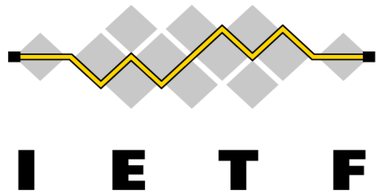
IETF 110

Constrained RESTful Environments WG (core)

Chairs:
Marco Tiloca <marco.tiloca@ri.se>
Jaime Jiménez <jaime.jimenez@ericsson.com>

Mailing list: core@ietf.org

Jabber: core@jabber.ietf.org



Stand-in Chair for IETF 110: Carsten Bormann Thank you!

Jaime is (again) busy working on the supply of future Working Group Chairs

- We assume people have read the drafts
- Meetings serve to advance difficult issues by making good use of face-to-face communications
- Note Well: Be aware of the IPR principles, according to RFC 8179 and its updates

- Blue sheets – Automatic
- Jabber Scribe(s)
- Note Taker(s)

Note Well

Any submission to the IETF intended by the Contributor for publication as all or part of an IETF Internet-Draft or RFC and any statement made within the context of an IETF activity is considered an "IETF Contribution". Such statements include oral statements in IETF sessions, as well as written and electronic communications made at any time or place, which are addressed to:

- **The IETF plenary session**
- **The IESG, or any member thereof on behalf of the IESG**
- **Any IETF mailing list, including the IETF list itself, any working group or design team list, or any other list functioning under IETF auspices**
- **Any IETF working group or portion thereof**
- **Any Birds of a Feather (BOF) session**
- **The IAB or any member thereof on behalf of the IAB**
- **The RFC Editor or the Internet-Drafts function**

All IETF Contributions are subject to the rules of RFC 5378 and RFC 8179.

Statements made outside of an IETF session, mailing list or other function, that are clearly not intended to be input to an IETF activity, group or function, are not IETF Contributions in the context of this notice. Please consult RFC 5378 and RFC 8179 for details.

A participant in any IETF activity is deemed to accept all IETF rules of process, as documented in Best Current Practices RFCs and IESG Statements.

A participant in any IETF activity acknowledges that written, audio and video records of meetings may be made and may be available to the public.

https://www.ietf.org/about/note-well/

I E T F

Practicalities

- Use the queue request on Meetecho
- Use of queuing at core@jabber.ietf.org
  - mic: to ask for relaying a question
- This meeting is recorded
- Bluesheets are automatically filled

Area Director: hand-off of the baton

- Lisa Dusseault
  (chartered us)
- Peter Saint-Andre
  (from 2010)
- Barry Leiba
  (from 2012)
- Alexey Melnikov
  (from 2016)
- Barry Leiba
  (from 2020)
- Francesca Palombini
  (from 2021)






All times are in UTC

Monday (120 min)

- 16:00–16:10 Intro, Agenda, Document status
- 16:10–17:10 Group communication
- 17:10–17:20 Cacheable OSCORE
- 17:20–17:30 OSCORE with EDHOC
- 17:30–17:50 AEAD limits in OSCORE
- 17:50–18:00 Flextime

All times are in UTC

Friday (120 min)

- 12:00–12:05 Intro, Agenda
- 12:05–12:20 CORECONF Comi
- 12:20–12:30 SenML Data Versions
- 12:30–12:40 SenML Data CT
- 12:40–12:55 New Block
- 12:55–13:10 Dynlink
- 13:10–13:25 RD Extensions and protocol negotiations
- 13:25–13:35 Data Centric deployment for CoAP
- 13:35–14:00 Flextime

Agenda Bashing

Document status

RFC 8974

- Extended Tokens and Stateless Clients in the Constrained Application Protocol (CoAP)

- Was draft-ietf-core-stateless

RFC Queue / IESG Processing

- draft-ietf-core-dev-urn-11
  - In RFC Editor Queue
- draft-ietf-core-resource-directory-28
  - In Approved-Announcement Sent
- draft-ietf-core-echo-request-tag-12
  - In Approved-Announcement to be Sent::Revised ID Needed

IESG processing

- draft-ietf-core-yang-cbor-15
  - In Last Call, until 2021-03-17
  - Requested reviews from YANG Doctors

- draft-ietf-core-sid-15
  - In Last Call, until 2021-03-17
  - Requested reviews from YANG Doctors

In Post-WGLC processing

- draft-ietf-core-yang-library-03
  - Waiting for Shepherd Write-Up
- draft-ietf-core-comi-11
  - Waiting for Shepherd Write-Up
- draft-ietf-core-oscore-groupcomm-11
  - Processed WGLC comments and one more review
  - Ongoing work on some open points
- draft-ietf-core-senml-data-ct-03
  - Processed WGLC comments
  - Synch with draft-bormann-core-media-content-type-format
- draft-ietf-core-new-block-07
  - Processed WGLC comments

Group communication

Cacheable OSCORE

OSCORE with EDHOC

AEAD limits in OSCORE

Flextime

Thank you!
Comments/questions?

I E T F

Group Communication for the Constrained Application Protocol (CoAP)

draft-ietf-core-groupcomm-bis-03

Esko Dijk, IoTconsultancy.nl
Chonggang Wang, InterDigital
Marco Tiloca, RISE

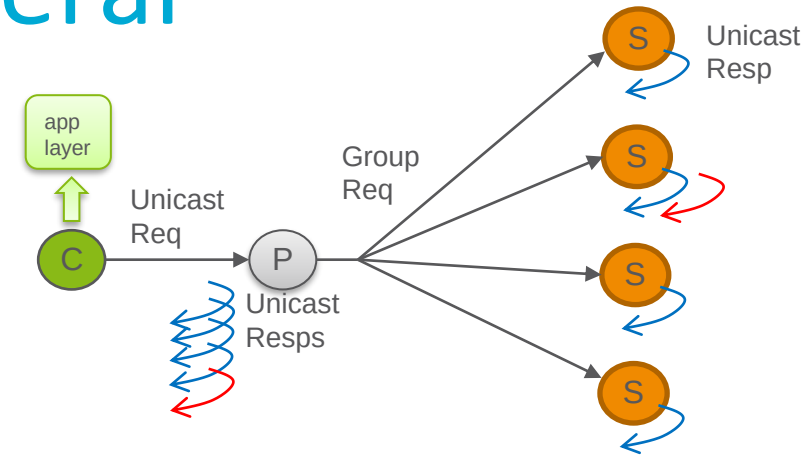
IETF 110, CoRE WG, March 8th, 2021

Goal

- Intended normative successor of experimental RFC 7390 (if approved)
  - As a Standards Track document
  - Obsoletes RFC 7390; Updates RFC 7252 and RFC 7641
- Be standard reference for implementations that are now based on RFC 7390, e.g.:
  - “Eclipse Californium 2.0.x” (Eclipse Foundation)
  - “Implementation of CoAP Server & Client in Go” (OCF)
- What’s in scope?
  - CoAP group communication over UDP/IP, including latest developments
  - (Observe/Blockwise/Security ...)
  - Caching and re-validation of responses
  - Unsecured CoAP or group-OSCORE-secured communication
  - Principles for secure group configuration
  - Use cases (appendix)

Updates in -03 – General

- Multiple CoAP responses to the same group request from the **same server** ( )
  - Handling moved from the CoAP layer to the application
  - Based on interop experience; the application has more context to decide what to do

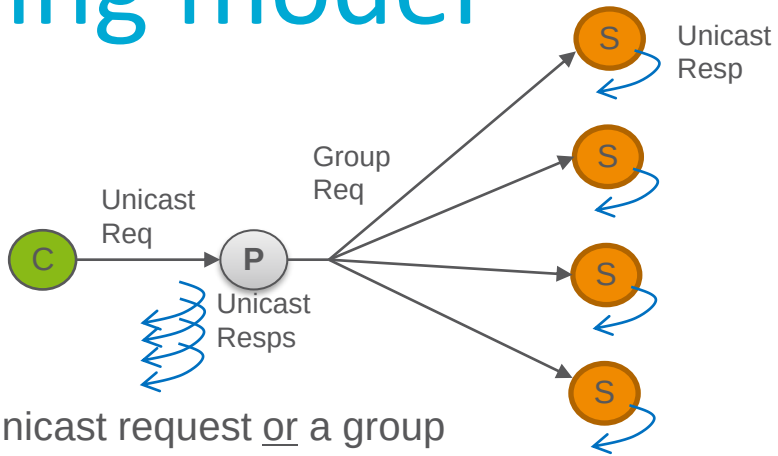


Updates in -03 – General

- Revised guidelines on **forward-proxies** and related issues
- Added guidelines and issues on **reverse-proxies**
  - Stand-in for the whole group of servers, optionally also for each individual server
  - Same and additional issues, compared to a forward-proxy
    - The client may need additional configuration to handle multiple responses
    - The proxy may need additional configuration on the duration of group exchanges
    - The client should get an error, if reusing a Token while a group exchange is still ongoing
- The signaling protocol of *draft-tiloca-core-groupcomm-proxy* is referenced
  - Addresses all known issues with both forward-proxies and reverse-proxies

Updates in -03 – Caching model

- Caching of responses at proxies (P)
  - Two types of cache entries:



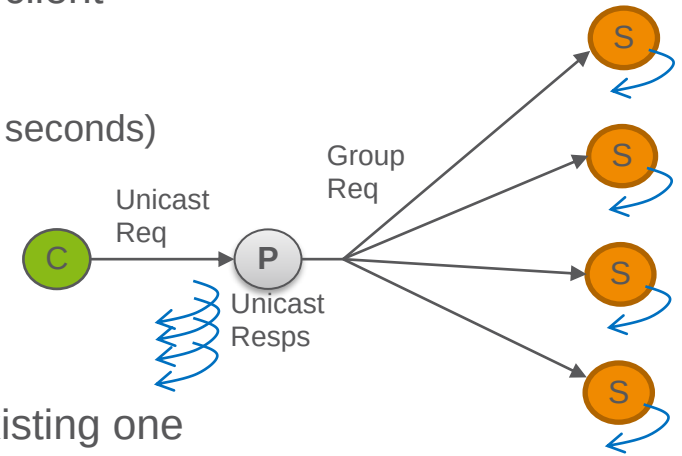
- "**Individual**" cache entry
  - Populated with the response from one server (to a unicast request or a group request)
  - Hit by a matching unicast request intended to that server

- "**Aggregated**" cache entry
  - Populated with all the responses to a group request, from any server in the group
  - Hit by a matching group request intended to all servers

Updates in -03 – Caching model

› As it receives responses to a group request, the proxy:

1. Forwards each response from the origin server S to the client
2. Adds each response to the individual cache entry for S
   – Same lifetime as Max-Age of the response (or default to 60 seconds)
3. Adds the response to a list L



› After forwarding back all the responses, the proxy:

1. Creates an aggregated cache entry, or cleans up the existing one
2. Copies the responses from the list L to the cache entry
3. Set the cache entry lifetime to the smallest Max-Age of the added responses
4. Set the cache entry as active

Updates in -03 – Caching model

› When it receives a response to a unicast request, the proxy:

1. Forwards back the response from the origin server S to the client
2. Creates an Individual cache entry for S, or updates the existing one
   - Same lifetime as Max-Age of the response (or default to 60 seconds)
3. Looks for existing Aggregated cache entries, such that:
   - They would produce a hit, if receiving a group request matching the forwarded unicast request
4. In each found Aggregated cache entry:
   - Store the response, possibly overwriting a currently stored one
   - Set the lifetime of the cache entry to *min*(current entry lifetime, Max-Age of the response)

› Same when the proxy sends requests to the servers, to refresh its cache

Updates in -03 – Validation model

› *Section 8.2.1 of RFC 7252 left this for further study*

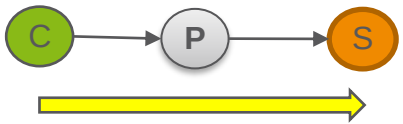
› **Between Client and Servers**

The Multi-ETag Option

No.	C	U	N	R	Name	Format	Length	Default
TBD1				x	Multi-ETag	(*)	any	(none)

C=Critical, U=Unsafe, N=NoCacheKey, R=Repeatable

› New Multi-Etag option
  – Only for group requests
  – One instance per server in group to revalidate against



› Option value: CBOR sequence of 1+M elements
  – First element: addressing information of the server, encoded as in *groupcomm-proxy*
  – The following M elements are entity-tag values, as CBOR byte strings

› A server processes only the Multi-Etag option pertaining to itself, unlike ETag
  – What follows uses ETag, as in RFC 7252

Updates in -03 – Validation model

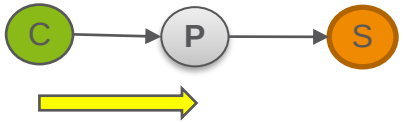
- **Between Client and Proxy**
- New Group-Etag Option
  - Only for Aggregated cache entries
  - For group requests and related responses

No.	C	U	N	R	Name	Format	Length	Default
TBD2				x	Group-ETag	opaque	1-8	(none)

C=Critical, U=Unsafe, N=NoCacheKey, R=Repeatable

The Group-ETag Option

- Option value: an entity-tag value, as CBOR byte string
  - Basically, a version number of the Aggregated cache entry (maintained by the proxy)



- A 2.05 (Content) response may include one Group-ETag Option
- In a GET/FETCH group request
  - One option instance per e-tag value to revalidate against the proxy's Aggregated cache entries
- A 2.03 (Valid) response revalidates all responses in the Aggregated cache entry
  - MUST include one Group-Etag Option indicating the revalidated responses set

Caching/validation with e2e security

- **Caching at a proxy**
  - Possible, by using deterministic requests
  - Limited to (REST) safe requests with no side-effects on resource at the servers
  - See *draft-amsuess-core-cachable-oscore*

- **Response re-validation**
  - Possible between origin client and origin servers, with Multi-ETag options
    - Caveat: different set of Multi-ETag options → Different deterministic request
    - Different deterministic requests → Different cache entries at the proxy
    - Trade off between flexibility for the client and caching efficiency at the proxy
  - Not possible between proxy and origin servers, with Multi-ETag options
  - Not possible between origin client and proxy, with Group-ETag options

Open point – Github issue #11

› **What's the most appropriate place for below new items?**

1. General mechanics & rules on cacheability of responses at proxies
   – Appropriate to be in this document?

2. Validation of individual responses, with the new Multi-ETag option
   – Appropriate to be in this document? Or in a separate dedicated document?

3. Validation of a set of response cached at the proxy, with the new Group-ETag option
   – Appropriate to be in this document? Or instead in *draft-tiloca-core-groupcomm-proxy*?

Next steps

- Address comments from John [1] – Thanks!
  - More reviews would be good! Promised @IETF108: Christian
- Address open point from today (issue #11) and other Github issues [2]
- Test selected functions in CoAP implementations
  - E.g. “Observe + multicast” extension of RFC 7641
  - Report results

[1] https://mailarchive.ietf.org/arch/msg/core/xy3ImeWkbqziBhqs4NCGwNP6R7U/

[2] https://github.com/core-wg/groupcomm-bis/issues

Thank you!

Comments/questions?

https://github.com/core-wg/groupcomm-bis/

Motivation (backup slide)

- RFC 7390 was published in 2014
  - CoAP functionalities available by then were covered
  - No group security solution was available to indicate
  - It is an Experimental document (started as Informational)

- What has changed?
  - More CoAP functionalities have been developed (Block-Wise, Observe)
  - RESTful interface for membership configuration is not really used
  - Group OSCORE provides group end-to-end security for CoAP

- Practical considerations
  - Group OSCORE clearly builds on RFC 7390 normatively
  - However, it can refer RFC 7390 only informationally

Group OSCORE - Secure Group Communication for CoAP

draft-ietf-core-oscore-groupcomm-11

Marco Tiloca, RISE
Göran Selander, Ericsson
Francesca Palombini, Ericsson
John Mattsson, Ericsson
Jiye Park, Universität Duisburg-Essen

IETF 110, CoRE WG, March 8th, 2021

Update since the November meeting

- Version -11 submitted
  - Addressed review of version -10 from Christian [1] – Thanks! Reply at [2]
  - Addressed more points discussed at the IETF 109 meeting

- Two main open points
  - Admitting to recycle Group IDs in the same group (Christian)
  - Security of using one identity key for both signing and Diffie-Hellman (Ben [3][4])

[1] https://mailarchive.ietf.org/arch/msg/core/pXEyxhbf-s2wgGDzrDhUNPsHZZc/
[2] https://mailarchive.ietf.org/arch/msg/core/quxfWG2mZnp--5gP10PAZOfPwbU/
[3] https://mailarchive.ietf.org/arch/msg/core/ujj_I-LlqW9fq__quh-YqKS0fF0/
[4] https://mailarchive.ietf.org/arch/msg/core/YRNXvtiFmHLk5YkXK8-uJg-t3NU/

Updates from -11

- Single format for the *external_aad*
  - For both encrypting and signing operations
  - Removed '*par_countersign_key*'
  - Improved description of last two fields

```
aad_array = [
   oscore_version : uint,
   algorithms : [alg_aead : int / tstr,
                 alg_countersign : int / tstr,
                 par_countersign : [countersign_alg_capab,
                                    countersign_key_type_capab]],
   request_kid : bstr,
   request_piv : bstr,
   options : bstr,
   request_kid_context : bstr,
   OSCORE_option: bstr
]
```

- Today, COSE algorithms have only "Key Type" as capability
  - In general, 0 or 2+ capabilities; that can happen with future algorithms

- New Appendix H, with future-friendly templates
  - For parameters of the Security Context
  - For '*par_countersign*' in the external_aad
  - An instance with today's algorithms produces the formats used in the document body

par_countersign [

countersign_alg_capab [c_1 : any,
c_2 : any,
…,
c_N : any],
countersign_capab_1,
countersign_capab_2,
…
countersign_capab_N

]

Updates from -11

- Usage of '*kid*' in response messages
  - Must be included only if the request was protected in group mode
  - The mode used to protect the response plays no role
- Relaxed rules on recycling Sender IDs in a group
  - Now forbidden only under the same Group ID
- Revised examples of protected messages

Updates from -11

- Additional reason to lose part of the Security Context – Section 2.4.1.2
  - Reached the limit of Recipient Contexts, due to memory availability
  - Delete a current Recipient Context, to make room for a new one

- Hereafter, each new Recipient Context will start with an invalid Replay Window
  - Get rekeyed by the Group Manager; or
  - Run the Echo exchange in Appendix E, achieving also freshness as byproduct

- Overall, improved distinction between anti-replay and freshness
  - Server “synchronization” with a client is related to freshness, and achievable with Echo

Updates from -11

Some “major editorial” changes

- Reorganized Sections 2.4.* , to better stress cause-effect relations
  - Causes: loss of mutable Security Context; exhaustion of Sender Sequence Number
  - Effect: ask the Group Manager for new keying material; reset Sender Sequence Number
- Section 9 – Message processing in pairwise mode
  - Rewritten as delta from OSCORE (RFC 8613), plus few additions from the Group Mode
- Removed old Appendix E.1 and Appendix E.2 as moot
  - Revised Appendix E (was E.3), on the Echo exchange as only synchronization method

Open point – Observations and GIDs

- Text to explicitly add
  - If a group member re-joins the group, it MUST terminate all its ongoing observations

- Recycling of Group IDs in a same group
  - Currently forbidden, to avoid possible issues with long-lasting observations
  - Reminder: observations survive a change of Sender ID and Group ID

- A client C1 starts an observation with (GID1, KID1, PIV1)
  - C1 obtains a new 'kid' = KID2; its observation continues as (GID1, KID1, PIV1)
  - … The group is rekeyed many times … The Gid "wraps" and becomes GID1 again
  - A client C2 with 'kid' = KID1 legitimately starts an observation (GID1, KID1, PIV1)

→ One notification would match and decrypt against two observations /!\

Open point – Recycling Group IDs

- Solution to enable Group ID recycling
  - The Group Manager (GM) retains the Gid that a node obtains upon group joining, i.e. its “birth Gid”
  - Before rekeying the group, the GM checks if the new Gid is any current member’s “birth Gid”

- If such members are found, the GM removes them from the group and rekeys accordingly

- Those evicted nodes will ask the GM for the latest keying material
  - Since they are not group members anymore, they receive error responses
  - Eventually, they will re-join the group, terminating their observations

- If any of those nodes re-joins before another rekeying has happened
  - The Group Manager MUST NOT rekey the group again upon its joining

Recycling Group IDs is safe → A group can live forever – Objections?

