## IETF 111

# 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

- 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)

# 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

# Agenda (120 min)

- 19:00–19:10 Intro, Agenda, Status
- 19:10–19:20 HREF
- 19:20–19:30 Dynlink and Conditional Attributes
- 19:30–19:45 Transport indication
- 19:45–19:55 CoAP Attacks
- 19:55–20:05 Groupcomm-bis
- 20:05–20:20 Group OSCORE
- 20:20–20:30 Combining EDHOC with OSCORE
- 20:30–20:40 Key update for OSCORE
- 20:40–20:50 OSCORE-capable proxies
- 20:50–21:00 Flextime

## Document status

# RFC 9030

- “Uniform Resource Names for Device Identifiers”

- Was *draft-ietf-core-dev-urn*

## RFC Queue

- draft-ietf-core-resource-directory-28
  - In RFC Ed Queue : MISSREF
- draft-ietf-core-new-block-14
  - In RFC Ed Queue : MISSREF
- draft-ietf-core-senml-versions-05
  - In RFC Ed Queue : AUTH48

## IESG Processing

- draft-ietf-core-echo-request-tag-13
  - In Approved-Announcement to be Sent::AD Followup

- draft-ietf-core-yang-cbor-16
  - Shepherding transferred to Marco Tiloca
  - IESG Evaluation::Revised I-D Needed

- draft-ietf-core-sid-16
  - Shepherding transferred to Jaime Jiménez
  - IESG Evaluation::Revised I-D Needed

## Post-WGLC Processing

- draft-ietf-core-senml-data-ct-04
  - In “AD Evaluation”
- draft-ietf-core-comi-11
  - Waiting for Shepherd Write-Up; fixes required from authors
- draft-ietf-core-yang-library-03
  - Waiting for Shepherd Write-Up; need to synch with -comi
- draft-ietf-core-oscore-groupcomm-12
  - Addressed 1st WGLC comments
  - Worked on several recent open points (see later presentation)

## Rechartering ?

- The current text in charter-ietf-core-02 is pretty old

- Main reasons to revise
  - Charters are closely looked at when the IESG receives WG documents
  - Newcomers can better and easier understand what we did and what we are doing

- Proposed revision criteria
  - Keep following the same overall scope and direction
  - Ensure to reflect what was done, the work going on, and potential new work items

- Thoughts? Strong opinions against?

- The Chairs can work on a first draft to share with the WG

All times are in UTC

## Agenda (120 min)

- 19:00–19:10  Intro, Agenda, Status
- 19:10–19:20  HREF
- 19:20–19:30  Dynlink and Conditional Attributes
- 19:30–19:45  Transport indication
- 19:45–19:55  CoAP Attacks
- 19:55–20:05  Groupcomm-bis
- 20:05–20:20  Group OSCORE
- 20:20–20:30  Combining EDHOC with OSCORE
- 20:30–20:40  Key update for OSCORE
- 20:40–20:50  OSCORE-capable proxies
- 20:50–21:00  Flextime

## Flextime

## Thank you!
## Comments/questions?

I E T F

# CoRE: CRI

July 28th (Wednesday), 19:00-21:00 UTC
(21:00–23:00 CEST, 12:00–14:00 PDT)

## The Web (~ 1990)

| component | big web |
|---|---|
| hyperreferences | URI |
| transfer protocol | HTTP |
| representation format | HTML |

## The Thing Web (~ 2010): CoRE

| **component** | **thing web** |
|---|---|
| hyperreferences | URI |
| transfer protocol | **CoAP** |
| representation format | **(CBOR-based formats)** |

## Time for a cleanup? (~ 2020)

| **component** | **thing web** |
|---|---|
| hyperreferences | URI → **CRI** |
| transfer protocol | CoAP |
| representation format | (CBOR-based formats) |

# URI

**[…] many implementations […] support only an ad-hoc, informally-specified, bug-ridden, non-interoperable subset of half of RFC 3986.**
— Klaus Hartke

# RFC 3986, RFC 7252

RFC 3986: syntax of URIs, (implicit) data model
RFC 7252 maps that data model to CoAP options

| **Component** | **Structure** | **Separators** |
|---|---|---|
| scheme | http: | |
| authority | //tzi.de | |
| path | (path segments) | / |
| query | (query elements) | & — CoAP |
| fragment | #page-5 | |

## URI references: RFC 3986 "resolution procedure"

URIs relative to a base (document URI): occur in documents, not in transfer protocols

| Example | Resolution from https://tzi.de/pa/t/h | What happens to Path Segments? |
|---|---|---|
| foo | https://tzi.de/pa/t/foo | discard last 1 |
| /foo | https://tzi.de/foo | discard all |
| ../foo | https://tzi.de/pa/foo | discard last 2 |
| ?bar | https://tzi.de/pa/t/h?bar | discard 0 |
| //tzi.org/foo | e.g., https://tzi.org/foo | (new authority), discard all |

# CRI

Concise Resource Identifier:
Concise equivalent of URIs and URI references (RFC 3986)

New representation format for **URI data model**

draft-ietf-core-href defines **CRIs** and CRI references

## -04 ("new-syntax")

— developed with Jim Schaad, since further optimized → efficient!
— requires a number of decisions (if-statements) on ingest ("parsing")

Abstract content:

```
[ scheme, authority, discard, path, query, fragment ]
```

(path and query are arrays;
authority has address/name + optional port)

# -05 ("simplified-syntax")

- directly use the six-tuple, in two compacted variants:
  - [scheme, authority, path, query, fragment]
  - [discard, path, query, fragment]
- use cases **roughly** URI vs. relative reference

Cost of simplification: 2–4 bytes per relative CRI-reference

# -06 ("authority anomaly")

| URI | CRI |
|---|---|
| urn:x | ["urn", null, ["x"]] |
| urn:/x/y | ["urn", null, ["x", "y"]] |
| urn:/x | ["urn", null, ["x"]] |
| Solution: | special-case non-rooted opaque |
| urn:x | ["urn", true, ["x"]] |

# -06: parsed hostname

| **Component** | **representation** | **parsed out** |
|---|---|---|
| scheme | "http" or -2 | : |
| authority | ["tzi", "de", 4711] | . and : |
| path | ["pa", "th] | / |
| query | ["qu", "e=ry"] | ? and & |
| fragment | "fragment" | # |

http://tzi.de:4711/pa/th?qu&e=ry#fragment

# href-06: Status

— consistent design

— feature-complete

— implementations need updates to –06/publishing

— href was developed for CoRAL
  - Klaus Hartke is currently taking time off from CoRAL
  - Requirements from CRIs from other groups, e.g., attestation
  - "Finishing" editors: Carsten Bormann, Henk Birkholz

— (1) More implementer reviews, (2) WGLC?

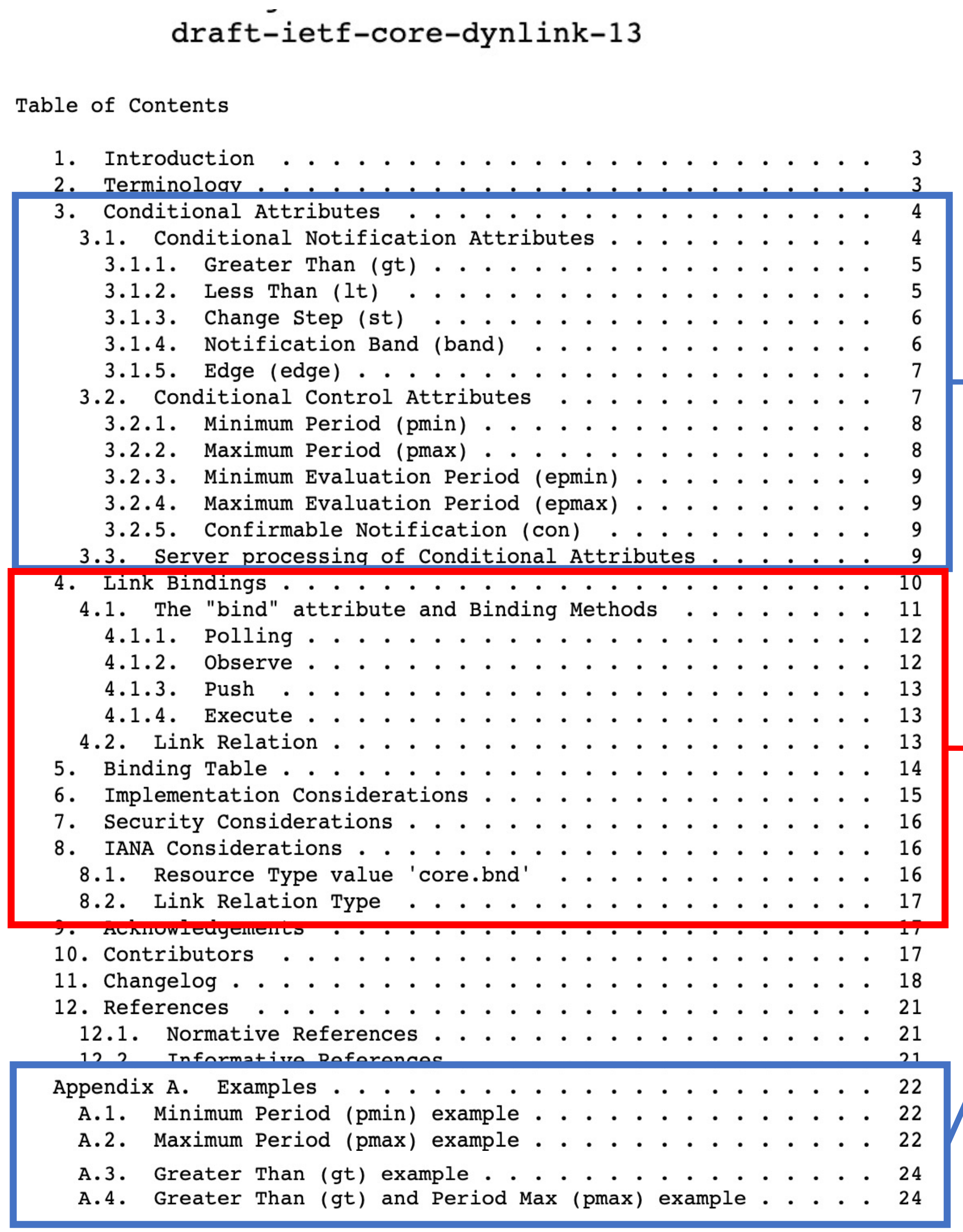
# Dynlink and Conditional Attributes

IETF 111 July 2021

Core WG

## Latest developments

- After discussion with AD, Chairs and authors, Dynlink draft version 13 was split into two WG drafts
  - draft-ietf-core-dynlink
  - draft-ietf-core-conditional-attributes
- Continuing to incorporating feedback received for updates, corrections and clarifications



draft-ietf-core-dynlink-13

Table of Contents

Text extracted verbatim to new WG draft: draft-ietf-core-conditional-attributes-00

The rest of the dynlink-13 extracted verbatim to dynlink-14

## Why the changes?

- While conditional observe attributes are now ready for some time, the section on link bindings and binding tables still under development particularly in relation to other work such as CoRAL
- OMA’s LWM2M specs has a dependency to Dynlink, but purely to the conditional observe attributes and not the link bindings
- Work on conditional attributes is almost finished
- Separating into 2 drafts will allow core-conditional-attributes to proceed rapidly towards RFC without creating pressure on link bindings

## core-conditional-attributes-00

- Available at https://github.com/core-wg/conditional-attributes
- Impact on behaviour from the possible presence of (multiple) proxies
    - This will be added into Implementation Considerations
- Text for possible security considerations will be added to -01
- Update reference code for server processing of Conditional Attributes
    - Include also a state machine to describe server-side processing, for example with epmin and epmax
- Other changes as needed
- Reviews are always welcome!
- Then ready for WG last call

## core-dynlink-14

- Shortcomings seen with using CoRE Link Format for describing link bindings in binding table
    - Exploration is still needed for better technique to describe link bindings
- Dynlink-15 will address these and evolve to include more examples and usage cases

# Dynlink and Conditional Attributes

Thank you!

