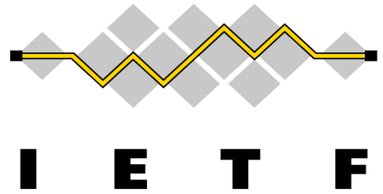
IETF 109

Constrained RESTful Environments WG (core)

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

Mailing list:
core@ietf.org

Jabber:
core@jabber.ietf.org



- 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

Tuesday (120 min)

- 05:00–05:10 Intro, Agenda, Document status
- 05:10–05:20 Resource Directory
- 05:20–05:30 Stateless
- 05:30–06:35 Group Communication
- 06:35–06:45 OSCORE with EDHOC
- 06:45–07:00 Flextime

All times are in UTC

Friday (120 min)

- 09:00–09:05 Intro, Agenda
- 09:05–09:10 SenML Versions
- 09:10–09:20 SenML Data CT
- 09:20–09:35 New Block
- 09:35–09:45 Dynlink
- 09:45–09:55 Fasor
- 09:55–10:10 CoAP over GATT
- 10:10–10:30 Rekeying for OSCORE
- 10:30–11:00 Flextime

Agenda Bashing

Document status

IESG Processing

- draft-ietf-core-resource-directory-26
  - In IESG Evaluation::AD Followup
  - Processed comments from IESG reviews

- draft-ietf-core-stateless-07
  - In IESG Evaluation::AD Followup
  - Processed comments from IESG reviews

In Post-WGLC processing

- draft-ietf-core-dev-urn-08
  - On AD Evaluation::AD Followup

- draft-ietf-core-echo-request-tag-11
  - On AD Evaluation::AD Followup

In Post-WGLC processing

- draft-ietf-core-yang-cbor-13
- draft-ietf-core-comi-10
- draft-ietf-core-sid-14
- draft-ietf-core-yang-library-02

Waiting for Shepherd Write-Up

In Post-WGLC processing

- draft-ietf-core-oscore-groupcomm-10
  - Processed WGLC comments
  - New review comments to process

Resource Directory

Stateless

Group communication

OSCORE with EDHOC

Flextime

Thank you!
Comments/questions?

I E T F

Resource Directory

draft-ietf-core-resource-directory

Christian Amsüss, Zach Shelby, Michael Koster, Carsten Bormann,
Peter van der Stok

2020-11-17

Picking highlights here; full changelog is detailed.

- "First Come First Remembered": example policy; implementatons may choose as default
- Host discovery *is not* secured, path discovery *is* if needed
- Move simple registration from `/.well-known/core` to `/.well-known/rd`
- More care for `rt` registered values

But treating them raised questions

"When DTLS is used like TLS, replay protection should be considered"

Event	Reqeust	Response
Power on	`POST /rd?ep=node1`	`/reg/reg1`
Power off	`DELETE /reg/1`[1]	Deleted
Power on	`POST /rd?ep=node1`	`/reg/reg1`
Replayed message	`DELETE /reg/1`	on but gone

Works with and without replay protection, even in OSCORE

[1] sniff sniff

Step to be taken

Server ensures freshness using Echo.

Mechanism already referenced for Simple Registration and amplification mitigation.

WIP text is in #291[2].

[2] See there for explored alternatives.

Server authorization

Q: “How much can you trust links from the RD?”

A: “As far as this is allowed by security policies.”

Server authorization

Q: “How much can you trust links from the RD?”

A: “As far as this is allowed by security policies.”

Intention $\rightarrow$ Request $\rightarrow$ Effect / Response

Server authorization

Q: "How much can you trust links from the RD?"

A: "As far as this is allowed by security policies."

Create a topic $\xrightarrow{\texttt{rt=topics}}$ $\underbrace{\texttt{POST /tpcs} \rightarrow \text{Topic gets created}}_{\text{security layers}}$

Trust an authorized RD that checks, or verify before use.

And this can be way more powerful with forms, or when the RS sets the ACE scope.

Possibilities to be explored

on the long term – outside RD

- Single-use servers: "If it's on this host, it's the PubSub server." Security contexts multiply.
- Deposited verifiable statments in the RD. Granularity? Audience?
- Finer grained authorization: "...is authorized to provide the PubSub service at its `/tpcs` resource." Linking to requests without multiplying contexts?

And by the way, your examples are needlessly large

Yes. RFC6690, ambiguity, different readings...

And by the way, your examples are needlessly large

Yes. RFC6690, ambiguity, different readings...

...but turned out mine was the worst.

- In many resource lookup results, `anchor` is superfluous.
- TBD: Give precise rules when anchor must still be set, rewrite examples.
- Existing RD servers stay compliant (just verbose).
- Pending recheck of the RFC6690 implementations.

Outlook

-27 around start of December:

Freshness on registration updates.

Relaxed anchor setting.

Continued exploration of protecting intention?

draft-ietf-core-stateless

Has been stuck since May on review comments
Eleven (11) issues identified that had not been resolved in some way.
https://github.com/core-wg/stateless/issues
Pull requests for four:
https://github.com/core-wg/stateless/pulls

https://www.rfc-editor.org/cluster_info.php?cid=C310 has 11 ROLL/6tisch/6lo IDs waiting on core-stateless
(and two ROLL documents, which are “done”)

Issues and suggestion resolution

We have done a “WONTFIX” on:

#3 is stateless updating 7252 on distinguishing unrecognized vs invalid extension?
https://github.com/core-wg/stateless/issues/3

#4 how does freshness window of client/intermediate interact?
https://github.com/core-wg/stateless/issues/4

#6 can larger tokens fill responder memory?
https://github.com/core-wg/stateless/issues/6

#7 how to size the replay window?
https://github.com/core-wg/stateless/issues/7

#9 look ma, no state!
https://github.com/core-wg/stateless/issues/9

https://mailarchive.ietf.org/arch/msg/core/Y8c0mSR6THFP97JIyQqllLFF3Ok

Issues and suggestion resolution

We have replaced confusing I-D text here:

#8 use automated key management due to AES-CCM/BCP107

https://github.com/core-wg/stateless/issues/8

https://github.com/core-wg/stateless/pull/11

Too long for the slide.

https://mailarchive.ietf.org/arch/msg/core/Y8c0mSR6THFP97JIyQqllLFF3Ok

Issues and suggestion resolution

We have generated better I-D text for:

#10 60 minutes for address change

https://github.com/core-wg/stateless/issues/10

https://github.com/core-wg/stateless/pull/12

#5 lack of integrity protection results in spoofed responses

https://github.com/core-wg/stateless/issues/5

https://github.com/core-wg/stateless/pull/13

https://mailarchive.ietf.org/arch/msg/core/Y8c0mSR6THFP97JIyQqllLFF3Ok

Document Status

- draft-ietf-core-stateless-07 uploaded on 2020-11-02
- Reply to AD at: https://mailarchive.ietf.org/arch/msg/core/fCXVcloF8j0R0tHKSPXbwhm_nRk/
  - Seems that Ben has replied today, and cleared the DISCUSS. There may be an additional cycle on getting the wording right

Group Communication for the Constrained Application Protocol (CoAP)

draft-ietf-core-groupcomm-bis-02

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

IETF 109 - CoRE WG, November 17, 2020

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 …)
  - Unsecured CoAP or group-OSCORE-secured communication
  - Principles for secure group configuration
  - Use cases (appendix)

Overview of -02 updates

- Clarify messaging/endpoint model – server may respond from different UDP port #1

- Consistency requirement for response suppression updated – now based on response code class #2

- 'Group definition' 2.1 expanded into subsections; relations between group types detailed.
  - Encoding application group using 'Uri-Host' Option added #3
  - Best practices for application group inclusion in request added

Overview of -02 updates

- Included FETCH method in many places, where applicable. Also in Observe 2.3.5 and Block-Wise 2.3.6.

- Various enhancements & editorial
  - A few more Observe re-registration details
  - ...

Server response from different UDP port

› Issue #1, now closed

Next steps

- Move handling of multiple CoAP responses from CoAP layer to the application layer, in 2.3.1
  - based on Interop experience

- Extend proxy operation 2.3.3 with caching of responses
  - explore caching scenarios – can we suppress the sending of multicast request in certain cases? Can Proxy do cache-refresh?

Next steps

- More reviews would be good!
  - Promised @IETF 108: Christian, Francesca

- Test selected functions in CoAP implementations
  - E.g. “Observe + multicast” extension of RFC 7641
  - Report results

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-10

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

IETF 109, CoRE WG, November 17th, 2020

Update since the July meeting

- Version -10 submitted before the cut-off
  - Addressed WGLC comments [1][2]
  - Addressed more points discussed around IETF 108
- 3rd interop during this Hackathon
  - Rikard Höglund, Peter van der Stok, Christian Amsüss
  - The pairwise mode was also successfully tested
- New review of -10 from Christian [3] – Thanks!

[1] https://mailarchive.ietf.org/arch/msg/core/VMhrAPEt4TE8jahatVd1EoDzdMI/
[2] https://mailarchive.ietf.org/arch/msg/core/tOHaMpTrWJ2CfsX2E5IGS8qpt-U/
[3] https://mailarchive.ietf.org/arch/msg/core/pXEyxhbf-s2wgGDzrDhUNPsHZZc/

Main updates in -10

- Common Security Context
  - Removed “Counter Signature Key Parameters”
  - Added parameters for the pairwise mode

- A server may respond with 5.03
  - Not having the public key of the client yet
  - Not possible to retrieve it right away

- Non-recycling policies for the Group Manager
  - Don’t reassign the same Sender ID in the same group
    - Open point about slightly relaxing it
  - Don’t reassign the same Group ID to the same group

Context Component	New Information Elements
Common Context	Counter Signature Algorithm Counter Signature Parameters *Secret Derivation Algorithm *Secret Derivation Parameters
Sender Context	Endpoint’s own private key *Pairwise Sender Keys for the other endpoints
Each Recipient Context	Public key of the other endpoint *Pairwise Recipient Key of the other endpoint

Main updates in -10

- Sender Sequence Number (SSN)
  - Keep one shared space, for group mode and pairwise mode
  - Reset to 0 when establishing a new context
    - Got a new Sender ID; or whole group rekeying

- Request protected with Ctx_old , response protected with Ctx_new
  - The server MUST use its SSN as Partial IV of that response

- Added ‘request_kid_context’ to the external_aad
  - Support observations beyond a group rekeying
  - Required now that the SSN is reset upon rekeying
  - A notification can’t match with 2 registration requests

```
external_aad = bstr .cbor aad_array

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

Figure 2: external_aad for Encryption

Main updates in -10

- More on supporting Observation

- The **client and server** store the ‘kid’ and ‘kid context’ from the registration request
  - Used to correctly build the external_aad of notifications

- The **client** stores ‘kid’ and ‘kid context’ from the registration request
  - Only if actually interested in continuing the observation beyond a group rekeying

- The **client** stores an invariant identifier of the group
  - Unchanged over group rekeyings, e.g. the “group name” of *ace-key-groupcomm-oscore*
  - Simpler to get updated key material from the Group Manager, if a rekeying was missed
  - Only if actually interested in continuing the observation beyond a group rekeying

From Christian's review

- Improve distinction between anti-replay and freshness
  - Clarify server "synchronization" with a client, as related to freshness
- Methods in Appendix E
  - E.1 "Best effort" and E.2 "Baseline" are not significant and can be removed
  - E.3 using Echo makes a Replay Window valid and brings freshness
- More reasons to lose part of the Security Context
  - 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 starts with an invalid Replay Window
- Get rekeyed by the Group Manager or run Echo (achieving also freshness)

From Christian's review

- Relax non-recycling of Sender IDs in the same group
  - Now: never-ever recycle → eventually leads to large KID sizes, with no way back
  - Proposal: never recycle under the same GID value. Issues with that?

- Converge to a single external_aad format ?
  - We have added 'request_kid_context'
  - Now both external_aad structures deviate from RFC 8613 anyway

aad_array for encryption [
  oscore_version,
  algorithms,
  request_kid,
  request_piv,
  options,
  request_kid_context
]

aad_array for signing [
  oscore_version,
  algorithms,
  request_kid,
  request_piv,
  options,
  request_kid_context,
  OSCORE_option
]

aad_array [
  oscore_version,
  algorithms,
  request_kid,
  request_piv,
  options,
  request_kid_context,
  OSCORE_option
]

From Christian's review

- More on the external_aad