Open point – Github issues #72 #73

- Using identity keys for both signing and Diffie-Hellman [3][4]
  - A DH secret is used to generate encryption keys for the pairwise mode
  - Both usages have the same goal and policy: group communication under a Security Context

- As deviating from common best practices, security has to be well proven
  - Ongoing work to prove this secure in Group OSCORE
  - Build on the paper at [5], as focused on (but not limited to) ECIES settings

- The pairwise mode per se is fine! This is actually about the derivation of pairwise keys
  - Problem alternatively solvable by providing and storing separate Diffie-Hellman keys
  - That's a last resort, since it would mean more provisioning and storage overhead

[3] https://mailarchive.ietf.org/arch/msg/core/ujj_I-LlqW9fq__quh-YqKS0fF0/
[4] https://mailarchive.ietf.org/arch/msg/core/YRNXvtiFmHLk5YkXK8-uJg-t3NU/
[5] https://eprint.iacr.org/2011/615.pdf

Next steps

- Address the two open points
  - Recycling of Group IDs in the same group
  - Usage of identity keys for both signing and Diffie-Hellman

- Submit v -12
  - If no further issues arise, it should be ready to move on

Thank you!

Comments/questions?

https://github.com/core-wg/oscore-groupcomm

Discovery of OSCORE Groups with the CoRE Resource Directory

draft-tiloca-core-oscore-discovery-08

Marco Tiloca, RISE
Christian Amsüss
Peter van der Stok

IETF 110, CoRE WG, March 8th, 2021

Recap

- A newly deployed device:
  - May not know the OSCORE groups and their Group Manager (GM)
  - May have to wait GMs to be deployed or OSCORE groups to be created
- Use web links for discovery – Typically through the Resource Directory (RD)
  - Discover an OSCORE group and retrieve information to join it
  - Practically, discover the links to join the OSCORE group at its GM
  - CoAP Observe supports early discovery and changes of group information
- Use resource lookup, to retrieve:
  - The name of the OSCORE group
  - A link to the resource at the GM for joining the group
- Full support for both Link-Format and CoRAL RD

Updates from -08

- Added target attributes related the pairwise mode of Group OSCORE
  - *ecdh_alg* , *ecdh_alg_crv* , *ecdh_key_kty* , *ecdh_key_crv*
  - To refer to for the derivation of pairwise symmetric keys
  - Same advantages as for the attributes on the signature algorithm
- Usage of the right content-format, for links to join OSCORE groups
  - “application/ace-groupcomm+cbor”
  - See Section 8.2 of draft-ietf-ace-key-groupcomm
- Group discovery intended not only to joining nodes
  - Relevant case: signature verifiers, e.g. intermediary gateways
  - They don’t join the OSCORE group, but retrieve public keys from the Group Manager

Updates from -08

- Revised all examples in Link Format and CoRAL

- Proof-of-concept implementation using Link Format [1]
  - The full set of operations from the draft is covered
    - Register an application group
    - Register Group Manager, security group and associated Authorization Server
    - Discover the security group, with descriptive target attributes
    - Discover the associated Authorization Server
    - Discover the application group, with the multicast IP address
  - Successfully tested with Christian's RD at coap://rd.coap.amsuess.com

[1] https://bitbucket.org/marco-tiloca-sics/ace-java/src/master/src/test/java/se/sics/ace/interopGroupOSCORE/CoAPEndpointToResourceDirectory.java

Open points

Mostly on security considerations

1. Denial (common issue)
   – The RD hides the presence of groups
2. Interaction leaking (common issue)
   – An endpoint advertises groups and learn addresses of joining nodes as they come
   – Possibly acting also as MITM between joining nodes and real Group Manager
3. Downgrade attack
   – Data from the RD are a hint here
   – Possible to mitigate, by directly checking also with the Group Manager

… and more, consistent with the latest version -27 of the RD

Next steps

- Update also based on the latest *draft-ietf-core-resource-directory-27*
  - Revised used of the ‘anchor’ attribute
  - Security considerations, covering also the specific open points

- Integrate implementation in the ACE Group Manager and joining node

- Need for more reviews

Thank you!

Comments/questions?

https://gitlab.com/crimson84/draft-tiloca-core-oscore-discovery

Backup

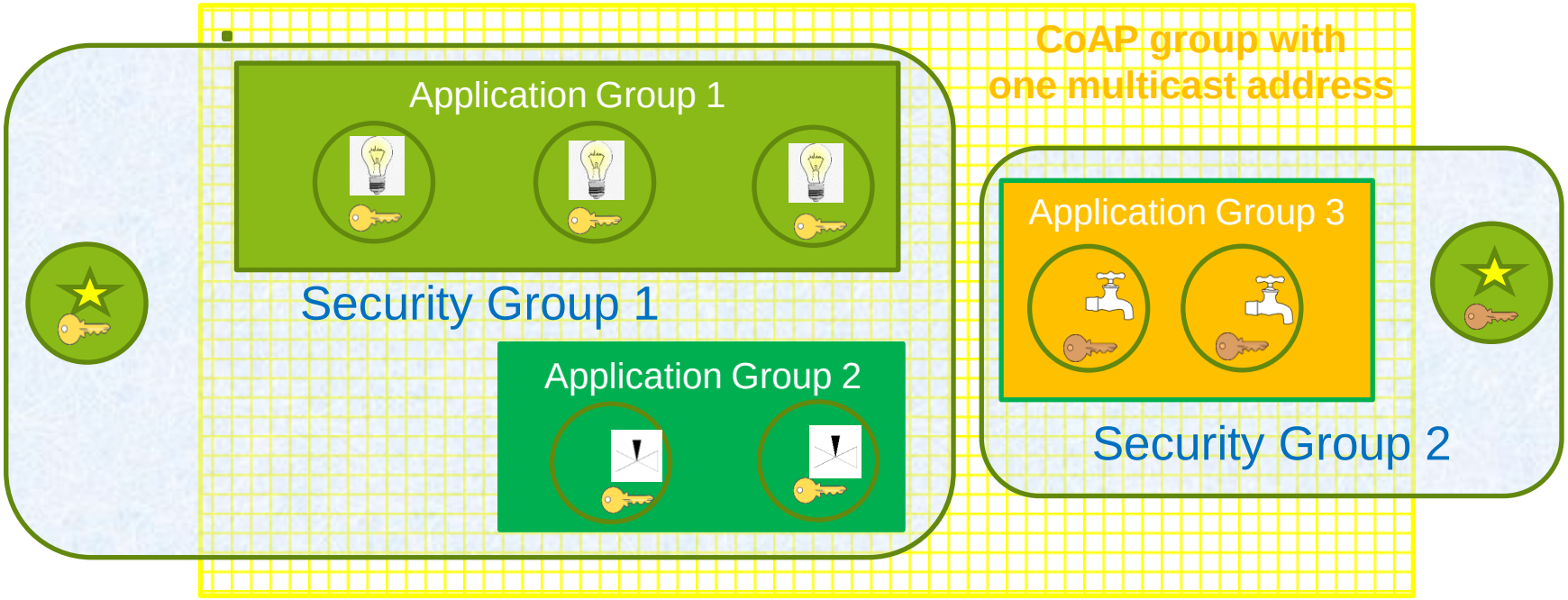
Application/CoAP/Security Groups

- Application group
  - Defined in {RD} and reused as is
  - Set of CoAP endpoints sharing a pool of resources
  - Registered and looked up just as per Appendix A of {RD}

- CoAP Group
  - Defined in *draft-ietf-core-groupcomm-bis*
  - Set of CoAP endpoints listening to the same IP multicast address
  - The IP multicast address is the ‘base’ address of the link to the application group

- (OSCORE) Security Group
  - Set of CoAP endpoints sharing common security material (e.g. OSCORE Ctx)
  - A GM registers the group-membership resources for accessing its groups

Application vs. Security Groups



Client of application group

Different key sets

Resources for given function

Registration

- The GM registers itself with the RD
  - MUST include all its join resources, with their link attributes
  - rt="core.osc.gm" , if="ace.group"

```
Request: GM -> RD

Req: POST coap://rd.example.com/rd?ep=gm1
Content-Format: 40
Payload:
</ace-group/feedca570000>;ct=65000;rt="core.osc.gm";if="ace.group";
                          sec-gp="feedca570000";app-gp="group1";
                          cs_alg="-8";cs_alg_crv="6";
                          cs_kenc="1";ecdh_alg="-27";
                          ecdh_alg_crv="4",
<coap://as.example.com/token>;
      rel="authorization-server";
      anchor="coap://[2001:db8::ab]/ace-group/feedca570000"

Response: RD -> GM
Res: 2.01 Created
Location-Path: /rd/4521
```

Discovery (1/2)

- The device performs a resource lookup at the RD
  - Known information: name of the Application Group, i.e. “group1”
  - Need to know: name of the OSCORE Group; Join resource @ GM; Multicast IP address
  - ‘*app-gp*’ → Name of the Application Group, acting as tie parameter in the RD

```
Request: Joining node -> RD

Req: GET coap://rd.example.com/rd-lookup/res
  ?rt=core.osc.gm&app-gp=group1

Response: RD -> Joining node

Res: 2.05 Content
Payload:
<coap://[2001:db8::ab]/ace-group/feedca570000>;ct=65000;
    rt="core.osc.gm";if="ace.group";sec-gp="feedca570000";
    app-gp="group1";cs_alg="-8";cs_alg_crv="6";
    cs_kenc="1";ecdh_alg="-27";ecdh_alg_crv="4";
    anchor="coap://[2001:db8::ab]"
```

Discovery (2/2)

› The device performs an endpoint lookup at the RD
  – Still need to know the Multicast IP address
  – ‘ep’ // Name of the Application Group, value from ‘*app-gp*’
  – ‘base’ // Multicast IP address used in the Application Group

```
Request: Joining node -> RD

Req: GET coap://rd.example.com/rd-lookup/ep
  ?et=core.rd-group&ep=group1

Response: RD -> Joining node

Res: 2.05 Content
Payload:
</rd/501>;ep="group1";et="core.rd-group";
          base="coap://[ff35:30:2001:db8::23]";rt="core.rd-ep"
```

Alg/key related parameters

- New optional parameters for a registered group-membership resource
  - (*)(**) *cs_alg* : countersignature algorithm, e.g. “EdDSA”
  - (*) *cs_alg_crv* : countersignature curve (if applicable), e.g. “Ed25519”
  - (*) *cs_key_kty* : countersignature key type, e.g. “OKP”
  - (*) *cs_key_crv* : countersignature curve (if applicable), e.g. “Ed25519”
  - (*) *cs_kenc* : encoding of public keys, e.g. “COSE_Key”
  - (*)(**) *ecdh_alg* : ECDH algorithm to derive pairwise keys, e.g. “ECDH-SS + HKDF-256”
  - (*) *ecdh_alg_crv* : ECDH curve, e.g. “X25519”
  - (*) *ecdh_key_kty* : ECDH key type, e.g. “OKP”
  - (*) *ecdh_key_crv* : ECDH curve, e.g. “X25519”
  - (**) *alg* : AEAD algorithm, e.g. “AES-CCM-16-64-128”
  - (**) *hkdf* : HKDF algorithm, e.g. “HKDF SHA-256”

- Benefits for a joining node, when discovering the OSCORE group
  - (*) No need to ask the GM or to have a trial-and-error when joining the group
  - (**) Decide whether to join the group or not, based on the supported algorithms

Observe Notifications as CoAP Multicast Responses

draft-tiloca-core-observe-multicast-notifications-05

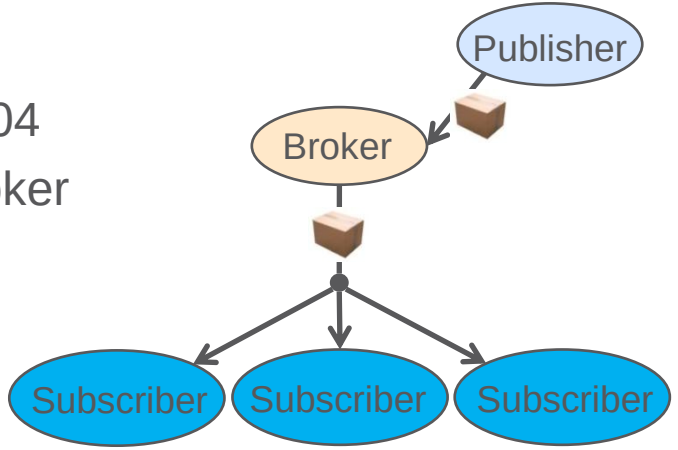
Marco Tiloca, RISE
Rikard Höglund, RISE
Christian Amsüss
Francesca Palombini, Ericsson

IETF 110, CoRE WG, March 08th, 2021

Recap

- Observe notifications as multicast responses
  - Many clients observe the same resource on a server S
  - Improved performance due to multicast delivery
  - Multicast responses are not defined yet --- Token binding, security, …

- Example of relevant use case
  - Pub-Sub scenario, also discussed at IETF 104
  - Many subscribers to a same topic on the Broker
  - Better performance
  - Subscribers can remain clients only



Single notification response over multicast

How

- Define multicast responses, in particular Observe notifications

- 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*

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
  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
  - The serialization of the latest multicast notification (optional)

Updates from -05

› New payload format for the informative response

```
informative_response_payload = {
   1 => array, ; 'tp_info', i.e. transport-specific information
   2 => bstr,  ; 'ph_req' (transport-independent information)
 ? 3 => bstr   ; 'last_notif' (transport-independent information)
}
```

› The same '*tp_info*' content applies to both '*ph_req*' and '*last_notif*'

› '*ph_req*' - Serialization of the phantom request

› '*last_notif*' - Serialization of latest sent multicast notification

– Now only optional to include

Updates from -05

› New payload format for the informative response

```
informative_response_payload = {
    1 => array, ; 'tp_info', i.e. transport-specific information
    2 => bstr,  ; 'ph_req' (transport-independent information)
  ? 3 => bstr   ; 'last_notif' (transport-independent information)
}
```

```
tp_info = [
    srv_addr  ; Addressing information of the server
  ? req_info  ; Request data extension
]

srv_addr = (
    tp_id : int,  ; Identifier of the used transport protocol
  + elements      ; Number, format and encoding
                  ; based on the value of 'tp_id'
)

req_info = (
  + elements  ; Number, format and encoding based on
              ; the value of 'tp_id' in 'srv_addr'
)
```

```
tp_info = [
    tp_id    : 1,              ; UDP as transport protocol
    srv_host : #6.260(bstr),   ; Src. address of multicast notifications
    srv_port : uint,           ; Src. port of multicast notifications
    token    : bstr,           ; Token of the phantom request and
                               ; associated multicast notifications
    cli_addr : #6.260(bstr),   ; Dst. address of multicast notifications
  ? cli_port : uint            ; Dst. port of multicast notifications
]
```

Concrete encoding for this document, where 'tp_id' = 1 (UDP)

› Defined new IANA registry, for 'tp_id' values, and formats of 'srv_addr' and 'req_info'

› Format reused for the Response-Forwarding option in *draft-tiloca-core-groupcomm-proxy*

Updates from -05

- There is no client-server negotiation of multicast notification service
  - The proposed mechanisms is used in situations where:
    - Individual notifications are not feasible; or
    - Individual notifications are not preferred beyond a certain number of observers
  - Future applications can define negotiation mechanisms if need be

- Signaling of multicast notification service
  - A web link can include the target attribute “grp_obs”, as a simple hint

- Revised processing in the presence of forward proxies
  - Improved mechanics without and with Group OSCORE (Section 9 and Section 10)
  - Updated examples in Appendix E and Appendix F

Updates from -05

- Appendix C – OSCORE group self-managed by the server
  - The client's observation request works as a joining request
  - The informative response includes also group keying material
  - This mirrors the case where the client joins an OSCORE group only as silent server
  - Not suitable when backward security and forward security are required

- Appendix D – Phantom request as deterministic request
  - Each client builds the same phantom request, see *draft-amsuess-core-cachable-oscore*
  - No need to include the phantom request in the '*ph_req*' of informative responses

Summary

- Latest additions
  - New flexible and extensible encoding of the informative response
  - No negotiation with clients; just signaling of support for group observations
  - Revised processing with proxies; updated examples in Appendix E and F
  - New Appendix C: OSCORE group self-managed by the server
  - New Appendix D: phantom request as deterministic request
- Next steps
  - Case with reverse proxy – Mechanics and example
  - Case with deterministic request and proxy – Mechanics and example
- Ready for WG adoption ?

Thank you!

