Observe Notifications as CoAP Multicast Responses

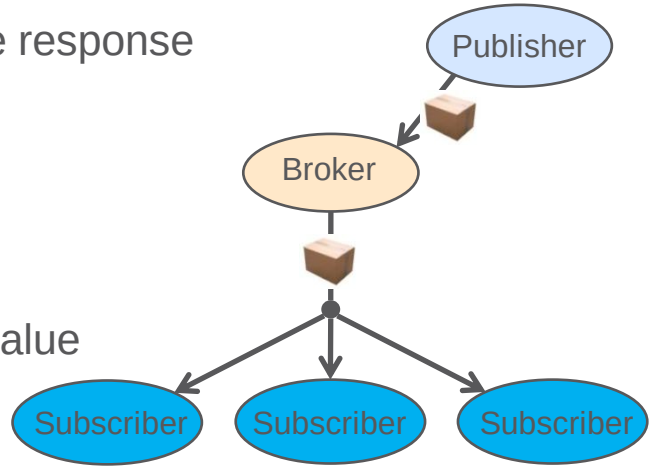
draft-ietf-core-observe-multicast-notifications-02

Marco Tiloca, RISE
Rikard Höglund, RISE
Christian Amsüss
Francesca Palombini, Ericsson

CoRE WG interim meeting, October 27th, 2021

Recap

- Observe notifications as multicast responses
  - Many clients observe the same resource on a server (e.g., pub-sub)
  - Improved performance due to multicast delivery
  - Clients configured by the server, with a 5.03 informative response

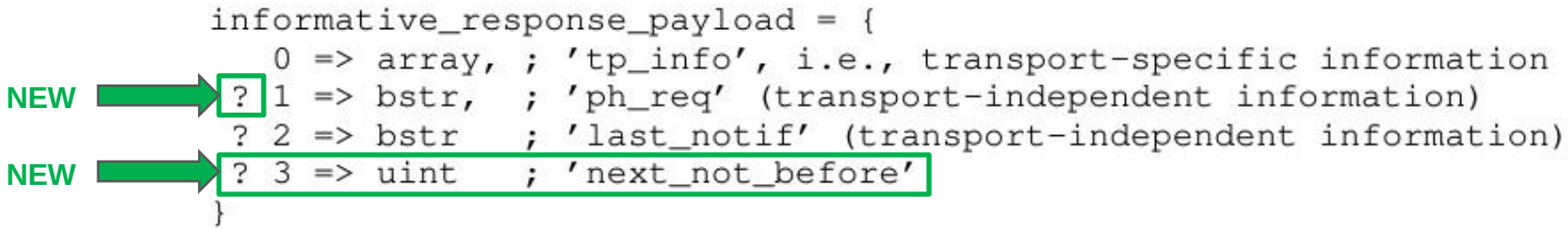


- Token space managed by the server
  - The Token space belongs to the group (clients)
  - The group entrusts the management to the server
  - All clients in a group observation use the same Token value

- Group OSCORE to protect multicast notifications
  - The server aligns all clients of an observation on a same *external_aad*
  - All notifications for a resource are protected with that *external_aad*

Updates since IETF 110

› New payload format for the informative response



› '*ph_req*' – Serialization of the phantom request – Now optional to include

  › May be omitted if the phantom request is "transport-independent-equivalent" to the client's request

  › This is most likely the case when Group OSCORE is not used for end-to-end security

  › This is almost never the case when Group OSCORE is used for end-to-end security

    › Unless the phantom request is a Deterministic Request, see Appendix D

› '*next_not_before*' – Minimum amount of time before the next multicast notification

  – Synch clients as they come, before starting a content transfer; enable a new client to "catch up"

Updates since IETF 110

- Simplified cancellation of the group observation
  - Not using a “phantom cancellation request” anymore
  - The server just cancels the observation and sends a multicast 5.03 error response

- More optimizations
  - Appendix C – OSCORE group self-managed by the server
  - The server can rekey the clients with a protected a 5.03 error response
  - Removed information elements that do not change during the observation’s lifetime

- Secure association between client and proxy
  - Relevant if e2e security is used, to protect the additional Ticket Request to the proxy
  - Possible with (D)TLS or nested OSCORE (see *draft-tiloca-core-oscore-capable-proxies*)