```
external_aad = bstr .cbor aad_array

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

Figure 2: external_aad for Encryption

- Can we remove 'par_countersign_key' ?
  - It's repeating what in 'par_countersign'
  - Redundancy removed from the Common Context

- Can we further generalize 'par_countersign' ?
  - Today, algorithms have only "Key Type" as capability
  - COSE admits algorithms with 0 or 2+ capabilities
  - Possible future-friendly format

par_countersign [
  countersign_alg_capab [C1, C2, …, CN],
  countersign_C1_capab [C1, …],
  countersign_C2_capab [C2, …],
  …
  countersign_CN_capab [CN, …]
]

Summary and next steps

- Addressed comments from WGLC and IETF 108

- Successful tests at the Hackathon
  - Message exchange in group mode and pairwise mode

- Next steps
  - Submit version -11 addressing Christian's review
  - More interop tests, covering also error cases

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-07

Marco Tiloca, RISE
Christian Amsüss
Peter van der Stok

IETF 109, CoRE WG, November 17th, 2020

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 -07

- Addressed review of -06 at [1]
- Closed open points raised at IETF 108
- Retrieval for the link to the ACE AS, if also registered
  - Need for a separate lookup (simpler if/when using CoRAL queries)
  - Shown in later example
- Rewording about Link-Format as not typed (-2 vs. “-2”)
  - Limitation for target attributes indicating algorithms
  - Strings that look like integers are not supported
  - Not an issue with current registered algorithms

[1] https://mailarchive.ietf.org/arch/msg/core/3M-ASJxDvrMrSi26-Jk4tI3cmOs/

Updates from -07

- Alignment with other documents
  - rt="core.osc.gm": Moved to *draft-ace-key-groupcomm-oscore*
  - if="ace.group": Used from *draft-ace-key-groupcomm*
  - *"OSCORE groups" renamed as "Security groups"*
- All examples in Link-Format and CoRAL revised accordingly

Updates from -07

Registration

```
Request: GM -> RD

Req: POST coap://rd.example.com/rd?ep=gm1
Content-Format: 40
Payload:
</ace-group/feedca570000>;ct=41;rt="core.osc.gm";if="ace.group";
                            sec-gp="feedca570000";app-gp="group1";
                            cs_alg="-8";cs_alg_crv="6";
                            cs_key_kty="1";cs_key_crv=6;
                            cs_kenc="1",
<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
```

Example involving the link to the ACE Authorization Server

Discovery

```
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>;rt="core.osc.gm";
    if="ace.group";sec-gp="feedca570000";app-gp="group1";
    cs_alg="-8";cs_alg_crv="6";cs_key_kty="1";cs_key_crv=6";
    cs_kenc="1";anchor="coap://[2001:db8::ab]"
```

```
Request: Joining node -> RD

Req: GET coap://rd.example.com/rd-lookup/res
  ?rel=authorization-server
  &anchor=coap://[2001:db8::ab]/ace-group/feedca570000

Res: 2.05 Content
Payload:
<coap://as.example.com/token>;rel=authorization-server;...
```

Updates from -07

- Bridge with *draft-ietf-ace-oscore-gm-admin*
  - Followed a suggestion from [2] and discussed at IETF 108
  - Names of application groups specified when creating the security group at the GM
  - The GM knows those names, to use them as value of the ‘app-gp’ attribute with the RD

- Multiple security groups may be retrieved for a same application group
  - Useful when different joining nodes support different algorithms
  - A client can join any security group; a server has to join all security groups
  - More details and guidelines are in *draft-ietf-core-groupcomm-bis*

[2] https://mailarchive.ietf.org/arch/msg/core/BoYGYmEpJMUS8bk4PNHOEaFFcdU/

Summary and next steps

- Addressed review of -06 and open points from IETF 108

- Next steps
  - Add target attributes related to the pairwise mode of Group OSCORE
  - Revise the usage of ‘anchor’, based on upcoming *core-resource-directory-27*
  - Extend security considerations, based on upcoming *core-resource-directory-27*

- Plan to run first tests against Christian’s RD

- Need for more reviews

Thank you!

Comments/questions?

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

Backup

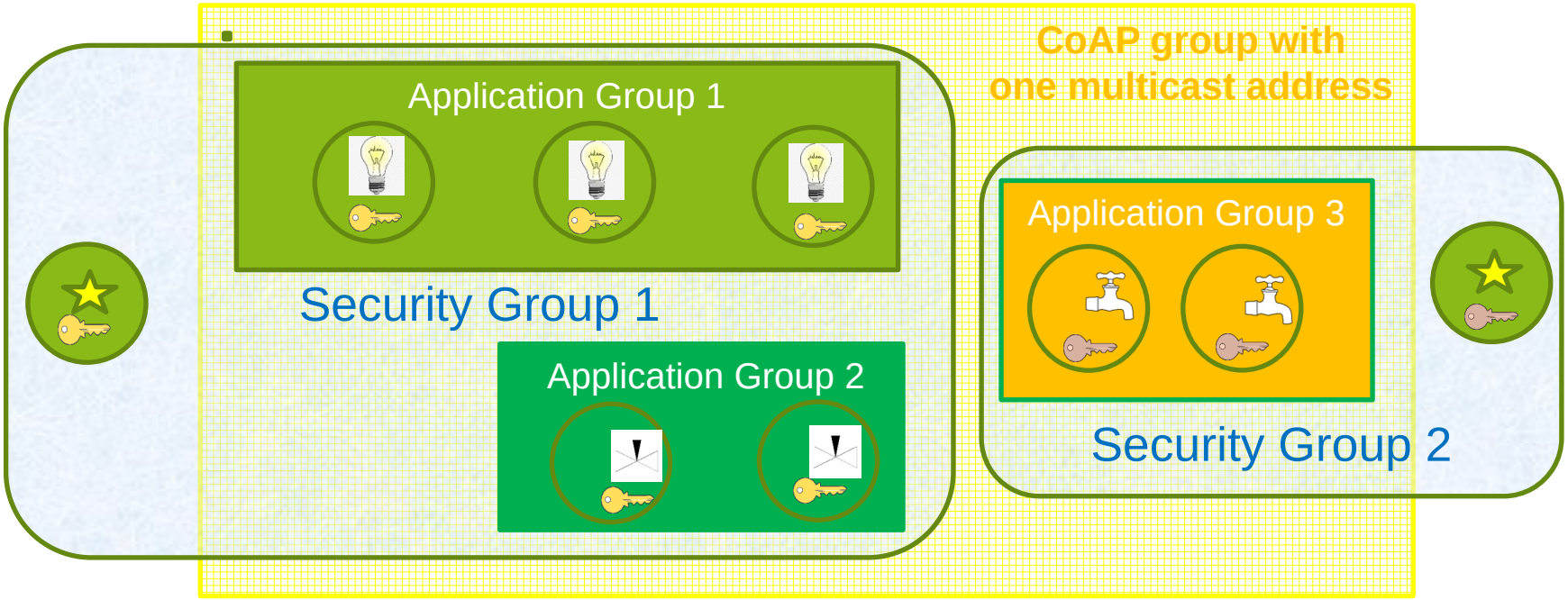
Application/CoAP/Security Groups

- 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 a 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=41;rt="core.osc.gm";if="ace.group";
                           sec-gp="feedca570000";app-gp="group1";
                           cs_alg="-8";cs_alg_crv="6";
                           cs_key_kty="1";cs_key_crv=6";
                           cs_kenc="1",
<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>;rt="core.osc.gm";
    if="ace.group";sec-gp="feedca570000";app-gp="group1";
    cs_alg="-8";cs_alg_crv="6";cs_key_kty="1";cs_key_crv=6";
    cs_kenc="1";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]"
```

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”
  - (**) *alg* : AEAD algorithm
  - (**) *hkdf* : HKDF algorithm

- 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-04

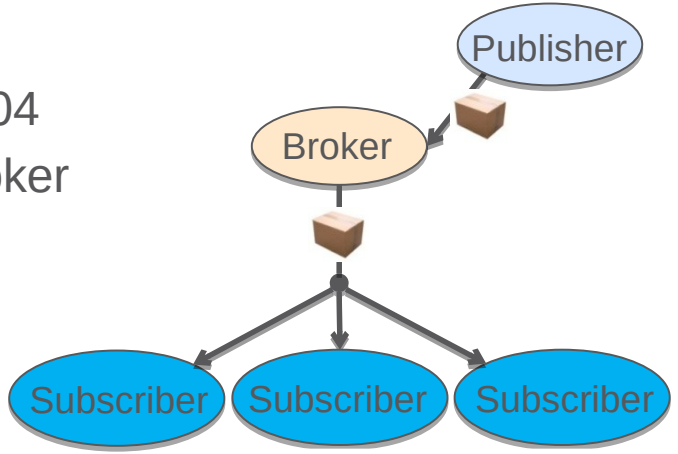
Marco Tiloca, RISE
Rikard Höglund, RISE
Christian Amsüss
Francesca Palombini, Ericsson

IETF 109, CoRE WG, November 17th, 2020

What, Why

- Observe notifications as multicast responses
  - Many clients observe the same resource on a server
  - 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 “informative response” with:
  - The serialization of the phantom request
  - The IP multicast address where notifications are sent to
  - The serialization of the latest multicast notification (i.e., the current resource status)

Updates from -04

- Improved and extensible encoding of the informative error response
  - Transport-independent information, for the phantom request and the latest notification
  - Common transport-specific information; detailed specification for CoAP over UDP

```
informative_response_payload = {
  1 => bstr,   ; phantom request (transport-independent information)
  2 => bstr,   ; latest notification (transport-independent information)
  3 => array   ; transport-specific information
}
```

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

OLD

```
Payload: { ph_req     : bstr(PH_REQ.CoAP),
           last_notif : bstr(LAST_NOTIF.CoAP)
           cl_addr    : bstr(GROUP_ADDR),
           cl_port    : GROUP_PORT,
           srv_addr   : bstr(SERVER_ADDR),
           srv_port   : SERVER_PORT,
         }
```