# CoAP Protocol Indication

`draft-amsuess-core-transport-indication-01`

which you may know from having piggybacked on resource-directory-extensions on IETF110

Christian Amsüss

2021-07-28, IETF111

## A brief history of CoAP schemes

| | | |
|---|---|---|
| 2014 | RFC7252 | `coap` for UDP |
| 2015–2017 | `coap-tcp-tls`≤08 | `coap+tcp` for TCP |
| 2017 | `coap-tcp-tls-09` | `coap` for TCP or UDP |
| | `coap-tcp-tls`≥10 | `coap+tcp` for TCP |

## A brief history of CoAP schemes

| | | |
|---|---|---|
| 2014 | RFC7252 | `coap` for UDP |
| 2015–2017 | `coap-tcp-tls`$\leq$08 | `coap+tcp` for TCP |
| 2017 | `coap-tcp-tls-09` | `coap` for TCP or UDP |
| | `coap-tcp-tls`$\geq$10 | `coap+tcp` for TCP |

### From No-Objection ballots

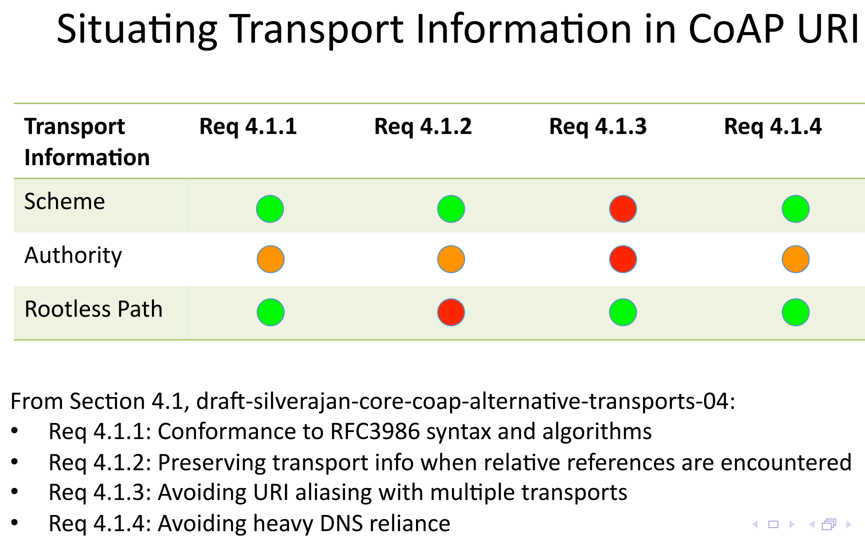
...these scheme registrations [...] present an “antipattern” ...

This runs counter to the principle that a URI identifies a resource ...

I am perplexed that no concrete mechanism for UDP/TCP failover is provided ...

## It was known not to be easy

From Bill’s 2014 (IETF 89) presentation



### Situating Transport Information in CoAP URI

| Transport Information | Req 4.1.1 | Req 4.1.2 | Req 4.1.3 | Req 4.1.4 |
|---|---|---|---|---|
| Scheme | 🟢 | 🟢 | 🔴 | 🟢 |
| Authority | 🟠 | 🟠 | 🔴 | 🟠 |
| Rootless Path | 🟢 | 🔴 | 🟢 | 🟢 |

From Section 4.1, draft-silverajan-core-coap-alternative-transports-04:

- Req 4.1.1: Conformance to RFC3986 syntax and algorithms
- Req 4.1.2: Preserving transport info when relative references are encountered
- Req 4.1.3: Avoiding URI aliasing with multiple transports
- Req 4.1.4: Avoiding heavy DNS reliance

## URI aliasing pain points

$$\texttt{coaps://[2001:db8::1]/cfe} \overset{?}{=} \texttt{coaps+tcp://[2001:db8::1]/cfe}$$

Pick one side:

- Multiple entries in discovery
- Multiple entries in caches

- Transports stay unused
- Devices can not connect[1]

Proxies can't pick transports according to their abilities.

No established terminology to describe URI aliasing.

[1] CoAP-over-UDP was never described as mandatory-to-implement

## Tools we have

- Easy resource metadata

```
<coap+tcp://[2001:db8::1]>;rel=...
```

(with some indirection to make site-wide statements)

## Tools we have

- Easy resource metadata

  ```
  <coap+tcp://[2001:db8::1]>;rel=...
  ```

  (with some indirection to make site-wide statements)

- Cheap proxying

  ```
   CON GET
   Observe: 0
   Uri-Path: "cfe"
  +Proxy-Scheme: "coap"
  ```

  (with triviality bonus points for implementations ignoring the 'critical' flag)

## Putting it together

```
</cfe>;rt="tag:...:coffemachine";rel=hosts;anchor="/",
<coap+tcp://[2001:db8::1]>;rel=has-proxy;anchor="/"
```

### Goals (1-2/5)

**Enablement** Inform clients of the availability of other transports of servers.

**No Aliasing** Any URI aliasing must be opt-in by the server. Any defined mechanisms must allow applications to keep working on the canonical URIs given by the server.

Server implementation: Just accept provided Proxy-Scheme options.

Client implementation: Ignore, or use indicated protocol and add Proxy-Scheme (and, if needed, Uri-Host) option.

## Message overhead kills

```
 CON GET
 Observe: 0
 Uri-Path: "cfe"
+Proxy-Scheme: "coap"
```

$\sim$ 5 bytes per request. More if host names are involved.

**Goals (3/5)**

**Optimization** Do not incur per-request overhead from switching protocols. This may depend on the server's willingness to create aliased URIs.

`rel=has-unique-proxy` additionally means you can skip Proxy-Scheme and Uri-Host

## Proxy interaction

### Goals (4-5/5)

**Proxy usability** All information provided must be usable by aware proxies to reduce the need for duplicate cache entries.
**Proxy announcement** Allow third parties to announce that they provide alternative transports to a host.

...which I'll be happy to elaborate on in hallway discussions.

## Security Considerations

# Just As With Any Proxy.

OK, there's more in the text, but that's the gist.

Problematic with third-party protocol translation services:
What's done by (D)TLS users here? Do they use proxies at all?
Are all-valid certificates common there? Do we want to endorse them?

## Take-home message

- It can probably be just this simple.
- No URI aliasing introduced in applications.

Questions? Comments? Interest?

## Backup slide / FAQ

*Didn't we want to do this with DNS?*

We[2] still can, just need to phrase the equivalent statements in DNS.

Straw man for "`coap://device.example.com` has CoAP-over-TCP running on port 1234":

```
_has-coap-proxy._tcp.device.example.com SRV 0 0 device.example.com 1234
device.example.com AAAA 2001:db8::1
```

*How does this relate to HTTP's `Alt-Svc`?*

Generally similar; links instead of headers (as common in CoAP), and no need for protocol-id because we have schemes already.

[2] Whoever wants to use it will need to volunteer as coauthor.

# CoAP Attacks

draft-mattsson-core-coap-attacks

IETF 111, CORE, John Preuß Mattsson

## draft-mattsson-core-coap-attacks

- In his IESG review of draft-ietf-core-echo-request-tag, Security AD Benjamin Kaduk suggested that it would be good to publish also draft-mattsson-core-coap-actuators.

- Based on this suggestion we have resubmitted draft-mattsson-core-coap-actuators as draft-mattsson-core-coap-attacks. The draft has been rebranded and expanded with a new section on amplification attacks.

- The draft gives background on the attacks motivating draft-ietf-core-echo-request-tag
  - Updated client Token processing requirements
  - Request-Tag option to match block-wise message fragments
  - Echo option to verify the freshness of a request
  - Echo option to to demonstrate reachability at its claimed network address

- **We aim to publish draft-mattsson-core-coap-attacks as an informational RFC.**

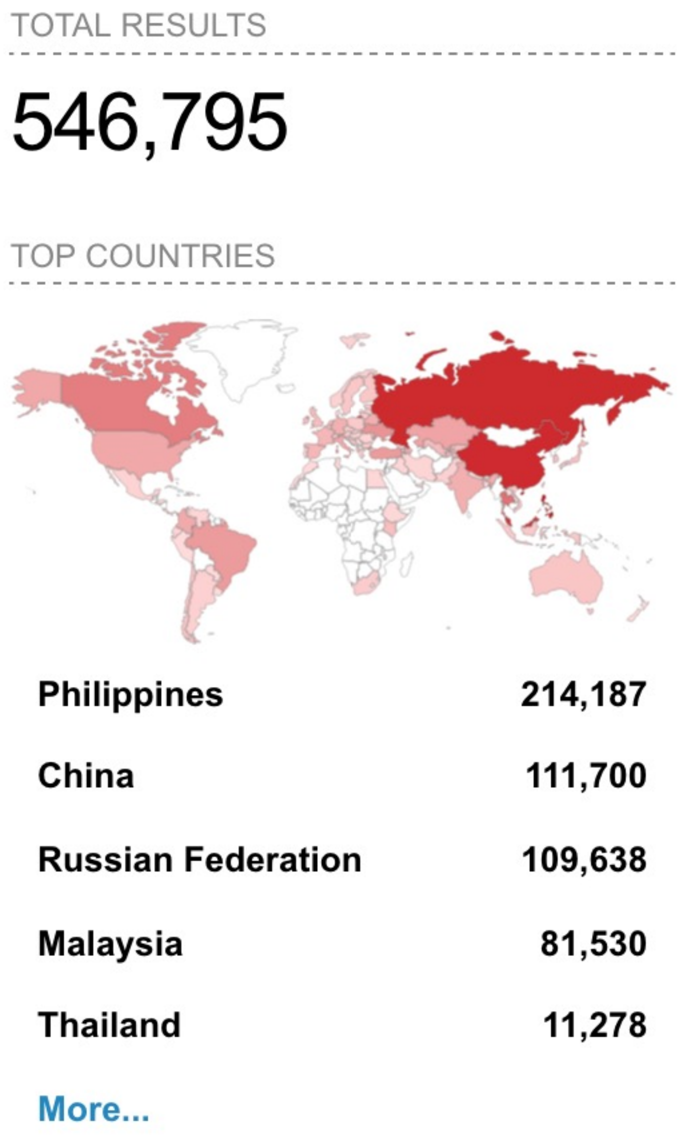
- **We think CORE need to discuss and take more concrete action against amplification attacks.**

