protect group communications

Group security context (per device)

- 1 Common Context, including also:
  - Group Identifier (*)
  - Counter signature algorithm
- 1 Sender Context, including also:
  - Public-private key of the device
- Multiple Recipient Contexts, each of which:
  - Including also the public key of the sender device
  - Created at runtime, when receiving the first message from that sender device

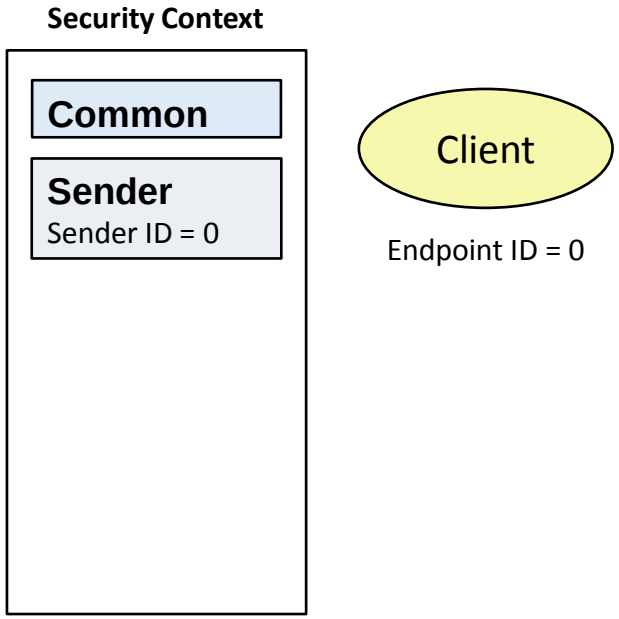


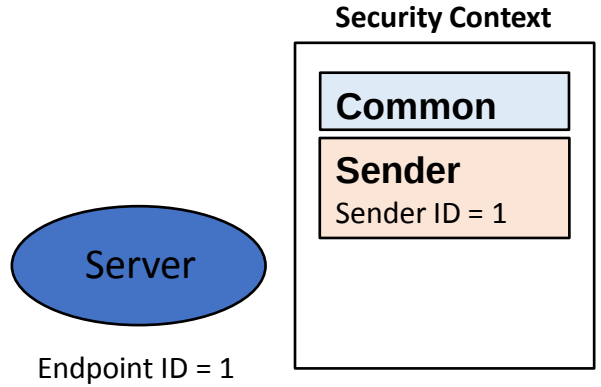
Yes, I have an example ☺

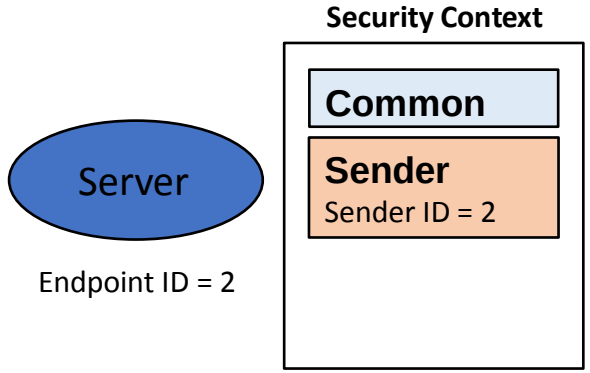
* Unique under the same Group Manager, it identifies the OSCORE “Security Group”.

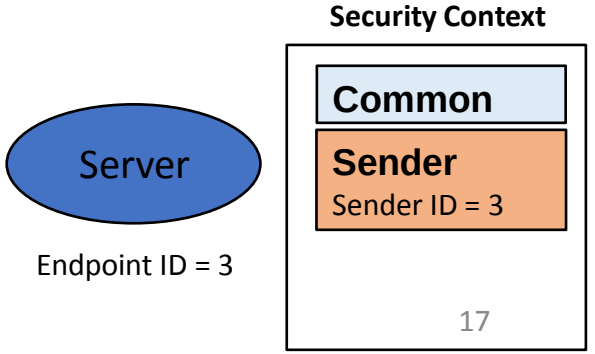
Message processing (1/6)

No previous communication has occurred.





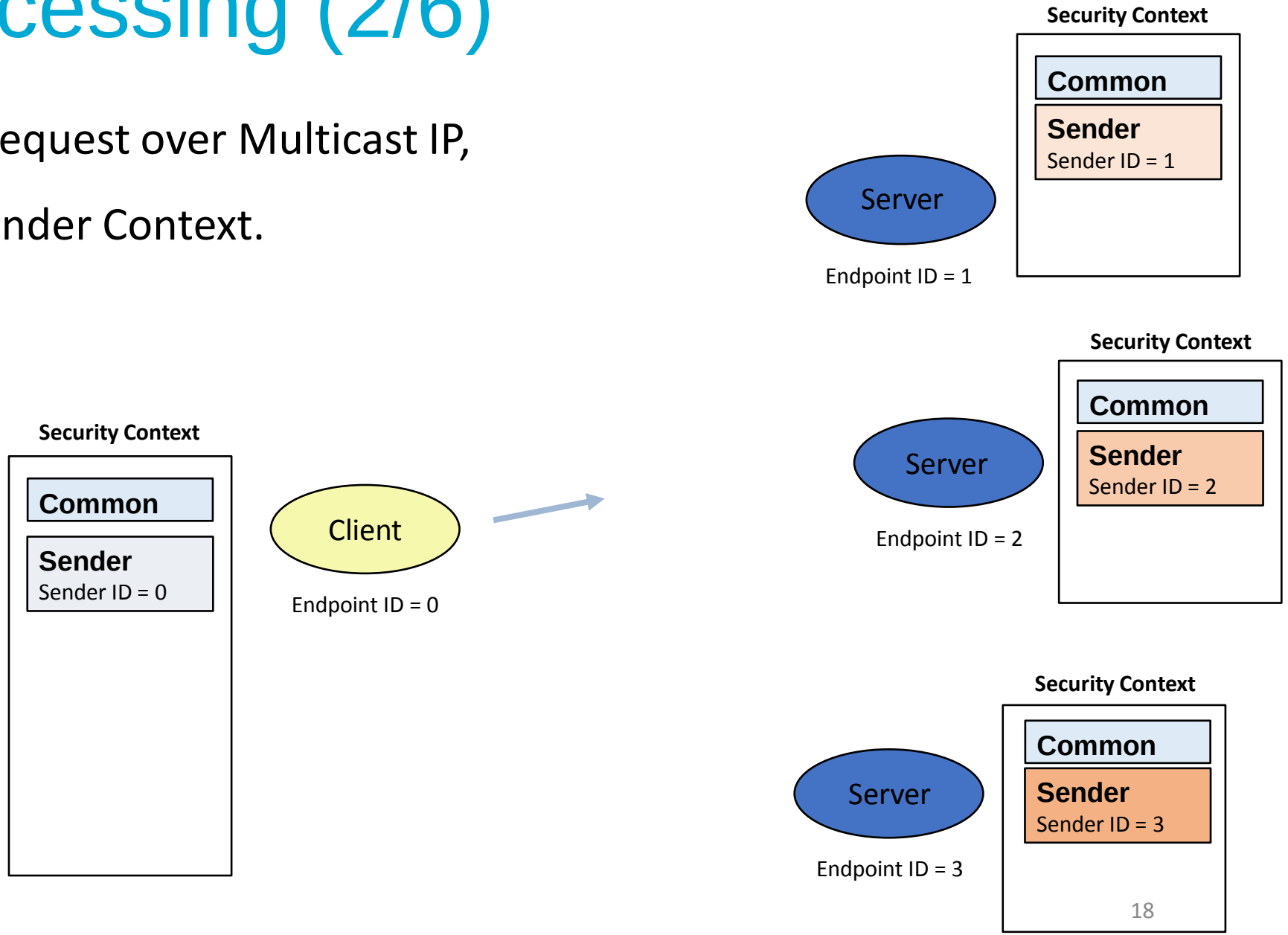




Message processing (2/6)

The Client sends a group request over Multicast IP,

securing it with its own Sender Context.