```
Payload: {
  ph_req     : bstr(0x01 | OPT),
  last_notif : bstr(0x25 | OPT | 0xff | PAYLOAD),
  tp_info    : [1, 0x7b, bstr(SRV_ADDR), SRV_PORT,
                bstr(GRP_ADDR), GRP_PORT]
}
```

NEW

Updates from -04

- Improved rough counting of active clients, by poll for interest
  - New CoAP option in successful multicast notifications ✉ **Now the length is max 1 byte!**

- Server current rough estimate: *COUNT*
  - *N* = max(*COUNT*, 1)
  - *M* desired confirmations
  - *L = ceil(log2(N / M))*
  - Option value: *Q* = max(*L*, *0*)
  - Each client picks a random value *I* : [0, *2^Q*)
  - If *I* == 0, the client sends a re-registration request
    - Non Confirmable; w/ No-Response; w/ the new Option having empty value
  - The server receives *R* of such requests; meanwhile, the estimate has become *COUNT '*
  - *F = R * (2^Q)* ; then *COUNT* ← *COUNT '* + *((F – N) / D)* , with *D* > 0 as dampener

No.	C	U	N	R	Name	Format	Len.	Default
TBD		x			Multicast-Response-Feedback-Divider	uint	0-1	(none)

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

- Pseudo-code in Appendix B, also optimized for constrained clients

Updates from -04

- Added support for intermediary proxies
  - 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 8)
  - The proxy can retrieve the phantom request from the informative response
  - The informative response is still forwarded back to each new client
- With end-to-end security (Section 9)
  - 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

Updates from -04

- The client sends the rebuilt Phantom request as addressed to the proxy
  - The request itself already provides the transport-specific information
- The sent request includes a new CoAP option Listen-To-Multicast-Responses
  - This provides the transport-independent information
  - Value: serialized CBOR array, i.e. ‘tp_info’ from the informative response

No.	C	U	N	R	Name	Format	Len.	Default
TBD	x	x			Listen-To-Multicast-Responses	(*)	3-1024	(none)

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

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

- Now the proxy has all it needs to receive multicast notifications
- Appendix C and Appendix D include interaction examples

Open points

- Proxy example fixes
- Negotiation: What to do when there is no common way?
- '*last_notif*' in the informative response
  - Serialization of latest sent multicast notification
  - To be made optional (like an empty ACK)