Updates since IETF 110

- Security considerations when Group OSCORE is not used for e2e security
  - Not possible to authenticate clients, but also …
  - … not possible to intersperse CON and NON notifications (as per RFC7641)
  - The rough counting of active clients can still provide acknowledgments to the server
  - The server should do a rough counting at least after every X multicast notifications

- Clarifications and editorial fixes
  - Group-OSCORE-related parameters aligned with other CoRE/ACE documents
  - Revised Appendix C: OSCORE group self-managed by the server
  - Revised Appendix D: phantom request as Deterministic Request
  - Revised all the examples w/ and w/o end-to-end security, w/ and w/o a proxy

Updates since IETF 110

- New Appendix F
  - Example with Group OSCORE + proxy + phantom request as Deterministic Request
  - Each client sends the same request as the actual phantom request
  - Pro1: no need for clients to later send a separate Ticket Request to the proxy
  - Pro2: no need to include the phantom request in the informative responses
  - The server recognizes **byte-by-byte** the protected phantom request from the clients
    - No Group OSCORE decryption is performed on it!
    - The informative response can be and is unprotected, hence the advantages above
      - See also the entry "Q: When should my CoAP server send an unprotected Response to an OSCORE-protected Request?" of https://github.com/core-wg/wiki/wiki/CoAP-FAQ

Summary

- Latest updates
  - Extensions and optimizations for the payload of the informative response
  - Simplified cancellation of group observations
  - Extended security considerations
  - Revised Appendices C and D; Group-OSCORE-related parameters; and examples
  - New Appendix G: example with Group OSCORE + proxy + Deterministic Request
- Next steps
  - Encode addressing information using CRIs – see *draft-ietf-core-href*
  - Add discussed examples with a reverse-proxy
- Need for reviews – Previously promised: Göran, Esko, Jaime, Carsten, Thomas

Thank you!

Comments/questions?

https://github.com/core-wg/observe-multicast-notifications

Backup

Phantom request and error response

- The server requests the observation on its own, e.g. when:
  1. A first traditional registration request comes from a first client; or
  2. Some threshold is crossed – clients can be shifted to a group observation

- Consensus on Token & external_aad , by using a phantom observation request
  - Generated inside the server, it does not hit the wire
  - Like if sent by the group, from the multicast IP address of the group
  - Multicast notifications are responses to this phantom request

- The server sends to clients a 5.03 ***error response*** with:
  - Transport-specific information, e.g. the IP multicast address where notifications are sent to
  - The serialization of the phantom observation request (optional)
  - The serialization of the latest multicast notification (optional)
  - Minimum amount of time after which the next multicast notification will be sent (optional)

Server side

1. Build a GET phantom request; Observe option set to 0

2. Choose a value T, from the Token space for messages …
   - … coming from the multicast IP address and addressed to target resource

3. Process the phantom request
   - As coming from the group and its IP multicast address
   - As addressed to the target resource