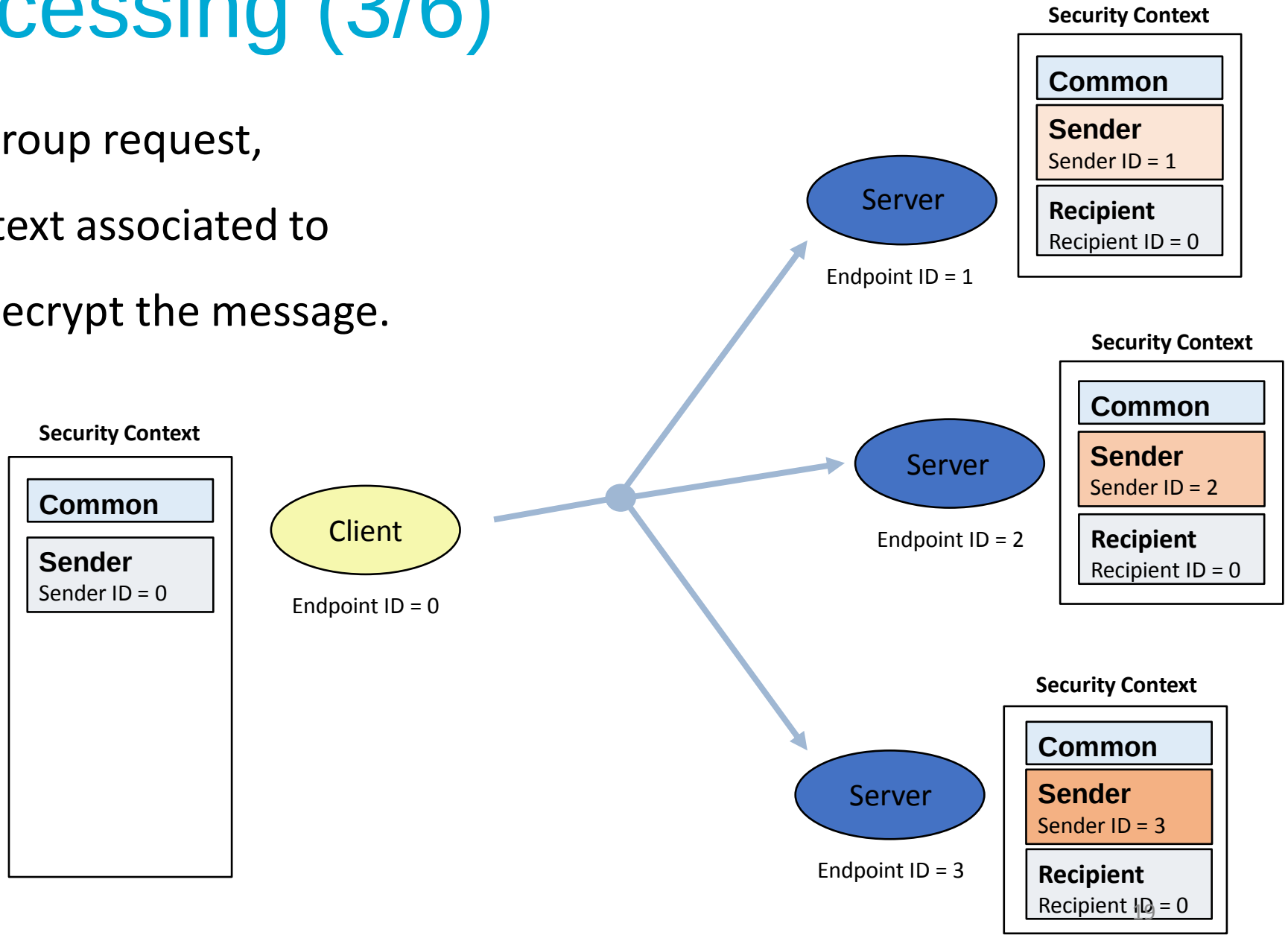


Message processing (3/6)

Each Server receives the group request,

derives the Recipient Context associated to

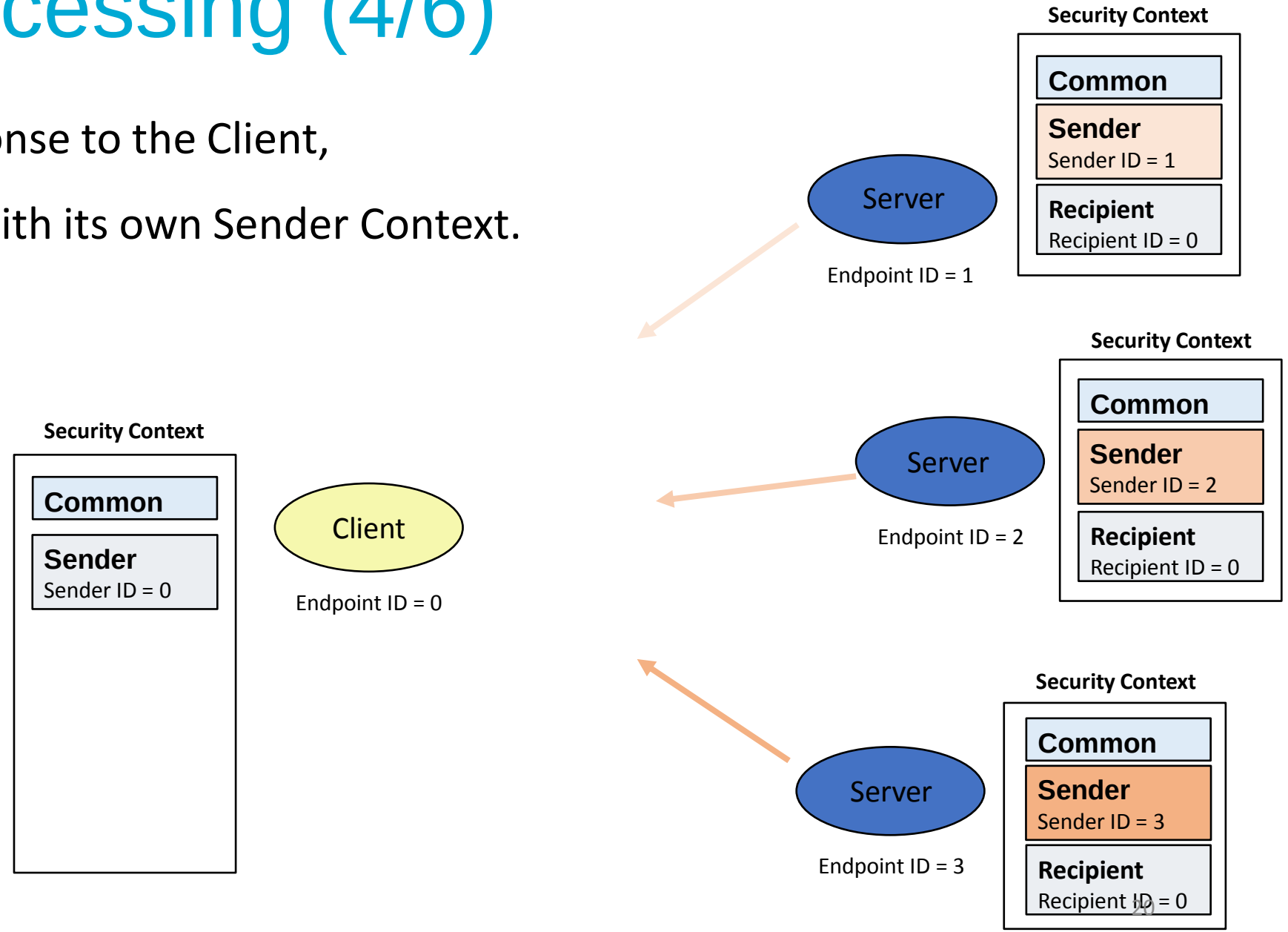
the Client, and uses it to decrypt the message.



Message processing (4/6)

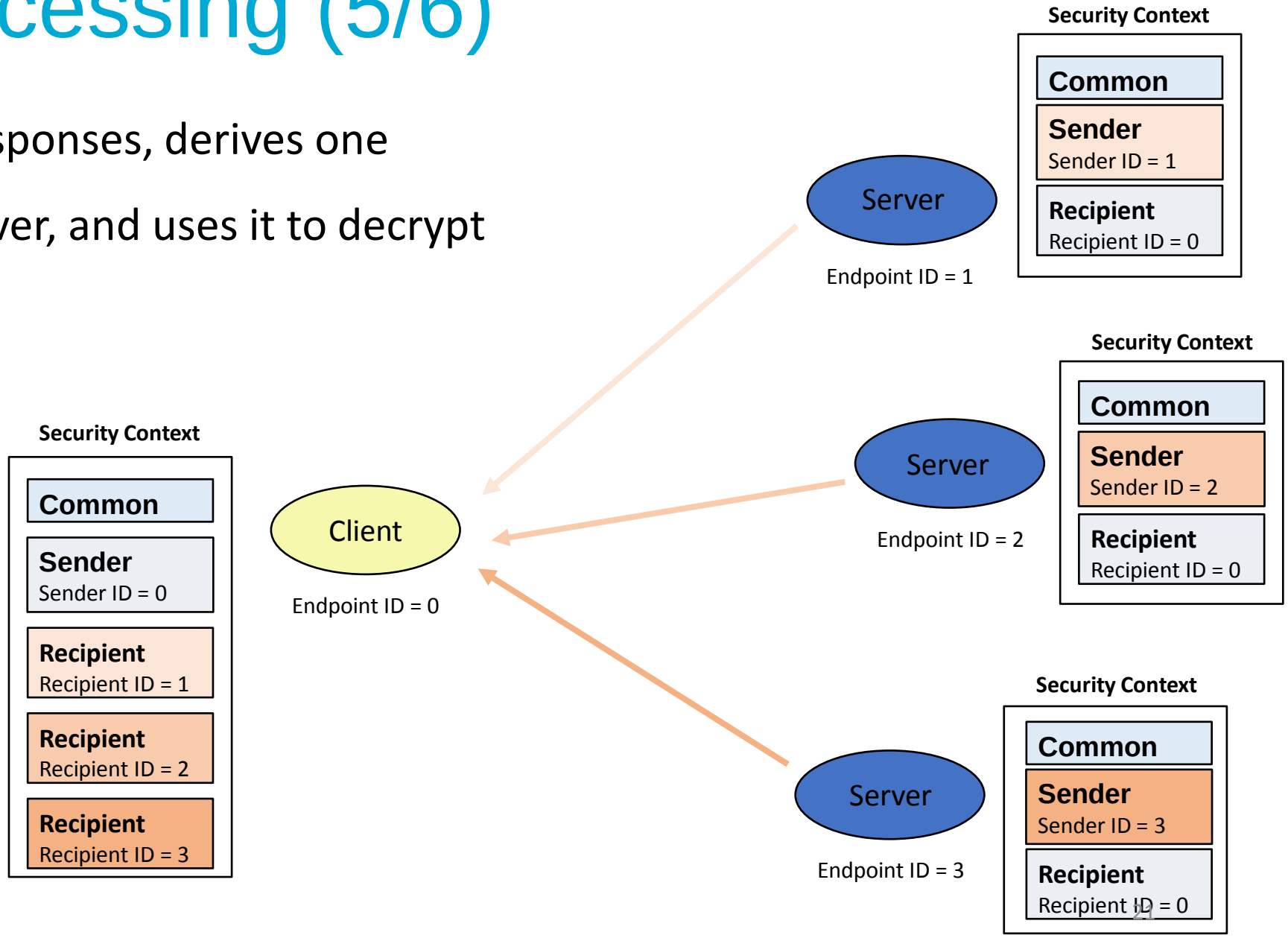
Each Server sends a response to the Client,

protecting the message with its own Sender Context.