Comments/questions?

https://gitlab.com/crimson84/draft-tiloca-core-observe-responses-multicast

Backup

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

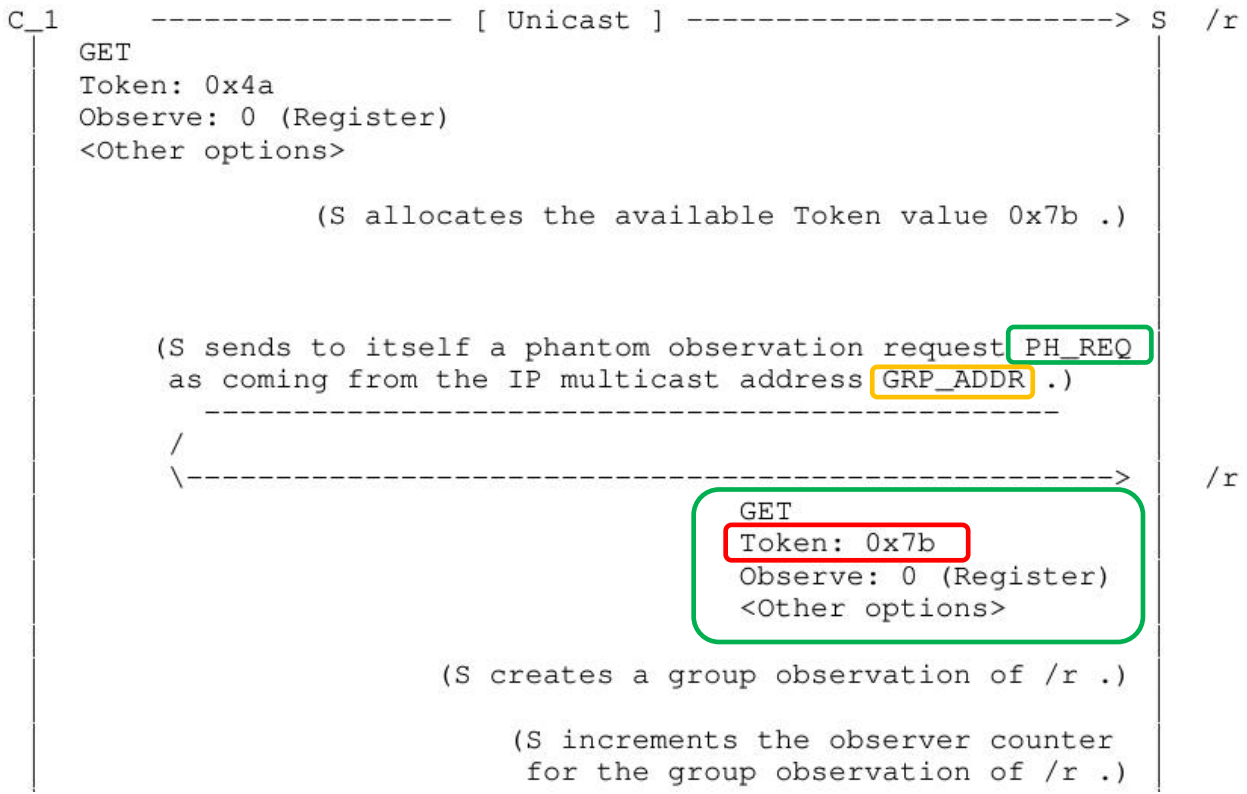
Interaction with clients

- 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

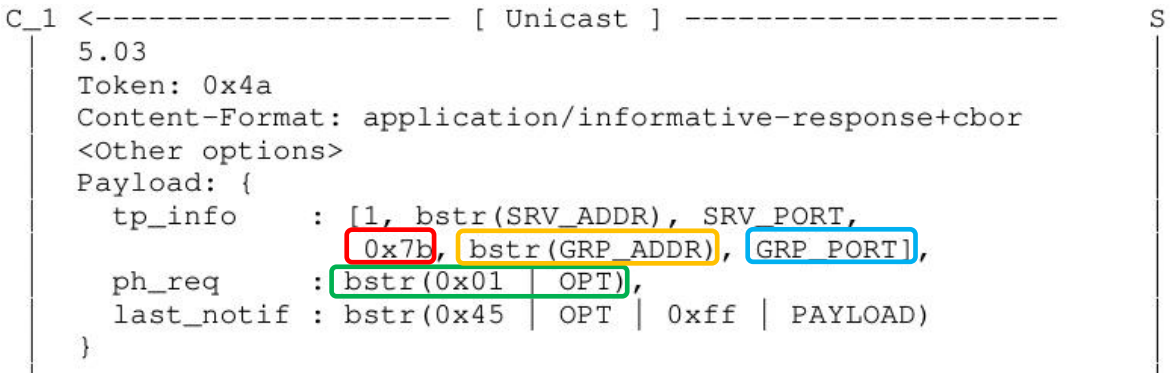
- When the value of the target resource changes:
  - The server sends an Observe notification to the IP multicast address '*cli_addr*'
  - The notification has the Token value T of the phantom request

- 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

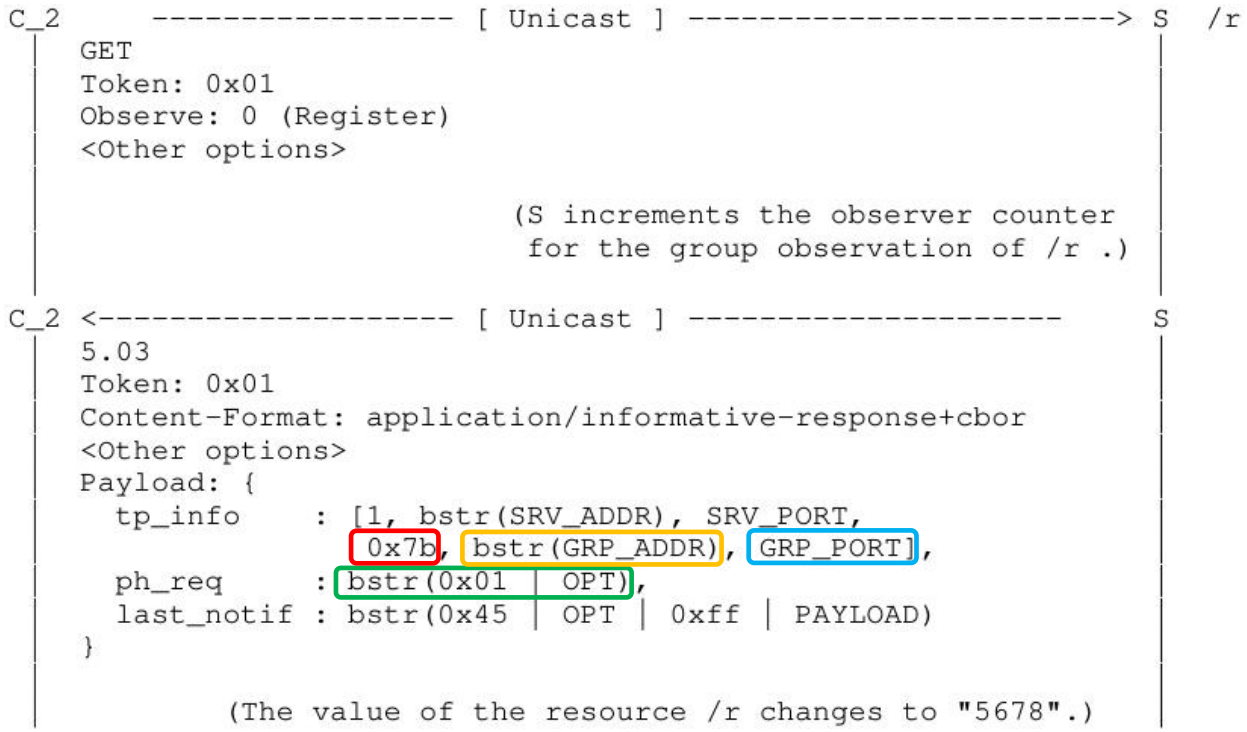
C1 registration



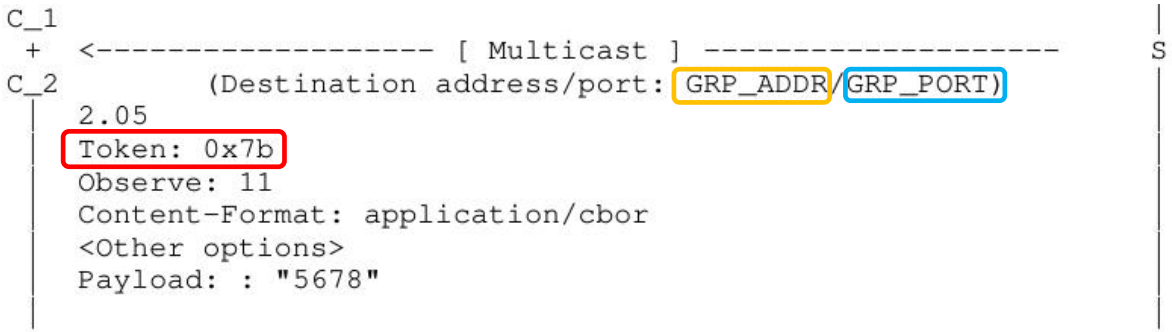
C1 registration



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

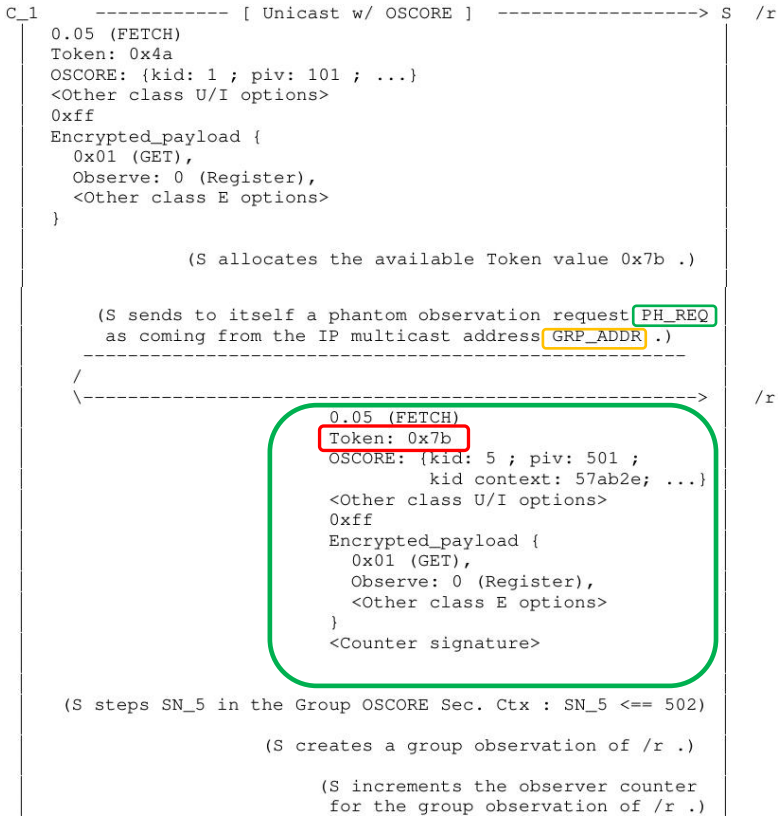
Security with Group OSCORE

- 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
  - '*cs-alg*' : countersignature algorithm
  - '*cs-alg-crv*' : countersignature curve of the algorithm
  - '*cs-key-kty*' : countersignature key type
  - '*cs-key-crv*' : countersignature curve of the key
  - '*cs-kenc*' : countersignature key encoding
  - '*alg*' : AEAD algorithm
  - '*hkdf*' : HKDF algorithm

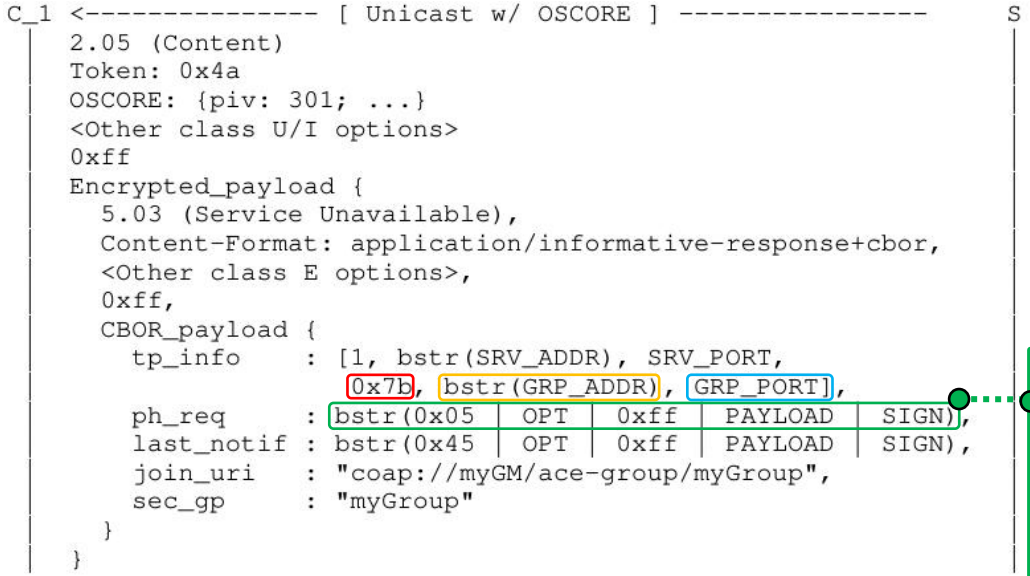
MUST

MAY

C1 registration w/ security



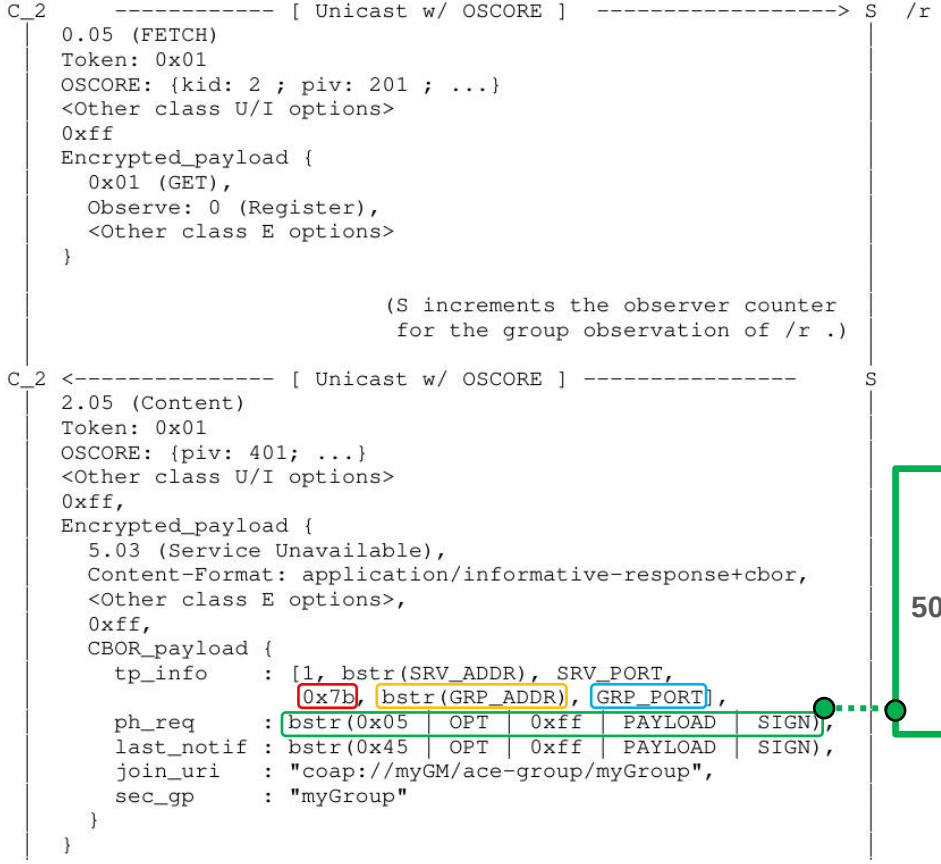
C1 registration w/ security



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

501: Sequence Number of S in the OSCORE group <u>when S created the group observation</u>

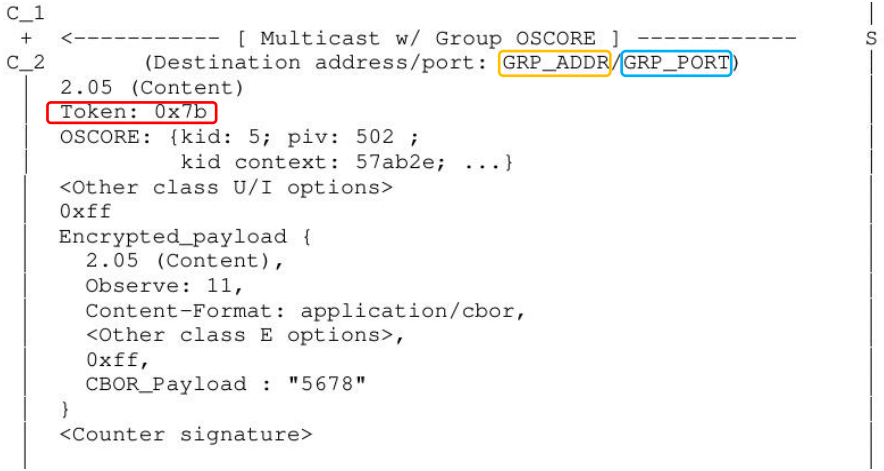
C2 registration w/ security



5: 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' = **5** 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

Proxy Operations for CoAP Group Communication

draft-tiloca-core-groupcomm-proxy-03

Marco Tiloca, RISE
Esko Dijk, IoTconsultancy.nl

IETF 110, CoRE WG, March 8th, 2021

Recap

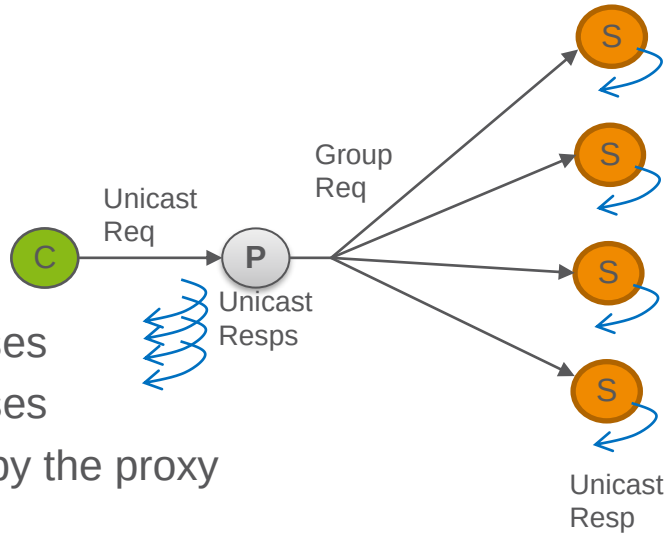
- CoAP supports group communication over IP multicast
  - Section 3.4 of *draft-ietf-core-groupcomm-bis* discusses issues when using a proxy
  - The proxy forwards a request to the group of servers, over IP multicast
  - Handling responses and forwarding them back to the client is not trivial