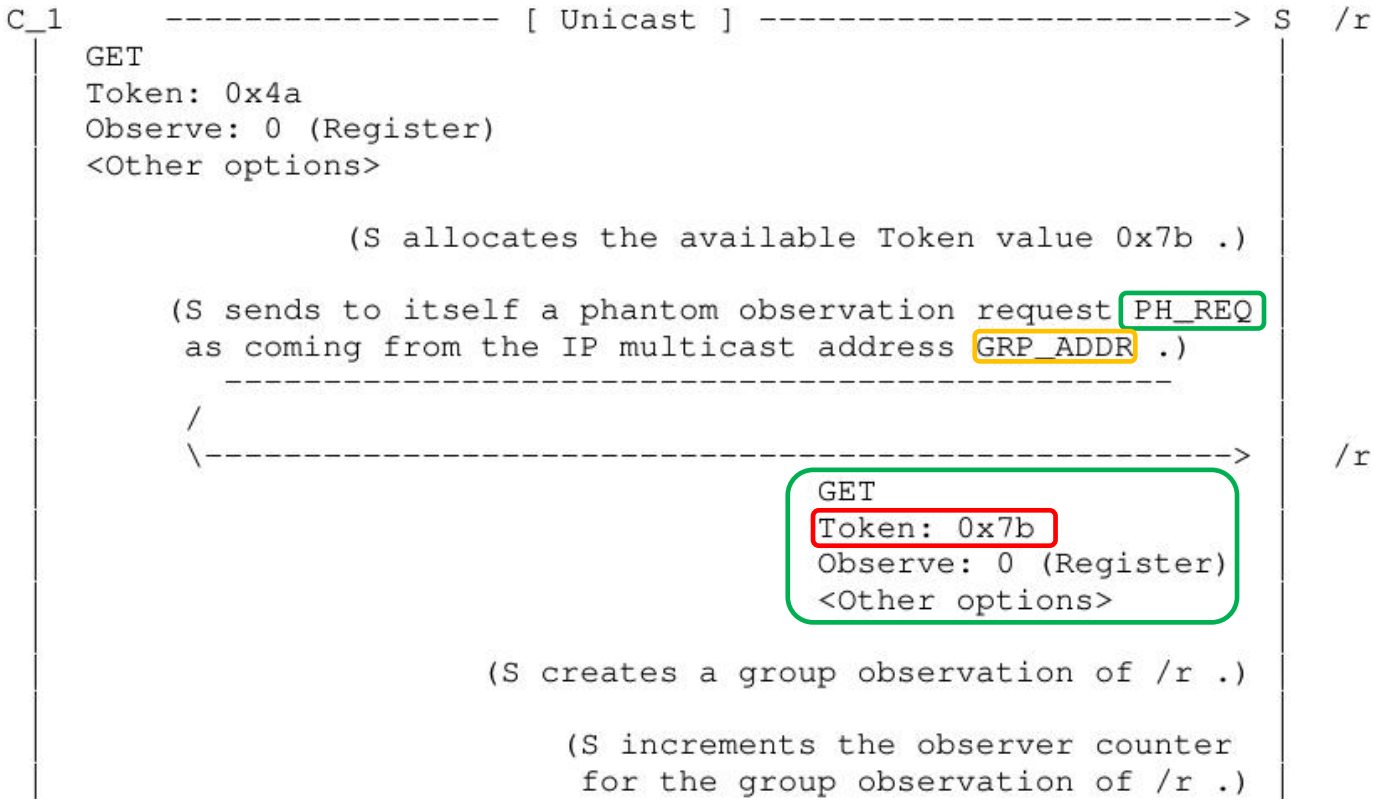
4. Hereafter, use T as token value for the group observation

5. Store the phantom request, store (not send) reply for '*last_notif*'

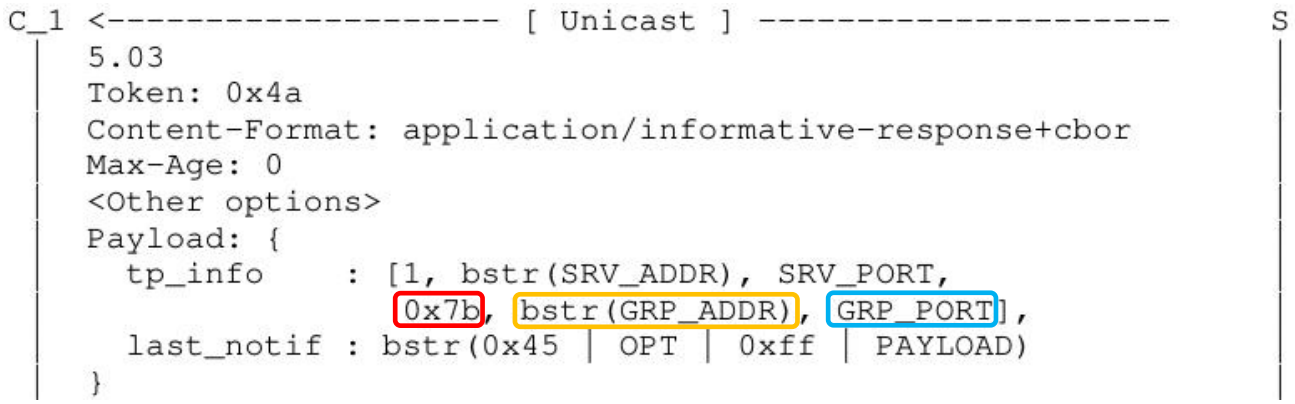
Interaction with clients

- The server sends to new/shifted clients an ***error response*** with
  - '*tp_info*': transport-specific information
    - '*srv_addr*' and '*srv_port*': destination address/port of the phantom request
    - '*token*': the selected Token value T, used for '*ph_req*' and '*last_notif*'
    - '*cli_addr*' and '*cli_port*': source address/port of the phantom request
  - '*ph_req*': serialization of the phantom request
  - '*last_notif*': serialization of the latest sent notification for the target resource
  - '*next_not_before*': minimum amount of time after which the next multicast notification will be sent
- When the value of the target resource changes:
  - The server sends an Observe notification to the IP multicast address '*cli_addr*'
  - The notification has the Token value T of the phantom request
- When getting the error response, a client:
  - Configures an observation for an endpoint associated to the multicast IP address
  - Accepts observe notifications with Token value T, sent to that multicast IP address