```
Payload: {
  ph_req     : bstr(0x01 | OPT),
  last_notif : bstr(0x25 | OPT | 0xff | PAYLOAD),
  tp_info    : [1, 0x7b, bstr(SRV_ADDR), SRV_PORT,
                bstr(GRP_ADDR), GRP_PORT]
}
```

- OSCORE group possibly self-managed by the server
  - The observation request might even double as joining request of a Monitor member
  - The informative response would include also key material, as in a Joining Response
  - Thoughts? Objections?

Summary

- Latest additions
  - Improved and extensible encoding of the informative error response
  - Improved rough counting of clients; added pseudo-code for constrained clients
  - Added support for intermediary proxies, with or without end-to-end security
  - Re-organization of sections and editorial improvements
- Next steps
  - Address open points
  - Investigate possible optimization with a deterministic phantom request
- Need for document reviews
  - Potential reviewers: Göran, Jaime, Carsten

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

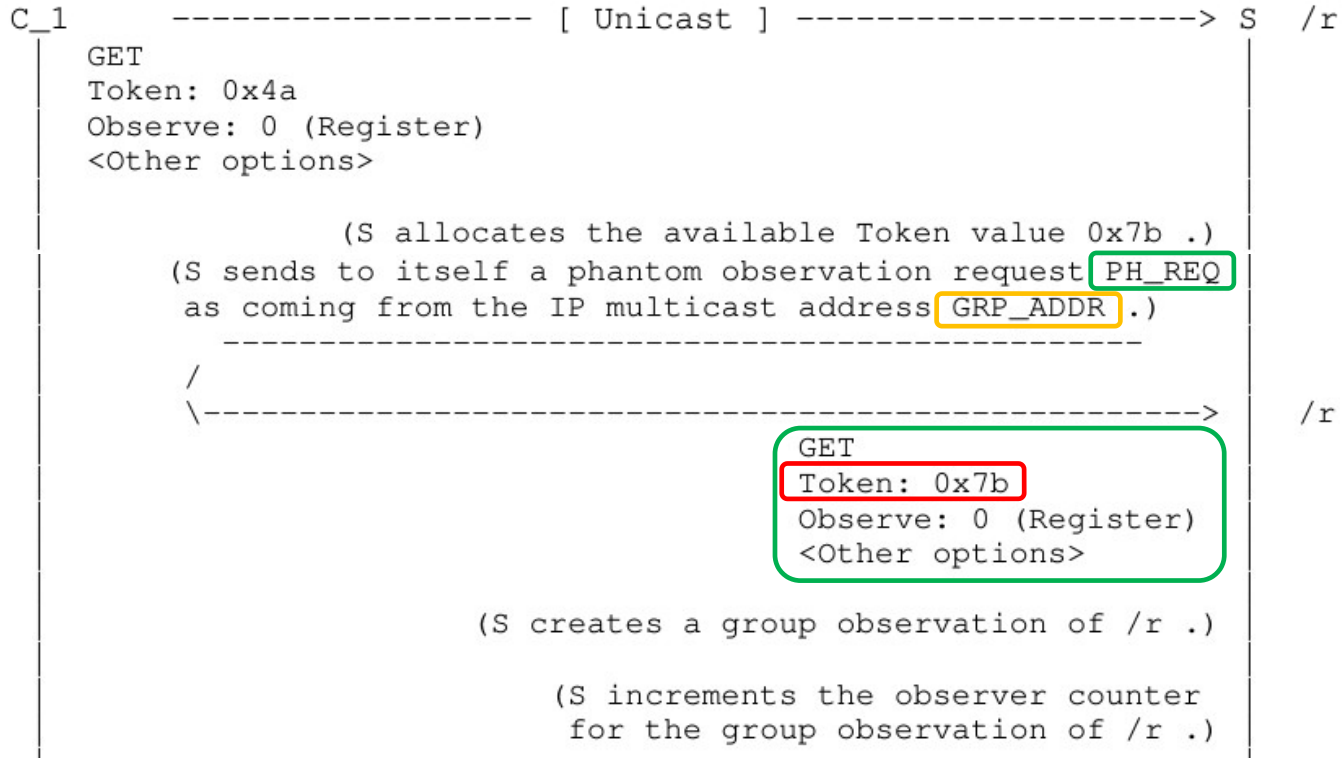
4. Hereafter, use T as token value for the group observation

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

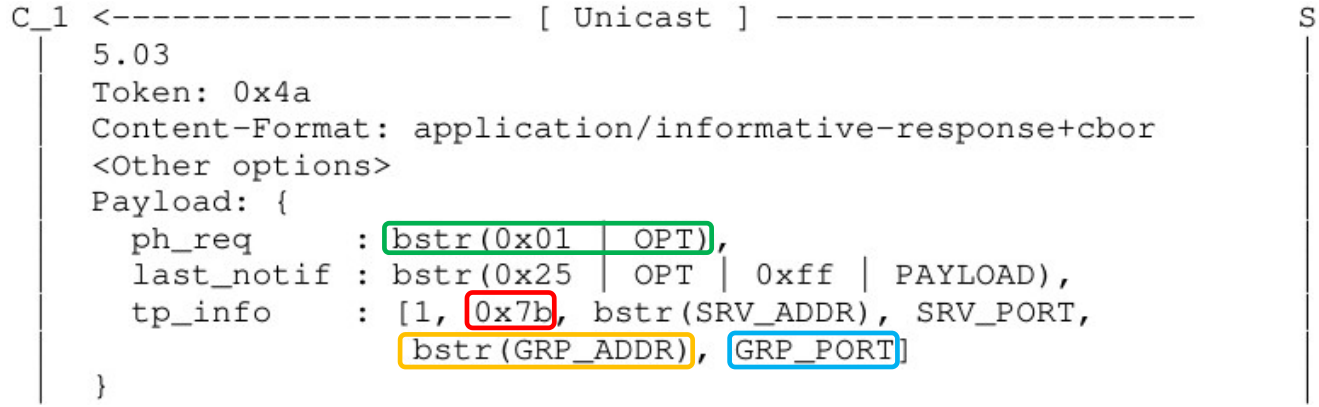
Interaction with clients

- The server sends to new/shifted clients an ***error response*** with
  - '*ph_req*': serialization of the phantom request
  - '*last_notif*': serialization of the latest sent notification for the target resource
    - '*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
    - '*srv_addr*' and '*srv_port*': destination address/port of the phantom request
- 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

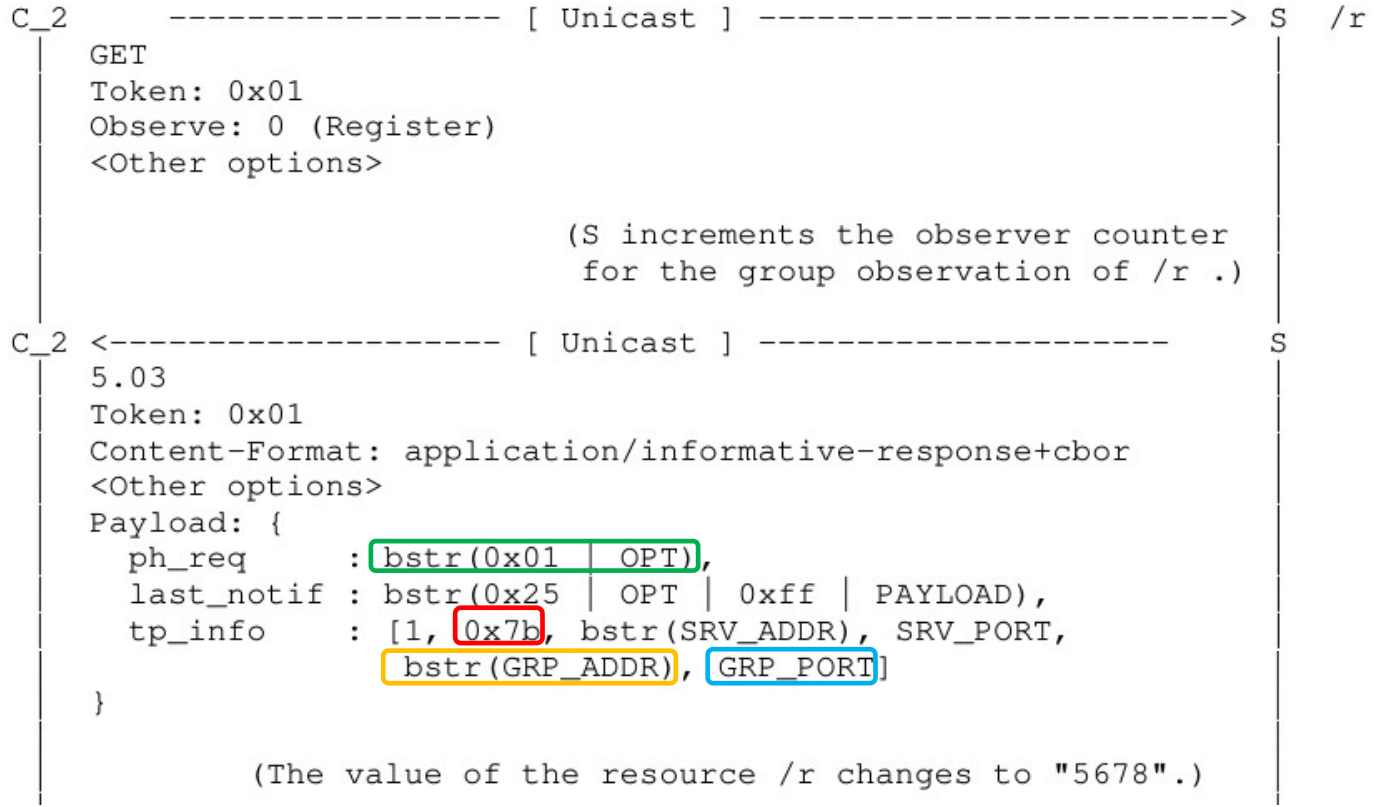
C1 registration



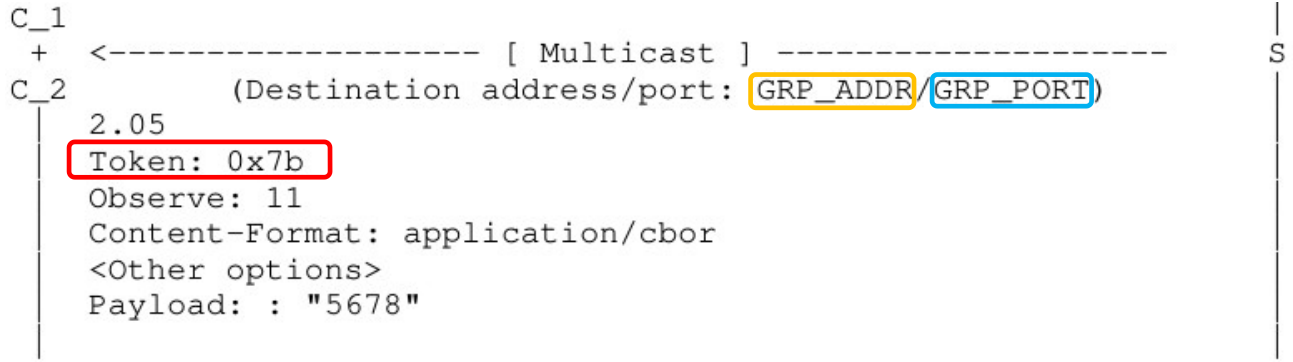
C1 registration



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

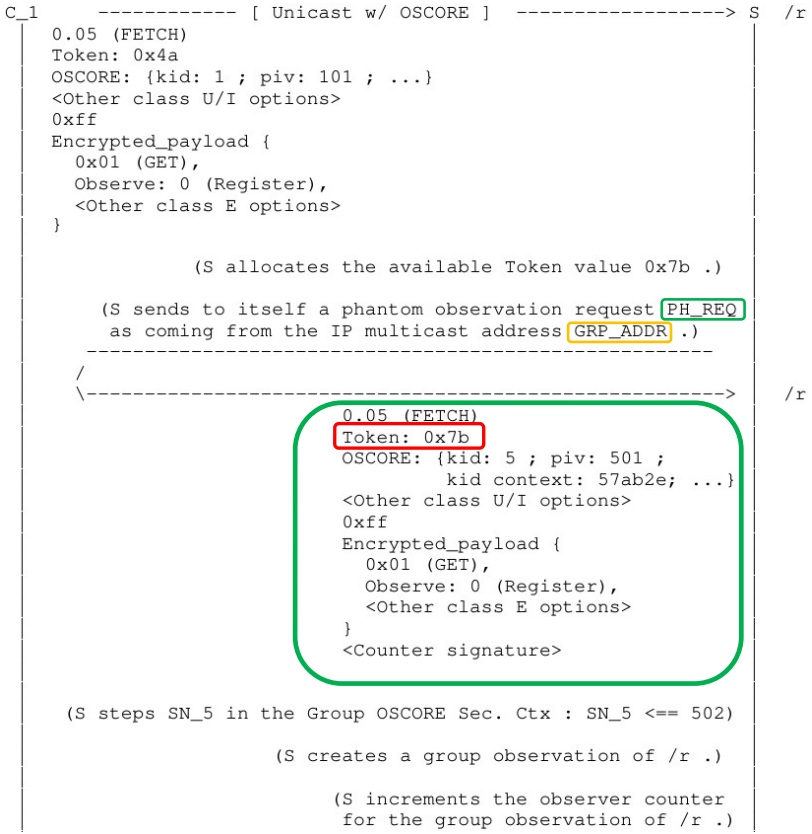
Security with Group OSCORE

- 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

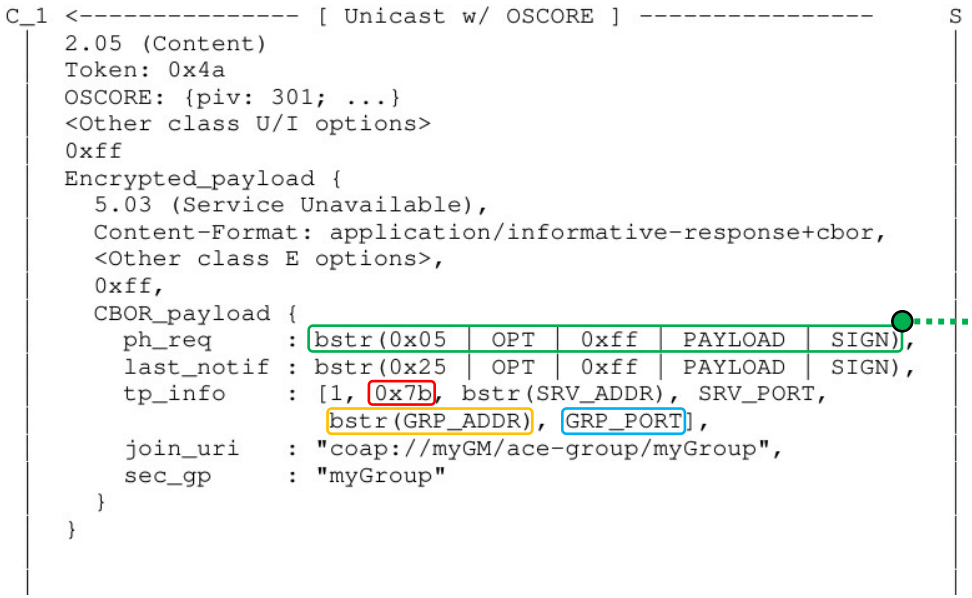
MUST

MAY

C1 registration w/ security