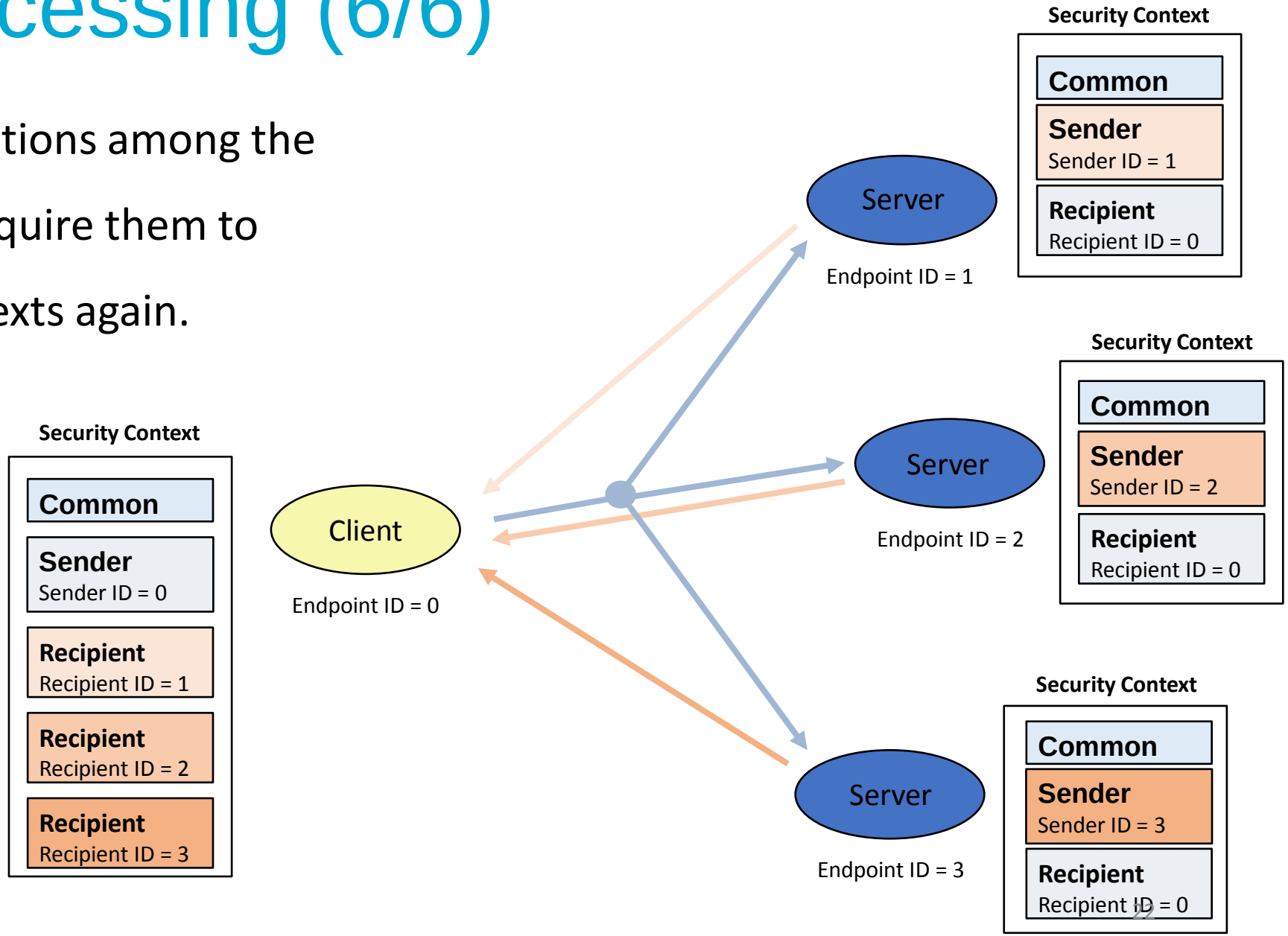
Message processing (5/6)

The Client receives the responses, derives one Recipient Context per Server, and uses it to decrypt the respective response.



Message processing (6/6)

Further group communications among the same endpoints do not require them to derive the Recipient Contexts again.

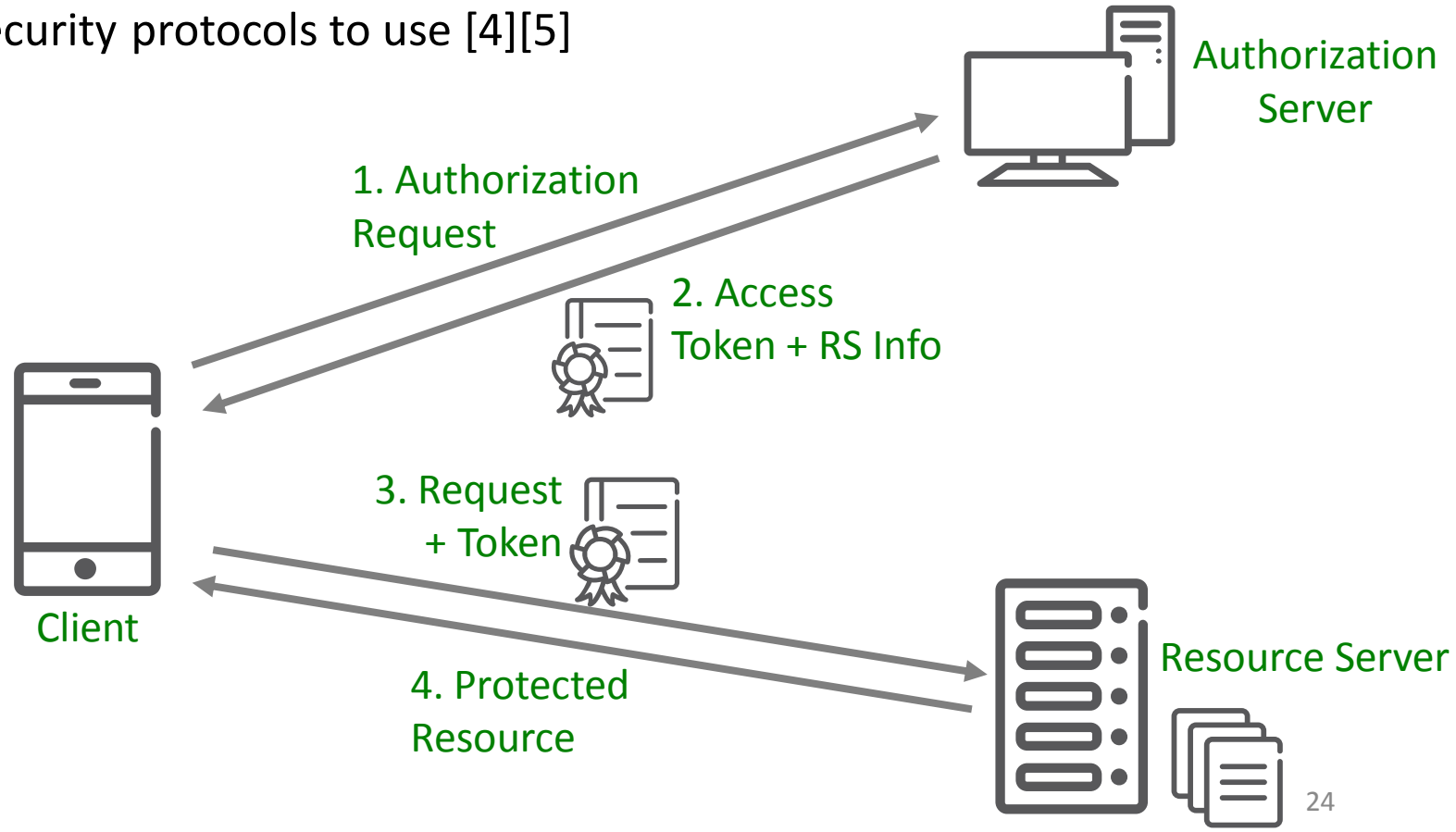


OSCORE Group Joining over the ACE Framework

ACE Framework

Authentication and Authorization in IoT environments [3]

- Builds on OAuth 2.0 (RFC 6749)
- Profiles specify communication/security protocols to use [4][5]



Goals

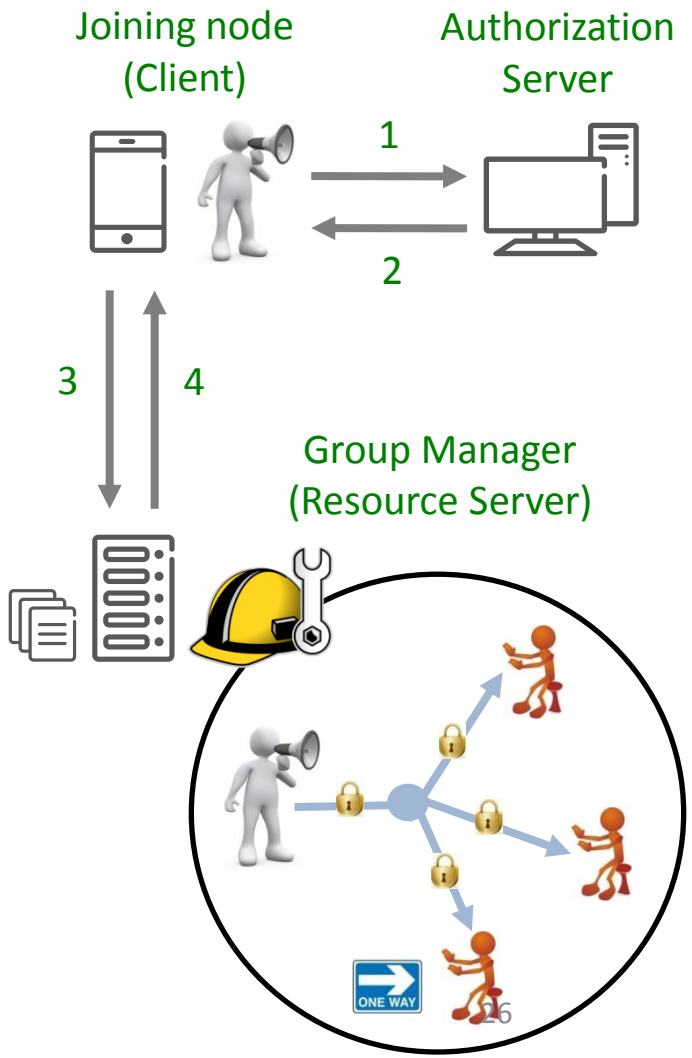
- Join OSCORE groups through a Group Manager
  - Pair group joining with enforcement of access control
  - Use the ACE framework [3] and its profiles [4][5] for this specific scenario
  - Keep the approach as such oblivious to the specific underlying profile