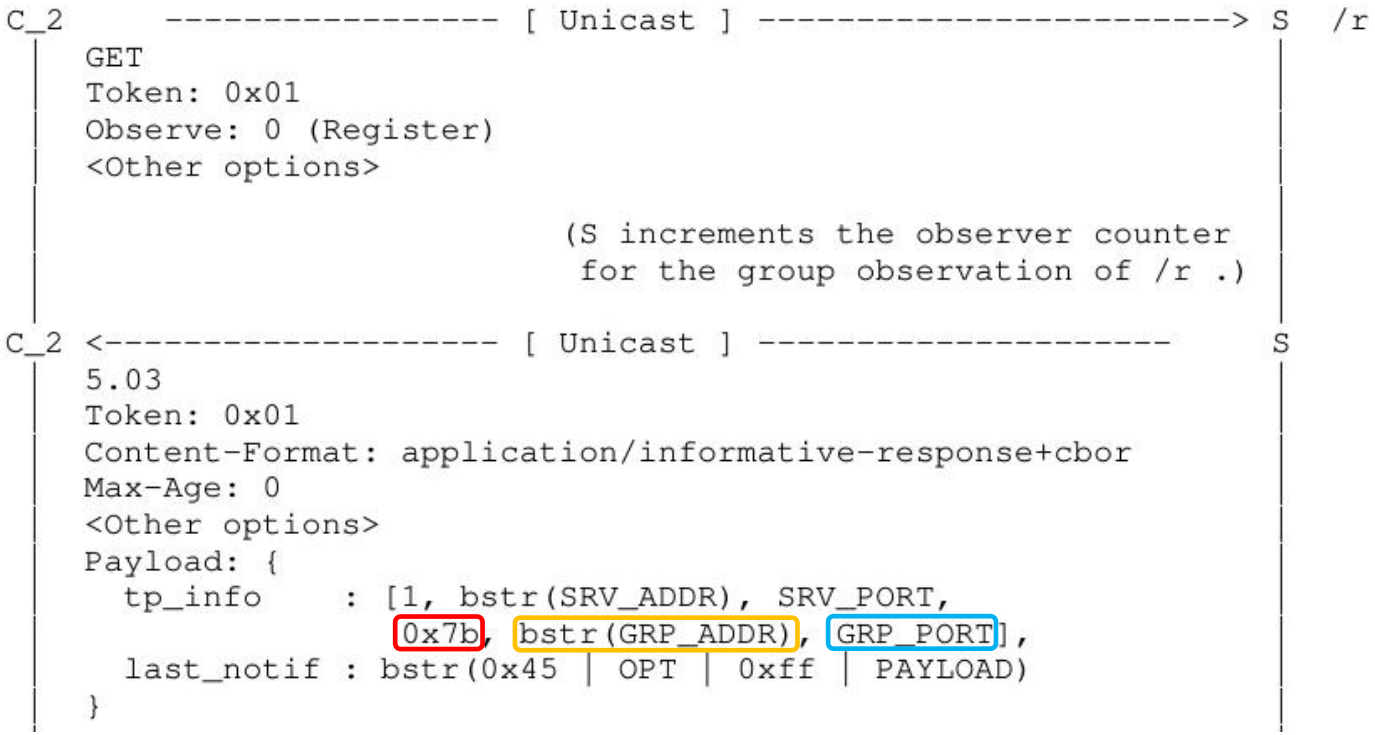
C1 registration



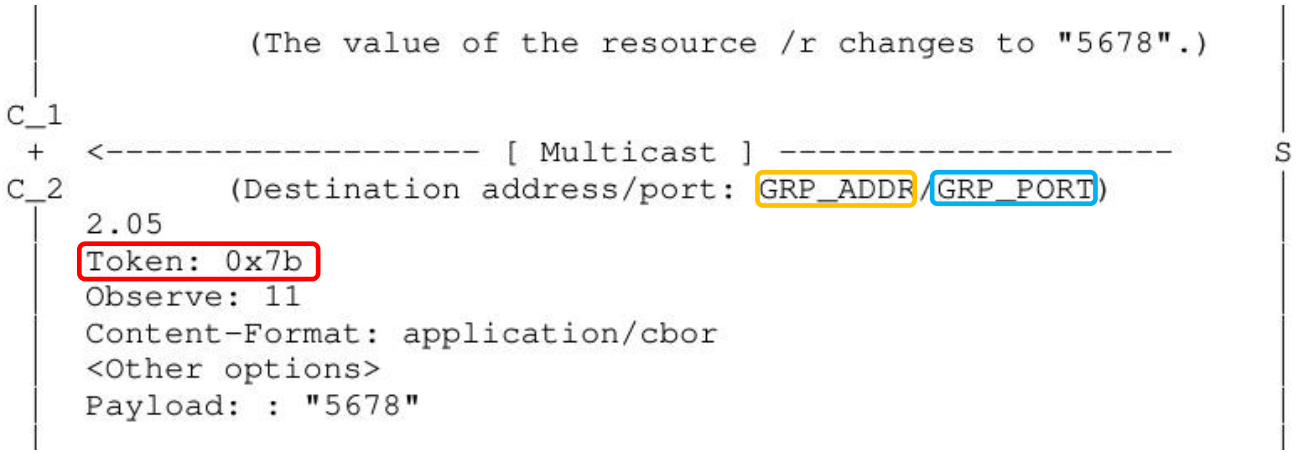
C1 registration



C2 registration



Multicast notification



- Same Token value of the Phantom Request
- Enforce binding between
  - Every multicast notification for the target resource
  - The (group) observation that each client takes part in

Security with Group OSCORE

- The phantom request is protected with Group OSCORE
  - ***x*** : the Sender ID ('kid') of the Server in the OSCORE group
  - ***y*** : the current SN value ('piv') used by the Server in the OSCORE group
  - Note: the Server consumes the value ***y*** and does not reuse it as SN in the group

- To secure/verify all multicast notifications, the OSCORE *external_aad* is built with:
  - 'req_kid' = ***x***
  - 'req_piv' = ***y***

- The phantom request is still included in the informative response
  - Each client retrieves ***x*** and ***y*** from the OSCORE option