- Contribution – Description of proxy operations for CoAP group communication
  - Addressed all issues in *draft-ietf-core-groupcomm-bis*
  - Signaling protocol between client and proxy, with two new CoAP options
  - Responses individually forwarded back to the client

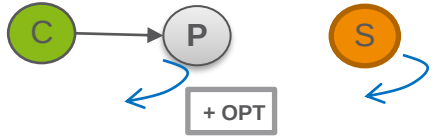
- The proxy is explicitly configured to support group communication
  - Clients are allowed-listed on the proxy, and identified by the proxy

How it works

- In the unicast request addressed to the proxy, the client indicates:
  - To be interested / capable of handling multiple responses
  - How long the proxy should collect and forward responses
  - The new CoAP option **Multicast-Signaling**, removed by the proxy
- In each response to a group request, the proxy includes the server address
  - In the new CoAP option **Response-Forwarding**
  - The client can distinguish responses and different servers
  - The client can contact an individual server (directly, or again via the proxy)
- Group OSCORE can be used for e2e security between client and servers
- OSCORE can be used between Client and Proxy (Appendix A), or DTLS



Updates from -03



- Only for P → C responses
  - Value: addressing information about the server (from the original response)
  - The proxy adds the option, before forwarding the response to the client
  - Presence: the client can distinguish responses and origin servers

No.	C	U	N	R	Name	Format	Length	Default
TBD2			-		Response-Forwarding	(*)	9-24	(none)

C=Critical, U=Unsafe, N=NoCacheKey, R=Repeatable

Response-Forwarding Option

- Format of option value aligned with [1]
  - Serialization of a '*tp_info*' array
  - Thanks to Christian for the suggestion!

```
tp_info = [
      tp_id : 1,              ; UDP as transport protocol
   srv_host : #6.260(bstr),   ; IP address where to reach the server
 ? srv_port : uint / null     ; Port number where to reach the server
]
```

- Assumed '*srv_port*' values when '*tp_id*' = 1 (UDP)
  - If present with value Null → default CoAP port number 5683
  - If not present → same as destination port number of the proxied group request

[1] *draft-tiloca-core-observe-multicast-notifications-05*

Updates from -03

- New registrations, for '*tp_id*' different than 1 (UDP)
  - "CoAP Transport Information" Registry [1]
  - Useful here already, when using a reverse-proxy
- "CoAP Transport Information" Registry [1]
- Section 3.1
  - Encoding of elements in "Srv Addr" and "Req Info"
- Section 3.2
  - Default value to use, if 'srv_port' has value Null

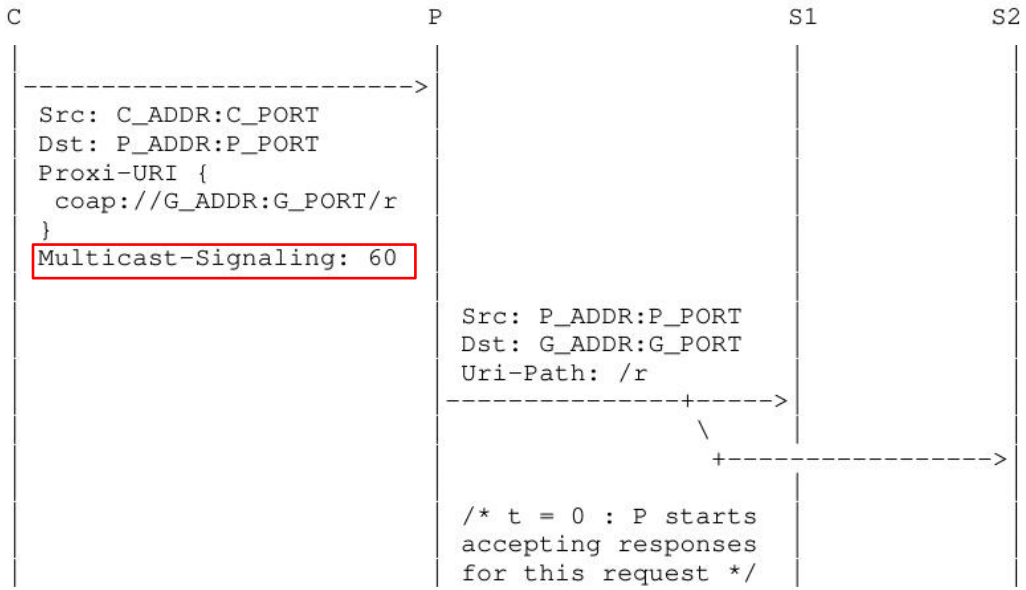
Transport Protocol	Description	Value	Srv Addr	Req Info	Reference
UDP secured with DTLS	UDP with DTLS is used as per RFC8323	2	tp_id srv_host srv_port	token cli_host ?cli_port	[This document]
TCP	TCP is used as per RFC8323	3	tp_id srv_host srv_port	token cli_host ?cli_port	[This document]
TCP secured with TLS	TCP with TLS is used as per RFC8323	4	tp_id srv_host srv_port	token cli_host ?cli_port	[This document]
WebSockets	WebSockets are used as per RFC8323	5	tp_id srv_host srv_port	token cli_host ?cli_port	[This document]
WebSockets secured with TLS	WebSockets with TLS are used as per RFC8323	6	tp_id srv_host srv_port	token cli_host ?cli_port	[This document]

[1] *draft-tiloca-core-observe-multicast-notifications-05*

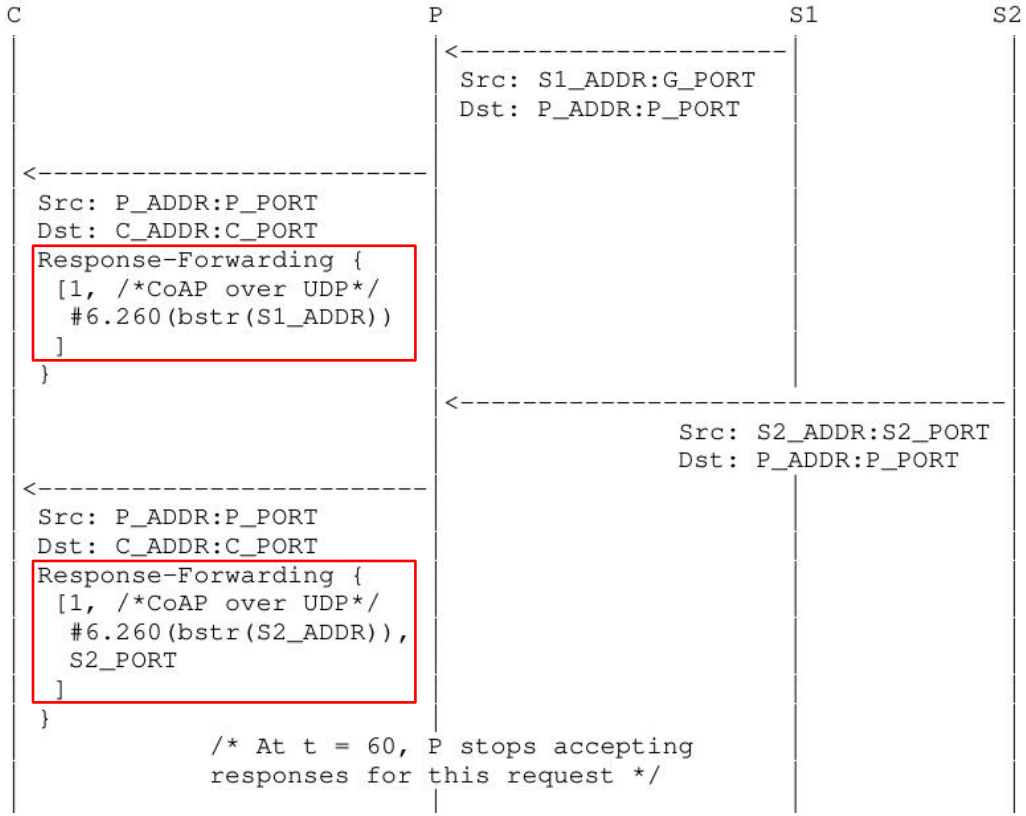
Updates from -03

- Support for reverse-proxies added
  - Thanks to Christian also for this suggestion!
- A client aware of server being a reverse-proxy:
  - MUST use the Multicast-Signaling Option (under discussion: issue #19)
  - Client acts similar to the case of the forward-proxy
- The reverse-proxy:
  - Processes the new options like a forward-proxy
  - Possibly “reveals itself” as a reverse-proxy
    - If it receives a group request without Multicast-Signaling option, and one is required, then …
    - it returns a 4.00 response, including an empty Multicast-Signaling option
- No difference for the servers

Example with forward-proxy (1/2)

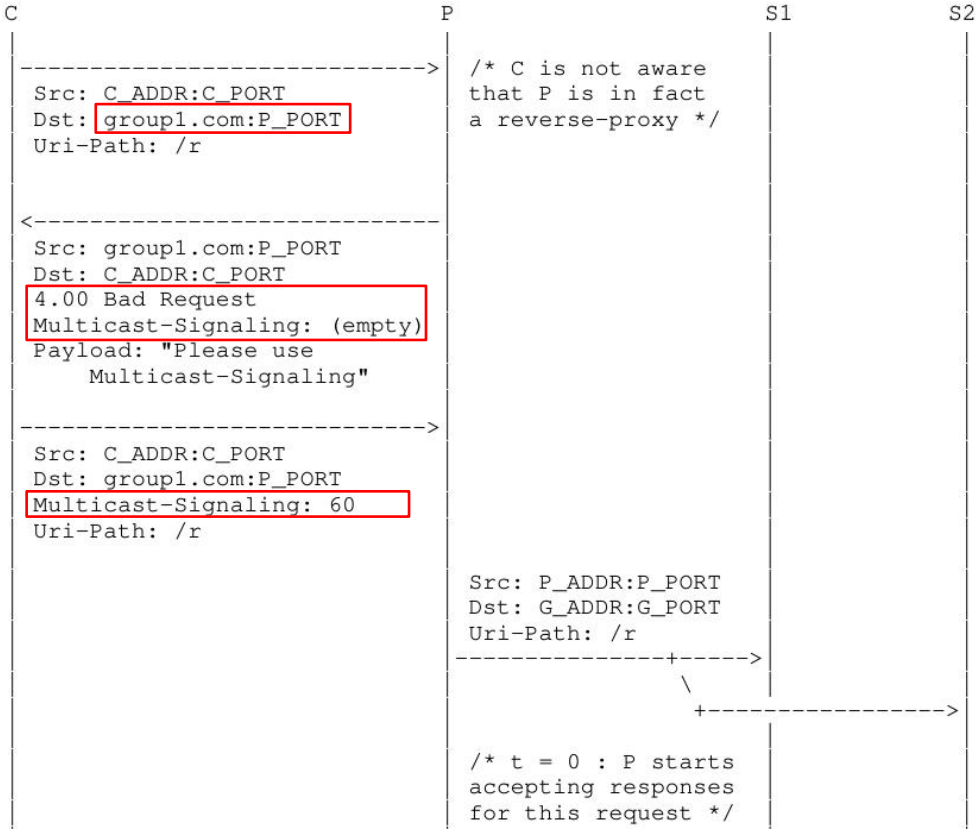


Example with forward-proxy (2/2)



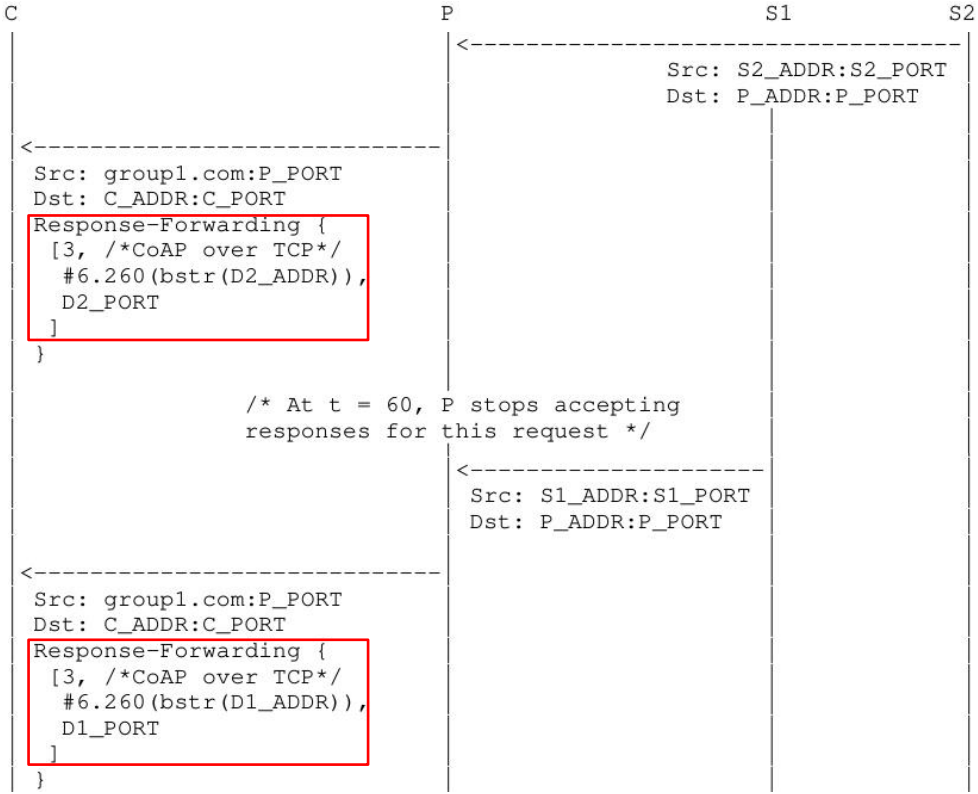
Example with reverse-proxy (1/3)

- C→P: CoAP over TCP
- group1.com resolves to the address of P
- The proxy hides the group as a whole and the individual servers



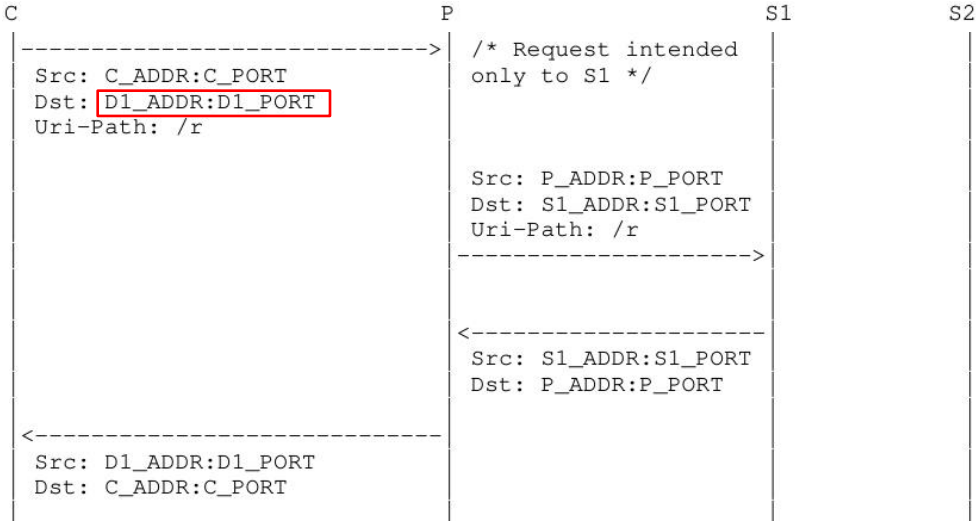
Example with reverse-proxy (2/3)

- C→P: CoAP over TCP
- group1.com resolves to the address of P
- The proxy hides the group as a whole and the individual servers
- Dx_ADDR:Dx_PORT is mapped to address and port of server Sx



Example with reverse-proxy (3/3)

- C→P: CoAP over TCP
- group1.com resolves to the address of P
- The proxy hides the group as a whole and the individual servers
- Dx_ADDR:Dx_PORT is mapped to address and port of server Sx



OSCORE between Client and Proxy

- Can co-exist with Group OSCORE between client and servers

- Some class **U** options are then treated by Proxy as if class **E**
  - Proxy-URI, Proxy-Scheme, Uri-Host, Uri-Port Options
  - OSCORE Option, if Group OSCORE is used end-to-end
  - Multicast-Signaling and Response-Forwarding Options (from this document)

- More options may come → Revised general rule, from Appendix A:
  - *This generally applies to all options that the proxy needs to understand and process in its exchange with the origin client. Further options can be added and treated as class U, e.g. related to routing information.* Accurate and simple enough?

OSCORE between Client and Proxy

- Nested OSCORE can have a broad applicability
  - A proxy forwarding to a group of CoAP servers – Like in this document
  - A server in a local domain, also acting as (cross-)proxy to servers in external domains
    - E.g., the local-domain server can be the Device Manager in a LWM2M setup

- Nested OSCORE is currently forbidden – There used to be no use case …
  - RFC 8613: *Nested use of OSCORE is not supported: If OSCORE processing detects an OSCORE option in the original CoAP message, then processing SHALL be stopped.*
  - This would require proper amendment, design and analysis.

- Move it from Appendix A of this document to a separate, dedicated document?

Summary

- Proxy operations for CoAP group communication
  - Embedded signaling protocol, using two new CoAP options
  - The proxy forwards individual responses to the client for a signaled time
  - The client can distinguish the origin servers and corresponding responses
- Latest additions
  - Revised encoding of server address information
  - Added support for reverse-proxies
  - Workflow examples, for forward-proxies and reverse-proxies
- Next steps
  - Define HTTP headers for Cross-Proxies → enable a HTTP client to talk to a CoAP group
  - ... and other issues listed
- Need for reviews – Promised: Christian, Carsten

Thank you!