- Focus on
  1. Authorizing a node to join the group according to enforced policies
  2. Securing channel establishment with the Group Manager
  3. Providing joining nodes with Security Context and public keys of group members

- Out of scope and covered in other documents
  - Secure communication in the group through Group OSCORE
  - Authorization to access resources at group members, possibly trough ACE

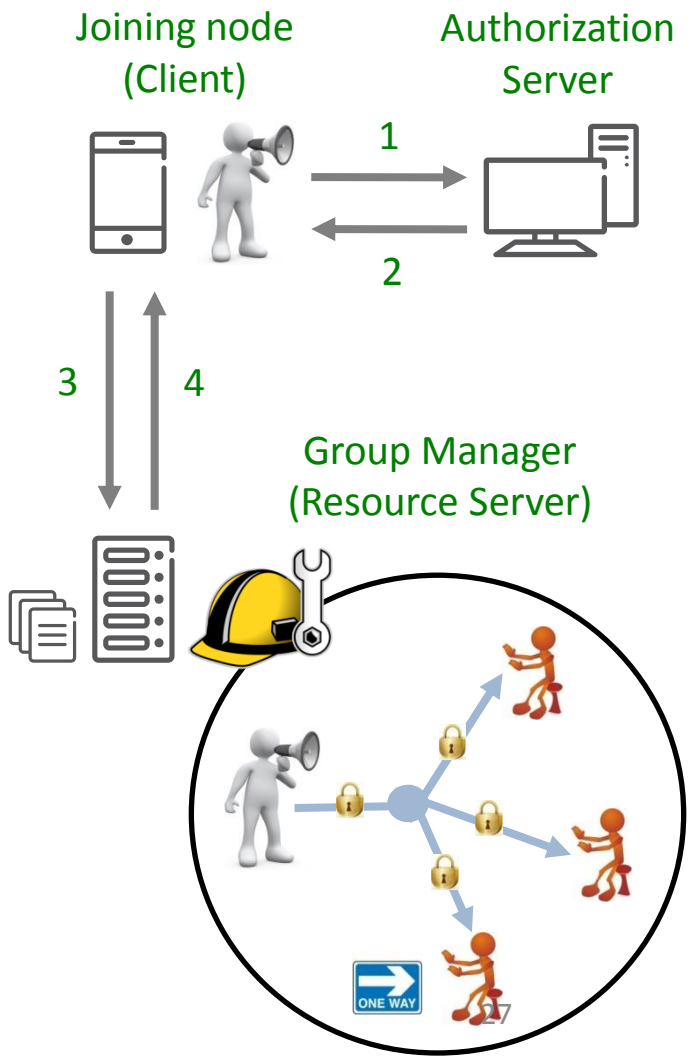
Protocol overview

- Join OSCORE groups over the ACE framework [3]
  - Joining node → Client
  - Group Manager → Resource Server (one *join resource* per group)
  - The AS enforces join policies on behalf of the Group Manager
- Use protocol-specific profiles of ACE
  - DTLS profile [4]
  - OSCORE profile [5]
- CoAP is used as application-layer protocol
- Full protocol description in [6]
  - Referred in Appendix D.3 of Group OSCORE [2]



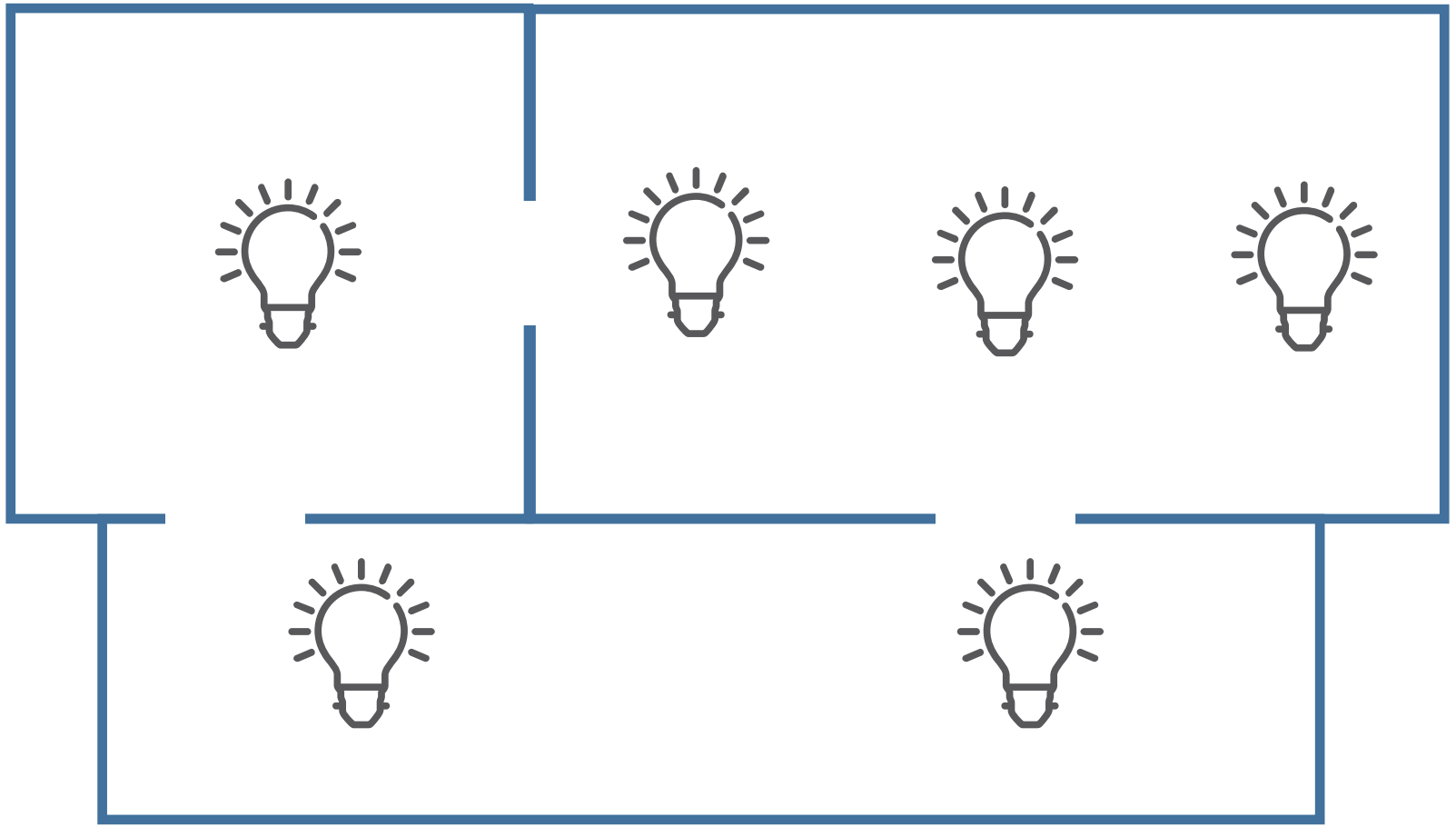
Joining process (Steps 3-4)

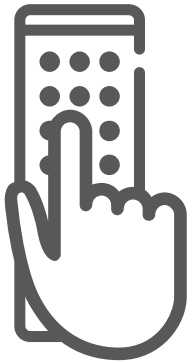
- One separate POST request for each group to join
  - Specify the intended role(s) in the group
  - “Requester” (client); “Listener” (server); “Pure Listener” (silent server)
- The GM admits the joining node to the group
  - Provides the Endpoint ID
  - Provides the OSCORE Common Security Context
- If storing member’s public keys (recommended), the GM:
  - Receives the joining node’s public key, if not received at earlier joins
  - Provides the public keys of the current members, if requested to