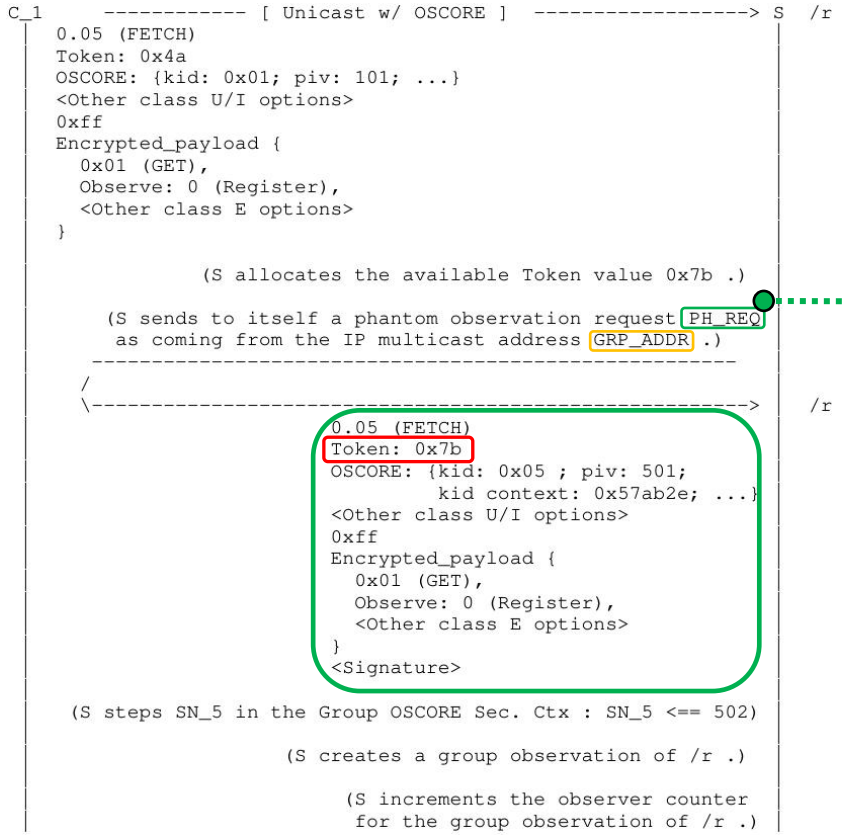
Security with Group OSCORE

- In the error response, the server can ***optionally*** specify also:
  - '*join_uri*' : link to the Group Manager to join the OSCORE group
  - '*sec_gp*' : name of the OSCORE group
  - '*as_uri*' : link to the ACE Authorization Server associated to the Group Manager
  - '*hkdf*' : HKDF algorithm
  - '*pub_key_enc*' : encoding used in the OSCORE group for the public keys
  - '*sign_enc_alg*' : AEAD algorithm
  - '*sign_alg*' : signature algorithm
  - '*sign_params*' : parameters of the signature algorithm and signing key

MUST

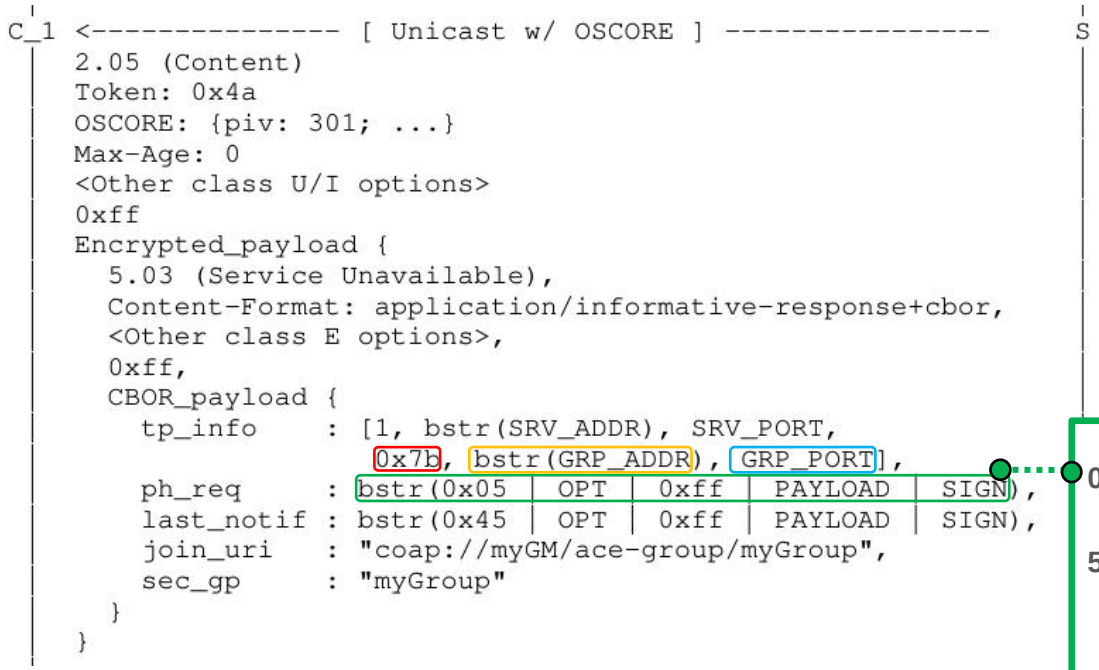
MAY

C1 registration w/ security



The server protects the Phantom Request with Group OSCORE, using its Sender Context, as if it was the sender.