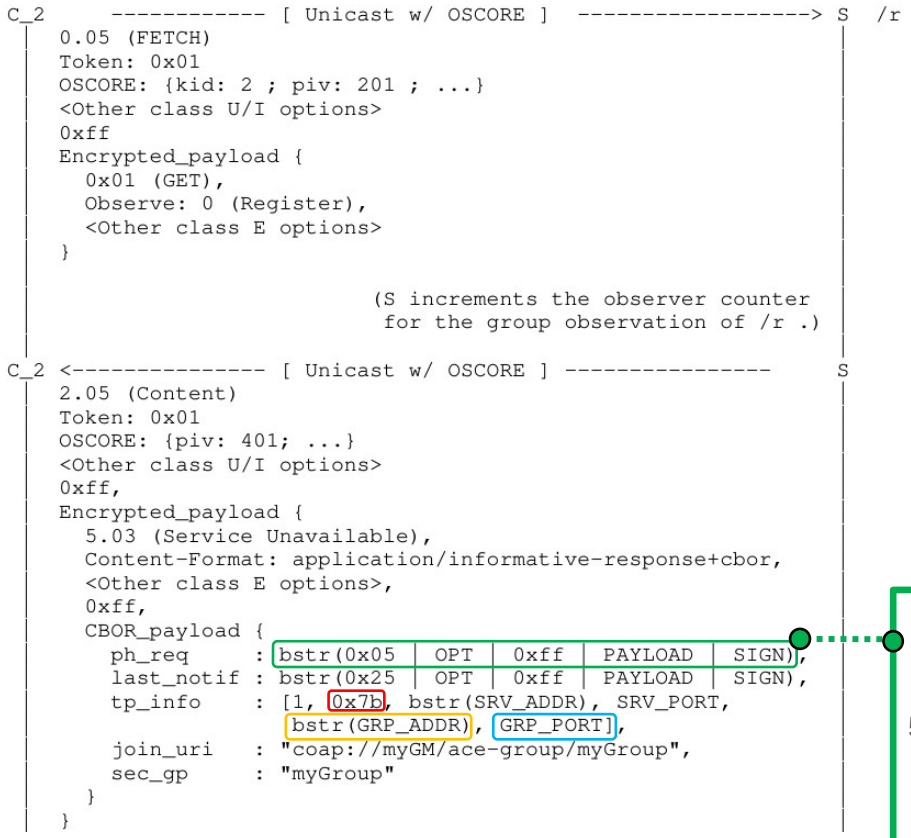
C1 registration w/ security



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

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

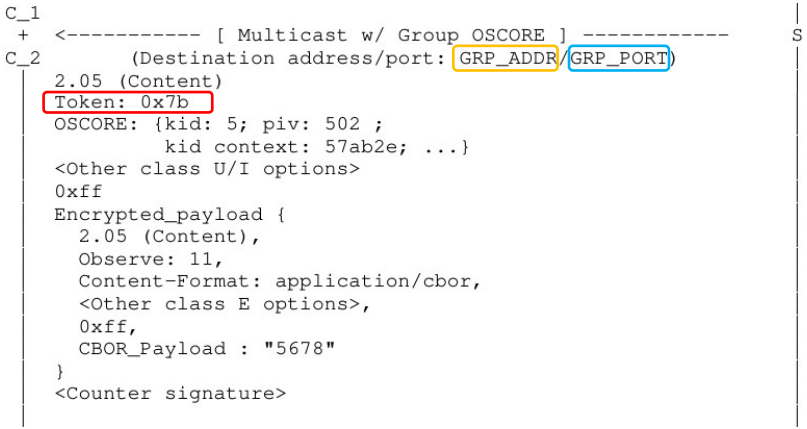
C2 registration w/ security



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>

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

Proxy Operations for CoAP Group Communication

draft-tiloca-core-groupcomm-proxy-02

Marco Tiloca, RISE
Esko Dijk, IoTconsultancy.nl

IETF 109, CoRE WG, November 17th, 2020

Rationale

- **Background** – CoAP supports group communication over IP multicast
  - *draft-ietf-core-groupcomm-bis* discusses also 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

Recap groupcomm-proxy

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

Updates from -02

- Editorial re-organization of text for Observation
  - Now as dedicated subsections, throughout the protocol workflow
  - The proxy keeps forwarding notifications back, until the observation terminates
- Revised security considerations
- Updated semantics and usage of the new CoAP options
- Added support for a chain of proxies
  - Same principles, extended through multiple hops

Multicast-Signaling option

- Only in C → P requests
  - Presence: explicit claim of support and interest from the client
  - Value: T′ s, i.e. for how long the proxy should forward back responses
  - The proxy removes the option, before forwarding the request to the servers

No.	C	U	N	R	Name	Format	Length	Default
TBD1		x	-		Multicast-Signaling	uint	0-5	(none)

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

- Now the value T′ can also be 0
  - Still ok to forward the request to the servers, no interest in proxy responses
  - SHOULD be used with the No-Response Option, with value 26

- Issues or comments?

Response-Forwarding option

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

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

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