# CoAP amplification attacks gets media attention

- https://www.netscout.com/blog/asert/coap-attacks-wild
- https://www.shadowserver.org/news/accessible-coap-report-scanning-for-exposed-constrained-application-protocol-services/
- https://www.zdnet.com/article/the-coap-protocol-is-the-next-big-thing-for-ddos-attacks/
- https://www.zdnet.com/article/fbi-warns-of-new-ddos-attack-vectors-coap-ws-dd-arms-and-jenkins/
- https://www.helpnetsecurity.com/2019/03/08/iot-coap-ddos-weapon/
- https://blog.mazebolt.com/understanding-the-coap-ddos-attack-vector
- https://www.securityweek.com/attackers-use-coap-ddos-amplification
- https://medium.com/nsc42/what-is-coap-and-is-it-the-next-ddos-for-iot-de8ee97e57e6
- https://www.globaldots.com/resources/blog/iot-devices-using-coap-increasingly-used-in ddos-attacks/

## Amplification attacks with IP address spoofing

- In an amplification attack, the amplification factor is the ratio between the size of the request and the total size of the response(s) to that request.
- Amplification attacks are often combined with the attacker spoofing the source IP address of the targeted victim to create a distributed denial-of-service attack on the target.
- When transported over UDP, the CoAP NoSec mode is susceptible to source IP address spoofing.
- [CoAP-Report] and [CoAP-Wild] report average amplification factor of 27 and 34 respectively from a single response to a GET request for /.well-known/core to the default UDP port 5693.
- The open CoAP servers are mostly concentrated to a few countries and a few implementations, which do not follow the recommendations in Section 11.3 of [RFC7252] (but the requirements are a bit soft).



TOTAL RESULTS

546,795

TOP COUNTRIES

| | |
|---|---|
| Philippines | 214,187 |
| China | 111,700 |
| Russian Federation | 109,638 |
| Malaysia | 81,530 |
| Thailand | 11,278 |

More...

# Amplification attack using a single response

- If the response is $a$ times larger than the request, the **amplification factor is $a$.**

- Amplification factors can be significantly worse when combined with observe [RFC7641] and multicast [I-D.ietf-core-groupcomm-bis].

```
Client   Foe   Server
   |      |      |
   |      +----->|      Code: 0.01 (GET)
   |      | GET  |     Token: 0x77
   |      |      |  Uri-Path: random quote
   |      |      |
   |<------------+      Code: 2.05 (Content)
   |      | 2.05 |     Token: 0x77
   |      |      |   Payload: "just because you own half the county
   |      |      |             doesn't mean that you have the power
   |      |      |             to run the rest of us. For twenty-
   |      |      |             three years, I've been dying to tell
   |      |      |             you what I thought of you! And now...
   |      |      |             well, being a Christian woman, I can't
   |      |      |             say it!"
   |      |      |
```

# Amplification attack using observe

- If each response have an amplification factor of $a$, and the server sends $n$ responses, the total **amplification factor is $an$.**

```
Client   Foe   Server
   |      |      |
   |      +----->|     Code: 0.01 (GET)
   |      | GET  |    Token: 0x83
   |      |      |  Observe: 0
   |      |      | Uri-Path: stock market index 1 min
   |      |      |
   |<------------+     Code: 2.05 (Content)
   |      | 2.05 |    Token: 0x83
   |      |      |  Observe: 217362
   |      |      |  Payload: 3749.7
   |      |      |
   |<------------+     Code: 2.05 (Content)
   |      | 2.05 |    Token: 0x83
   |      |      |  Observe: 217363
   |      |      |  Payload: 3745.33
   |      |      |
   |<------------+     Code: 2.05 (Content)
   |      | 2.05 |    Token: 0x83
   |      |      |  Observe: 217364
   |      |      |  Payload: 3747.65
   |      |      |
     ....   ....
```

## Amplification attack using multicast

- If each response have an amplification factor of $a$, and there there are $m$ servers, the total **amplification factor is $am$.**

- Note that the servers usually do not know the variable $m$.

```
Client    Foe    Server
   |       |       |
   |       +------>|      Code: 0.01 (GET)
   |       | GET   |     Token: 0x69
   |       |       |  Uri-Path: </c>
   |       |       |
   |<-------------+      Code: 2.05 (Content)
   |       | 2.05  |     Token: 0x69
   |       |       |   Payload: { 1721 : { ...
   |       |       |
   |<-------------+      Code: 2.05 (Content)
   |       | 2.05  |     Token: 0x69
   |       |       |   Payload: { 1721 : { ...
   |       |       |
     ....    ....
```

# Amplification attack using multicast and observe

- If each response have an amplification factor of $a$, and there there are $m$ servers, and each server sends $n$ responses, the total **amplification factor is $amn$.**

- Note that the servers usually do not know the variable $m$.

```
Client   Foe   Server
   |      |      |
   |      +----->|      Code: 0.01 (GET)
   |      | GET  |     Token: 0x44
   |      |      |   Observe: 0
   |      |      |  Uri-Path: temperature
   |      |      |
   |<------------+      Code: 2.05 (Content)
   |      | 2.05 |     Token: 0x44
   |      |      |   Observe: 217
   |      |      |   Payload: "301.2 K"
   |      |      |
   |<------------+      Code: 2.05 (Content)
   |      | 2.05 |     Token: 0x44
   |      |      |   Observe: 363
   |      |      |   Payload: "293.4 K"
   |      |      |
     ....   ....
   |<------------+      Code: 2.05 (Content)
   |      | 2.05 |     Token: 0x44
   |      |      |   Observe: 218
   |      |      |   Payload: "301.5 K"
   |      |      |
   |<------------+      Code: 2.05 (Content)
   |      | 2.05 |     Token: 0x44
   |      |      |   Observe: 364
   |      |      |   Payload: "293.6 K"
   |      |      |
     ....   ....
```

# There is a need for harder requirements

- CORE has considered amplification attacks since the start, but the current recommendations are a bit soft:
  - RFC 7252: *"**large** amplification factors **SHOULD NOT** be provided in the response if the request is not authenticated"*
  - RFC 7252: *"**SHOULD NOT** accept multicast requests that can not be authenticated in some way"*
  - RFC 7252: *"**If possible**, a CoAP server **SHOULD** limit the support for multicast requests"*
  - RFC 7641: *"**MUST** strictly limit the number of notifications that it sends between receiving acknowledgements that confirm the actual interest of the client in the data; i.e., any notifications sent in non-confirmable messages **MUST** be interspersed with confirmable messages. **Note that an attacker may still spoof the acknowledgements"***
  - draft-ietf-core-groupcomm-bis-04: *"it is **generally NOT RECOMMENDED** to use CoAP group communication in NoSec mode"*

- QUIC [RFC9000] mandates "an endpoint **MUST limit the amount of data it sends to the unvalidated address to three times the amount of data received from that address**" without exceptions. This approach should be seen as current best practice.

- DDoS is a major problem. Networks and services are targeted by CoAP Amplification attacks. This tarnishes CoAP's reputation. Even if we can not fix existing deployments, CORE should make sure to not make it worse.

- **RFC 7252, RFC 7641, and draft-ietf-core-groupcomm-bis needs to be augmented with strict normative requirements (MUST) on implementations similar to QUIC with a specified anti-amplification limit.**
  It should be clear that devices used in DDoS attacks are violating IETF requirements.
  - **How?, Where?, When?**

# Group Communication for the Constrained Application Protocol (CoAP)

draft-ietf-core-groupcomm-bis-04

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

IETF 111, CoRE WG, July 28th, 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 (e.g., 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)

## Update from v -04

**Revised caching model – based on feedback from the June CoRE interim [1]**

- Freshness model, for the origin client
  - New members can join the group at any time; a local cache entry for responses to a group request may not cover all the responses sent since the latest cache refresh. This needs rules.
  - The client always sends out a group request, unless the client has fresh responses cached for all group servers. This is possible only when the client has a full, up-to-date knowledge of the group membership.

- Validation model, between origin client and origin servers
  - Simple and based on the ETag Option in group request/response, as normally used.
  - The server SHOULD (but is not required to) embed a compact, server-specific ID as ETag value.
  - The client needs to handle potential cases of 'value conflict' in ETags from different servers.
    - If responses from two servers have the same ETag value, it's not possible to validate only one
  - «Legacy» servers not aware of this ETag feature will just ignore the option (=ok)

[1] https://datatracker.ietf.org/doc/minutes-interim-2021-core-07-202106091600/

## Update from v -04

**Revised caching model – based on feedback from the June CoRE interim [1]**

- Caching model at a proxy
  - Creation and maintenance of cache entries
  - Freshness model like for the origin client, with more details about how/when serving from a cache entry
  - Case with end-to-end security based on Cacheable OSCORE
    - https://datatracker.ietf.org/doc/draft-amsuess-core-cachable-oscore/

- Response re-validation between proxy and group Servers
  - Based on the ETag option, like between the origin client and the group Servers

- Response re-validation between client and proxy
  - Based on a new Group-ETag option

- All the above moved to *draft-tiloca-core-groupcomm-proxy* as more appropriate

[1] https://datatracker.ietf.org/doc/minutes-interim-2021-core-07-202106091600/

# Update from v -04

**Processed review and comments from John Mattsson [2] – To be completed**

1. Make more general to cover group communication – Done
   - Not necessarily UDP over IP multicast, although it is the default transport

2. Make more general to about security group communication – Done
   - Not necessarily Group OSCORE, although it is the default security solution

3. Expectations from Group OSCORE and Echo Option about amplification / DoS – Done
   - The problem is mitigated by using Echo, but not prevented altogether

4. Make it clearer what is added/replaced in the updated/obsoleted documents – TODO

5. Explicit dedicated considerations on amplification attacks and DoS
   - Added new Section 6.3 "Risk of amplification" – Need for feedback and possible additional input
   - The NoSec mode is NOT RECOMMENDED and strongly discouraged; examples are given when it can still be acceptable, as discussed in the June CoRE interim. In any other case, security MUST be used.