Comments/questions?

https://gitlab.com/crimson84/draft-tiloca-core-groupcomm-proxy

Backup

Issues with proxies

- From Section 3.4 of *draft-ietf-core-groupcomm-bis*

- Issues when using proxies
  - Clients to be allow-listed and authenticated on the proxy
  - The client may receive multiple responses to a single *unicast* request
  - The client may not be able to distinguish responses and origin servers
  - The proxy does not know when to stop handling responses

- Possible approaches for proxy to handle the responses
  - **Individually forwarded back to the client**
  - Forwarded back to the client as a single aggregated response

Workflow: C -> P

- C prepares a request addressed to P
  - The group URI is included in the Proxi-Uri option or the URI-* options
- C chooses T seconds, as token retention time
  - T < Tr , with Tr = token reuse time
  - T considers the processing time at the proxy and the involved RTTs
- C includes the Multicast-Signaling option, with value T′ < T
- C sends the request to P via unicast
  - C retains the token beyond the reception of a first matching response

Workflow: P -> S

- P identifies C and verifies it is allowed-listed
- P verifies the presence of the Multicast-Signaling option
  - P extracts the timeout value T′
  - P removes the Multicast-Signaling option
- P forwards the request to the group of servers, over IP multicast
- P will handle responses for the following T′ seconds
  - Observe notifications are an exception – they are handled until the Observe client state is cleared.

Workflow: S -> P

- S processes the request and sends the response to P

- P includes the Response-Forwarding option in the response
  - The option value is absolute URI of the server
  - IP address: source address of the response
  - Port number: source port number of the response

Workflow: P -> C

- P forwards responses back to C, individually as they come
- P frees-up its token towards the group of servers after T′ seconds
  - Later responses will not match and not be forwarded to C
  - Observe notifications are the exception
- C retrieves the Response-Forwarding option
  - C distinguishes different responses from different origin servers
  - C is able to later contact a server individually (directly or via the proxy)
- C frees-up its token towards the proxy after T seconds
  - Observe notifications are the exception

Support for chain of proxies (1/2)

- Each proxy forwards the group request to the next hop
  - Nothing changes for the last proxy or for the origin servers
- Each proxy has to allow-list and authenticate the previous hop
- Only the last proxy removes the Multicast-Signaling option altogether
- For each **non-last** proxy:
  - The time indication T′ from Multicast-Signaling is still used for the local timer
  - If T′ > 0, a new value T″ < T′ replaces the value of Multicast-Signaling
- If a good T″ can’t be determined, reply with 5.05 (Proxying not supported)
  - Include Multicast-Signaling, with the minimum acceptable value for T′

Support for chain of proxies (2/2)

- Each proxy forwards the response back to the previous hop
  - Nothing changes for the last proxy or for the origin servers
- Only the last proxy adds the Response-Forwarding option
- Each **non-last** proxy does **not** alter or remove the Response-Forwarding option

OSCORE between Client and Proxy

- P has to authenticate C
  - A DTLS session would work
  - If Group OSCORE is used with the servers
    - P can check the counter signature in the group request
    - P needs to store the clients' public keys used in the OSCORE group
    - P may be induced to forward replayed group requests to the servers

- Appendix A – OSCORE between C and P
  - If Group OSCORE is also used between C and the servers
    1. Protect the group request with Group OSCORE (C<->Servers context)
    2. Protect the result with OSCORE (C<->P context)
       - Some class U options are processed as class E options
    3. Reverse processing for responses

Deterministic Requests: Cacheable OSCORE

draft-amsuess-core-cachable-oscore

Christian Amsüss, Marco Tiloca

2021-03-08, IETF 110

Why caching?

- Reduce traffic

  Firmware updates
- Hide traffic

  Firmware updates, again
- Increase reliability
- Decrease latency
- Makes `multicast-notifications` work[1]

  Make protected case as simple as unprotected case

[1] It works without Deterministic Requests. It works *better* with.

Why is this hard in (even Group) OSCORE

POST / 2.01
Different PIVs
} uncacheable

Original KID / PIV unknown
Foreign KID request is untrusted[2]
} unverifiable

[2] It'd be a pity if someone requested `/whom-i-know`, but handed you a different request for `/whom-to-trust`

Proposed mechanism[4]

- Dedicated group member: Deterministic Client
- Request-Hash option: Hash of DC sender key || external AAD || plaintext
- "Pairwise" sender key of DC derived from DC sender key and Request-Hash[3]
- Server recognizes DC as requester, builds recipient key from Request-Hash, verifies Request-Hash
- Response bound to request using external AAD (%)

[3] Details pending processing of received comments
[4] including sneak peek at -02

Request-response binding: Details

~~Overriding the Request-KID-Context~~

- Request-Hash as an option in the response
- Request-Hash is Class I for responses
- Request-Hash may be elided from response on the wire transmission but is reconstructed by recipient before OSCORE processing

Limitations

- Only safe requests (GET, FETCH)
- Only resources every group member may access this way
- Algorithms limited to those doing AEAD deterministically

  Currently, all are.

- Security properties traded for cacheability
  - No order between request and response
  - Limited request confidentiality
  - No source authentication
  - No replay protection

Status

- Two implementations interop'd at version between -01 and -02

  The things you learn...
- Addressing pending comments on security.
  Then: review with security in mind.
- Practical testing
- Further WG input?

Combining EDHOC and OSCORE

draft-palombini-core-oscore-edhoc-02

Francesca Palombini, Ericsson
Marco Tiloca, RISE
Rikard Höglund, RISE
Stefan Hristozov, Fraunhofer AISEC
Göran Selander, Ericsson

IETF 110, CoRE WG, March 8th, 2021

Recap

- Optimization for combining EDHOC (run over CoAP) with OSCORE
  - Combines EDHOC message_3 and the first subsequent OSCORE request
    - In a single EDHOC + OSCORE request, transporting both
  - Reduces the number of round trips required
    - To set up the OSCORE Security Context
    - To complete the first OSCORE transaction with that Context

- Detailed contribution
  - Method for signalling the combined message
  - Format and processing of the EDHOC + OSCORE request
  - Example of encoded EDHOC + OSCORE request

Original way: EDHOC then OSCORE

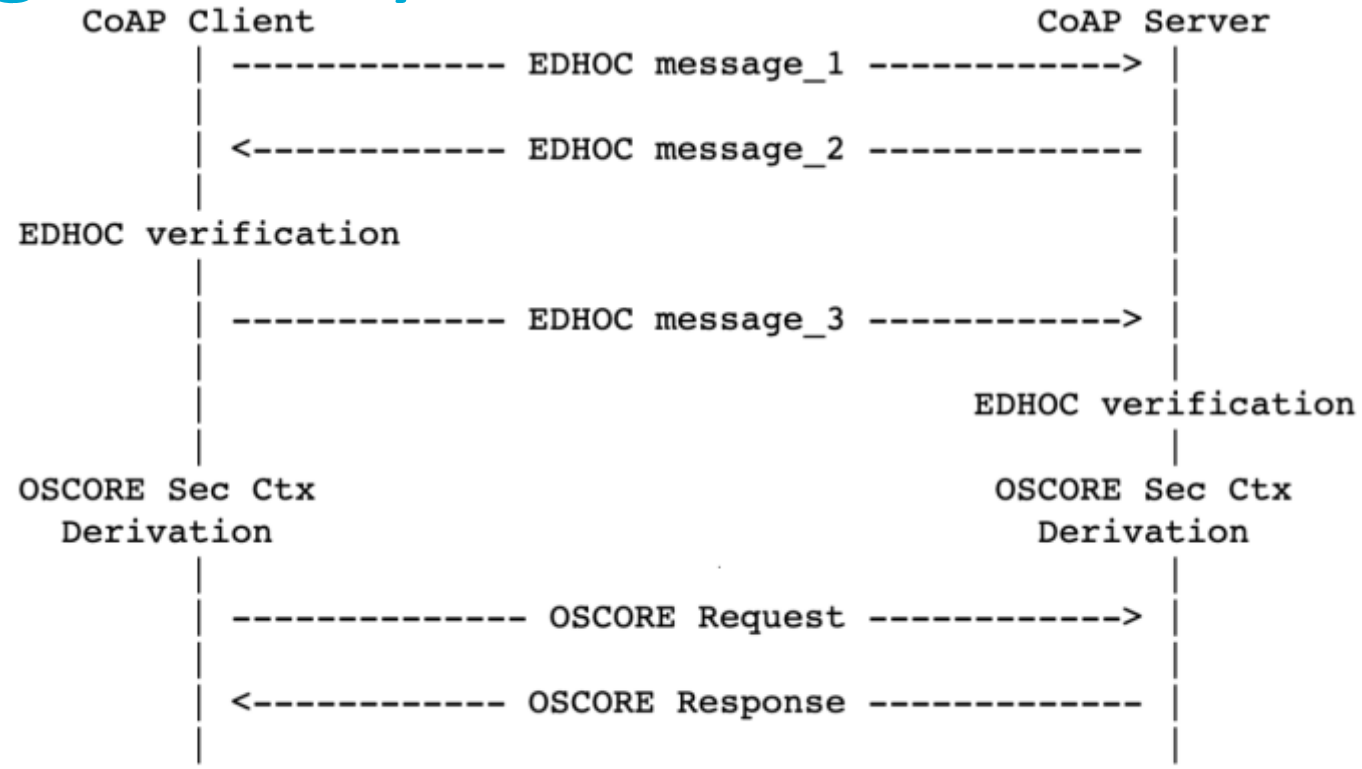


Figure 1: EDHOC and OSCORE run sequentially

New way: EDHOC + OSCORE Request

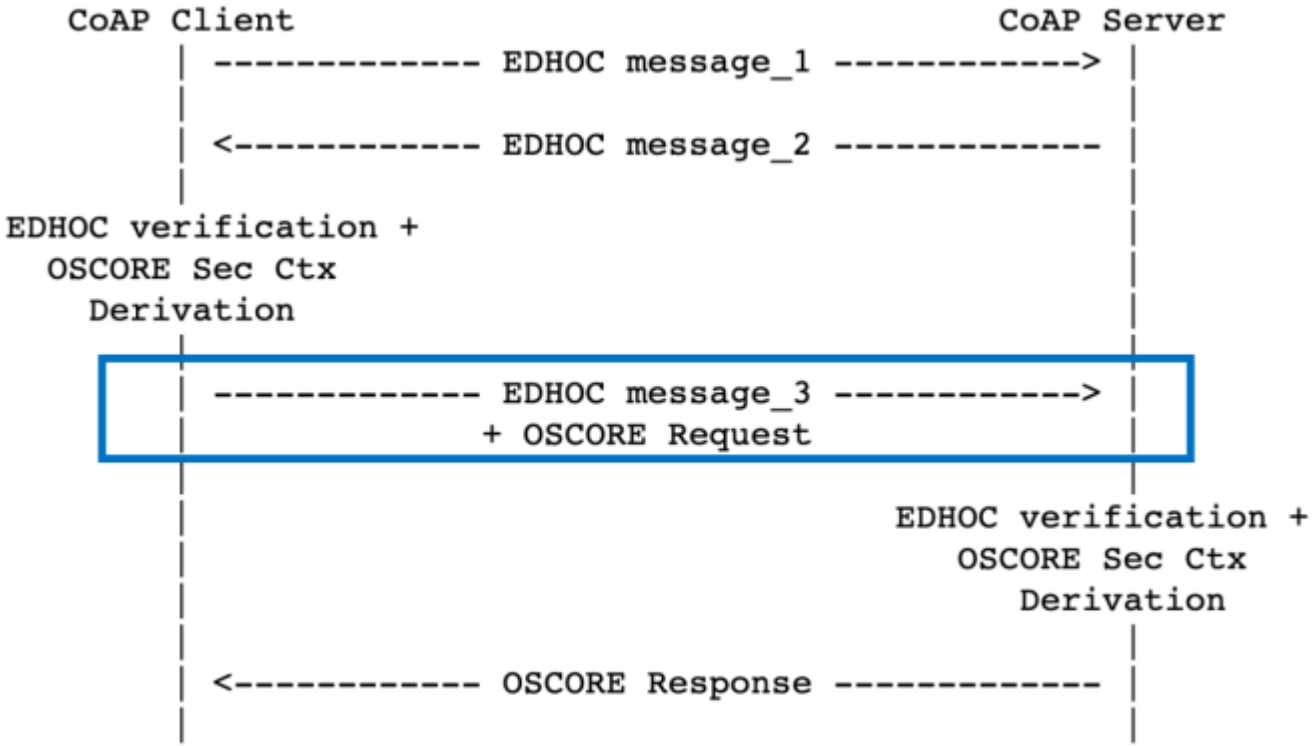


Figure 2: EDHOC and OSCORE combined

Updates from -02

- Single method for signalling the combined message
  - Use a new EDHOC option (zero-length); class U for OSCORE
  - Intended only for the EDHOC + OSCORE request
  - Based on preference from IETF 109, and feedback from implementers

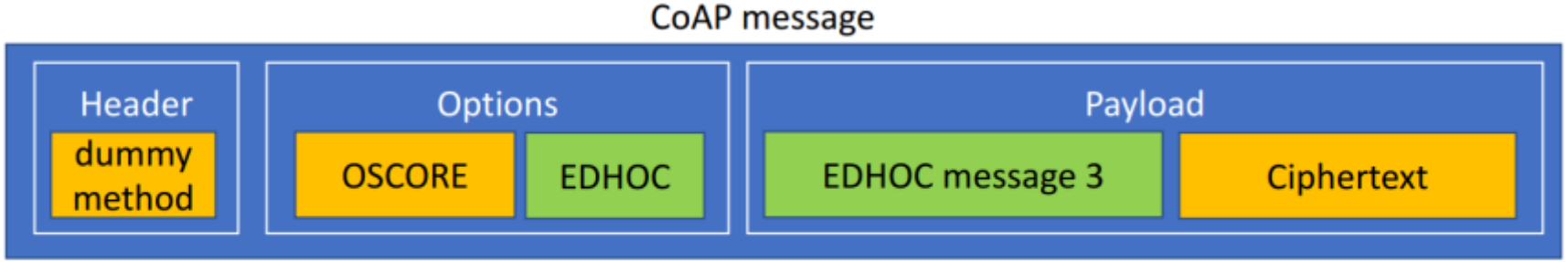
- Proposed suitable option number 13 to keep the overall option size of 1 byte
  - That's because the OSCORE option (9) is always present
  - Hence, the delta for the EDHOC option is less than 12
  - Note: option number 21 would work fine as well

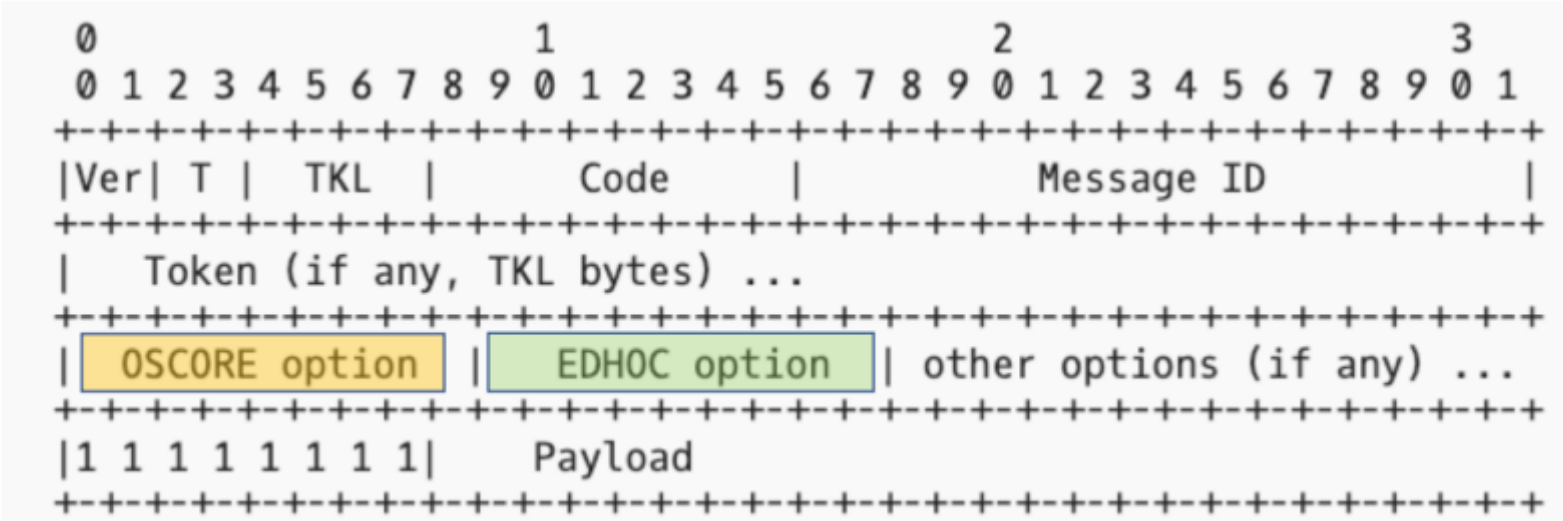
No.	C	U	N	R	Name	Format	Length	Default
TBD13	x				EDHOC	Empty	0	(none)

C=Critical, U=Unsafe, N=NoCacheKey, R=Repeatable

Figure 3: The EDHOC Option.

EDHOC + OSCORE request





Updates from -02

- Section restructuring and editorial improvement
  - Consistent with the signalling using the EDHOC option
- Improved step-by-step description of message processing
  - Detailed steps on client and server side
- Client (EDHOC Initiator):
  - Prepare EDHOC message_3 and OSCORE request; combine and send
- Server (EDHOC Responder):
  - Receive combined request; extract and process EDHOC message_3; derive OSCORE context; process the OSCORE request

Updates from -02

- Further optimization in the EDHOC + OSCORE request
  - Avoid the Sender ID of the Client to be redundant information!
    - C_R in the full EDHOC message_3 (always present in this setup)
    - 'kid' field in the OSCORE option (always present in a request)
- The combined request has a partial EDHOC message_3, that:
  - Does not include C_R
  - Includes just CIPHERTEXT_3 as a CBOR byte string
  - This saves at least 2-4 bytes on the wire
- The server rebuilds the full EDHOC message_3
  - Takes 'kid' from the OSCORE option
  - Encodes it as a bstr_identifier, as per EDHOC
  - Rebuilds the CBOR Sequence [C_R , CIPHERTEXT_3]

```
message_3 = (
  data_3,
  CIPHERTEXT_3 : bstr,
)

data_3 = (
  ? C_R : bstr_identifier,
)
```

EDHOC Message_3

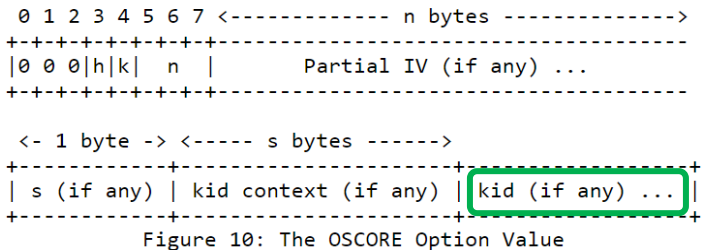


Figure 10: The OSCORE Option Value

Updates from -02

- Improved error handling on the server side
  - Details on behavior when EDHOC processing fails
  - Considerations on error code and content format to use

- EDHOC processing failure
  - Return an EDHOC Error Message
  - This will be a non-protected response to an OSCORE protected request
  - Unlike in the EDHOC draft, need to use CoAP error codes, i.e. 4.00 or 5.00
  - Use content format application/edhoc, to distinguish from OSCORE errors

- OSCORE processing failure
  - Same as in RFC 8613

Next Steps

- Keep in sync with the main EDHOC document
  - Specific points on CoAP and OSCORE may fit better in this draft
- More feedback is welcome
- WG adoption ?