```
srv_info = [
  srv_addr : #6.260(bstr),   ; IP address of the server
  ? srv_port : uint,         ; Port number of the server
]
```

- Address and port in the value
  - If port is omitted, assume the dst port of group URI in the Request – most common

- It used to be an absolute URI, with scheme & hostname
  - Pro: now it's a smaller option, less parsing, handy for constrained clients
  - Con: excludes scenarios where Proxy inserts DNS hostname. **Can we live with it?**

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 Option, 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
- N**on-last** proxies do **not** alter or remove the Response-Forwarding option

OSCORE between Client and Proxy

- Can co-exist with Group OSCORE between client and servers
- Can be used between each pair of hops, until the last proxy
- Some class **U** options are treated as class **E**
  - Proxy-URI, Proxy-Scheme, Uri-Host, Uri-Port
  - OSCORE, if Group OSCORE is used end-to-end
  - Multicast-Signaling and Response-Forwarding from this document
- More options may come. **Any general rule to identify them?**

OSCORE between Client and Proxy

- Proposal: process an option X as class E rather than U if:

- X is intended (also) for the recipient hop and its processing
  - E.g., Uri-Host option, Multicast-Signaling option, …

OR

- X is intended for the final endpoint, but more instances will be added as intended for the recipient hop and its processing
  - E.g., OSCORE option, when Group OSCORE is used end-to-end

- **Accurate enough? Anything simpler?**

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
  - This version adds also support for a chain of proxies

- Next steps
  - Define HTTP headers for HTTP/CoAP Cross-Proxies
  - Enable a HTTP client to talk to a CoAP group

- Need for reviews
  - Promised: Christian, Carsten, Francesca

Thank you!

Comments/questions?

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

Backup

Issues with proxies

- *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 processing at the proxy and 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

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

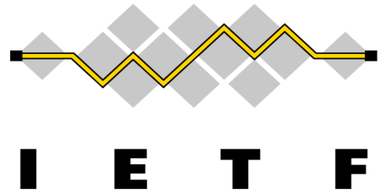
IETF 109

Constrained RESTful Environments WG (core)

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

Mailing list:
core@ietf.org

Jabber:
core@jabber.ietf.org



- 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)

- 09:00–09:05 Intro, Agenda
- 09:05–09:20 SenML (Versions + Data CT)
- 09:20–09:35 New Block
- 09:35–09:45 Dynlink
- 09:45–09:55 Fasor
- 09:55–10:05 OSCORE with EDHOC
- 10:05–10:20 CoAP over GATT
- 10:20–10:40 Rekeying for OSCORE
- 10:40–11:00 Flextime

Agenda Bashing

Document status

- draft-ietf-core-stateless-08
  - Approved as Proposed Standard!
- draft-ietf-core-dev-urn-08
  - In Last Call
- draft-ietf-core-echo-request-tag-11
  - In Last Call

SenML

New Block

Dynlink

Fasor

OSCORE with EDHOC

CoAP over GATT

Rekeying for OSCORE

Flextime

Thank you!
Comments/questions?

I E T F

SenML Features and Versions

Draft-ietf-core-senml-versions-01

Carsten Bormann

IETF 109, 2020-11-20, in the cloud

Summary of IETF108 slides

RFC 8428, SenML: Version 10
Objective: extensibility
Version numbers are stupid
Proposal: interpret version number as bits
53: wasn’t that an evil number?
draft-ietf-core-senml-versions-00
Next steps

Next steps

IETF108

- Need more reviews!
  This is just about the interpretation for one field…
- Proposal: Process these reviews, check if we are done, WGLC

- IETF108 notes:
  »People who will review: Bill, Jaime«
- —01 makes minor edits (math presentation),
  updates RFC 8798 reference

IETF109

- **So, how about some reviews?**
- **And, how about implementations?**

Backup slides

RFC 8428, SenML: Version 10

- RFC 8428 SenML evolution path: allows for version upgrade
- Default version: 10 (accounting for previous development versions)
- Can set higher: [{**“bver":11**, “v”:4711}, …]
- Semantics to be defined by RFC updating RFC 8428

Objective: extensibility

- Over time, new specifications will add features to SenML
- Version number is a unitary declaration:
  implementation of certain features is needed by the receiver to process SenML pack

- Version number N+1 includes all features of version number N (total order)
  - Except for features that are **deprecated**

Version numbers are stupid

- Well, they work well for document revisions and software releases
- Not so great for protocols and other interface specifications
- Long discussion in T2TRG:
  Version numbers force creating a total order on a set of new features
- Better: declare individual features
  - Could do with must-understand fields: `bfeature1_: true`
  - But maybe can leverage the version number?

Proposal: interpret version number as bits

- A number can be used as a bit array
- Version 10 = 1010_2, i.e. features 1 and 3 ($2^1 + 2^3 = 10$)
- Add bits for additional features
- Proposed feature 4: use of Secondary Units ($2^4 = 16$)
  Version number with that additional feature would thus be 26
- Feature code can go up to 52 (53-bit integers in JSON):
  48 remaining now (after secondary units)

53: wasn’t that an evil number?

- Yes.
- But it could be all we need:
  - As the number of features that can be registered has a hard limit (48 codes left at the time of writing), the designated expert is specifically instructed to maintain a frugal regime of code point allocation, keeping code points available for SenML Features that are likely to be useful for non-trivial subsets of the SenML ecosystem.
  - Quantitatively, the expert could for instance steer the allocation to not allocate more than 10 % of the remaining set per year.

draft-ietf-core-senml-versions-00

- Defines the feature system:
  New Registry under the SenML registry
  Reserving feature code 0..3 for “10 = 1010_2”
  Specification required, frugality mandate to designated expert
- **Updates** the RFC 8428 version number to use that system
- Registers **feature code 4**: Use of secondary units

- Now WG draft, submitted 2020-05-13
  - Referenced from RFC 8798 (senml-more-units)
- No technical changes from 2020-03-06 draft-bormann-core-senml-versions-01

SenML Data Value Content-Format Indication

draft-ietf-core-senml-data-ct-02

Carsten Bormann

IETF 109

Content-Format indication

- SenML Records can contain (binary) "data values" in a "vd" field

```
[
 {"bn":"urn:dev:ow:10e2073a01080063:", "n":"temp", "v":7.1},
  {"n":"open", "vb":false},
  {"n":"nfc-reader", "vd":"aGkgCg"}
]
```

- This draft: new Content-Format indication ("ct") field to indicate the Content-Format of the data in the SenML Record

Example SenML Record with data value and Content-Format indication

```
{"n":"nfc-reader", "vd":"gmNmb28YKg", "ct":"60"}
```

CBOR CoAP Content Format

```
base64(
        82          # array(2)
          63        # text(3)
            666F6F  # "foo"
          18 2A     # unsigned(42)
)
```

Updates for draft-ietf-senml-data-ct-02

- 108: must-understand ct field (“ct_”) proved puzzling
  - Don’t know how to manage interaction with normal field (“ct”) in resolution process
  - So we got rid of it again (thanks for the suggestion, Klaus)
  - Would have been nice to have, but don’t know how to do it

- WGLC 2020-07-30..2020-08-17; no reviews by end of WGLC
- Since: reviews by Jim Schaad, Thomas Fossati, Alexey Melnikov

Need ABNF for contents of ct field (#2)

- Import from draft-bormann-core-media-content-type-format? (Revived 2020-08-18; would need to consider WG adoption)
- Or define here?
- content-coding = token ; [since RFC2616; RFC 7231]
- **Content-Format-String** = Content-Type ["@" content-coding]
- **Content-Format** = 1*DIGIT