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

## Update from v -04

- New Section 5.3 – Valid security cases with forward/reverse proxies
  - With forward/reverse proxy → Group OSCORE for e2e security over client ↔ servers
  - With a totally trusted reverse proxy acting entirely on behalf of the client, admit also:
    - Hop-by-hop security over client ↔ proxy
    - Group OSCORE over proxy ↔ servers
  - Further details on security in different legs are left to *draft-tiloca-core-groupcomm-proxy*

- Clarified interaction between Observe and No-Response Options

- Added informative reference to *draft-ietf-core-new-block*
  - Servers MUST ignore multicast requests that contain the Q-Block2 Option.

- Open point on terminology – Issue #24
  - Change "backward/forward security" to "backward/forward secrecy" ? Opinions/input ?

# Next steps

- Finish addressing the comments from John Mattsson [2]
  - Consider the latest points on amplification raised for *draft-mattsson-core-coap-attack*
  - Make it clearer what is added/replaced in the updated/obsoleted documents

- (Finish to) address the few remaining Github issues [3], also covering the points above

- Some specific functionalities left for testing in the CoAP implementation
  - Block2 in a multicast request, followed by Block2 unicast requests to each server

- Next version can be ready for WGLC

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

[3] 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-12

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

IETF 111, CoRE WG, July 28th, 2021

## Update since the March meeting

- Version -12 submitted

- Main updates
  - Recycling of Group IDs in the group (Christian)
  - Security of using an identity public key for both signing and Diffie-Hellman (Ben [1][2])
  - Major changes to the message processing, especially in group mode (John)
  - Clarified security properties

[1] https://mailarchive.ietf.org/arch/msg/core/ujj_I-LlqW9fq__quh-YqKS0fF0/

[2] https://mailarchive.ietf.org/arch/msg/core/YRNXvtiFmHLk5YkXK8-uJg-t3NU/

## Updates from -12

- **Recycling of Group IDs in a group – Reminder: the Group ID changes when rekeying**
  - It used to be forbidden, to prevent a notification from matching two long-lived observations

- The Group Manager (GM) retains the Gid obtained by a node when 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 and thus terminate 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

# Updates from -12

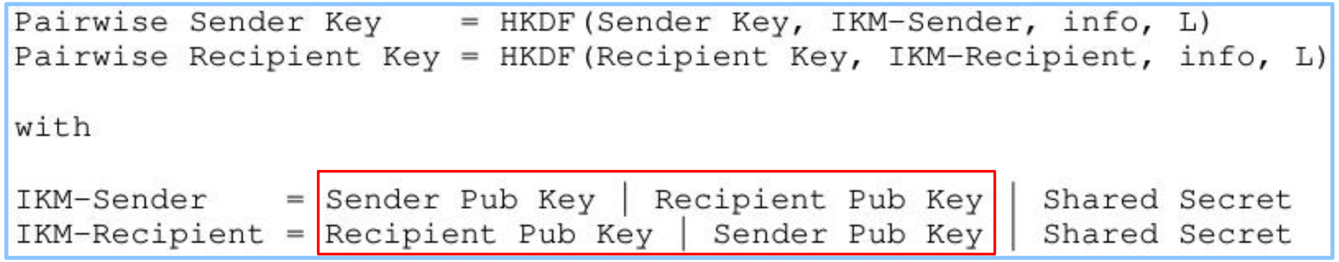
- **Security of using an identity key for both signing and Diffie-Hellman** [3][4] – #72 #73
  - If signing keys use Ed25519 (Ed448) they are converted to Curve25519 (Curve448) for DH
  - The computed DH secret is used to generate encryption keys for the pairwise mode
  - Both uses have the same goal and policy: group communication under a Security Context

- This double-use deviates from common best practices $\rightarrow$ Security has to be well proven
  - Build on and extend the proof at [5], as focused on (but not limited to) ECIES settings
  - Now proven to be secure specifically in Group OSCORE, see [6] – Thanks to Erik Thormarker!

[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
[6] https://eprint.iacr.org/2021/509.pdf

# Updates from -12

› **Security of using an identity key for both signing and Diffie-Hellman**
  – Adapted derivation of pairwise keys, explicitly involving the two peers' public keys (see Section 2.4.1)



```
Pairwise Sender Key    = HKDF(Sender Key, IKM-Sender, info, L)
Pairwise Recipient Key = HKDF(Recipient Key, IKM-Recipient, info, L)

with

IKM-Sender    = Sender Pub Key | Recipient Pub Key | Shared Secret
IKM-Recipient = Recipient Pub Key | Sender Pub Key | Shared Secret
```

› (Cryptographic) security considerations supporting the procedure's correctness
  – For ECDSA signing keys, [6] proves that the proof from [5] is applicable also to Group OSCORE
  – For EdDSA signing keys, [6] builds a new proof from the one in [5], specific to Group OSCORE
  – Dedicated discussion on converting from Ed25519 (Ed448) to Curve25519 (Curve448)

[5] https://eprint.iacr.org/2011/615.pdf
[6] https://eprint.iacr.org/2021/509.pdf

## Updates from -12

- **Admitted formats of public keys (see Section 2.3)**
  - Must provide the full set of information about the public key algorithm
  - Relevant examples:
    - CWT, see *RFC8392*
    - Unprotected CWT claim set, see *draft-ietf-rats-uccs*
    - X.509 certificates, see *RFC7925*
    - C509 certificates, see *draft-ietf-cose-cbor-encoded-cert*

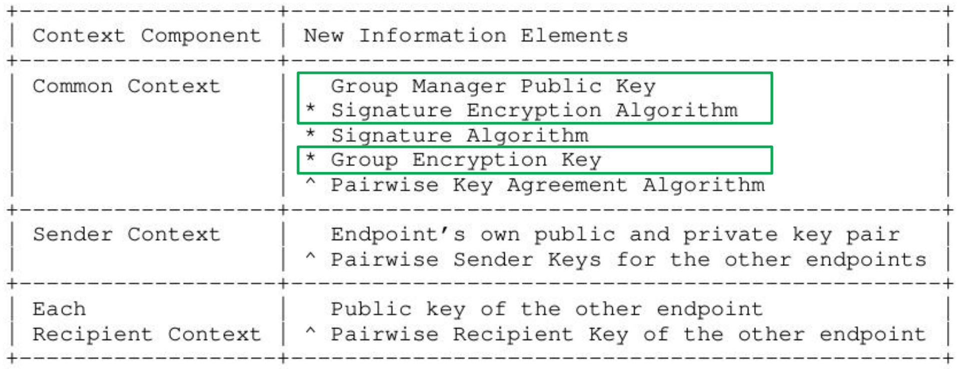
```
{ /CWT claims list/
  2: "42-50-31-FF-EF-37-32-39", /sub/
  8: { /cnf/
      1: { /COSE_Key/
      1: 1,
     -1: 4, /X25519/
     -2: h'b1a3e89460e88d3a8d54211dc95f0b
           903ff205eb71912d6db8f4af980d2db83a',
         }
  }
}
```

- All public keys used in the group
  - Must have the same single format used in the group
  - Must be compatible with the algorithm and related parameters used in the group

# Updates from -12

- **Now an OSCORE group can work in three ways**
  - “Group mode” only; “Pairwise mode” only (NEW); both modes
  - Group members know what is used in the group – Learned when joining, or at early group discovery

- **Consistent revision of the Security Context – <u>Additions</u> to OSCORE (RFC8613)**
  - (*) Specific to the group mode
  - (^) Specific to the pairwise mode
  - Some recently added parameters



| Context Component | New Information Elements |
|---|---|
| Common Context | Group Manager Public Key<br>* Signature Encryption Algorithm<br>* Signature Algorithm<br>* Group Encryption Key<br>^ Pairwise Key Agreement Algorithm |
| Sender Context | Endpoint's own public and private key pair<br>^ Pairwise Sender Keys for the other endpoints |
| Each Recipient Context | Public key of the other endpoint<br>^ Pairwise Recipient Key of the other endpoint |

Figure 1: Additions to the OSCORE Security Context. The optional elements labeled with * (with ^) are present only if the group uses the group mode (the pairwise mode).

- **In group mode**
  - Encrypt with Signature Encryption Algorithm
  - Sign with Signature Algorithm
  - <u>1 common</u> Group Encryption Key derived like a Sender/Recipient Key, through HKDF()

- **In pairwise mode**
  - Encrypt with AEAD Algorithm (RFC 8613)
  - Derive pairwise keys with Pairwise Key Agreement Algorithm

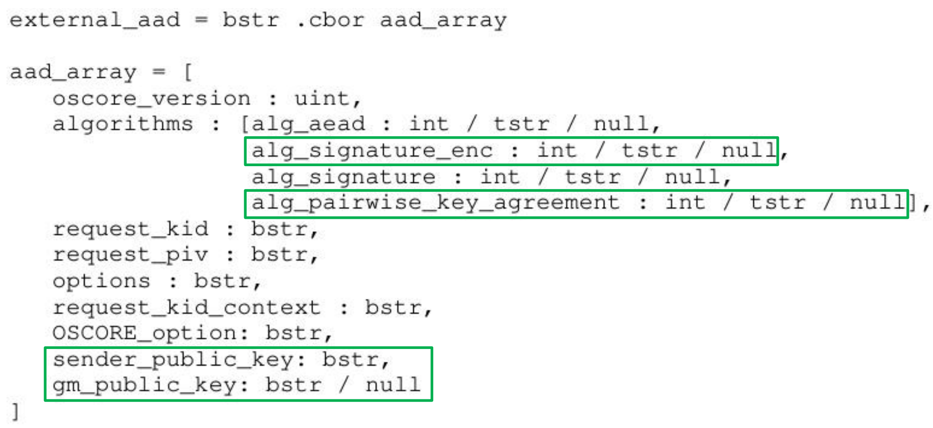
# Updates from -12

- **Now an OSCORE group can work in three ways**
  - “Group mode” only; “Pairwise mode” only (NEW); both modes
  - Group members know what is used in the group – Learned when joining, or at early group discovery