Thank you!

Comments/questions?

https://github.com/EricssonResearch/oscore-edhoc

AEAD key usage limits in OSCORE

draft-hoeglund-core-oscore-key-limits-00

Rikard Höglund, RISE
Marco Tiloca, RISE

IETF 110, CoRE WG, March 8th, 2021

Problem Overview

- OSCORE uses AEAD algorithms to provide security properties
  - Confidentiality
  - Integrity

- Forgery attack against AEAD algorithms
  - Adversary may break the security properties of the AEAD algorithm
  - See **draft-irtf-cfrg-aead-limits-01**

- Need to describe relevant limits for OSCORE
  - How the forgery attack and the limits affect OSCORE
  - Necessary steps to take during message processing
  - What to do if the limits are exceeded

Limits on key usage

- What you need to count
  - ‘q’: the number of messages protected with a specific key, i.e. the number of times the algorithm has been invoked to encrypt data with that key
  - ‘v’: the number of forgery attempts that have been made against a specific key, i.e. the amount of failed decryptions that has been done with the algorithm for that key
- When a peer uses OSCORE
  - The key used to protect outgoing messages is its Sender Key
  - The key used to decrypt and verify incoming messages is its Recipient Key
- Relevant counters for OSCORE
  - Counting number of times Sender Key has been used for encryption (q value)
  - Counting number of times Recipient Key has been used for failed decryption (v value)
  - Counters and limits can be added to the OSCORE Security Context

Limits for ‘q’ and ‘v’

- Formula for limits for AES-CCM-16-64-128 See **draft-irtf-cfrg-aead-limits-01**

```
q <= sqrt((p * 2^126) / l^2)

v * 2^64 + (2l * (v + q))^2 <= p * 2^128
```

- Depends on assumptions for the p probability values
  - Considering the values $p_q = 2^{-60}$ and $p_v = 2^{-57}$
  - Same values used in [I-D.ietf-tls-dtls13]
- Exact limits calculated

```
q <= sqrt(((2^-60) * 2^126) / 1024^2)

q <= 2^23
```

‘q’: encryptions with a key

```
v * 2^64 + (2*1024 * (v + 2^23))^2 <= 2^-57 * 2^128

v <= 112
```

‘v’: failed decryptions with a key

New information in OSCORE Context

- Sender Context
  - ‘count_q’: Initialized to 0; incremented after encrypting with the Sender Key
  - ‘limit_q’: Limit for 'count_q'
- Recipient Context
  - ‘count_v’: Initialized to 0; incremented upon a failed decryption with the Recipient Key
  - ‘limit_v’: Limit for 'count_v'
- If 'limit_v' or 'limit_q' are reached
  - The nodes must stop using that Security Context and must rekey
- This updates RFC 8613

Methods for rekeying OSCORE

- Reasoned overview of available methods
- Appendix B.2 of OSCORE (RFC 8613)
  - https://datatracker.ietf.org/doc/html/rfc8613#appendix-B.2
- OSCORE Profile of ACE
  - https://datatracker.ietf.org/doc/draft-ietf-ace-oscore-profile/
- EDHOC protocol
  - https://datatracker.ietf.org/doc/draft-ietf-lake-edhoc/
- Manual re-configuration (not practical)

Open Points

- Best location to provide the limits for more algorithms?
- Consider the assumed probabilities for p_q and p_v
  - Are the same values relevant for OSCORE as [I-D.ietf-tls-dtls13] defines?
- Adding an expiration timer to the OSCORE Security Context
  - An “expires in” element can be added to the OSCORE Security Context, similar to the ACE ‘exi’ parameter. This would hold the lifetime of the Context in seconds
  - Is there a value in doing this from a security perspective?

Summary and next steps

- AEAD limits and their impact on OSCORE
  - Introduce counting of ‘q’ and ‘v’ values for OSCORE
  - Stop and rekey if the limits are reached
  - Overview of current rekeying methods

- Next steps
  - Cover more AEAD algorithms
  - Synchronize with other ongoing work
    - EDHOC in the LAKE WG
    - Broader relevance (e.g. (D)TLS, QUIC …) - Next presentation from John
  - Optimizations for constrained devices and implementation guidelines

Thank you!

Comments/questions?

https://gitlab.com/rikard-sics/draft-hoeglund-oscore-rekeying-limits/

Usage Limits on AEAD Algorithms

IETF 110
John Preuß Mattsson

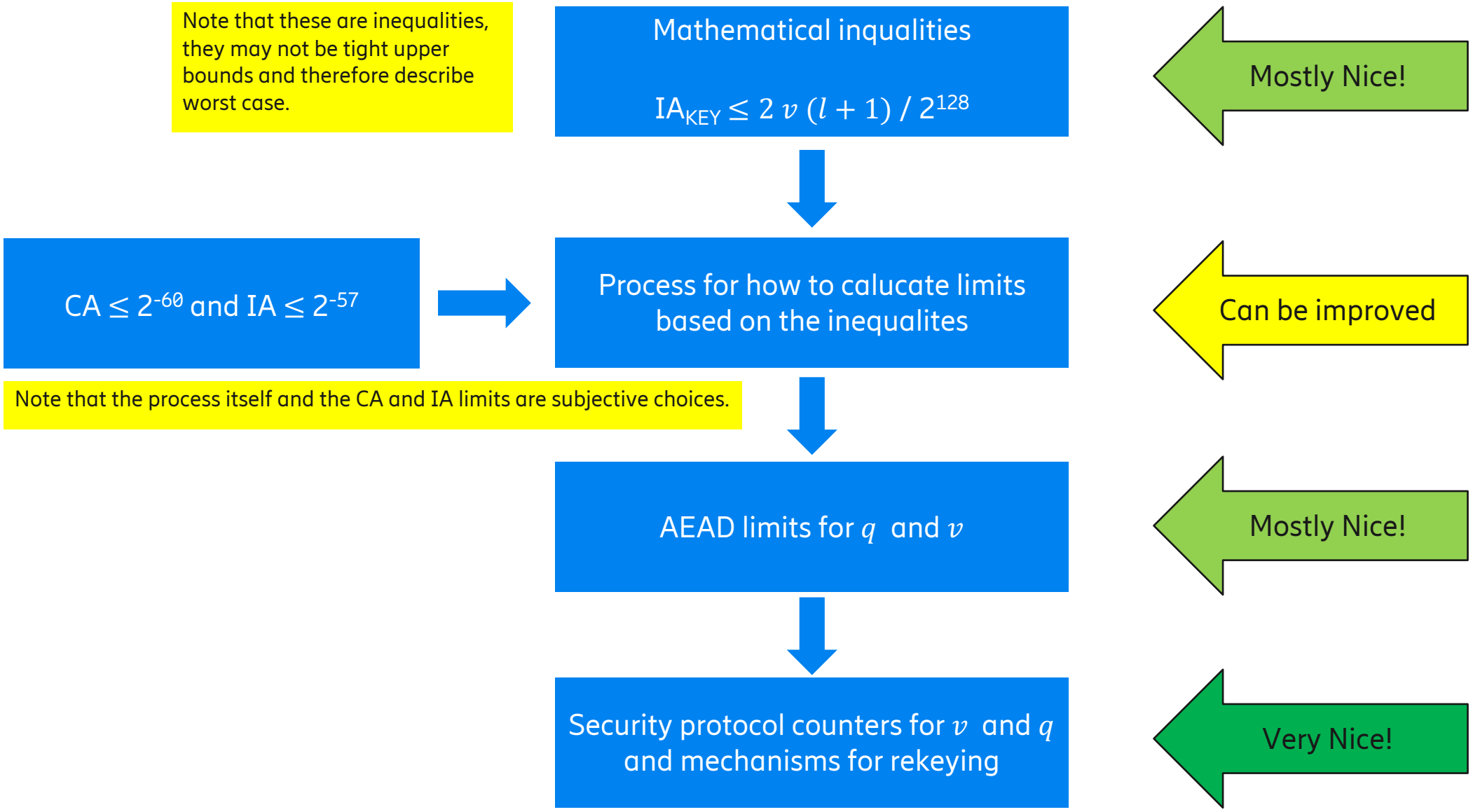
Usage Limits on AEAD Algorithms Overview

- AEAD limits have recently been discussed for TLS, DTLS, QUIC, and OSCORE:
  - https://datatracker.ietf.org/doc/draft-irtf-cfrg-aead-limits/
  - https://datatracker.ietf.org/doc/rfc8446/
  - https://datatracker.ietf.org/doc/draft-ietf-tls-rfc8446bis/
  - https://datatracker.ietf.org/doc/draft-ietf-tls-dtls13/
  - https://datatracker.ietf.org/doc/draft-ietf-quic-tls/
  - https://datatracker.ietf.org/doc/draft-hoeglund-core-oscore-key-limits/

- Use mathematical single-key and multi-key inequalities for CA (Confidentiality Advantage) and IA (Integrity Advantage) to calculate limits for:
  - v the number of attacker forgery attempts (failed AEAD decryption invocations)
  - q the number of protected messages (AEAD encryption invocations)
  - l the maximum length of each message (in blocks)
- Rekeying must be done before the v and q limits are met.

- If done correctly, rekeying gives also forward secrecy, which limits the impact of key compromise.
  - Rekeying with (EC)DHE gives additional protection by forcing attackers to keep being active.

The AEAD limits work consists of 4 steps



Analysis of the inequalities (single-key)

- The inequalities for AES-GCM [AEBounds] and AES-CCM [CCM-ANALYSIS] assumes that AES is a PRP (Pseudo-Random Permutation). Gordon Procter [ChaCha20Poly1305Bounds] assumes ChaCha20 is a PRF (Pseudo-Random Function).

- The inequality CA ≤ $v\ l$ / 2^{103} used in DTLS, TLS, QUIC, CFRG does ChaCha20 great injustice. This would suggest that the ChaCha20 stream cipher provides much worse confidentiality than AES-CTR for small q, which is not true.

- We recommend treating ChaCha20 as a PRF similar to the way AES is treated as a PRP which would imply CA = 0 for ChaCha20.

- It should be noted that CA and IA are practically very different:
  - CA is typically used for an offline attack, while IA is typically used for online attacks.
  - IA is directly related to a practical attack (forgery) while CA is more theoretical (distinguishing) and might not be directly related with any practical attack.

Analysis of the suggested "calculating limits" step

- Current suggested process is to set limit for CA and IA per key and based on the inequalities calculate limits for q and v. TLS, DTLS and QUIC use approximately CA ≤ 2^{-60} and IA ≤ 2^{-57}.

- This mostly leads to practically usable limits that improves security. The process do however also give strange and misleading results.

- The suggested process lead to the recommendation that the ideal MAC needs to be rekeyed. This does not make sense and does of course not improve security. The suggested process suggests that the ideal 64-bit MAC and CCM_8 needs to be rekeyed extremely often. DTLS 1.3 more of less forbids CCM_8 due to the rekeying requirement. For low v and q, CCM_8 behaves very close to the ideal 64-bit MAC.

- The suggested process misleadingly gives the idea that frequent rekeying can keep security high.
  - While CA for the whole connection is bounded, IA for the whole connection is unbounded.
  - For some of the advantages (AES-GCM IA, ChaCha20-Poly1305 CA and IA) rekeying it not shown to lower advantages or security levels at all.

Linear and superlinear inequalities

It is easy to see from the inequalities if rekeying improves security for the connection. Superlinear equations e.g. $(v + q)^2$ needs to be rekeyed before the security level gets to low.

— ChaCha20-Poly1305 IA	$IA \leq v \cdot l / 2^{103}$	Rekeying does not improve connection advantage
— AES-GCM IA	$IA \leq v \cdot l / 2^{127}$	Rekeying does not improve connection advantage
— AES-CCM IA	$IA \leq v / 2^{128} + l^2 (v + q)^2 / 2^{126}$	Rekeying does improve connection advantage*
— AES-CCM_8 IA	$IA \leq v / 2^{64} + l^2 (v + q)^2 / 2^{126}$	Rekeying does improve connection advantage*
— ChaCha20-Poly1305 CA	$CA = 0$	Rekeying does not improve connection advantage
— AES-GCM CA	$CA \leq (l + q)^2 / 2^{129}$	Rekeying does improve connection advantage
— AES-CCM CA	$CA \leq l^2 \cdot q^2 / 2^{126}$	Rekeying does improve connection advantage

Analysis of the suggested "calculating limits" step

- It is trivial to see that rekeying limits CA and IA per key. But with the attacker cost measured in encryption or decryption queries, rekeying might actually surprisingly increase IA for the whole connection.

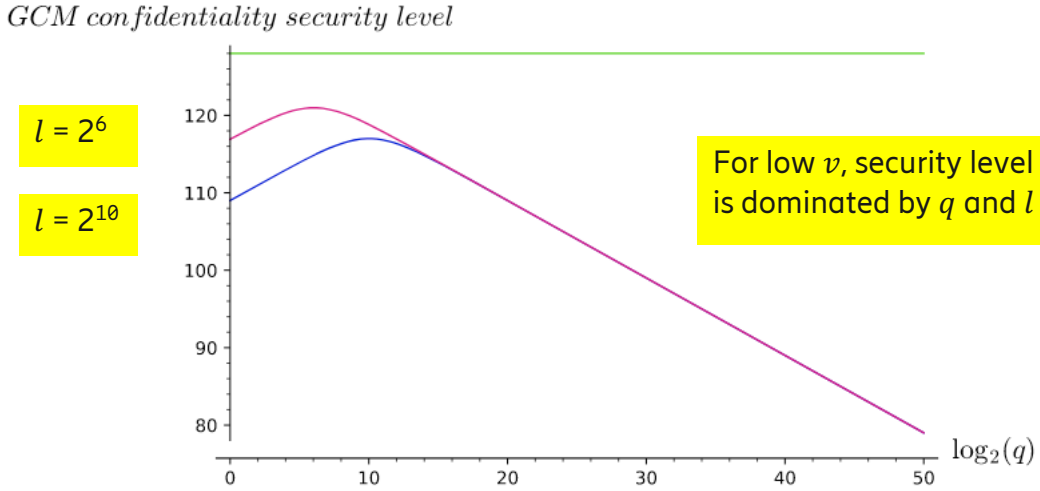
- The limits $q = 2^{23}$ and $l = 2^{10}$ makes CCM_8 deviate from an ideal MAC also for small values of v. An application/protocol using CCM_8 should probably chose smaller values, e.g. $q = 2^{20}$ and $l = 2^{8}$. With these values CCM_8 performs like an ideal 64-bit MAC to until $v = 2^{35}$.

- The process of limiting CA and IA per key does not seem like the right thing to do for a security protocol where each connection has many keys, communication between two parties can use many connections, and adversaries can often trick the parties to tear down the old connection and set up a new connection.

- An easier process seems to be to just calculate the security levels (attacker cost / advantage) and put limits on the security level for distinguishing and forgery. The security level is minimized over all possible adversaries. This seems to avoid the misleading results of the currently suggested process (rekeying ideal MAC, rekeying gives arbitrary small CA and IA per key, and rekeying increases IA for the connection).

- The suggested process has lead to quite arbitrary but practically useful limits for (D)TLS and QUIC
  - People are taking the specific process and exact limits a bit too serious.
  - We don't think the process as currently specified should be an IETF/IRTF recommendation.

