OPEN CONNECTIVITY FOUNDATION™

OSCORE: OCF Status and comments

Joint OCF IRTF T2TRG Call 2020-03-18 (CET)

Phil Hawkes (Qualcomm) phawkes@qti.qualcomm.com from OCF

Agenda

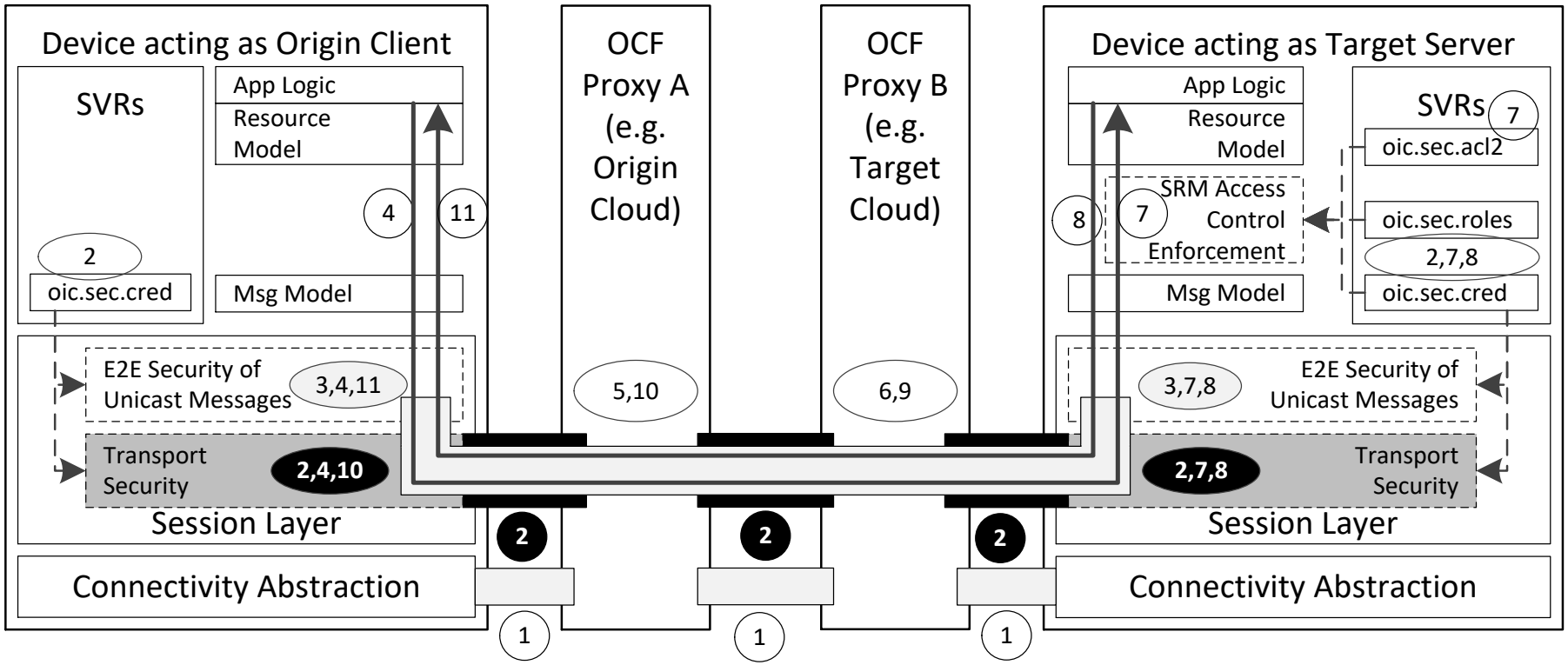
- Explain how OCF plans to use OSCORE for End to End Security of Unicast Messages.
- Explain how OCF expects to use ACE, EDHOC
  - Q for T2TRG: what is expected timeline for publication?
- OCF interim plan: "Directly Provisioned" OSCORE Security Contexts

- The content in this slide deck is not confidential

Terminology (for this discussion)