```
token       = 1*tchar;  [RFC 7230]
tchar       = "!" / "#" / "$" / "%" / "&" / "'" / "*"
            / "+" / "-" / "." / "^" / "_" / "`" / "|" / "~"
            / DIGIT / ALPHA
```

Meaning (allowedness) of "ct" without "vd" #1

- Can happen when “bct” is in effect
- Probably innocuous to allow ct without vd (no consequences)

Next steps

- Re-check reviews

- Resulting changes are probably not technical
- Perform changes, submit to IESG

New CoAP Block-Wise Transfer Options For Faster Transmission

draft-ietf-core-new-block-02

IETF CoRE Meeting, 20th Nov 2020

Mohamed Boucadair

Jon Shallow

Agenda

- Changes since last interim
- One Pending Question
- Next Steps

Updates in -02 (10/2020) (1 of 2)

- Add a statement that either both or neither options can be supported
- Add an implementation note about how tokens can be handled to reduce number to be tracked
- Add a clarification about the behaviour when multiple instances of Q-Block2 are included
- Remove the "MUST NOT" restriction on 2.31 (Continue)
- Handling of requests that cannot be fulfilled due to packet size limitations now return 4.13

Updates in -02 (10/2020) (2 of 2)

- Update the CDDL and add an implementation note to suggest the use of indefinite-length arrays
- Add a note about the ACK_TIMEOUT delay (2s) after MAX_PAYLOADS
- Not recommended to be used in a NoSec security mode
- Editorial
  - Clarify what is meant by "repeat request“ by updating use of ‘M’ bit in requests
  - Change the name of the options to Q-Block1 and Q-Block2

Option Naming

- Poll set up for naming: https://doodle.com/poll/2uv4vfez9sq77fa9
  - ~~Quick-Block~~
  - **Q-Block**
  - **Resilient-Block**
  - ~~Fast-Block~~
  - **Robust-Block**
  - ~~FLLF-Block~~
  - ~~LL-Block~~
  - **Tough-Block**
  - **A-Block (Alternative Block)**

Question: Congestion Control (1 of 2)

- Background: MAX_PAYLOADS (default every 10 packets)
  - Default wait of ACK_TIMEOUT before proceeding
  - Use of CON every MAX_PAYLOAD for reduction of turnaround times
  - CON fails if unidirectional traffic loss
  - NON will wait for ACK_TIMEOUT before next packet sent
- How to reduce NON turnaround times if network/peer OK?
  - Signal something in the MAX_PAYLOAD packet to indicate immediate acknowledge response required
  - if response fails to get through there still will be ACK_TIMEOUT wait which is OK

Question: Congestion Control (2 of 2)

- Possibilities: No-Response Option works for Q-Block1
  - ISE, not Standards Track
  - How to handle Q-Block2?
    - May make sense to do the same for Q-Block1
- What about?
  - Update the Q-Block option format to "NUM **R** M SZX" where R bit set means:
    - Q-Block1: Respond with 2.31
    - Q-Block2: Issue GET for next block
- Is it worth to be solved?

Next Steps

- Prepare -03 with the outcome of the discussion to address the pending question
- Update the implementation
- If no major issue, target a WGLC

- Please review and share comments:
  https://github.com/core-wg/new-block

Thank You

draft-ietf-core-dynlink

IETF 109

Dynlink developments

- Current draft is at version -11
- Incorporating feedback received for updates, corrections and clarifications

- This slideset describes what is in the editor’s copy now, and what the new proposals are (from the OMA)

Updates in dynlink-latest

- Editor's draft is always available in Github as dynlink-latest (https://core-wg.github.io/dynlink/draft-ietf-core-dynlink.html)
- dynlink-latest contains two new attributes, epmin and epmax
- epmin: *the minimum evaluation period indicates the minimum time, in seconds, the client recommends to the server to wait between two consecutive measurements of the conditions of a resource*
- epmax: *the maximum evaluation period indicates the maximum time, in seconds, the server MAY wait between two consecutive measurements of the conditions of a resource*

New Proposal: Allow pmin == pmax

- Issue found at https://github.com/core-wg/dynlink/issues/25

- Current wording in Dynlink:
  - “The maximum period MUST be greater than zero and MUST be greater than the minimum period parameter (if present)”
  - pmin: Minimum time between 2 consecutive notifications even if resource state has changed
  - pmax: Maximum time between 2 consecutive notifications even if resource state has not changed

- Suggestion from OMA:
  - Have pmin equal pmax if the client wants the notification to be sent exactly every N seconds
  - Change text to “The maximum period MUST be greater than zero and MUST be greater than **or equal to** the minimum period parameter (if present)”

- Comments?

Proposal: New Attribute “edge”

- Issue found at https://github.com/core-wg/dynlink/issues/22
- OMA LwM2M Core Spec:
  - The Edge Attribute indicates either the falling edge ("0") or the rising edge ("1") transition of a Boolean Resource. When this Attribute is present, the LwM2M Client MUST notify the Server each time the Observed Resource value goes from "true" to "false" (edge = "0"), or from "false" to "true" (edge = "1") with respect to the pmin parameter and valid "Change Value Conditions”
- Comments?

Proposal: New Attribute “con”

- Issue found at https://github.com/core-wg/dynlink/issues/23
- OMA LwM2M Core Spec:
  - The Notification Confirmable Attribute indicates whether a Notification resulting from an Observation of a specific Object, Object Instance, Resource, Resource Instance MUST be sent over confirmable transport. If a Notification includes several Objects or Object Instances or Resources or Resource Instances or a combination thereof, then this Notification MUST be sent over confirmable transport if at least one of the Notification components has con=1."
- Comments?

Proposal: New attribute “hqmax”

- Issue found at https://github.com/core-wg/dynlink/issues/24
- OMA LwM2M Core Spec

:

  - "The Maximum Historical Queue Attribute indicates how many entries of historical data resulting from an Observation of a specific Object, Object Instance, Resource, Resource Instance MUST be stored, e.g. while the LwM2M Client is offline, or, the LwM2M Server account is disabled. If this attribute is present, only the data of Objects, Object Instances, Resources, Resource Instances with hqmax>0 will be included in notifications which were stored while disabled or offline. Historical notifications MAY be sent in a format as described in Section [SenML JSON] (). If the queue size reaches hqmax and a new reading is received, the oldest reading MUST be dropped. The LwM2M Client SHOULD empty the queue as soon it becomes aware that connectivity has been restored. The use of "hqmax" is dependent on notification storing being enabled via the "Notification Storing When Disabled or Offline" Resource of the LwM2M Server Object."

- Comments?

Dynlink-latest: Others

- Editorial changes to alter the language in order to reflect notifications as RESTful state changes and state transfer

draft-ietf-core-dynlink

Thank you!

Fast-Slow Retransmission Timeout and Congestion Control Algorithm for CoAP

draft-ietf-core-fasor-01

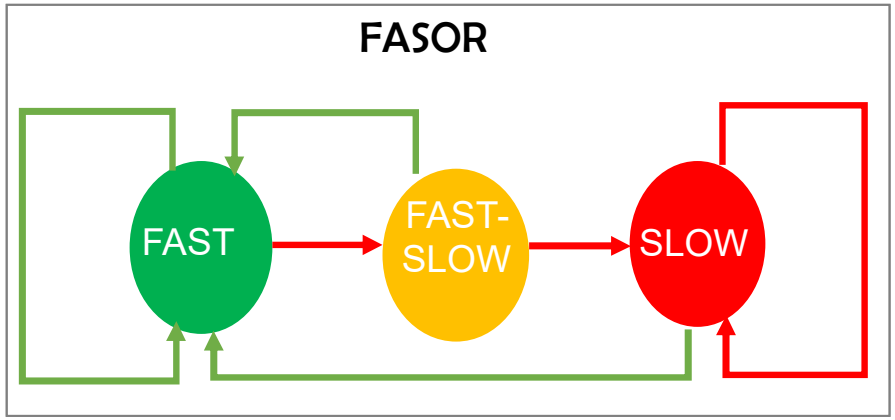
Ilpo Järvinen
Markku Kojo
Iivo Raitahila
Zhen Cao

IETF109

November 20, 2020

FASOR (Fast-Slow RTO)

- FASOR is an alternative Retransmission Timeout (RTO) and congestion control algorithm for CoAP
- Optional to implement in CoAP (like CoCoA)
- 3-state RTO logic each with its own back off series



I-D History

- -00 published in March, 2020 (after WG adoption)
  - Addressed feedback from Christer regarding the Retransmission Count option
    - Clarified limitations in the option use
    - Clarified the meaning of “new destination endpoint”
- -01 published in Oct 2020
  - Clarified the use of the Retransmission Count Option value
    - i.e., increment by one for each retransmission
  - Some editorial changes

Next Steps

- More reviews and feedback would be appreciated
  - Also from tsvarea
- Then ready for WGLC?

Combining EDHOC and OSCORE draft-palombini-core-oscore-edhoc-01

Francesca Palombini,
Marco Tiloca,
Rikard Höglund,
Stefan Hristozov,
Göran Selander

EDHOC then OSCORE over CoAP

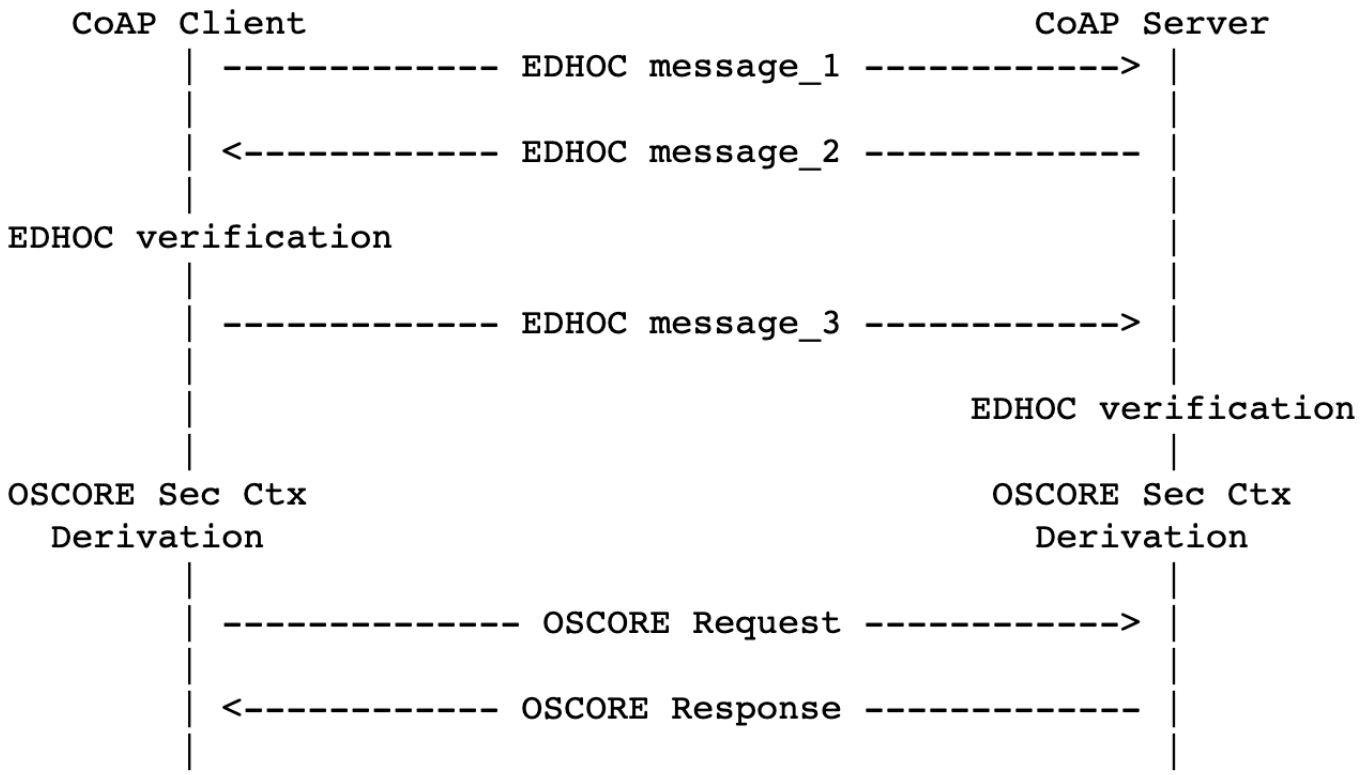


Figure 1: EDHOC and OSCORE run sequentially

Optimized – EDHOC message 3 + OSCORE Req

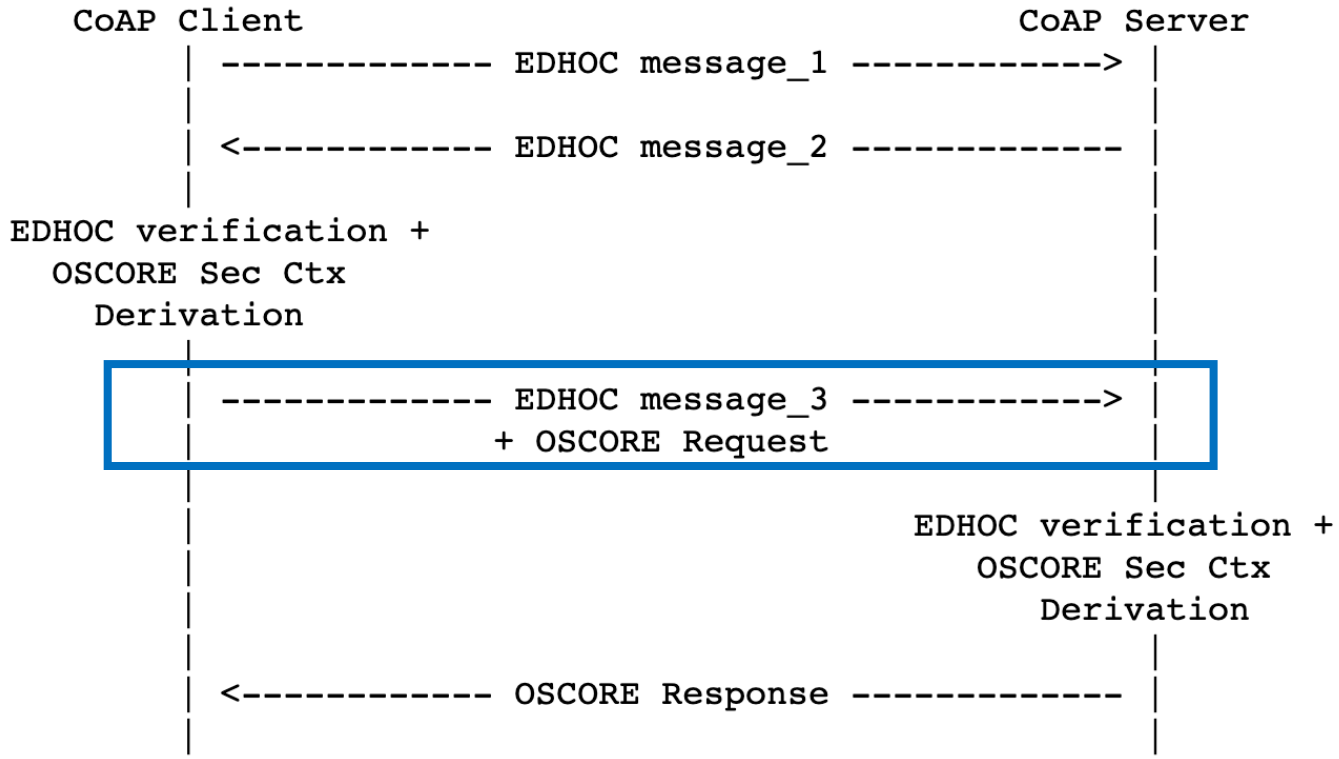


Figure 2: EDHOC and OSCORE combined

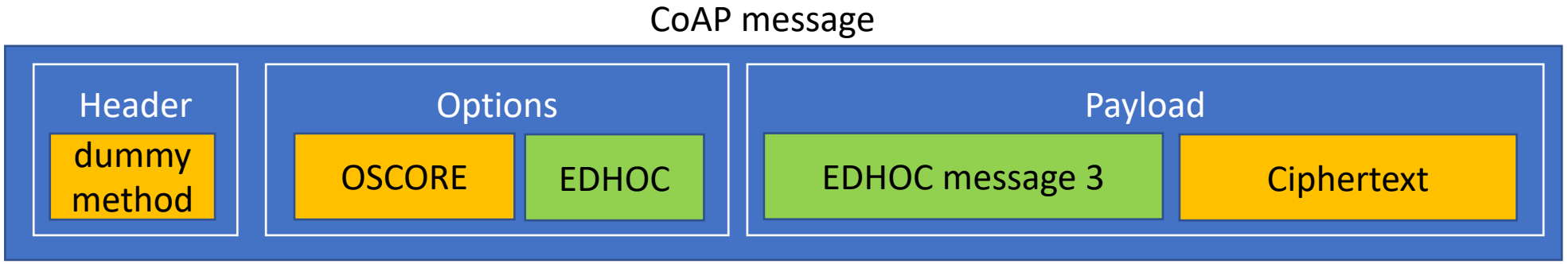
How to send those 2 messages together?

- **Send EDHOC in OSCORE ← preferred option from IETF 108**
  1. Signalling in a new CoAP option
  2. Signalling in the OSCORE option (use a bit in the flagbits)

- ~~**Send OSCORE in EDHOC**~~

EDHOC in OSCORE – Signalling in new option

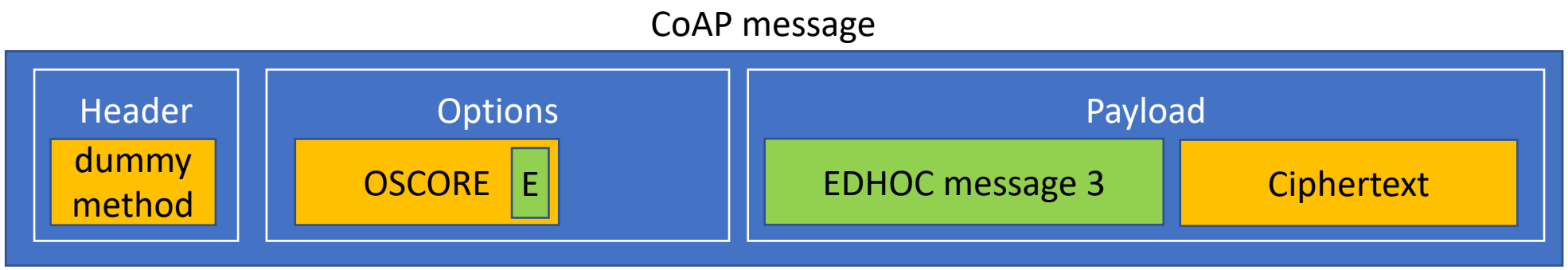
1.