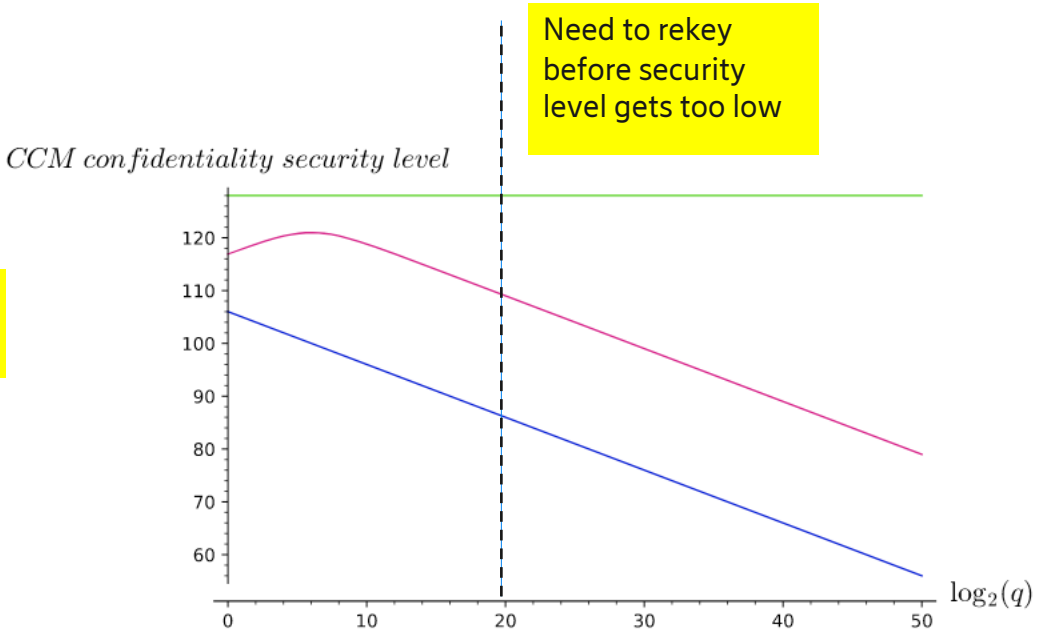
Lower bounds for security levels (in bits)



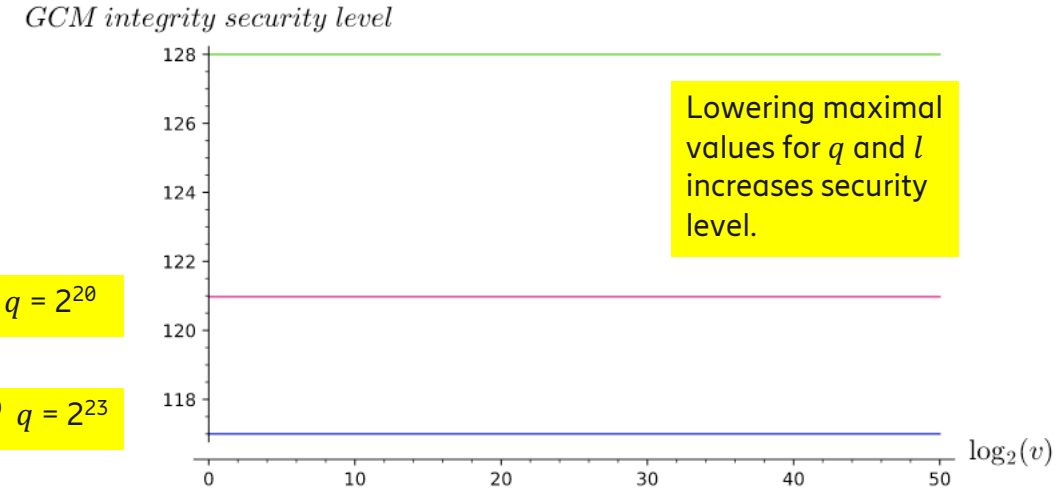
$l = 2^6$

$l = 2^{10}$

For low v, security level is dominated by q and l



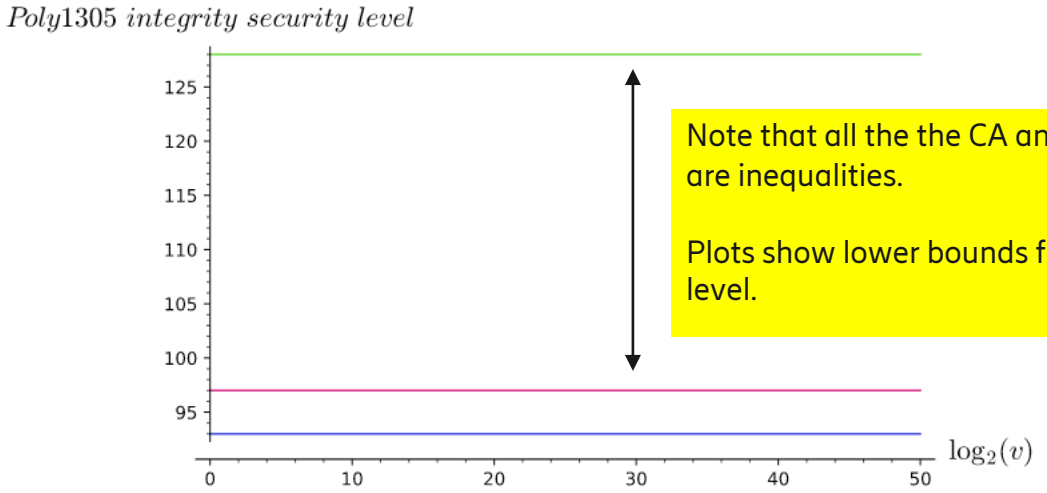
Need to rekey before security level gets too low



$l = 2^6$ $q = 2^{20}$

$l = 2^{10}$ $q = 2^{23}$

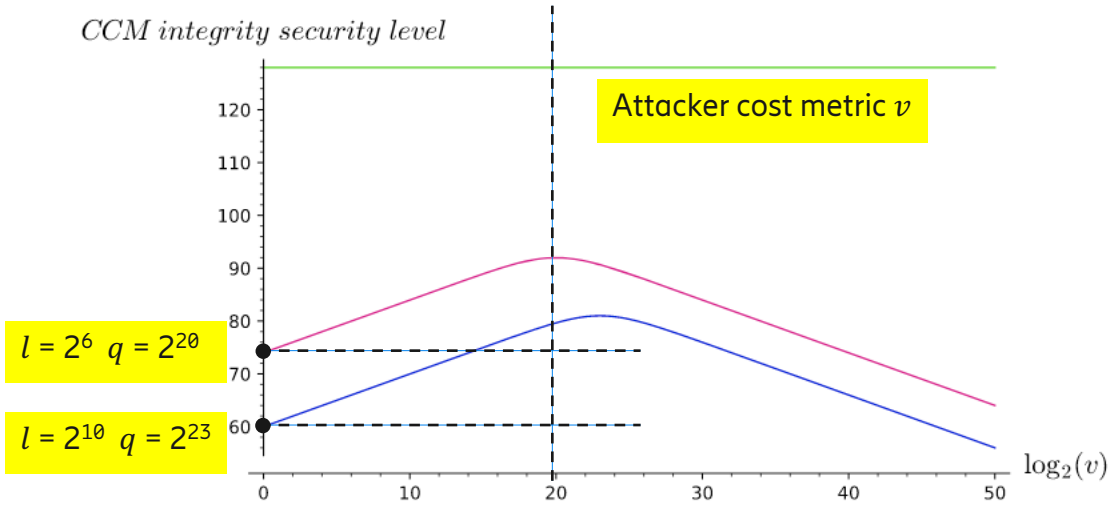
Lowering maximal values for q and l increases security level.

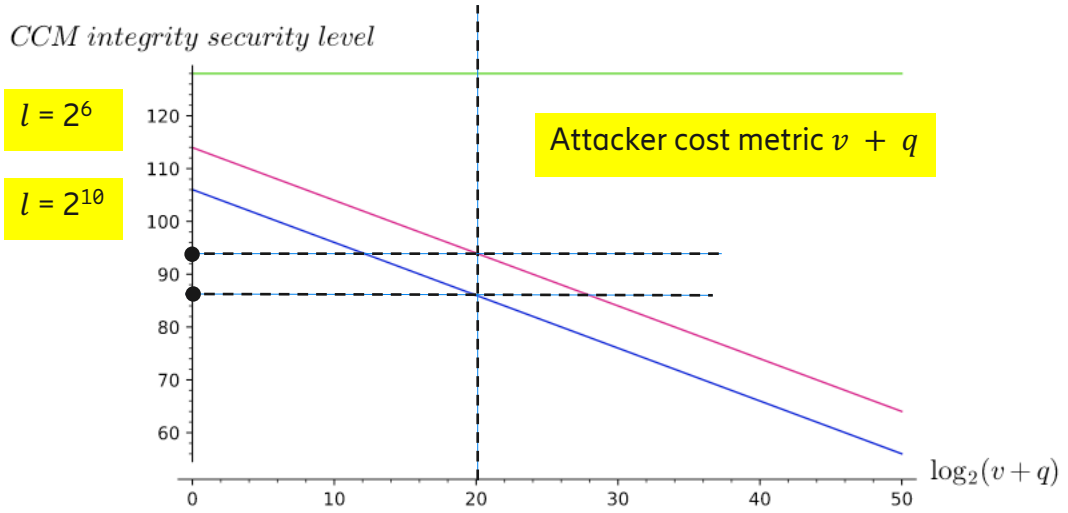


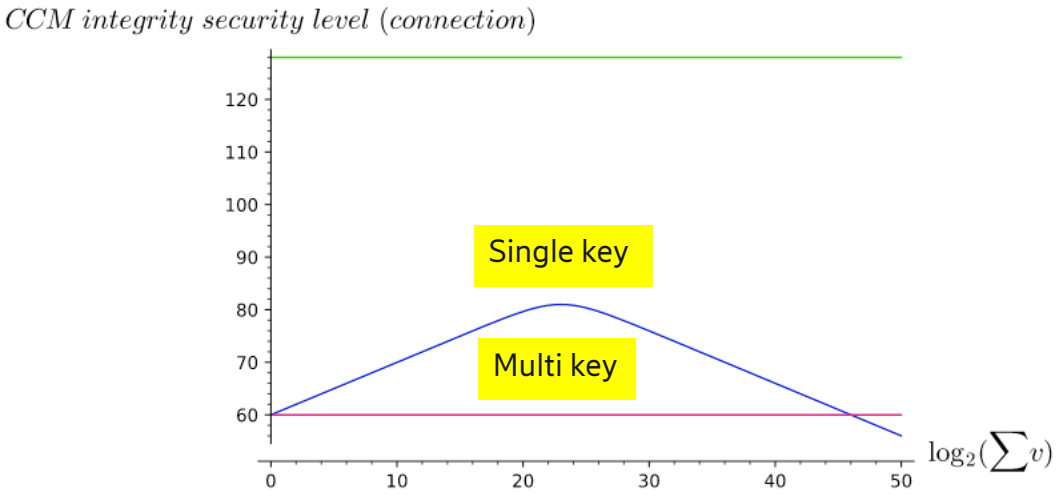
Note that all the the CA and IA Formulas are inequalities.

Plots show lower bounds for the security level.

Lower bounds CCM integrity security level (in bits)







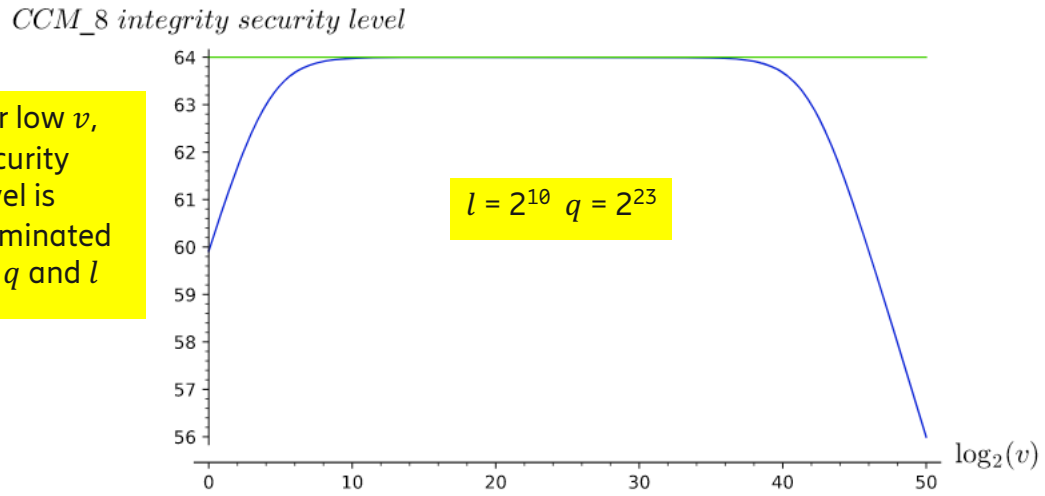
Security level is the minimized over all allowed v

If attacker cost is measured in only decryption queries or only encryption queries, rekeying can give higher advantage (but not a lower security level).

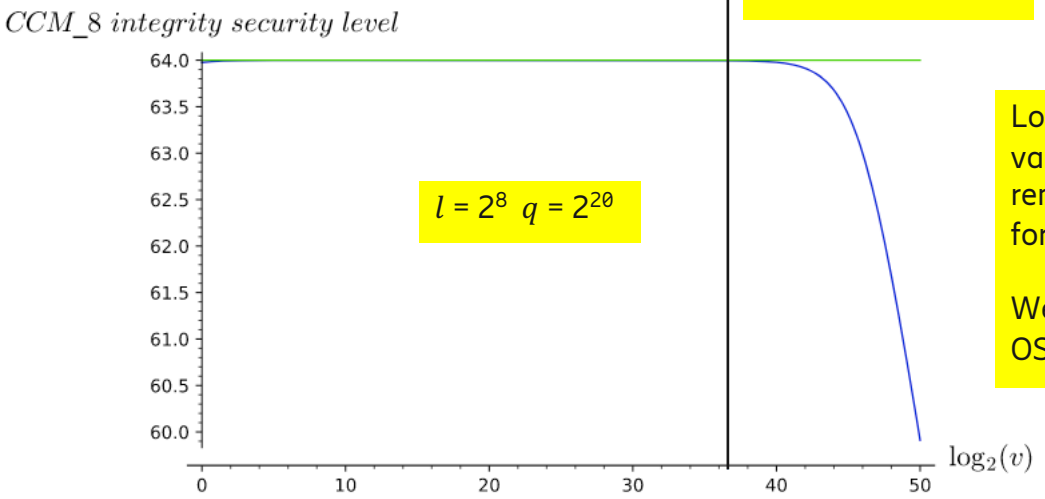
An optimal attacker do one forgery per key.

Lower bounds CCM_8 integrity security level (in bits)

For low v, security level is dominated by q and l

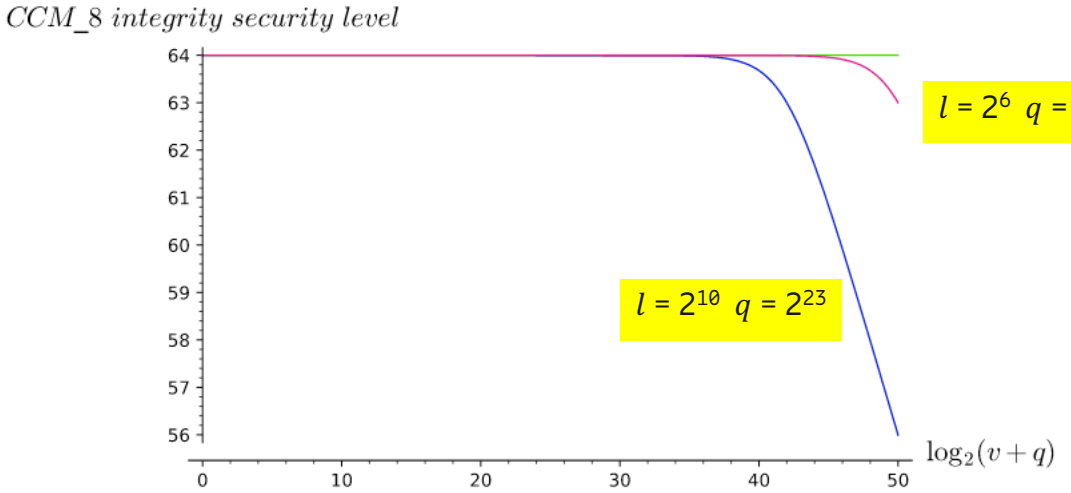


Need to rekey before security level gets too low



Lowering maximal values for q and l removes deviation for low v.

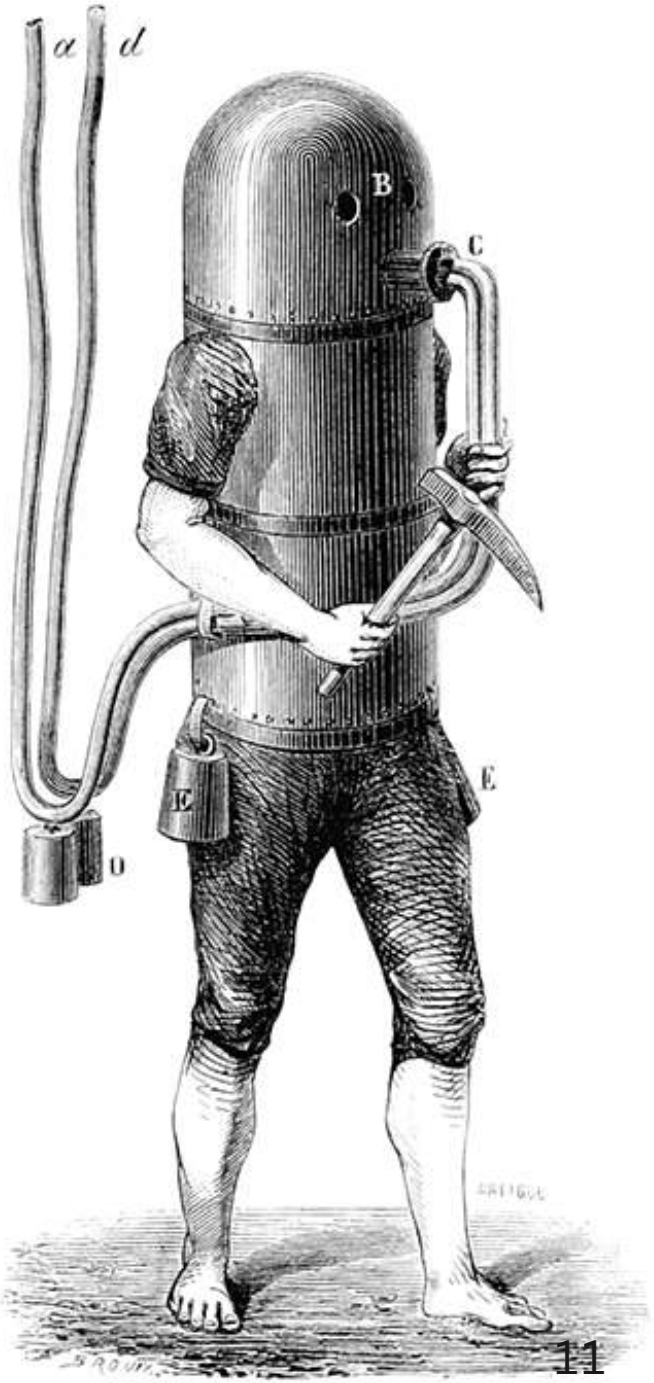
We recommend OSCORE to do so



CCM_8 behaves very much like the ideal 64-bit MAC.

Suggestions for (OS)CORE

- Security protocol counters for v and q and mechanisms for rekeying are necessary.
- Frequent rekeying with forward secrecy limits the impact of key compromise, this might be even more important that the AEAD advantages.
- Use CA = `0` for ChaCha20-Poly1305. Rekeying for linear inequalities (ChaCha20-Poly1305 CA/IA and AES-GCM IA) does not improve security level or the advantage for the connection.
- CCM_8 is a very close to a perfect 64-bit MAC for low values of q and v. No problem at all to continue using CCM_8 as long as 64-bit forgery probability is acceptable.
- 64-bit forgery probability is definitely acceptable in constrained IoT. To break 64-bit security against online brute force an attacker would on average have to send 4.3 billion messages per second for 68 years, which is infeasible in constrained IoT radio technologies.
- Consider using smaller limits than $q = 2^{23}$ and $l = 2^{10}$, this improves security for AES-GCM CA and AES-CCM(_8) CA/IA.
- The current process and limits should be taken with a pinch of salt. Suggestions for (OS)CORE:
  - Use a process that puts limits on the security level for distinguishing and forgery.
  - $q, v = 2^{20}$ for AES-GCM CA and AES-CCM CA/IA, $v = 2^{30}$ for CCM_8 IA, not limit for other?
  - $q, v = 2^{20}$ for all algorithms?
  - $l = 2^{8}$ (4 kB) ?