Lighting scenario

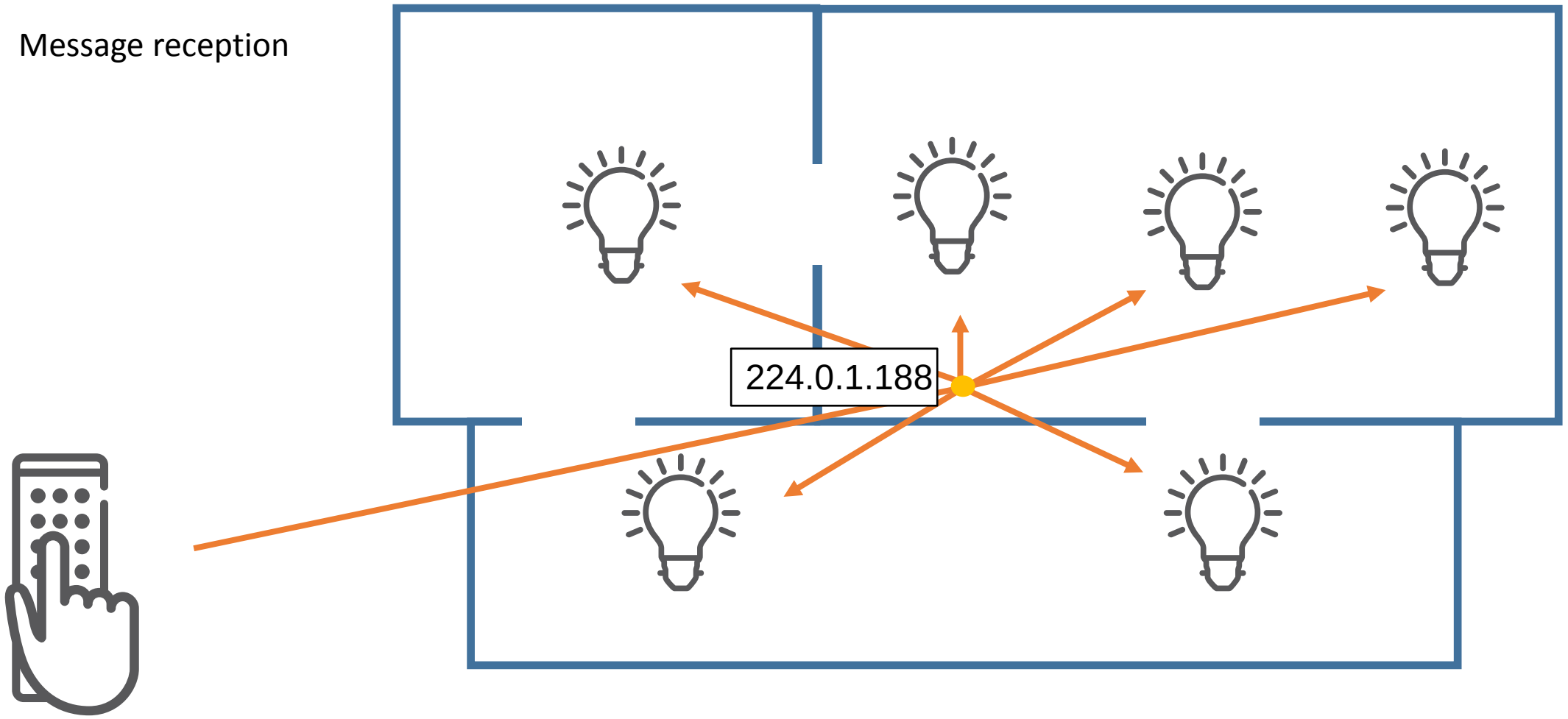
Example scenario





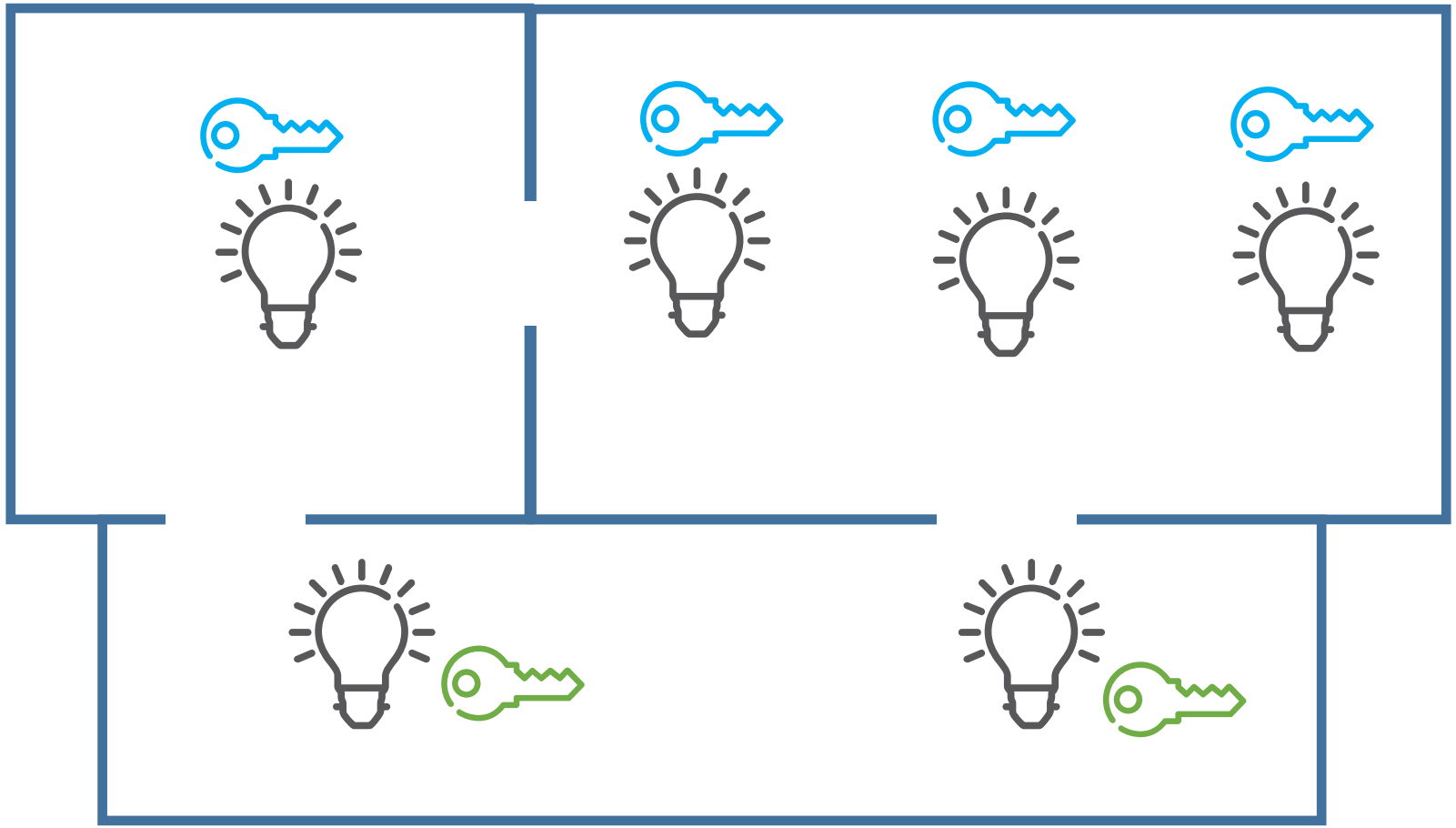
“Multicast group”

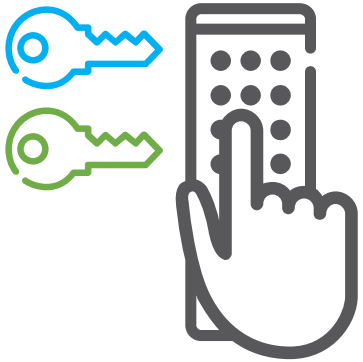
Message reception



“Security group”

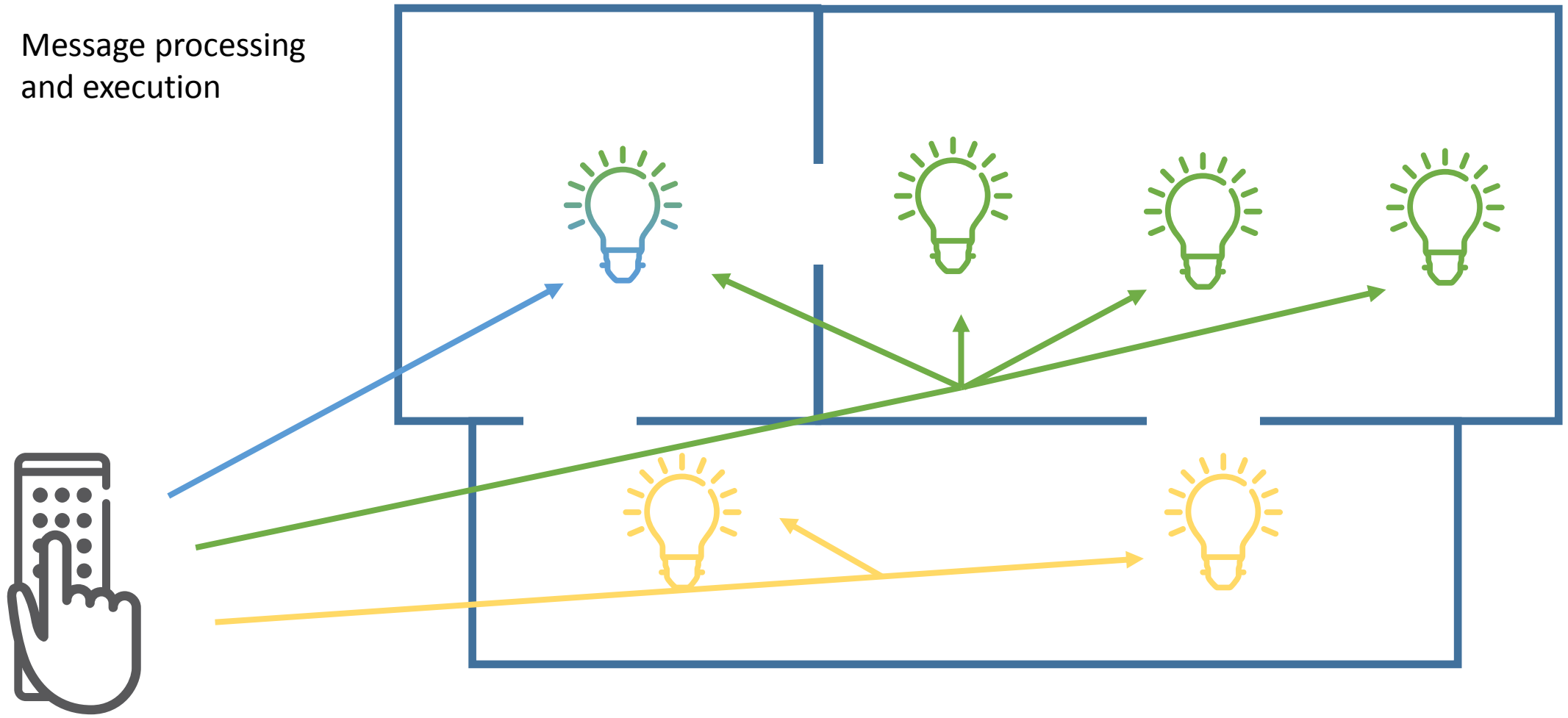
Message
decryption/verification





“Application group”