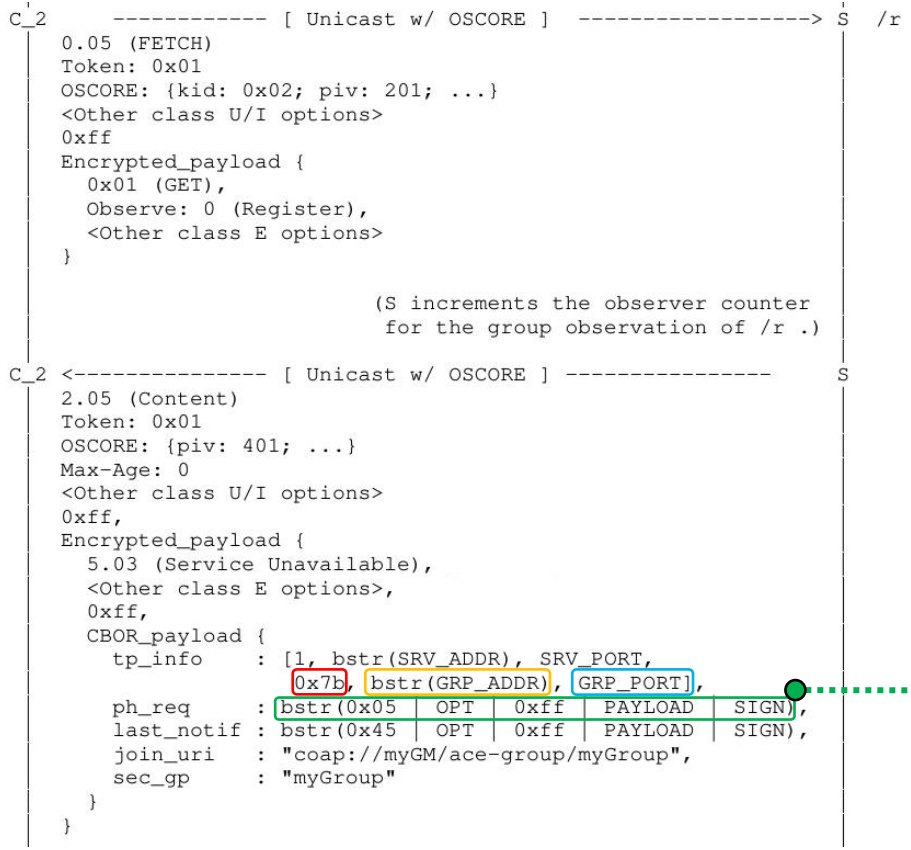
C1 registration w/ security



0x05: Sender ID ('kid') of S in the OSCORE group

501: Sequence Number of S in the OSCORE group when S created the group observation

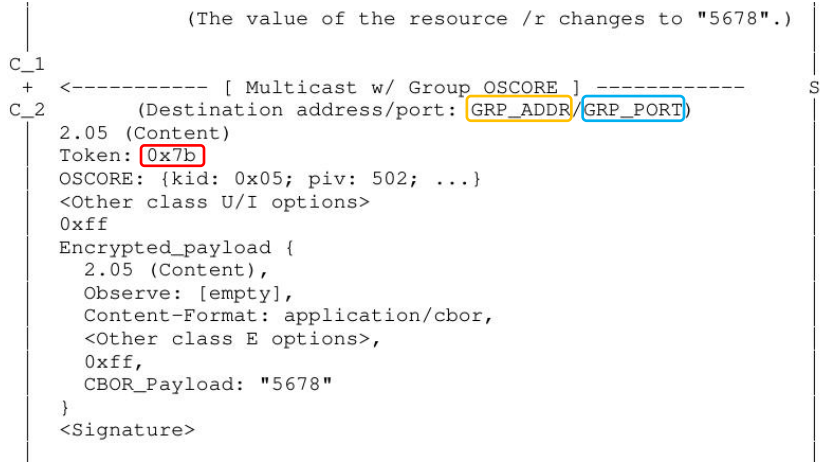
C2 registration w/ security



0x05: Sender ID ('kid') of S in the OSCORE group

501: Sequence Number of S in the OSCORE group when S created the group observation

Multicast notification w/ security



- When encrypting and signing the multicast notification:
  - The OSCORE *external_aad* has 'req_kid' = **0x05** and 'req_iv' = **501**
  - Same for all following notifications for the same resource
- Enforce secure binding between
  - Every multicast notification for the target resource
  - The (group) observation that each client takes part in

Support for intermediary proxies

- How it works
  - The proxy (next to the server) directly listens to the IP multicast address
  - The original Token of the phantom request has to match at the proxy
  - The proxy forwards multicast notifications back to each client
    - The proxy uses the Token values offered by the clients
- Without end-to-end security (Section 9)
  - The proxy can retrieve the phantom request from the informative response
  - No need to forward the informative response back to the clients
- With end-to-end security (Section 10)
  - The informative response is also protected with OSCORE or Group OSCORE
  - The proxy **cannot** retrieve the phantom request from the informative response
  - Each client has to explicitly provide the phantom request to the proxy
  - Exception: the phantom request is a Deterministic Request (see *core-cachable-oscore*)