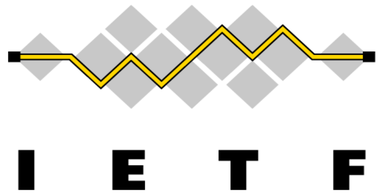
IETF 110

Constrained RESTful Environments WG (core)

Chairs:
Marco Tiloca <marco.tiloca@ri.se>
Jaime Jiménez <jaime.jimenez@ericsson.com>

Mailing list: core@ietf.org

Jabber: core@jabber.ietf.org



- **Stand-in Chair for IETF 110: Carsten Bormann**
- We assume people have read the drafts
- Meetings serve to advance difficult issues by making good use of face-to-face communications
- Note Well: Be aware of the IPR principles, according to RFC 8179 and its updates

- Blue sheets – Automatic
- Jabber Scribe(s)
- Note Taker(s)

Note Well

Any submission to the IETF intended by the Contributor for publication as all or part of an IETF Internet-Draft or RFC and any statement made within the context of an IETF activity is considered an "IETF Contribution". Such statements include oral statements in IETF sessions, as well as written and electronic communications made at any time or place, which are addressed to:

- **The IETF plenary session**
- **The IESG, or any member thereof on behalf of the IESG**
- **Any IETF mailing list, including the IETF list itself, any working group or design team list, or any other list functioning under IETF auspices**
- **Any IETF working group or portion thereof**
- **Any Birds of a Feather (BOF) session**
- **The IAB or any member thereof on behalf of the IAB**
- **The RFC Editor or the Internet-Drafts function**

All IETF Contributions are subject to the rules of RFC 5378 and RFC 8179.

Statements made outside of an IETF session, mailing list or other function, that are clearly not intended to be input to an IETF activity, group or function, are not IETF Contributions in the context of this notice. Please consult RFC 5378 and RFC 8179 for details.

A participant in any IETF activity is deemed to accept all IETF rules of process, as documented in Best Current Practices RFCs and IESG Statements.

A participant in any IETF activity acknowledges that written, audio and video records of meetings may be made and may be available to the public.

https://www.ietf.org/about/note-well/

I E T F

Practicalities

- Use the queue request on Meetecho
- Use of queuing at core@jabber.ietf.org
  - mic: to ask for relaying a question
- This meeting is recorded
- Bluesheets are automatically filled

All times are in UTC

Friday (120 min)

- 12:00–12:05 Intro, Agenda
- 12:05–12:20 CORECONF Comi
- 12:20–12:30 SenML Versions
- 12:30–12:40 SenML Data CT
- 12:40–12:55 New Block
- 12:55–13:10 Dynlink
- 13:10–13:25 RD Extensions and protocol negotiations
- 13:25–13:35 Data Centric deployment for CoAP
- 13:35–14:00 Flextime

Agenda Bashing

CORECONF Comi

SenML

New Block

Dynlink

RD Extensions and protocol negotiations

Data Centric deployments for CoAP

Flextime

Thank you!
Comments/questions?

I E T F

CORECONF status

Carsten Bormann, CoRE@IETF110, 2021-03-12

The four CORECONF documents

- Yang-cbor, Sid: In IETF Last Call (ends 2021-03-17), reviews TBD except:
  - secdir/yang-cbor (Shawn Emery): No security cons, and that is alright so.
- Comi, Yang-library: In WG, waiting for shepherd writeup (yours truly)
  - Comi: shepherd's comments, waiting for WG input and resolution
  - Yang-library: next

Core-sid comment

- Ben Kaduk (Security AD) upon the start of the last-call (well-observed):
  - This appears (on quick skim) to be a proposal to create a registry and assign globally unique uint64 values to items in (unrelated) YANG modules.
  - In a sense this would be a **new global naming system** and as such might have some unexpected properties. Please send comments to last-call@ as usual, if you take a look.
- This is indeed the intention, and core-sid has been carefully designed to fill this role. Looking forward to more security thinking about this!

Comi comments and issues (1)

- Issue in netmod: Type equivalence
  - Comi currently assumes e.g. uint8 is incompatible (NBC) with int8 by assigning different encodings in URI
  - This incompatibility seems to be held up in current netmod discussion
  - Still, do we need to **rely** on it?
  - No rationale for inconsistencies given in draft

Comi URI encoding of YANG data

YANG datatype	Uri-Query text content
uint8,uint16,unit32, uint64	int2str(key)
int8, int16,int32, int64	urlSafeBase64(CBORencode(key))
decimal64	urlSafeBase64(CBOR key)
string	key
boolean	"0" or "1"
enumeration	int2str(key)
bits	urlSafeBase64(CBORencode(key))
binary	urlSafeBase64(key)
identityref	int2str(key)
union	urlSafeBase64(CBORencode(key))
instance-identifier	urlSafeBase64(CBORencode(key))

- String may not be URL-safe
- Why uint≠int
  - Identityref, enumeration probably OK
- What is “CBOR key”?
- key vs. CBORencode(key)

Comi comments and issues (2)

- Text about block-wise transfers (§ 5) does not really add anything
  - All that is probably needed is a short mention of the need for attention
- Making pagination a SHOULD for /.well-known/core needs to indicate:
  - How this is done (it is defined in CoRE RD § 6, but **not** for /wkc)
  - When can that SHOULD be violated?
  - (Note that there is a general YANG problem here, too; https://tools.ietf.org/html/draft-wwlh-netconf-list-pagination-nc-01 ?)

Comi comments and issues (3)

Nits

- SID encoding: 0 ➔ A (and not the empty string)
- New yang module version needed after update of documentation reference from RFC 7049 to RFC 8949?
- 4.01 vs. 4.03 confusion

SenML etc.

Carsten Bormann, CoRE@IETF110, 2021-03-12

draft-ietf-core-senml-versions-02

- 2021-02-21: –02 — addressing Jaime's comments
  - More explanation about the approach bitmapping a number
  - Updated references to include C++20 (for `0b…`) and I-JSON (53-bit)
  - Added discussion of an allocation rate limit of ≦ 2 codes per year
- Implementation in SenML validator (Thanks, Ari):

```
$ curl -X POST http://wishi.nomadiclab.com:8086/senml-validate -d '[{"bver": 26, "v":42,"bn":"foo","bu":"km/h"}]'
Valid Pack.
$ curl -X POST http://wishi.nomadiclab.com:8086/senml-validate -d '[{"bver": 10, "v":42,"bn":"foo","bu":"km/h"}]'
Invalid Pack. Error: Using additional units without declaring feature in bver field
```

- Ready to go!

draft-ietf-core-senml-data-ct-03

- –03 (2021-01-15): A number of editorial clarifications, mention “decompression bombs” in the security considerations
- Done, but waiting for draft-bormann-core-media-content-type-format
  - Was on agenda at DISPATCH meeting on Monday
  - Looks like this will take a while
- ➔ remove normative reference, copy over the ABNF, and call it a day.

New CoAP Block-Wise Transfer Options For Faster Transmission

draft-ietf-core-new-block-07

IETF CoRE Meeting, 12th Mar 2021

Mohamed Boucadair

Jon Shallow

Agenda

- Changes since last Meeting
- Implementation Status
- Next Steps

Updates post -07

In response to Christian's recent comments

- Removed CSM Option
- Added disadvantage of not being able to mix CON/NON in same request / response
- Clarity over peers with different MAX_PAYLOADS

Updates -02 to -07 (1 of 2)

- Gone for naming Q-Block1 and Q-Block2
- Updated following WG Last Call reviews
  - Thanks Marco and Christian
- Additional disadvantages added
- Emphasis on Non-confirmable support
- Require use of Confirmable when testing for Q-Block
- Addition of Q-Block-Wise-Transfer CSM
  - Now dropped
- Clarity on mixing Q-Blockx and Blockx with OSCORE
- Clarity on use of ‘M’ bit in requests

Updates -02 to -07 (2 of 2)

- (redundant) Request-Tag removed from 4.08 response
- Congestion Control updated with new variables defined
  - Support for 2.31 Continue response (Q-Block1)
  - Support for ‘Continue’ Q-Block2 request
    - Special case ‘M’ bit
- Examples updated and added to
- Security considerations updated to include OSCORE
- RFC7049 -> RFC8949

Implementation Status

- Libcoap https://github.com/obgm/libcoap
- Q-Block PR for libcoap submitted, not yet merged
  - https://github.com/obgm/libcoap/pull/611
- No issues found in working DOTS environment

Next Steps

- We do believe that all the issues were considered and adequately handled
- We request publication of document

Thank You

draft-ietf-core-dynlink

IETF 110 CoRE

Dynlink developments

- Current draft is at version -13
- Continuing to incorporating feedback received for updates, corrections and clarifications

Editorial Changes

- Separation of Conditional Attributes into Conditional Notification Attributes and Conditional Control Attributes

Attribute	Parameter	Value
Greater Than	gt	xs:decimal
Less Than	lt	xs:decimal
Change Step	st	xs:decimal (>0)
Notification Band	band	xs:boolean
Edge	edge	xs:boolean

Table 1: Conditional Notification Attributes

Attribute	Parameter	Value
Minimum Period (s)	pmin	xs:decimal (>0)
Maximum Period (s)	pmax	xs:decimal (>0)
Minimum Evaluation Period (s)	epmin	xs:decimal (>0)
Maximum Evaluation Period (s)	epmax	xs:decimal (>0)
Confirmable Notification	con	xs:boolean

Table 2: Conditional Control Attributes

”Edge” conditional notification attribute

- Boolean attribute to indicate interest for receiving notifications of either the falling edge or the rising edge transition of a boolean resource state.
- When the value of the Edge attribute is 0, the server notifies the client each time a resource state changes from True to False.
- When the value of the Edge attribute is 1, the server notifies the client each time a resource state changes from False to True.

”Con” Conditional control attribute

- Boolean attribute
- When present with a value of 1 in a query, the con attribute indicates a notification MUST be confirmable
- When present with a value of 0 in a query, the con attribute indicates a notification can be confirmable or non-confirmable

Question to WG

- Right now, ”band” and “con” are defined to have Boolean type values
- Do we need to specify these query parameters as (key, value) pairs or is it enough to just to include these parameters into a query component as just values?


```
query       = *( pchar / "/" / "?" )
```

Implementation Considerations

- pmin == pmax now enabled, to allow the server to send notifications to clients at every *N* seconds
- Implementation considerations provides text that this is performed as a best effort service from the server

Changes being made from -13 to -latest

- Impact on behaviour from the possible presence of (multiple) proxies is still on TODO list
- More examples to be included in the appendices
- Section 3.3 contains reference code for server processing of Conditional Attributes
  - Include also a state machine to describe server-side processing, for example with epmin and epmax

draft-ietf-core-dynlink

Thank you!

Resource Directory Extensions

draft-amsuess-core-resource-directory-extensions-05
(and draft-amsuess-core-transport-indication-00)

Christian Amsüss

2021-03-12

Inside resource-directory-extensions

Exercising RD extension points

Reverse Proxy "Please give me a public address"

Infinite Lifetime to allow CTs to never come back

Zone identifier introspection to peek beyond the split horizon

Multicast aggregation Proxy poses as lookup interface, doing multicast discovery on demand

Opportunistic RD including soft hand-over

Reverse Proxy

simplistic TURN for CoAP

Discoverable version of existing practice (LwM2M)

`POST /rd?ep=timeserver` leading to lookup
`<coap://[2a02:b18:c13b:8010:8628:563f:7c92:13bc]/t>`

`POST /rd?ep=timeserver&proxy=on`
`<coap://timeserver.proxy.rd.coap.example.com/t>`

Valuable in IETF110 without VPN

Reverse Proxy: Downside, mitigation and difficulties

- Wild URI aliasing
  - Being a forward proxy
    - Discovery?

Proxy discovery in `transport-indication-00`

- Declare that there is a proxy

```
</res>,<coap+tcp://h1.example.com>;
    rel=has-proxy;anchor="/"
```

"CoAP-over-TCP on h1.example.com is a usable proxy for all resources hosted on /"

- No URI aliasing in the application (only on the wire if `has-unique-proxy` is set; then Uri-Host / Proxy-Scheme can be elided)

Next steps

- Talk about it
- Tune relationship semantics and directions to be practical
- Adjust and try out next-gen RD proxying that uses it

A Data-centric Deployment Option for CoAP

draft-gundogan-core-icncoap–00

CoRE WG @ IETF 110

Cenk Gündoğan[1] Christian Amsüss[2]

Thomas Schmidt[1] Matthias Wählisch[3]

[1]HAW Hamburg

[2]Unaffiliated

[3]FU Berlin

March 12, 2021

cenk.guendogan@haw-hamburg.de

Information-Centric Networking (ICN)

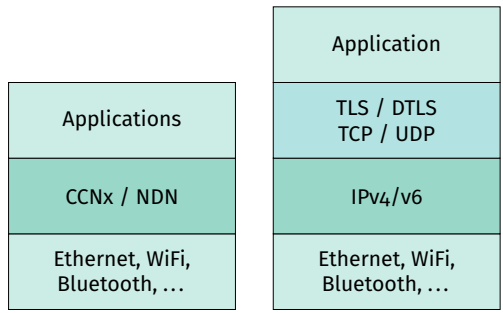
- Alternative networking paradigm
- Specialization on content delivery
- Loose coupling of data and host endpoints

Prominent architectures

- Named-Data Networking (NDN)
- Content-Centric Networking (CCNx)

Protocol features

- Name-based, stateful forwarding
- In-network content caching
- Content object security



Research indicates: promising candidate for IoT deployments

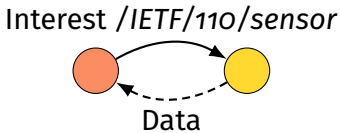
Benefits of Information-centric Properties for the IoT

- **Stateful forwarding** and **caching** shorten request paths and reduce link traversals on retransmissions
- **Content object security** enables end-to-end security and reduces session management complexity

Technical Aspects of NDN / CCNx

Communication Model

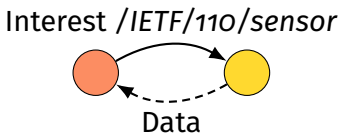
- Request–response paradigm
- Layer 3 primitives: Interest & Data



Technical Aspects of NDN / CCNx

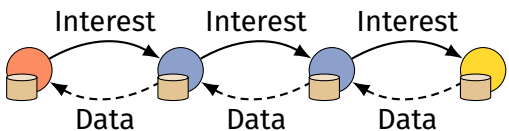
Communication Model

- Request–response paradigm
- Layer 3 primitives: Interest & Data



Forwarding & Flow Control

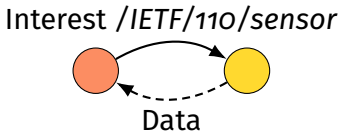
- Request state on each hop
- Hop-wise caching & retransmissions



Technical Aspects of NDN / CCNx

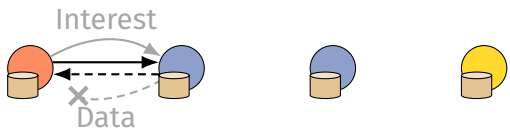
Communication Model

- Request–response paradigm
- Layer 3 primitives: Interest & Data



Forwarding & Flow Control

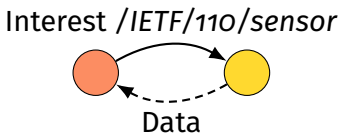
- Request state on each hop
- Hop-wise caching & retransmissions



Technical Aspects of NDN / CCNx

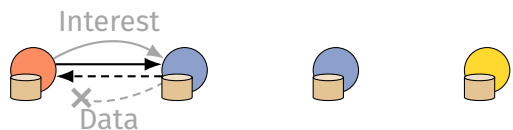
Communication Model

- Request–response paradigm
- Layer 3 primitives: Interest & Data



Forwarding & Flow Control

- Request state on each hop
- Hop-wise caching & retransmissions



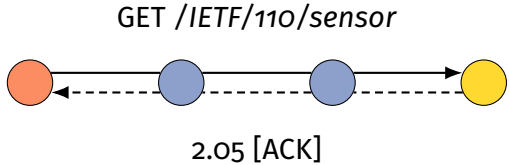
Content Object Security

- Autonomously verifiable data packets using HMAC or digital signatures
- End-to-end protection beyond untrusted gateways

Constructing a Data-centric CoAP Deployment

Standard deployment

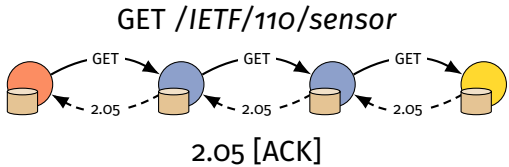
CoAP client / server + IPv6 forwarders
End-to-end retransmissions



Data-centric deployment

CoAP client / server + CoAP proxies
Hop-by-hop request state
Hop-wise caching & retransmissions
Forwarding decision on names

bonus: link-local IPv6 addresses for better 6LoWPAN compressibility

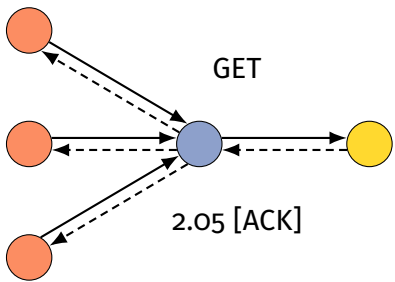


[ACM ICN'20] Toward a RESTful Information-Centric Web of Things [...]

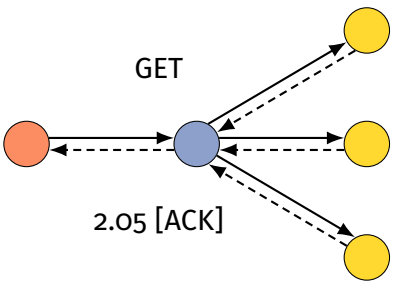
Multi-party Communication

- CCNx / NDN have integral support for multi-party communication
- Data-centric CoAP deployments inherit the same feature set

Request aggregation & Response fan-out



Request fan-out & Response deduplication



Conclusion & Outlook

Takeaways

- Improved network resiliency & reduced latency
- Location independence of content & mobility support
- Efficient multi-party communication
- New perspective for CoAP deployments

Future Work

- Dynamic proxy discovery