- **Consistent revision of the External AAD**
  - Some recently added parameters

- **Removed some parameters**
  - Capabilities and properties of algorithms
  - Now embedded in the public keys

- **Added Group Manager public key**
  - More accurate “group description” covered by the external_aad
  - Prevents a “group cloning attack”, discussed in the security considerations



```
external_aad = bstr .cbor aad_array

aad_array = [
   oscore_version : uint,
   algorithms : [alg_aead : int / tstr / null,
                 alg_signature_enc : int / tstr / null,
                 alg_signature : int / tstr / null,
                 alg_pairwise_key_agreement : int / tstr / null],
   request_kid : bstr,
   request_piv : bstr,
   options : bstr,
   request_kid_context : bstr,
   OSCORE_option: bstr,
   sender_public_key: bstr,
   gm_public_key: bstr / null
]
```

## Updates from -12

- **In group mode, the countersignature is <u>separately</u> encrypted**
  - Encrypt COSE plaintext → Sign the COSE ciphertext → Encrypt the countersignature <u>separately</u>
  - Group privacy: not possible to track a user across two groups, unless the tracker is a member of both

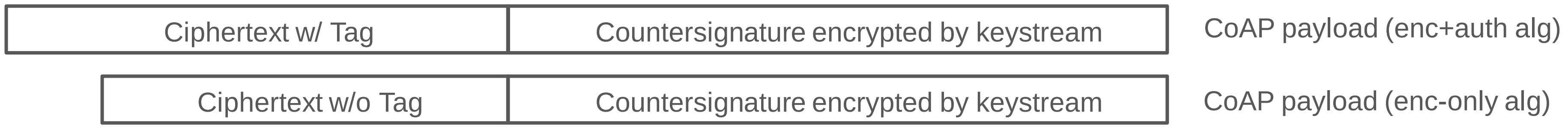
- **Countersignature encryption**
  - ENC_COUNTERSIGNATURE = SIGNATURE XOR KEYSTREAM
  - KEYSTREAM derived on a per-message basis from the new Group Encryption Key

- **Keystream derivation**
  - KEYSTREAM = HKDF(salt, IKM, info, length)
    - salt : Partial IV used to protect the message
    - IKM : Group Encryption Key
    - info : [Sender ID of the Partial IV generator; Group ID; True/False (req/resp); length]
    - length : Same length as the countersignature

- **Receiver side: verify the countersignature first, <u>then</u> decrypt the COSE ciphertext**

# Updates from -12

- **In group mode, admit also encryption-only algorithms**
  - Expected registration as COSE algorithms
  - Some applications may not desire/afford both a signature and an integrity Tag in each message
  - Source authentication still ensured, thanks to the signature and what is covered by the external_aad


Ciphertext w/ Tag
Countersignature encrypted by keystream
CoAP payload (enc+auth alg)
Ciphertext w/o Tag
Countersignature encrypted by keystream
CoAP payload (enc-only alg)

- **Even with encryption-only algorithms, i.e. without integrity Tag, we still want to …**
  - Admit external signature checkers <u>and</u> always verify group membership when receiving a message
  - But a group member might leave, become external signature checker and inject re-signed messages
  - Other nodes would believe the sender to still be a group member – They still have its public key!

**Therefore …**

# Updates from -12

- **Stricter management of group keying material, including group rekeying**
  - If a node leaves, the GM must rekey the group
  - Rekeying messages must specify the “stale Sender IDs” of the current key generation
    - Sender IDs earlier relinquished due to a change requested by the owning node; or
    - Sender IDs belonging to a leaving node

- **The rekeyed group members obtain the stale Sender IDs**
  - They delete the associated public keys used in the group
  - They delete the associated Recipient Contexts in the group

  → They rely on their owned public keys to assert the group membership of sender endpoints

- **A group member may miss a group rekeying instance**
  - The GM allows to retrieve an aggregate set of stale Sender IDs for the most recent key generations

- **Detailed mechanics for achieving this are out of scope for Group OSCORE**
  - The GM defined in *draft-ietf-ace-key-groupcomm-oscore* provides these functionalities

## Updates from -12

- **New Section 8.5 on external signature checkers**
  - Guidelines on message processing and corner cases

- **Removed three Appendices**
  - “No Verification of Signatures in Group Mode” – Can’t be considered secure
  - “Example Values with COSE Capabilities” – Moot (see admitted formats of public keys)
  - “Parameter Extensibility for Future COSE Algorithms” – Moot (see admitted formats of public keys)

- **Extended security considerations**
  - Revised considerations on the group mode, emphasizing its security properties
  - New Section 10.8 about the prevented “group cloning attack”
  - More cryptographic considerations, on using the same key pair for signing and for Diffie-Hellman

## Next steps

- Improve security considerations
  - Mostly on the group mode, based on the latest changes
- Double-check that the few open Github issues are well addressed
- Submit v -13
  - If no further issues arise, it should be ready for a 2nd WGLC

Thank you!

Comments/questions?

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

# Combining EDHOC and OSCORE

draft-ietf-core-oscore-edhoc-01

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

IETF 111, CoRE WG, July 28th, 2021

# Recap

- EDHOC is a lightweight authenticated key exchange developed in LAKE WG
  - Main use case: keying OSCORE for establishing a Security Context
  - Normal workflow: two round-trips

- This draft combines EDHOC (run over CoAP) with OSCORE
  - EDHOC message_3 combined with the first OSCORE-protected request
    - A single EDHOC + OSCORE request, transporting both
  - Achieve a minimum number of round trips required
    - To set up the OSCORE Security Context
    - To complete the first OSCORE transaction with that Context

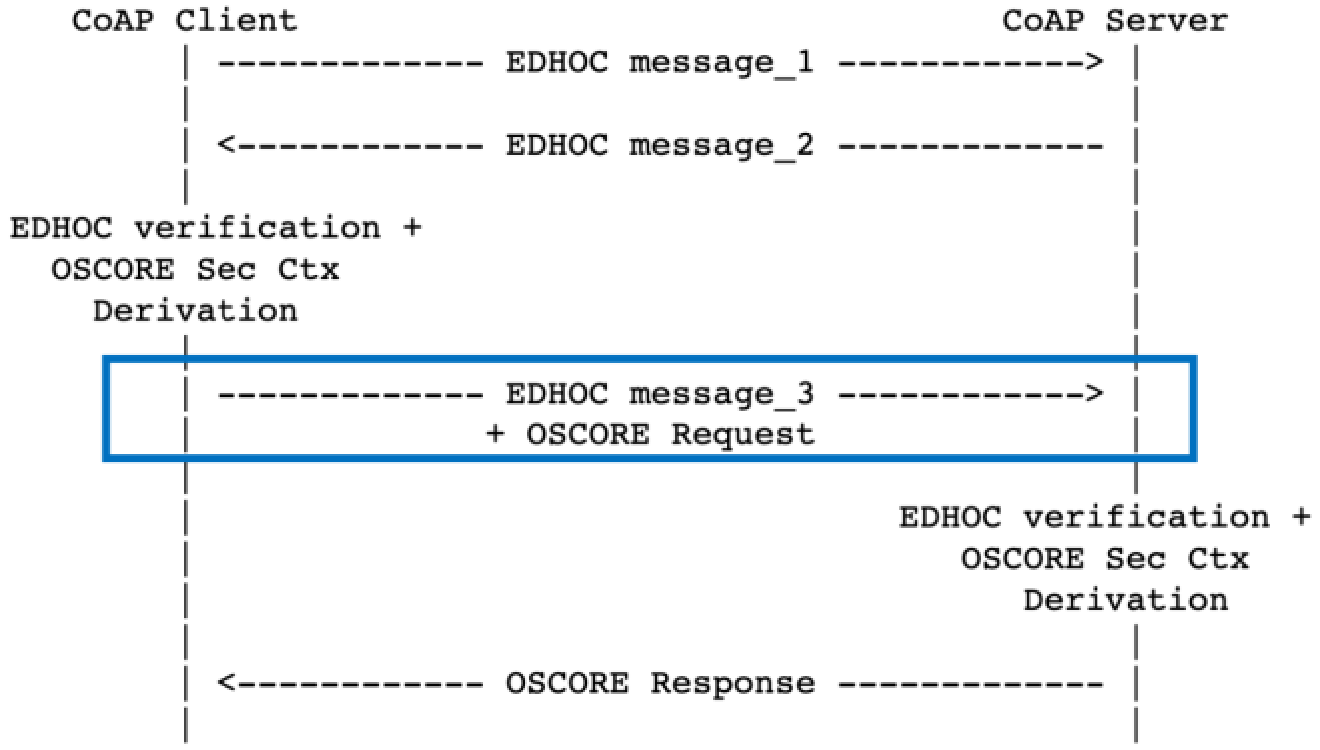
- More details that are too detailed for EDHOC? (To be discussed)

# Plain way: EDHOC then OSCORE

```
   CoAP Client                                          CoAP Server
        | ------------- EDHOC message_1 ------------> |
        |                                             |
        | <------------ EDHOC message_2 ------------- |
        |                                             |
EDHOC verification                                    |
        |                                             |
        | ------------- EDHOC message_3 ------------> |
        |                                             |
        |                                     EDHOC verification
        |                                             |
OSCORE Sec Ctx                                 OSCORE Sec Ctx
  Derivation                                     Derivation
        |                                             |
        | -------------- OSCORE Request ------------> |
        |                                             |
        | <------------ OSCORE Response ------------- |
        |                                             |
```