Message processing
and execution



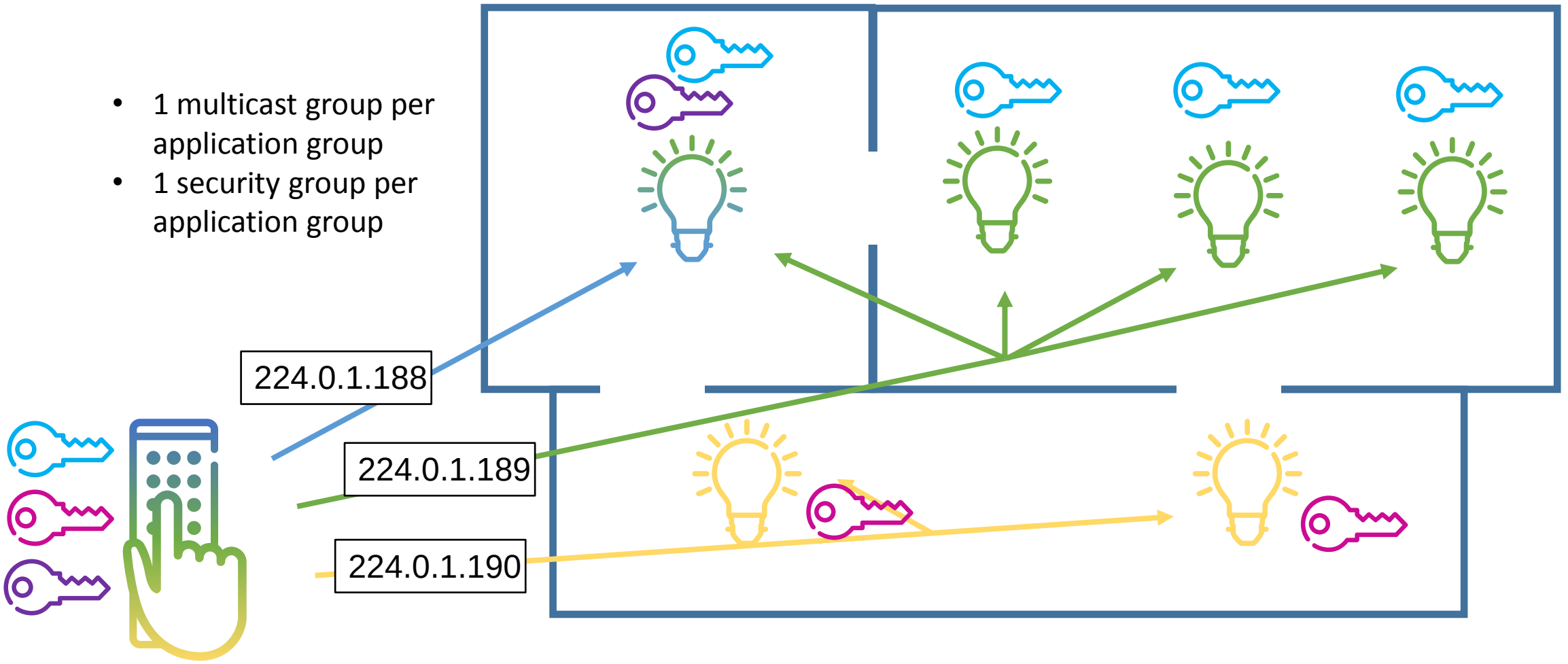
Requirements

- Only authorized members of a specific application group must be able to read and process messages

This can be solved in several different ways:

Solution 1: multicast group coincides with application group and security group

- 1 multicast group per application group
- 1 security group per application group



Requirements

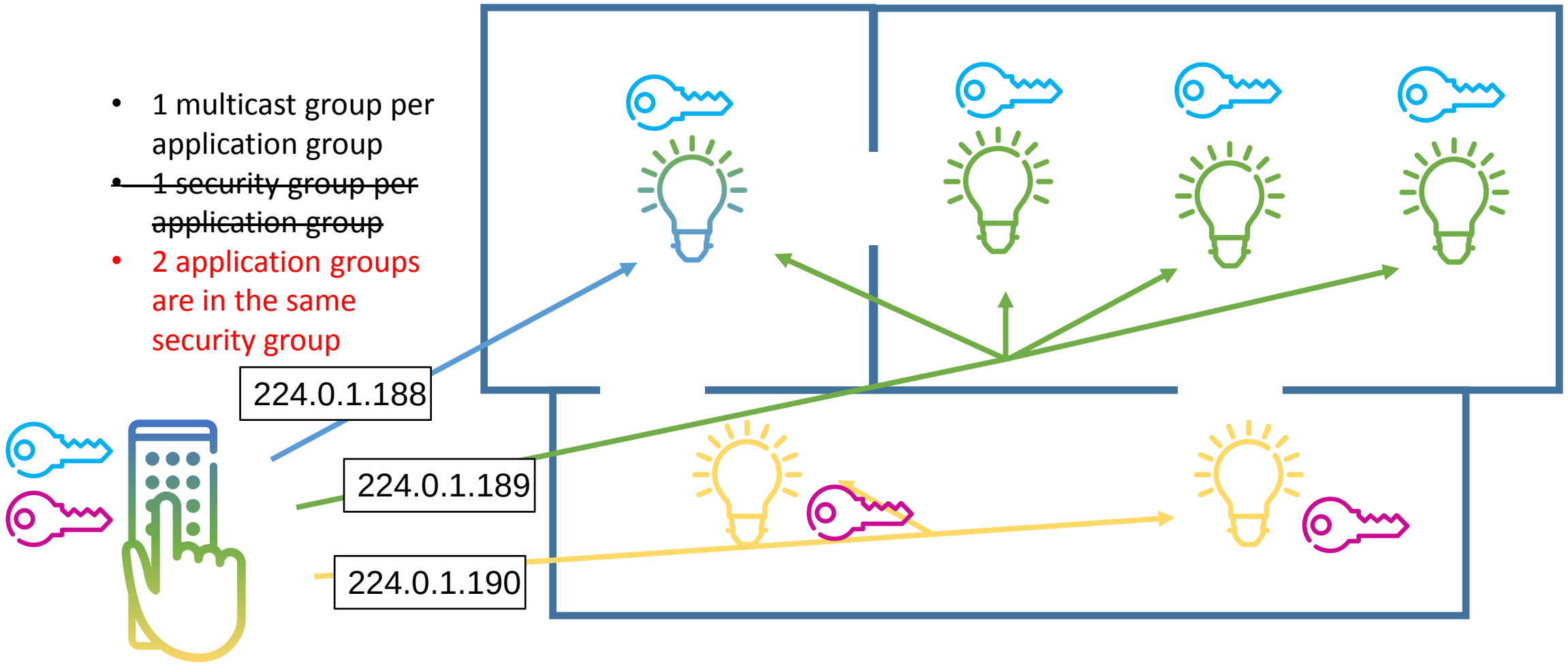
- Only members of a specific application group must be able to read and process messages

This is allowed by the Solution 1 but…

Maybe this is too heavy: lot of security material in the client sending requests over Multicast IP

Solution 2: using fewer security groups

- 1 multicast group per application group
- ~~1 security group per application group~~
- 2 application groups are in the same security group



Requirements

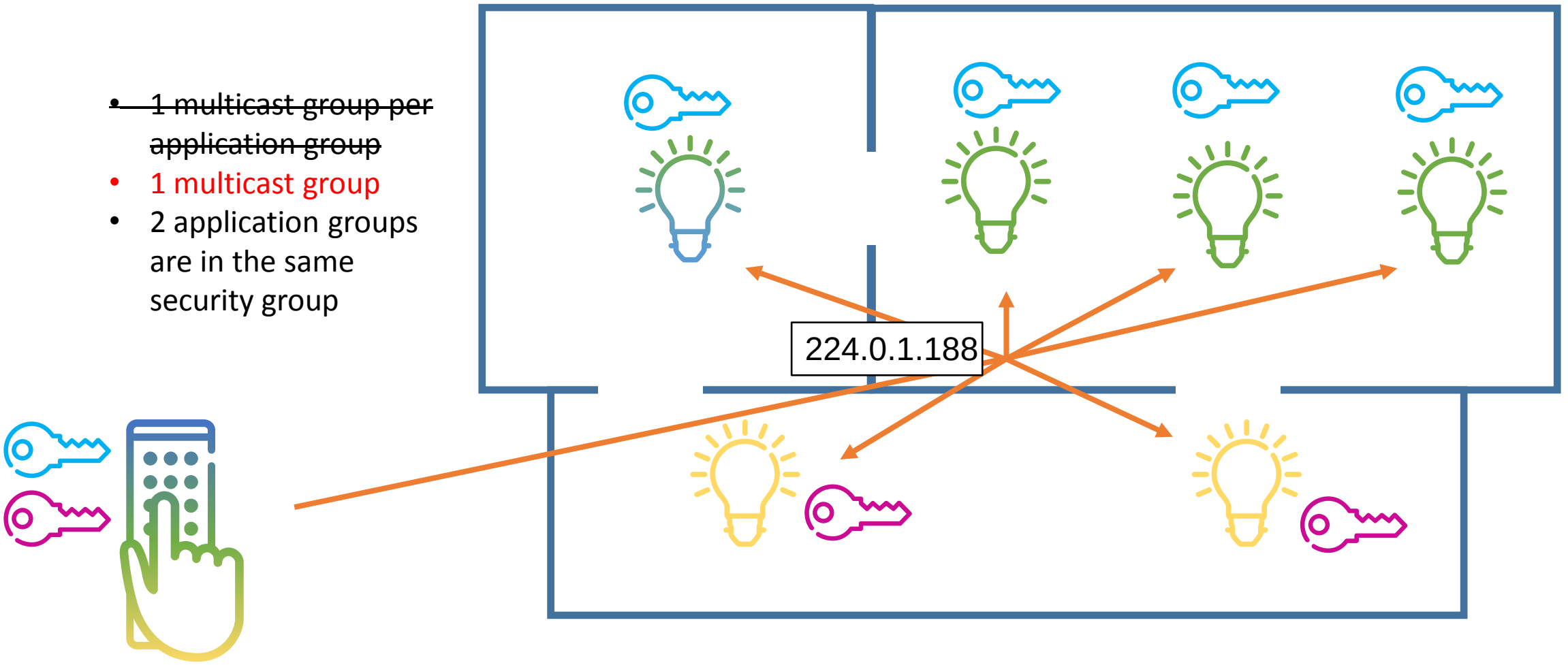
- Only members of a specific application group must be able to read and process messages