```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Ver| T |  TKL  |      Code     |          Message ID           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Token (if any, TKL bytes) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| OSCORE option | EDHOC option | other options (if any) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 1 1 1 1 1 1 1|    Payload
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

EDHOC in OSCORE – Signalling in OSCORE

2.




```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Ver| T |  TKL  |      Code     |          Message ID           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Token (if any, TKL bytes) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| OSCORE opt (with EDHOC bit set) |  other options (if any) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 1 1 1 1 1 1 1|    Payload
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Comparison

(based on Appendix C.4 of RFC8613 and Appendix B.2 of EDHOC test vectors)

1. 58 bytes in total (empty option)
2. 57 bytes in total (using bit 1 in the first byte of OSCORE Flag Byte)
   - 58 if we need to expand the flag byte, and use bit >7

Way Forward

- Pick one
- Get feedback + reviews
- WG adoption?

CoAP over GATT

draft-amsuess-core-coap-over-gatt

Christian Amsüss

IETF109, CoRE, 2020-11-20

What?

Exchange CoAP over GATT

GATT: Generic Attribute Profile of Bluetooth Low Energy

Why?

All the same reasons as for CoAP-over-WS

- Extend CoAP, especially OSCORE, to applications with limited APIs

  "Smart" phone apps, browsers[1]

- ~~Too lazy to implement IPSP~~
- ~~But with IPSP I'll have to have a full IP stack~~

- Explain non-CoAP GATT values to CoAP devices through SCHC?

[1] Not generally approved; future versions on future protocols (WebRTC data channel?).

How?

Request: Write *Method* || *Options* || [0*xff* || *Payload*]

Response: Read *Code* || *Options* || [0*xff* || *Payload*]

No message type / MID: Reliability handled by BLE

No Token: Multiplexing handled by descriptors[2]

$\sim$ 512 Byte MTU[3]: Block-wise up to szx=5

[2] May or may not be practical.

[3] May be lower per device, to be investigated.

URI questions

`coap+bluetooth://00-11-22-33-44-55-66-77-88-99/.well-known/core`

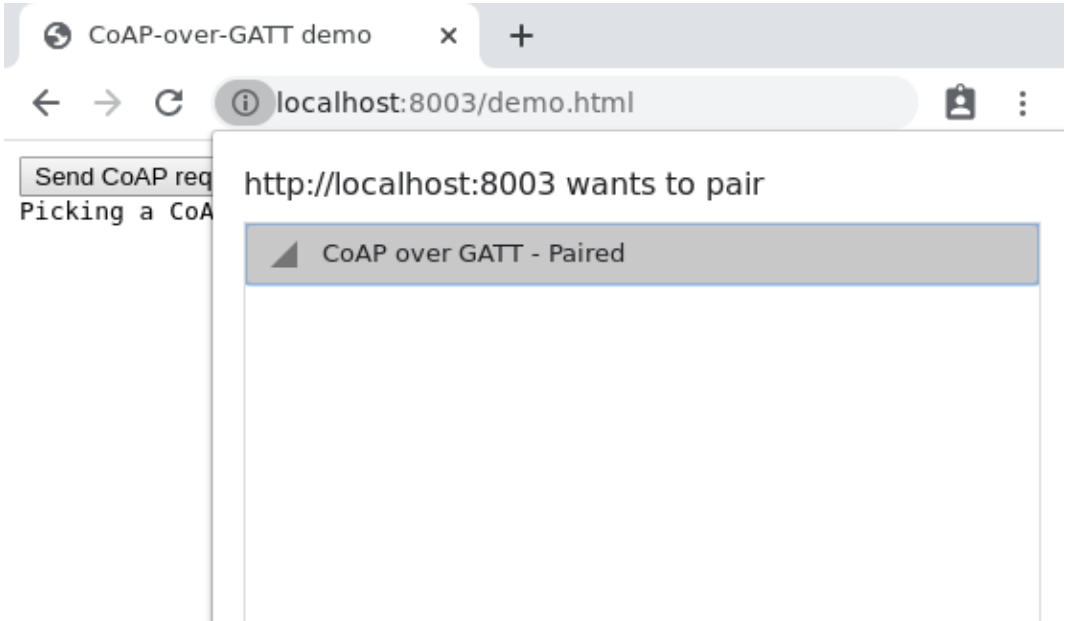
`coap+bluetooth://0x2000a140/.well-known/core`
`coap+bluetooth://[]%0x2000a140/.well-known/core`

`coap://device.example.com/.well-known/core`
where
`device.example.com.  IN BLUETOOTH 00-11-..-99`

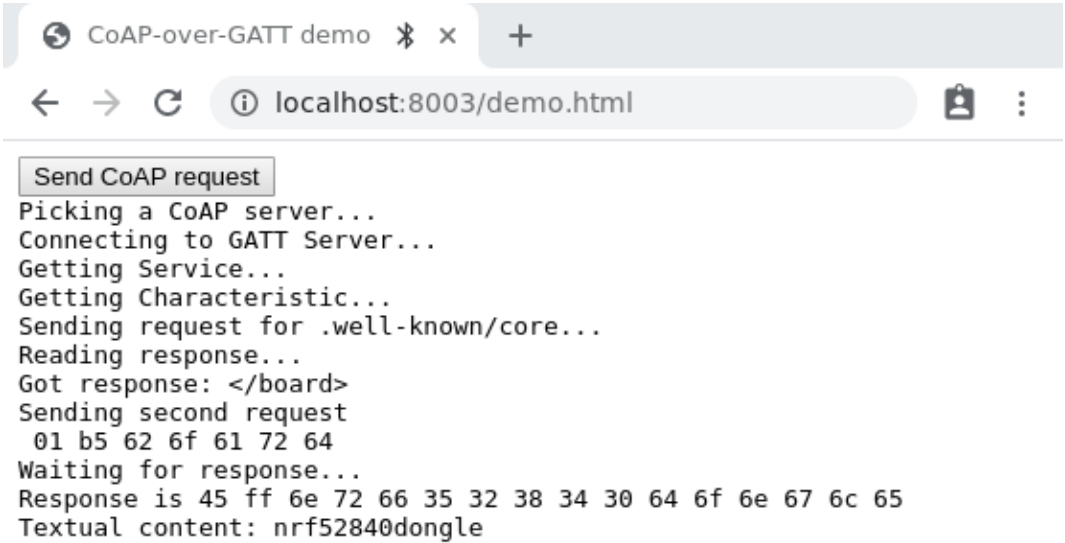
`coap://00-11-..-99.blue.arpa/.well-known/core`

. . . but basically this means continuing on `protocol-negotiation`

Implementations



Implementations



Roadmap

- Gather implementation experience
- Keep draft in sync
- WG feedback
- Experimental “This is how a few people do it” document

- If aiming for standards track, not before GATT is accessible from Mozilla browsers[4]

[4] https://github.com/mozilla/standards-positions/issues/95

AEAD Limits for OSCORE

CORE, IETF 109, November 2020

AEAD Limits

- CFRG works on equations to calculate AEAD limits when the key needs to be changed.
  - Limits based on best known bounds for adversary probability of forgery of a single packet or distinguishability from a random string.
  - https://tools.ietf.org/html/draft-irtf-cfrg-aead-limits

- The important equations for AEAD_AES_128_CCM_8 (AES-CCM-16-64-128, AES-CCM-64-64-128) are:

- Confidentiality Limit: $\mathbf{q} \leq \text{sqrt}((\mathbf{p} \cdot 2^{126}) / \mathbf{l}^2)$
- Integrity Limit: $\mathbf{v} \cdot 2^{64} + (2\mathbf{l} \cdot (\mathbf{v} + \mathbf{q}))^2 \leq \mathbf{p} \cdot 2^{128}$

AEAD limit based on
- **q** Number of packets to encrypt
- **v** Number of unsuccessful invocations of AEAD

- where
  - **q** = Number of protected messages (AEAD encryption invocations)
  - **v** = Number of attacker forgery attempts (failed AEAD decryption invocations)
  - **p** = Upper bound for adversary attack success probability
  - **l** = Input length (plaintext + additional data) in blocks

Note that it is always true that $\mathbf{v} \leq \mathbf{p} \cdot 2^{64}$

AEAD Limits

- TLS 1.3, DTLS 1.3, and QUIC have adopted strict limits based on the same equations. Having strict limits is not a problem if re-keying is easy.

- For example, DTLS 1.3 uses the parameters:
  - $\mathbf{p}_q = 2^{-60}$ $\mathbf{p}_v = 2^{-57}$, (upper bound for security level, different values can be chosen)
  - $\mathbf{l} = 2^{10}$ (2^{14} bytes = 16 kB) (maximum length of TLS record layer)
- Resulting in the following limits:
  - **CCM**: Limit for CCM set to $\mathbf{q} = 2^{23}$ and $\mathbf{v} = 2^{23.5}$
  - **CCM_8**: MUST NOT be used **without additional safeguards against forgery.**
- TLS 1.3 has $\mathbf{v} = 1$ so CCM_8 can be used.

- **What assumptions are relevant for the CoAP IoT setting:**
  - The CoAP IoT setting in general (where CCM_8 is commonly used)?

AEAD Limits Considerations

- Forgery attacks are a greater threat that distinguishing attacks. For CCM_8, the limits for number of failed decryptions **v** is the limiting factor. Limits smaller than 2^{40} likely needed also for other AEAD.
- Some IoT systems might accept a higher forgery probability than (D)TLS in general.
- If the attacker uses a high-speed broadband connection, a lot of forgeries can be sent quickly. On constrained IoT systems, bandwidth is restricted, and it would take long time to send forged messages.
- Any DoS attack resulting from low limits on **v** should be compared to other DoS attacks on the system. If there are even easier ways to reduce the availability of a system, a low limit on **v** does not matter.
- Replay attacks will never invoke AEAD decryption, thus not contribute to **v**.
- For most constrained IoT systems, input lengths are much smaller than 16 kB packets used for (D)TLS. Need to specify a use a maximum input length.
- Likely possible to use limits based on the maximum seen length, which would allow better limits in a non-attack scenario.
- A forgery itself is not a huge problem if the plaintext is not processed. Might not need to “close connection”. Note that the adversary gains some information with each successful forgery.
- Forgeries might look like random strings and may be discarded by CoAP. But it should probably be assumed that the adversary can do chosen prefix or postfix.

Mail thread in LAKE: https://mailarchive.ietf.org/arch/browse/lake/

OSCORE considerations

- "The maximum Sender Sequence Number is algorithm dependent (see Section 12) and SHALL be less than 2^40." (Section 7.2.1 of RFC 8613)
  - CCM_8 limit will apply before 2^{40} ($\approx 10^{12} \approx$ 3587 messages per second for 10 years)
    - Guidelines for setting max SSN are needed

- Current schemes for rekeying OSCORE
  - Appendix B.2
  - EDHOC asymmetric key authenticated DH
  - OSCORE profile of ACE
  - [provisioning of master secret . . .]

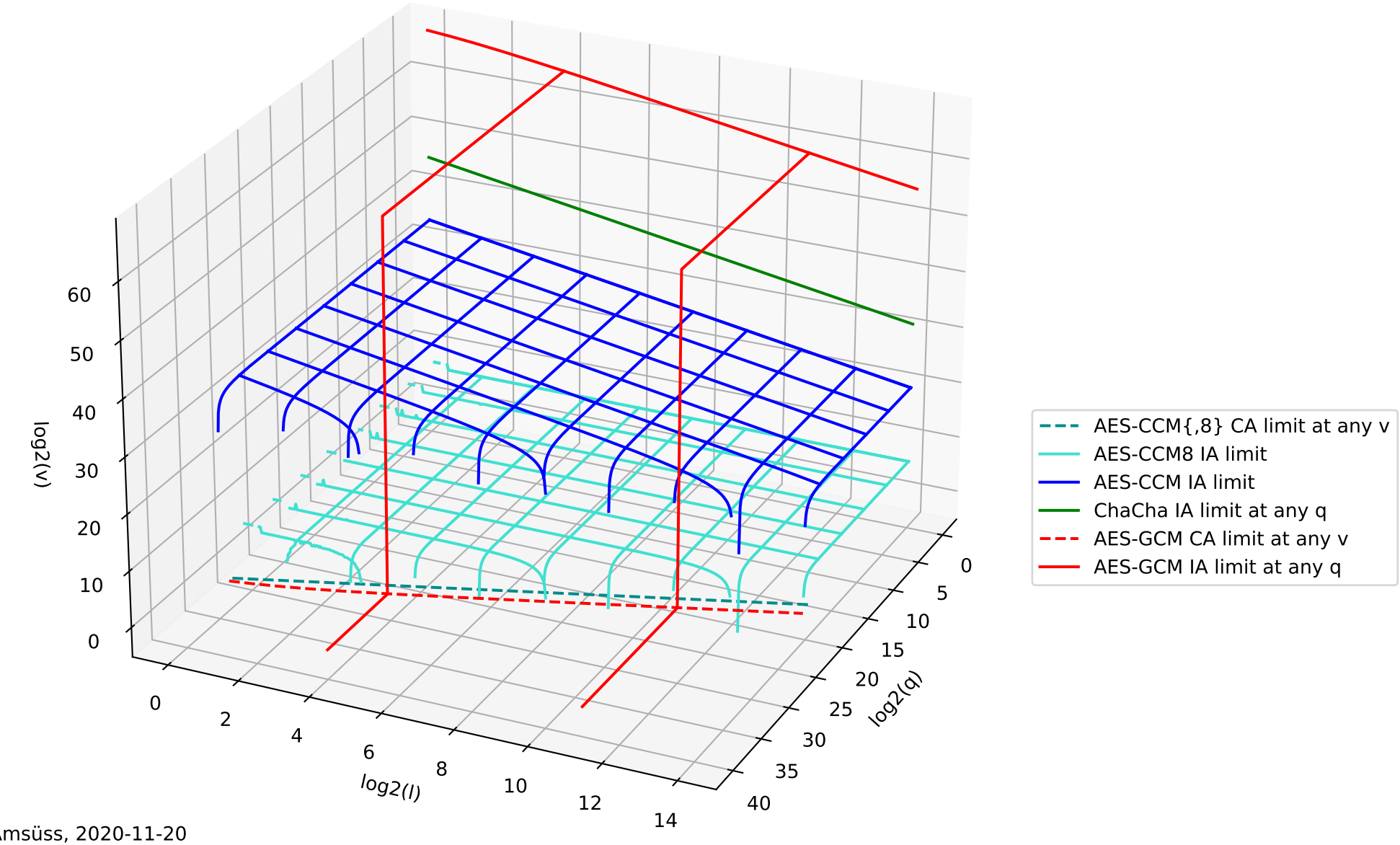
Potential actions

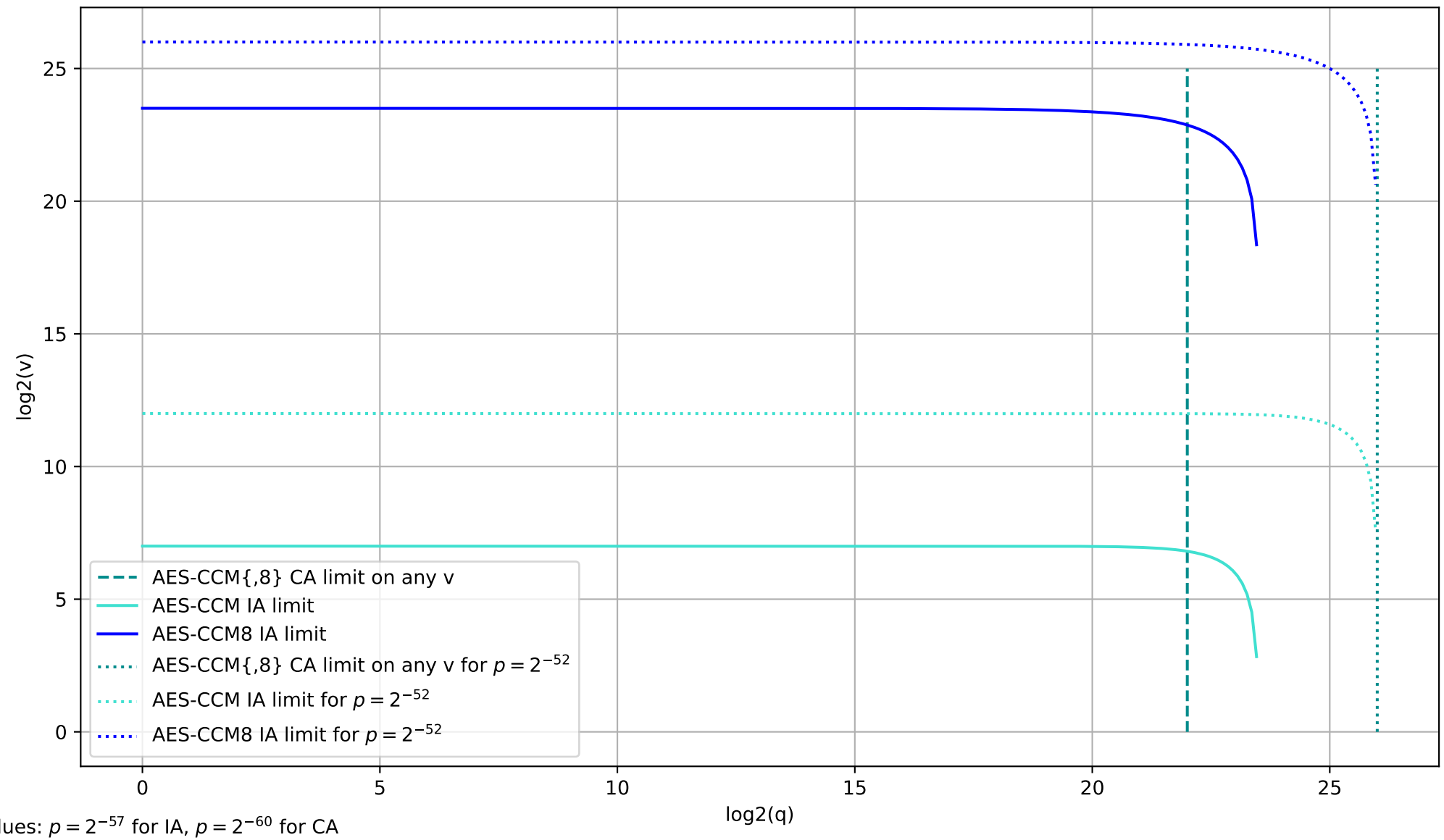
- General problem
  - There is a need to investigate CoAP IoT relevant limits
  - Parameters p and l needs to be determined
  - Trade-off between q and v
  - Discussion started in LAKE
  - For what protocols is this discussion relevant?

- Specific for OSCORE: potential new draft updating RFC 8613
  - Introduce count of q and v
    - Client already counts q
  - How to configure maximum q and v
  - Specify actions to take when max q or max v is reached (rekey)
  - Implementation guidance

- Corresponding adaptations needed for group OSCORE
- Maybe good idea to start investigating a lightweight symmetric-key rekeying mechanism?

Visualized equations from draft-irtf-cfrg-aead-limits-01





Default values: $p = 2^{-57}$ for IA, $p = 2^{-60}$ for CA