Figure 1: EDHOC and OSCORE run sequentially

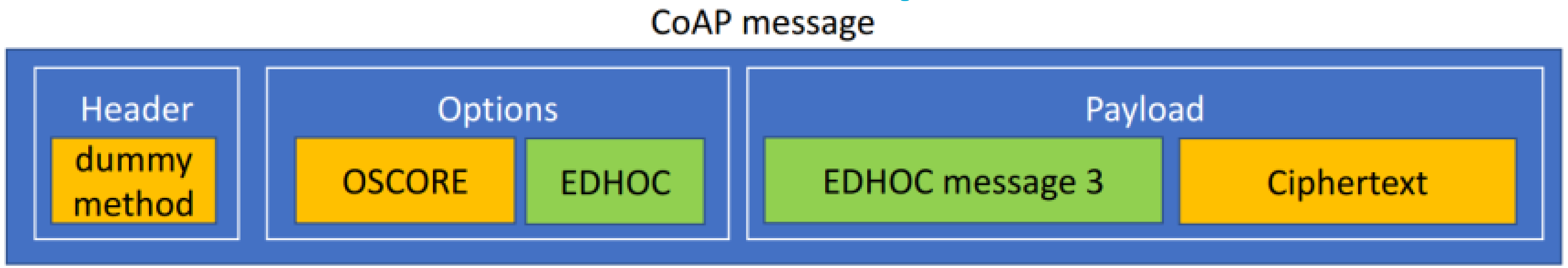
# New way: EDHOC + OSCORE Request

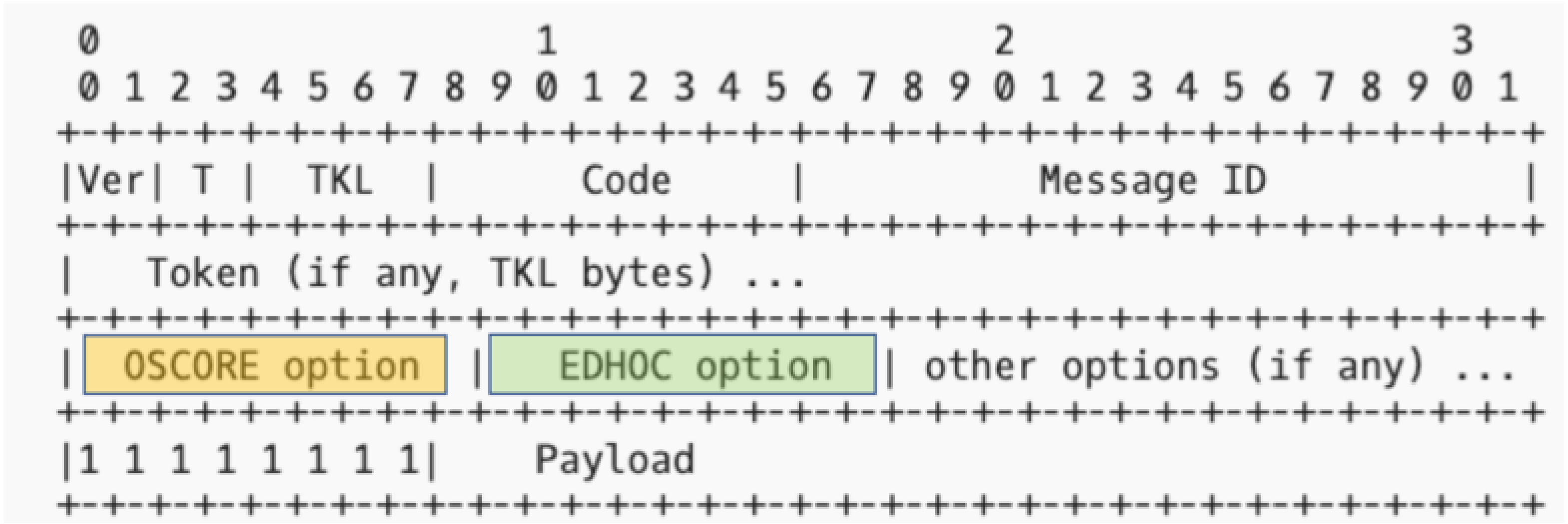


```
   CoAP Client                                  CoAP Server
        | ------------- EDHOC message_1 ------------> |
        |                                             |
        | <------------ EDHOC message_2 ------------- |
        |                                             |
EDHOC verification +                                  |
  OSCORE Sec Ctx                                      |
    Derivation                                        |
        |                                             |
        | ------------- EDHOC message_3 ------------> |
        |            + OSCORE Request                 |
        |                                             |
        |                                EDHOC verification +
        |                                   OSCORE Sec Ctx
        |                                     Derivation
        |                                             |
        | <------------ OSCORE Response ------------- |
        |                                             |
```

Figure 2: EDHOC and OSCORE combined

# EDHOC + OSCORE request





```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

## Updates from IETF 110

- Settled on EDHOC Option number 21
  - Instead of 13 in the previous version
  - Better since 21 works just as well, and other use cases may need 13 more
  - Request for early IANA allocation in the CoAP Option Numbers registry

- Extended and improved background about EDHOC
  - Based on feedback from the IETF 110 meeting

- Align with updates to EDHOC draft
  - Especially encoding of connection identifiers (can now be both CBOR bstr/int)

- Improvements and updating examples

## Updates from IETF 110

- As triggered by the EDHOC changes on connection identifiers

- Defined a conversion method from EDHOC IDs to OSCORE Sender/Recipient IDs
  - Initially defined here, from Christian's proposal
  - After discussions, this was moved to the EDHOC document in LAKE

- Defined a conversion method from OSCORE Sender/Recipient IDs to EDHOC IDs
  - Note: there are 2 "equivalent" EDHOC IDs for each OSCORE ID, i.e., CBOR *int* or *bstr*
  - This method deterministically picks either the *int* or the *bstr* EDHOC identifier
    - Required for the EDHOC+OSCORE request, as including an OSCORE Sender ID
    - Performance advantage: the selected identifier is the smallest of the two
  - Now in Appendix A. Move to the document body?

## Open point

- Previous attempts were made to have CoRE-specific content here
  - Derivation of OSCORE Security Context: EDHOC ==> Here ==> EDHOC
  - Conversion from EDHOC ID to OSCORE ID: Here ==> EDHOC

- Consider scope expansion: "Profiling the Use of EDHOC for CoAP and OSCORE"
  - Already have content about conversion from OSCORE ID to EDHOC ID (previous slide)
  - Use of a potential URI compression option (Christian's separate proposal)
  - Web linking
    - *rt=edhoc* might be already registered in the EDHOC draft
    - More target attributes aligned with the applicability statement can be added here
  - Any further things judged to be too detailed for the EDHOC draft

Opinions? More input?

## Next Steps

- Request for early IANA allocation: Option number 21 for the "EDHOC" CoAP option

- Planned updates to the draft
  - Make text/figures consistent with the use of content-format in the EDHOC draft
  - Notes on the EDHOC applicability statement
    - Support for EDHOC+OSCORE request and ID conversion method

- Update the implementation
  - We have running code, only based on EDHOC v –07
  - First need to update the EDHOC implementation to its v –08 (ongoing :-)

- Need for reviews

Thank you!

Comments/questions?

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

~~AEAD Key Usage Limits in OSCORE~~

# Key Update for OSCORE

draft-hoeglund-core-oscore-key-limits-01

**Rikard Höglund**, RISE
Marco Tiloca, RISE

IETF 111, CoRE WG, July 28th, 2021

# Draft Overview (1/2)

- OSCORE (RFC8613) uses AEAD algorithms to provide security
  - Confidentiality and Integrity

- Need to follow limits in key usage and failed decryptions, before rekeying
  - Otherwise, it is possible to break the security properties of the AEAD algorithm
  - Reference **draft-irtf-cfrg-aead-limits-03**

- (1) AEAD limits and their impact on OSCORE
  - Defining appropriate limits for OSCORE
  - Originally starting from the same assumptions in TLS
  - Revisited based on John Mattsson's input at the April CoRE interim
    - https://datatracker.ietf.org/meeting/110/materials/slides-110-saag-analysis-of-usage-limits-of-aead-algorithms-00.pdf

## Draft Overview (2/2)

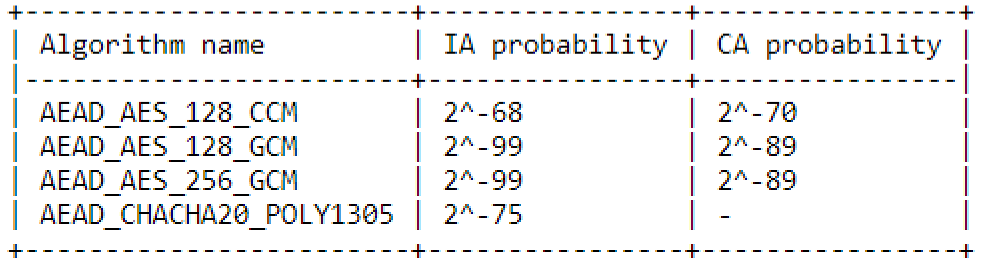
- (2) Updates to OSCORE
  - Counters in the Security Context: key encryption use (q) and invalid decryptions (v)
  - Necessary steps to take during message processing (counting)
  - Update the keys when the limits are exceeded (rekeying)

- (3) Defined a new method for rekeying OSCORE (**new**)
  - Loosely inspired by Appendix B.2 of OSCORE
  - Goal: renew the Master Secret and Master Salt; derive new keys from those
  - Achieves Perfect Forward Secrecy

# Key Limits (1/2)

- Selected fixed values for ‘q’, ‘v’, and ‘l’
  - q = 2^20, v = 2^20 and l = 2^8 // 'l' is the max message length in cipher blocks
  - Based on earlier discussions and John Mattsson’s presentation

- Table with ‘IA’ and ‘CA’ probabilities based on those values
  - These are based on the formulas in the CFRG document



| Algorithm name | IA probability | CA probability |
|---|---|---|
| AEAD_AES_128_CCM | 2^-68 | 2^-70 |
| AEAD_AES_128_GCM | 2^-99 | 2^-89 |
| AEAD_AES_256_GCM | 2^-99 | 2^-89 |
| AEAD_CHACHA20_POLY1305 | 2^-75 | - |

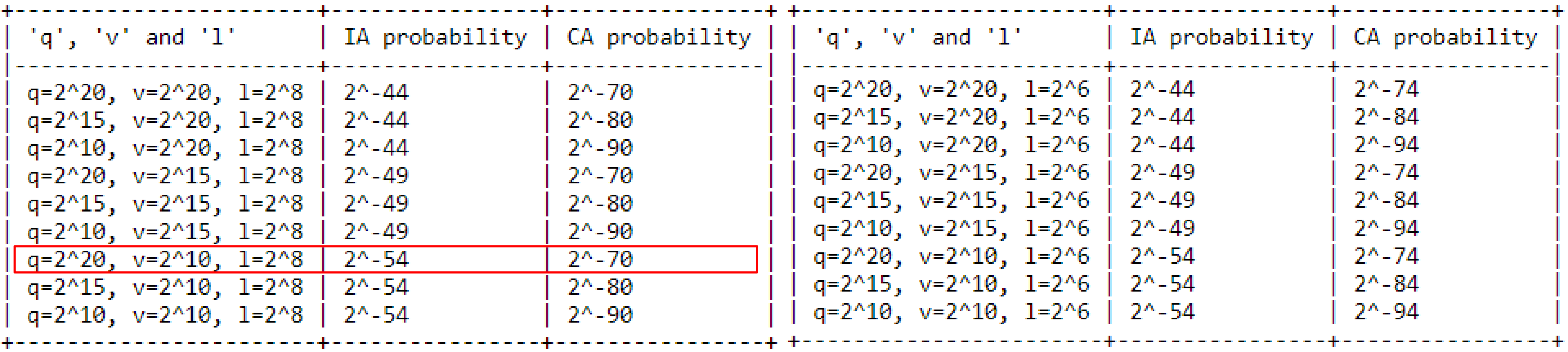
Figure 1: Probabilities for algorithms based on chosen q, v and l values.

Integrity Advantage (IA): Probability of breaking integrity properties

Confidentiality Advantage (CA): Probability of breaking confidentiality properties

# Key Limits (2/2)

- Specific look at AEAD_AES_128_CCM_8
  - Due to short Tag length the limits can be most problematic here
- Table with ‘IA’ and ‘CA’ probabilities for various values of ‘q’, ‘v’ and ‘l’



| 'q', 'v' and 'l' | IA probability | CA probability |
|---|---|---|
| q=2^20, v=2^20, l=2^8 | 2^-44 | 2^-70 |
| q=2^15, v=2^20, l=2^8 | 2^-44 | 2^-80 |
| q=2^10, v=2^20, l=2^8 | 2^-44 | 2^-90 |
| q=2^20, v=2^15, l=2^8 | 2^-49 | 2^-70 |
| q=2^15, v=2^15, l=2^8 | 2^-49 | 2^-80 |
| q=2^10, v=2^15, l=2^8 | 2^-49 | 2^-90 |
| q=2^20, v=2^10, l=2^8 | 2^-54 | 2^-70 |
| q=2^15, v=2^10, l=2^8 | 2^-54 | 2^-80 |
| q=2^10, v=2^10, l=2^8 | 2^-54 | 2^-90 |

| 'q', 'v' and 'l' | IA probability | CA probability |
|---|---|---|
| q=2^20, v=2^20, l=2^6 | 2^-44 | 2^-74 |
| q=2^15, v=2^20, l=2^6 | 2^-44 | 2^-84 |
| q=2^10, v=2^20, l=2^6 | 2^-44 | 2^-94 |
| q=2^20, v=2^15, l=2^6 | 2^-49 | 2^-74 |
| q=2^15, v=2^15, l=2^6 | 2^-49 | 2^-84 |
| q=2^10, v=2^15, l=2^6 | 2^-49 | 2^-94 |
| q=2^20, v=2^10, l=2^6 | 2^-54 | 2^-74 |
| q=2^15, v=2^10, l=2^6 | 2^-54 | 2^-84 |
| q=2^10, v=2^10, l=2^6 | 2^-54 | 2^-94 |

Figure 2: Probabilities for AEAD_AES_128_CCM_8 based on chosen q, v and l values.

- From the considered values the best triple is (q = 2^20, v= 2^10, l = 2^8)
  - The question is if an IA of 2^-54 is good enough?

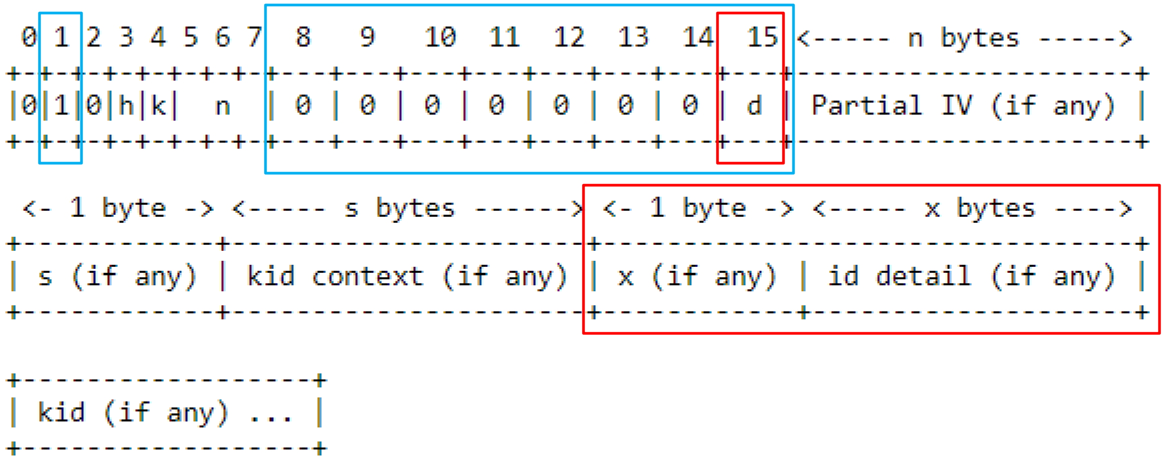
## OSCORE Key Update method (1/3)

- Defined a new method for rekeying OSCORE
  - Client and server exchange two nonces R1 and R2
  - *UpdateCtx()* function for deriving new OSCORE Security Context using the nonces
  - Current Sec Ctx (to renew) ==> Intermediate Sec Ctx ==> **New Sec Ctx**

- Properties
  - Only one intermediate Security Context is derived
  - The ID Context does not change
  - Can be initiated by either the client or server
  - It is robust and secure against a peer rebooting
  - The procedure completes in one round-trip (after that, the new context can be used)
  - Compatible with possible prior key establishment through the EDHOC protocol

# OSCORE Key Update method (2/3)

- Key update messages are OSCORE-protected and self-evident
- OSCORE Option: defined the use of flag bit 1 to signal presence of flag bits 8-15
- Defined flag bit 15 -- 'd' -- to indicate:
  - This is a OSCORE key update message
  - "id detail" is specified (length + value); used to transport a nonce for the key update