This is solved by the Solution 2 but…

Maybe this is still too heavy: several multicast addresses need to be set up

Solution 3: using fewer multicast groups

- ~~1 multicast group per application group~~
- 1 multicast group
- 2 application groups are in the same security group



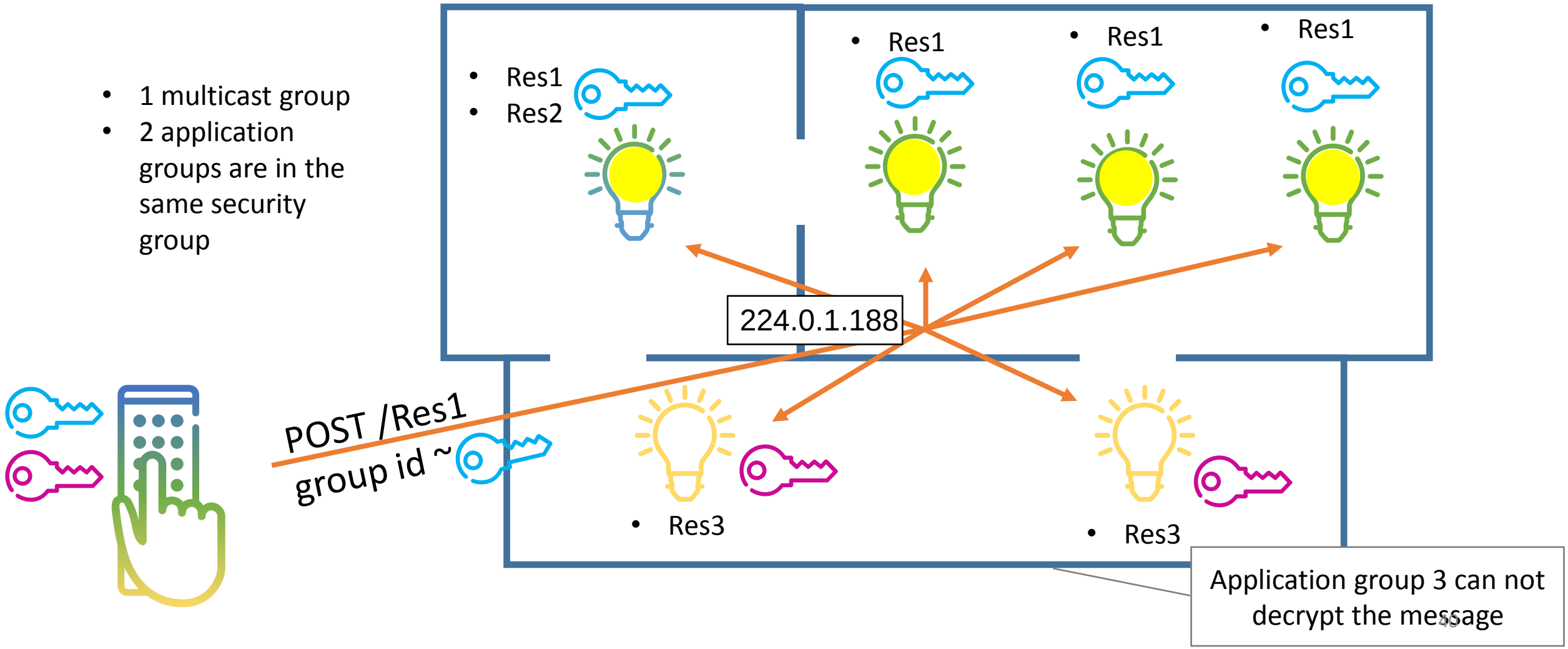
How it works with OSCORE

Every node in the multicast group receives the message but:

- OSCORE sends information about what security group to use (i.e. other members cannot decrypt the message)
- Application layer decides which application group it is (for example: hitting a specific resource, or using a specific method)

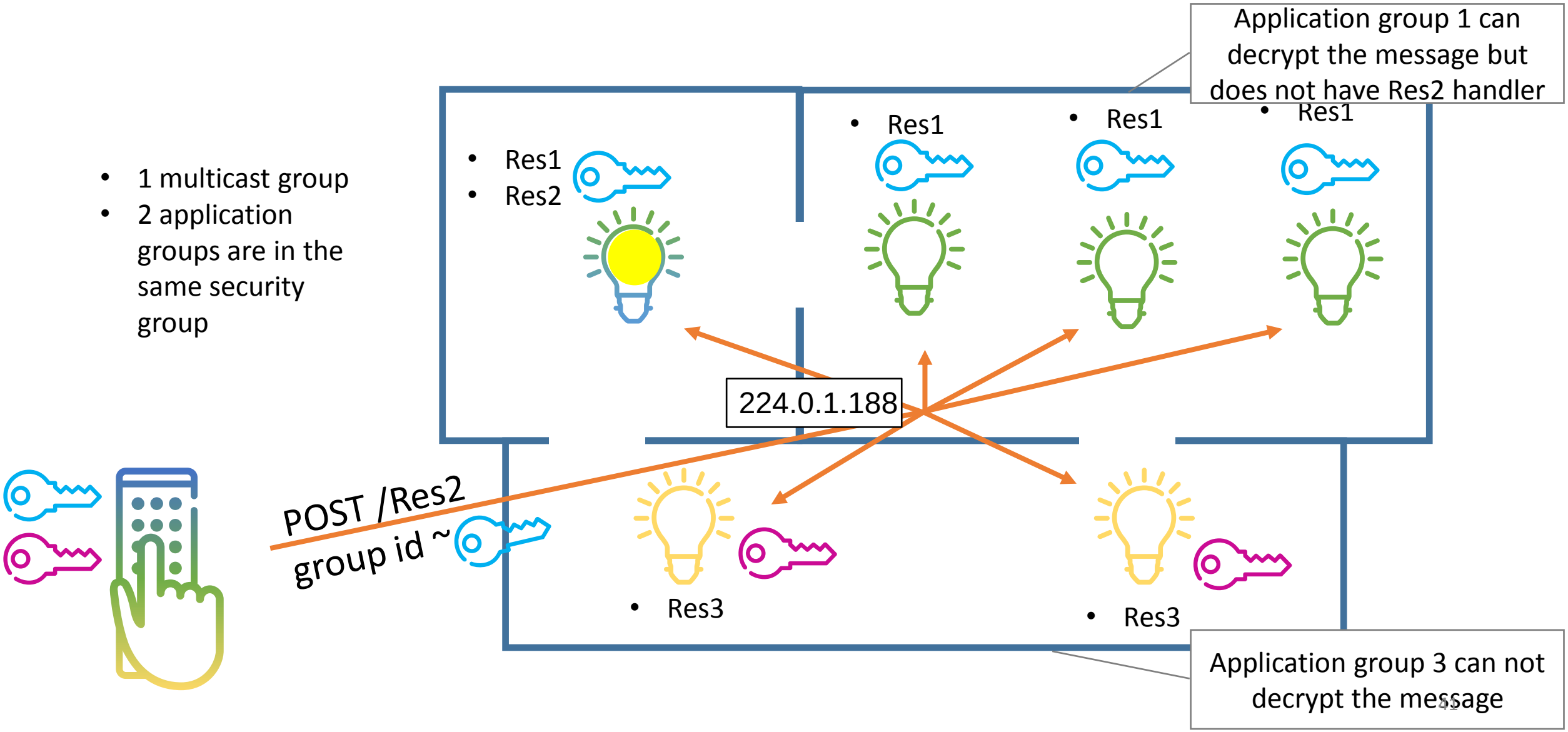
Solutions 3: using fewer multicast groups