```
 0 1 2 3 4 5 6 7  8   9   10  11  12  13  14  15 <----- n bytes ----->
+-+-+-+-+-+-+-+-+---+---+---+---+---+---+---+---+---------------------+
|0|1|0|h|k|  n  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | d | Partial IV (if any) |
+-+-+-+-+-+-+-+-+---+---+---+---+---+---+---+---+---------------------+

 <- 1 byte -> <----- s bytes ------> <- 1 byte -> <----- x bytes ---->
+------------+----------------------+---------------------------------+
| s (if any) | kid context (if any) | x (if any) | id detail (if any) |
+------------+----------------------+------------+--------------------+

+------------------+
| kid (if any) ... |
+------------------+
```

Figure 3: The OSCORE option value, including 'id detail'

# OSCORE Key Update method (3/3)

```
                        Client                  Server
                      (initiator)             (responder)
                           |                       |
Generate R1                |                       |
                           |                       |
CTX_1 =                    |                       |
  updateCtx(R1,            |                       |
            CTX_OLD)       |                       |
                           |                       |
                           |      Request #1       |
Protect with CTX_1         |---------------------->|
                           | OSCORE Option:        | CTX_1 =
                           |   ...                 |   updateCtx(R1,
                           |   d flag: 1           |             CTX_OLD)
                           |   ...                 |
                           |   ID Detail: R1       | Verify with CTX_1
                           |   ...                 |
                           |                       | Generate R2
                           |                       |
                           |                       | CTX_NEW =
                           |                       |   updateCtx(R1|R2,
                           |                       |             CTX_OLD)
                           |      Response #1      |
                           |<----------------------| Protect with CTX_NEW
CTX_NEW =                  | OSCORE Option:        |
  updateCtx(R1|R2,         |   ...                 |
            CTX_OLD)       |   d flag: 1           |
                           |   ...                 |
Verify with CTX_NEW        |   ID Detail: R1|R2    |
                           |   ...                 |
Discard CTX_OLD            |                       |
```

```
updateCtx( N, CTX_IN ) {

  CTX_OUT       // The new Security Context
  MSECRET_NEW   // The new Master Secret
  MSALT_NEW     // The new Master Salt
  if <the original Security Context was established through EDHOC> {
    EDHOC-KeyUpdate( N )
    // This results in updating the key PRK_4x3m of the EDHOC session,
    // i.e., PRK_4x3m = Extract( N, PRK_4x3m )

    MSECRET_NEW = EDHOC-Exporter( "OSCORE Master Secret", key_length )
      = EDHOC-KDF(PRK_4x3m, TH_4, "OSCORE Master Secret", key_length )

    MSALT_NEW = EDHOC-Exporter( "OSCORE Master Salt", salt_length )
      = EDHOC-KDF( PRK_4x3m, TH_4, "OSCORE Master Salt", salt_length )

  }
  else {
    Master Secret Length = < Size of CTX_IN.MasterSecret in bytes >

    MSECRET_NEW = HKDF-Expand-Label(CTX_IN.MasterSecret, Label,
                                    N, Master Secret Length)
                = HKDF-Expand(CTX_IN.MasterSecret, HkdfLabel,
                              Master Secret Length)

    MSALT_NEW = N;
  }
  < Derive CTX_OUT using MSECRET_NEW and MSALT_NEW,
    together with other parameters from CTX_IN >
  Return CTX_OUT;
}
```

## Summary and Next Steps

- Twofold update to OSCORE
  - Tracking and reacting to defined key limits, to preserve security of the AEAD cipher
  - New efficient key update procedure (rekeying) with Perfect Forward Secrecy
    - Initially planned as a separate draft
    - Preference at the April CoRE interim to have it in this document

- Take and adopt feedback on the new key limits and especially for CCM_8

- Main next steps are tracked as Gitlab issues in [1]

- Need for reviews, on both key limits and key update

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

Thank you!

Comments/questions?

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

# OSCORE-capable Proxies

*draft-tiloca-core-oscore-capable-proxies-00*

**Marco Tiloca**, RISE
Rikard Höglund, RISE

IETF 111, CoRE WG, July 28th, 2021

# Motivation

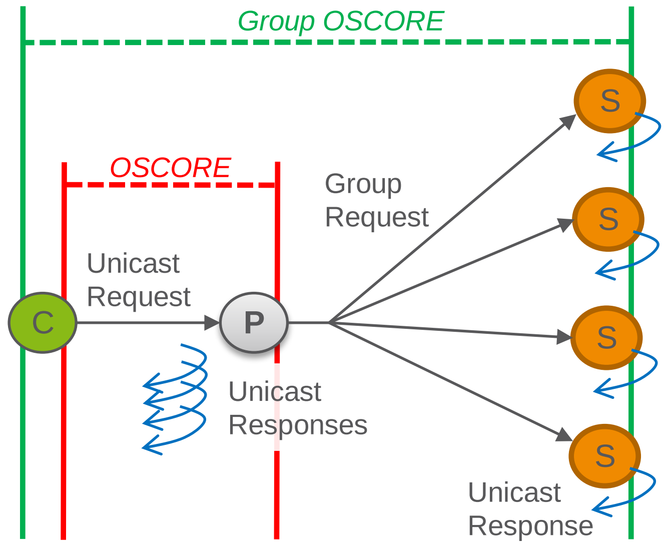
- › A CoAP proxy (P) can be used between client (C) and server (S)
  - – A security association might be required between C and P --- examples in next slides
- › It would be good to use OSCORE between C and P
  - – Especially, but not only, if C and S already use OSCORE also end-to-end
- › This is not defined and not admitted in OSCORE (RFC 8613)
  - – C and S are the only considered “OSCORE endpoints”
  - – It is forbidden to double-protect a message, i.e., both over C ↔ S and over C ↔ P
- › This started as an Appendix of *draft-tiloca-core-groupcomm-proxy*
  - – Agreed at IETF 110 [1] and at the June CoRE interim [2] to have a separate draft

[1] https://datatracker.ietf.org/doc/minutes-110-core-202103081700/

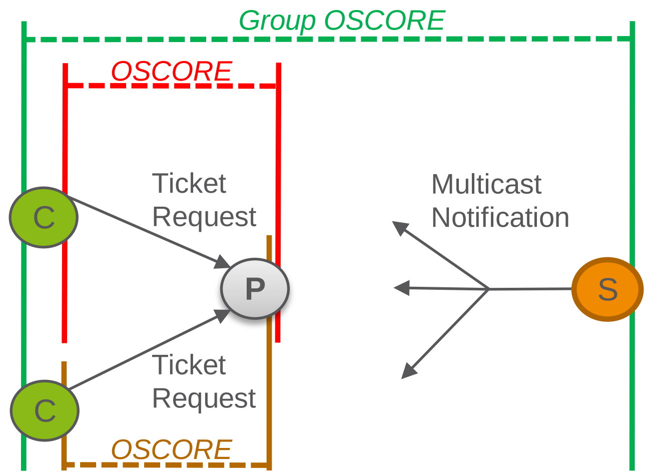
[2] https://datatracker.ietf.org/doc/minutes-interim-2021-core-07-202106091600/

# Use cases

- **CoAP Group Communication with Proxies**
  - *draft-tiloca-core-groupcomm-proxy*
  - CoAP group communication through a proxy
  - Possible e2e security with Group OSCORE
  - P must identify C through a security association before forwarding a request to the group



- **CoAP Observe Notifications over Multicast, with Group OSCORE for e2e security**
  - *draft-ietf-core-observe-multicast-notifications*
  - C provides P with a Ticket Request obtained from S
  - This allows P to correctly listen to multicast notifications sent by S
  - The provisioning of the Ticket Request to P should be protected over C ↔ P



# Use cases

- **OMA LwM2M Client and External Application Server**
  - *Lightweight Machine to Machine Technical Specification – Transport Binding*

    *OSCORE MAY also be used between LwM2M endpoint and non-LwM2M endpoint, e.g., between an Application Server and a LwM2M Client via a LwM2M server. Both the LwM2M endpoint and non-LwM2M endpoint MUST implement OSCORE and be provisioned with an OSCORE Security Context.*
  - The LwM2M Client may register to and communicate with the LwM2M Server using OSCORE
  - The LwM2M Client may communicate with an External Application Server, also using OSCORE
  - The LwM2M Server would act as CoAP proxy, forwarding outside the LwM2M domain

- **More generally, a proxy may want an OSCORE Security Context of its own**
  - E.g., it ensures the security of transport indication when OSCORE is used [3][4]

[3] https://datatracker.ietf.org/doc/draft-amsuess-core-transport-indication/

[4] https://mailarchive.ietf.org/arch/msg/core/RZH8pgyksEwtMYVE1MrPkj9opyg/

# Contribution

› Twofold update to RFC 8613

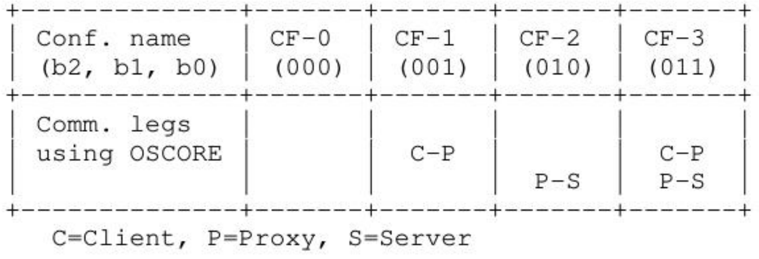
1. Define the use of OSCORE in a communication leg including a proxy
    › Between origin client/server and a proxy; or between two proxies in a chain
    › Not only an origin client/server, but also an intermediary can be an "OSCORE endpoint"

2. Explicitly admit double OSCORE protection – "OSCORE-in-OSCORE"
    – E.g., first protect end-to-end over C ↔ S, then further protect the result over C ↔ P
    – At most 2 OSCORE "layers" in the same message: 1 end-to-end; 1 between two adjacent hops

› Focus on OSCORE, but the same applies to Group OSCORE

# Leg independence

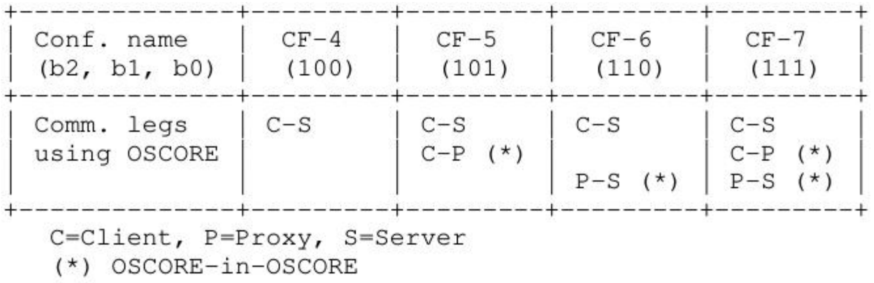
› Seamless support for different configurations
  – Configurations differ on whether OSCORE is used or not in a certain communication leg



| Conf. name (b2, b1, b0) | CF-0 (000) | CF-1 (001) | CF-2 (010) | CF-3 (011) |
|---|---|---|---|---|
| Comm. legs using OSCORE | | C-P | P-S | C-P P-S |

C=Client, P=Proxy, S=Server

Figure 1: Configurations without end-to-end security.



| Conf. name (b2, b1, b0) | CF-4 (100) | CF-5 (101) | CF-6 (110) | CF-7 (111) |
|---|---|---|---|---|
| Comm. legs using OSCORE | C-S | C-S C-P (*) | C-S P-S (*) | C-S C-P (*) P-S (*) |

C=Client, P=Proxy, S=Server
(*) OSCORE-in-OSCORE

Figure 2: Configurations with end-to-end security

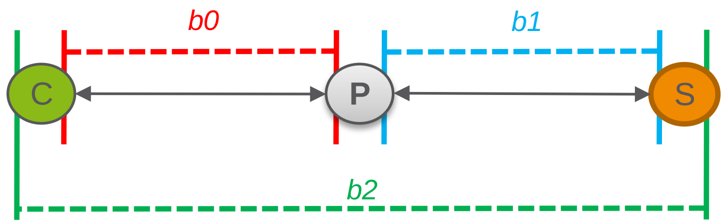
Naming convention: CF-X

X = b0 + (2 * b1) + (4 * b2)

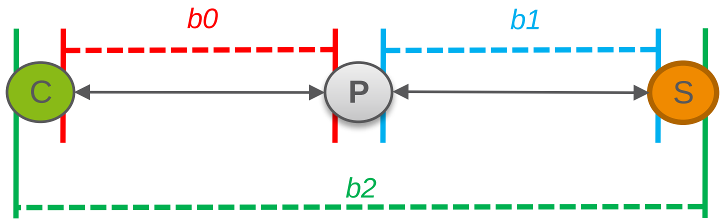
b0 : 1 if OSCORE over C ↔ P ; 0 otherwise

b1 : 1 if OSCORE over P ↔ S ; 0 otherwise

b2 : 1 if OSCORE over C ↔ S ; 0 otherwise



# High-level mechanics



- C request processing
  - (1) If b2 = 1, protect with OSCORE C ↔ S
  - (2) If b0 = 1, (further) protect with OSCORE C ↔ P
    - Encrypt options intended to P, e.g., Proxy-Scheme
    - Encrypt the OSCORE option from (1), if any

- P request processing
  - Visible proxy options → Forward to S
  - Absent proxy options && absent OSCORE option → Deliver to the application
  - Absent proxy options && Visible OSCORE option → Decrypt, as OSCORE C ↔ P
    - No proxy options in the decrypted request → Deliver to the application
    - Visible proxy options in the decrypted request → Forward to S
  - When forwarding to S
    - If b1 = 1, (further) protect with OSCORE P ↔ S
    - Encrypt the OSCORE option for C ↔ S, if any

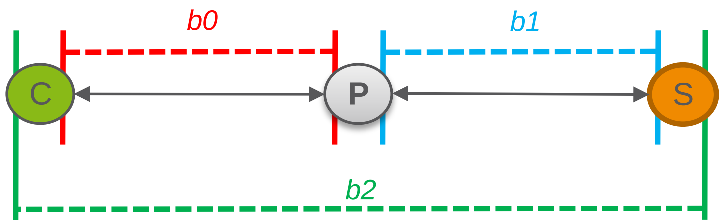
# High-level mechanics

- S request processing
  - Ready to find and process 1 or 2 OSCORE layers

- S response processing
  - (1) If b2 = 1, protect with OSCORE C ↔ S
  - (2) If b1 = 1, (further) protect with OSCORE P ↔ S
    - Encrypt options intended to P
    - Encrypt the OSCORE option from (1), if any

- P response processing
  - If b1 = 1, unprotect with OSCORE P ↔ S
  - When forwarding to C
    - If b0 = 1, (further) protect with OSCORE C ↔ P
    - Encrypt possible new added options intended to C
    - Encrypt the OSCORE option for C ↔ S, if any

- C response processing
  - Reverse of request processing; ready to find and process 1 or 2 OSCORE layers



## Summary and next steps

- › Proposed update to RFC 8613
  - – Define the use of OSCORE in a communication leg including a proxy
  - – Explicitly admit double OSCORE protection – “OSCORE-in-OSCORE”
  - – Useful for CoAP group communication, LwM2M external server, transport indication
- › Next steps
  - – Work on “Response caching” and “Chain of intermediaries”
  - – Mention the applicability for the security of transport indication [3]
- › The main mechanics are stable – Comments and reviews are welcome!

[3] https://datatracker.ietf.org/doc/draft-amsuess-core-transport-indication/

Thank you!

Comments/questions?

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