- 1 multicast group
- 2 application groups are in the same security group



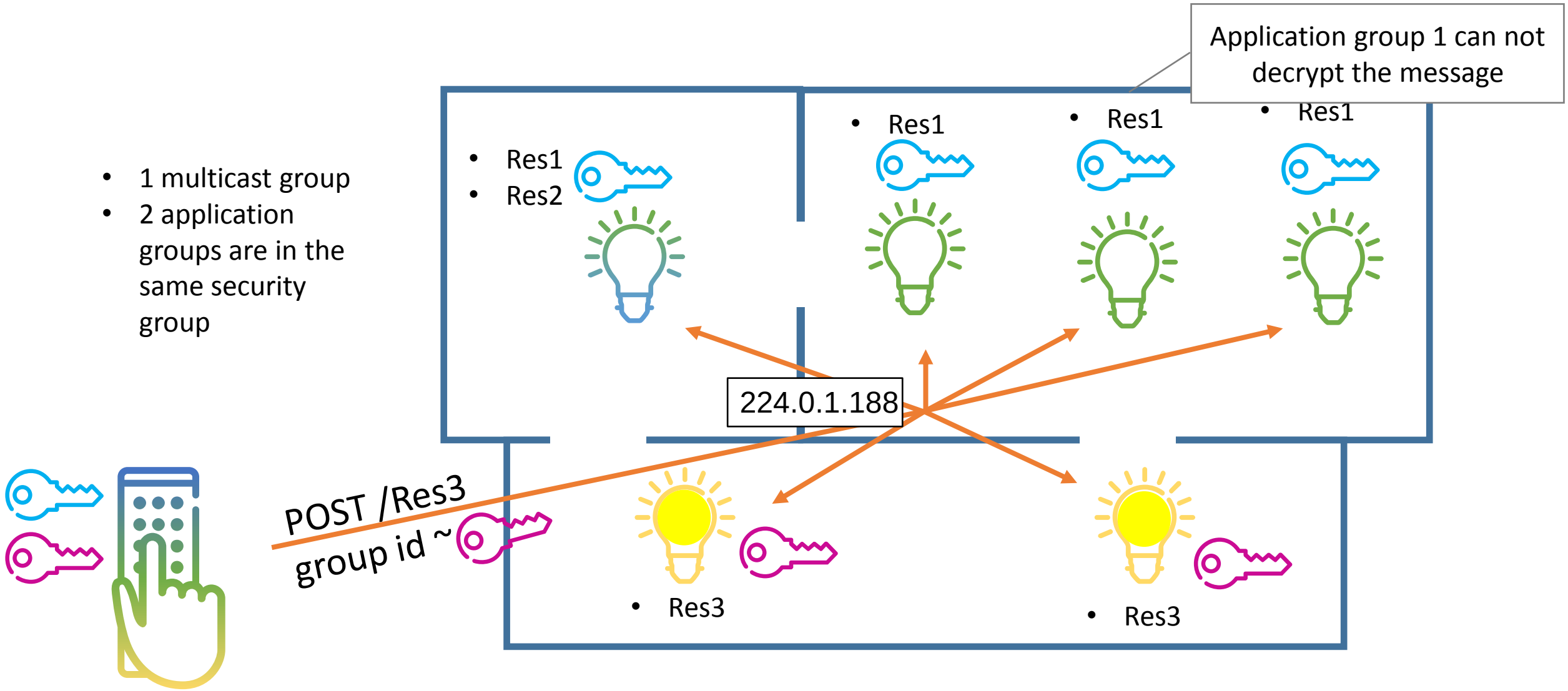
Solutions 3: using fewer multicast groups

- 1 multicast group
- 2 application groups are in the same security group



Solutions 3: using fewer multicast groups

- 1 multicast group
- 2 application groups are in the same security group



Remarks

- Given the application groups, the system designer can decide on multicast and security groups

  - Many multicast groups → better network performance but has to allocate more IP addresses.

    Fewer multicast groups → worse network performance, but less allocated addresses.

  - 1 security group per single application group → better processing/handling performance, but also more memory storage/handling of security contexts.

    1 security group for multiple application groups → worse processing/handling performance, but also less memory storage/handling of security contexts.

- Group OSCORE is flexible and works with all these different scenarios

Standardization status

- Group OSCORE
  - Adopted document in the CoRE WG
  - Implementation version of the draft expected soon
  - OSCORE is done (no more changes, waiting for formal approval)

- Group joining over ACE
  - It got a number of reviews and is eligible for WG adoption
  - Aligned with the message formats for key provisioning in ACE [7]
  - Adoption of new ACE documents only after current ones are in WGLC

- The two areas do not interfere with each other
  - The work in the CoRE WG will proceed in parallel with the work(s) in the ACE WG

Thank you!

References

[1] OSCORE --- *draft-ietf-core-object-security-15*

[2] Group OSCORE --- *draft-ietf-core-oscore-groupcomm-02*

[3] ACE Framework --- *draft-ietf-ace-oauth-authz-14*

[4] DTLS profile of ACE --- *draft-ietf-ace-dtls-authorize-04*

[5] OSCORE profile of ACE --- *draft-ietf-ace-oscore-profile-01*

[6] Group joining over ACE --- *draft-tiloca-ace-oscoap-joining-04*

[7] Key groupcomm --- *draft-palombini-ace-key-groupcomm-01*

Backup slides

Group Manager (GM)

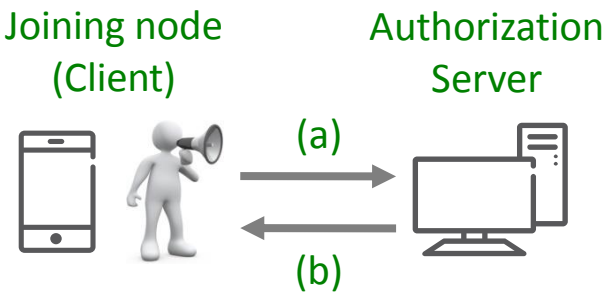
- Can be responsible of multiple groups
  - Drives the joining of new group members
  - Drives the renewal of group keying material

- Drive the joining process
  - Contact point for joining the group
  - Provides keying material to joining nodes (incl. Security Context)

- Recommended as key repository
  - Stores and provides the public keys of the group members

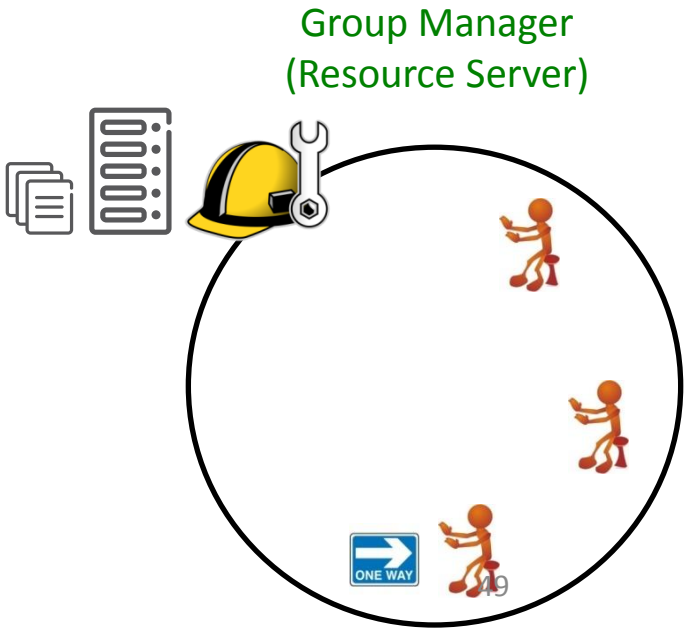
Protocol steps (1/3)

1. Joining node to Authorization Server
   - (a) Request an Access Token to access a join resource on the GM
   - (b) Provide also information to start a secure channel with the GM
   - Possibly update previously released Access Tokens



Access Token specified in the used profile of ACE.

The AS is not necessarily expected to release Access Tokens for any other purpose than accessing join resources on registered Group Managers.

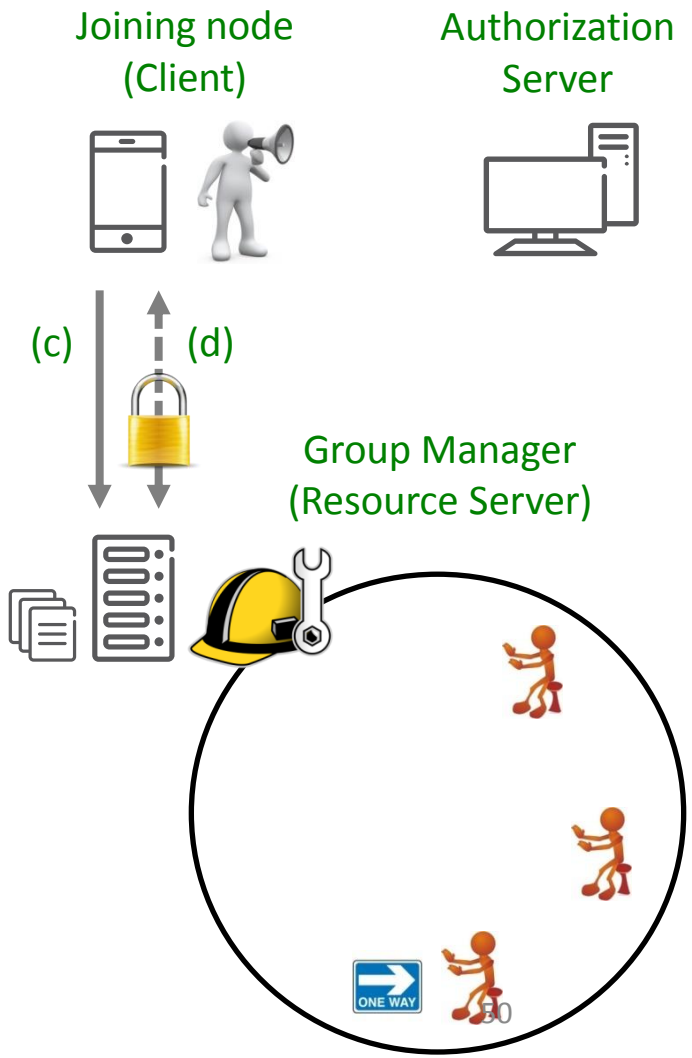


The AS may be configured also to release Access Tokens for accessing resources at members of multicast groups.

Protocol steps (2/3)

2. Joining node to Group Manager
   - (c) Transfer the Access Token
   - (d) Open a secure channel, if not already established

Access Token and secure channel establishment are specified in the used profile of ACE.



Protocol steps (3/3)

2. Joining node to Group Manager
   - (c) Transfer the Access Token
   - (d) Open a secure channel, if not already established

Access Token and secure channel establishment are specified in the used profile of ACE.

3. Joining node to Group Manager
   - Perform the joining process
   - (e) Access the join resource at the Group Manager
   - (f) Receive keying material and parameters to join the group

4. The new node is fully operative in the group