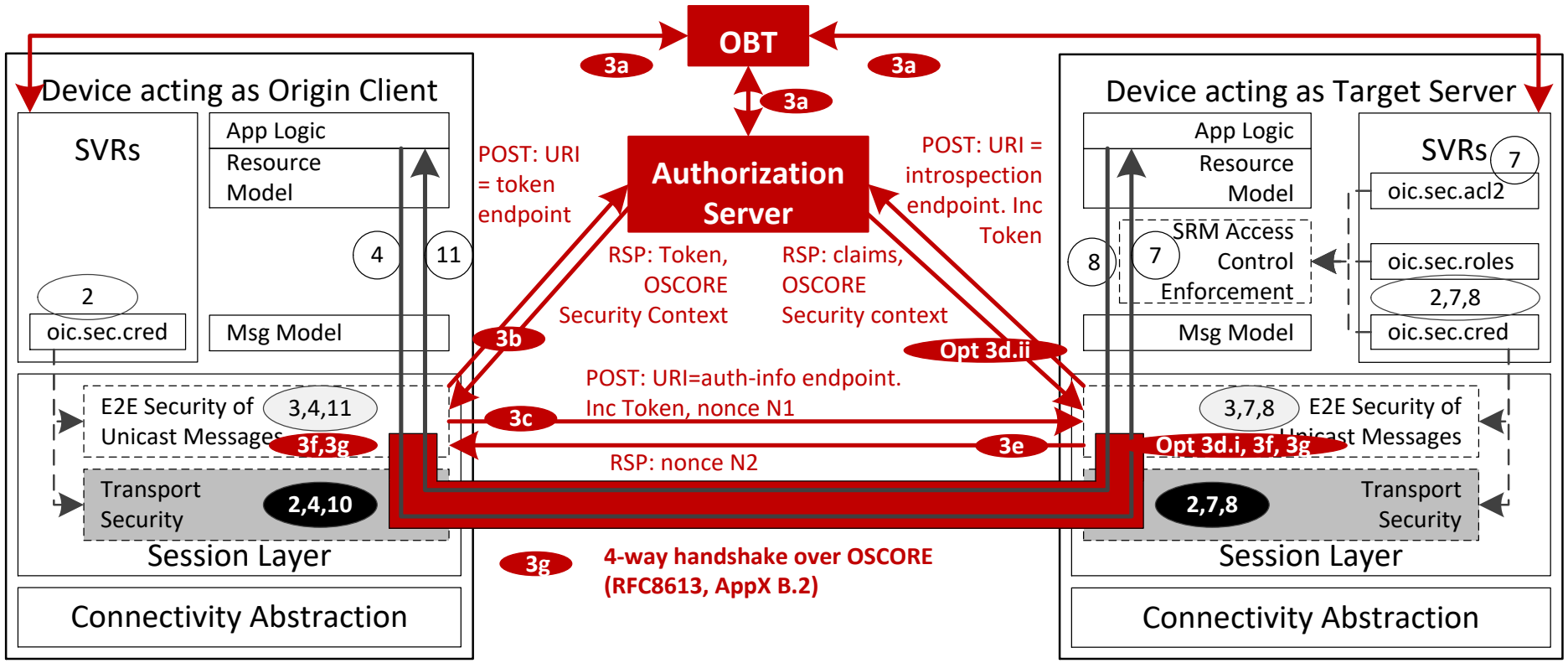
- **OCF Proxy:** functionality which can interpret the OCF compliant URIs of request messages intended for resources on another OCF Server, and can route those request messages accordingly
- **End-to-End Secure [Verb]** securely encapsulate information so that OCF Proxies on the end-to-end delivery path do not need to be trusted with the confidentiality, integrity and freshness of that information
- **End-to-End Security of Unicast Messages**: interoperable mechanism which End-to-End Secures the exchange of unicast OCF CRUDN messages
- **OSCORE** (not preceded by "Group") refers to protection of unicast CoAP messages using RFC 8613. OCF uses this for End-to-End Security of Unicast Messages.
  - **Group OSCORE**: refers to protection of messages using "Group Communication for CoAP" (IETF draft). The spec work for Group OSCORE is work in progress in the IETF core WG. Group OSCORE is an extension of OSCORE. This ppt does not address Group OSCORE.
  - **OSCORE Client/Server**: OCF Client/Server supporting OSCORE
- OSCORE protocol endpoint maps to OCF Endpoint. Purpose: protect msgs from OCF Proxies

E2E Security of Unicast Messages using OSCORE



1. Establish pairwise network connections
2. Pairwise connections may be secured by TLS/DTLS
3. Orig Client and Tgt Svr establish "initial OSCORE" security context (various options) and "derived" OSCORE security context (RFC8613, App'x B.2)
4. Orig Client generates CRUDN req. Protects in OSCORE req. Tgt Svr ID un-encrypted for routing. Sends to Pxy A
5. Pxy A gets Tgt Svr ID in OSCORE req. Subject to policy, fwd to Pxy B
6. Pxy B gets Tgt Svr ID in OSCORE req. Subject to policy, fwd to Tgt Svr
7. Tgt Svr verifies and decrypts OSCORE req to obtain CRUDN req, treated as "auth-crypt" rqe received from "deviceUUID" in "/oic/sec/cred". Tg svr applies access control as normal.
8. Target Svr generates CRUDN rsp Protect in OSCORE rsp. Send to Pxy B
9. Pxy B fwds OSCORE rsp to Pxy A
10. Pxy A fwds OSCORE rsp to Orig Client
11. Orig Client verifies and decrypts OSCORE rsp to obtain CRUDN rsp.

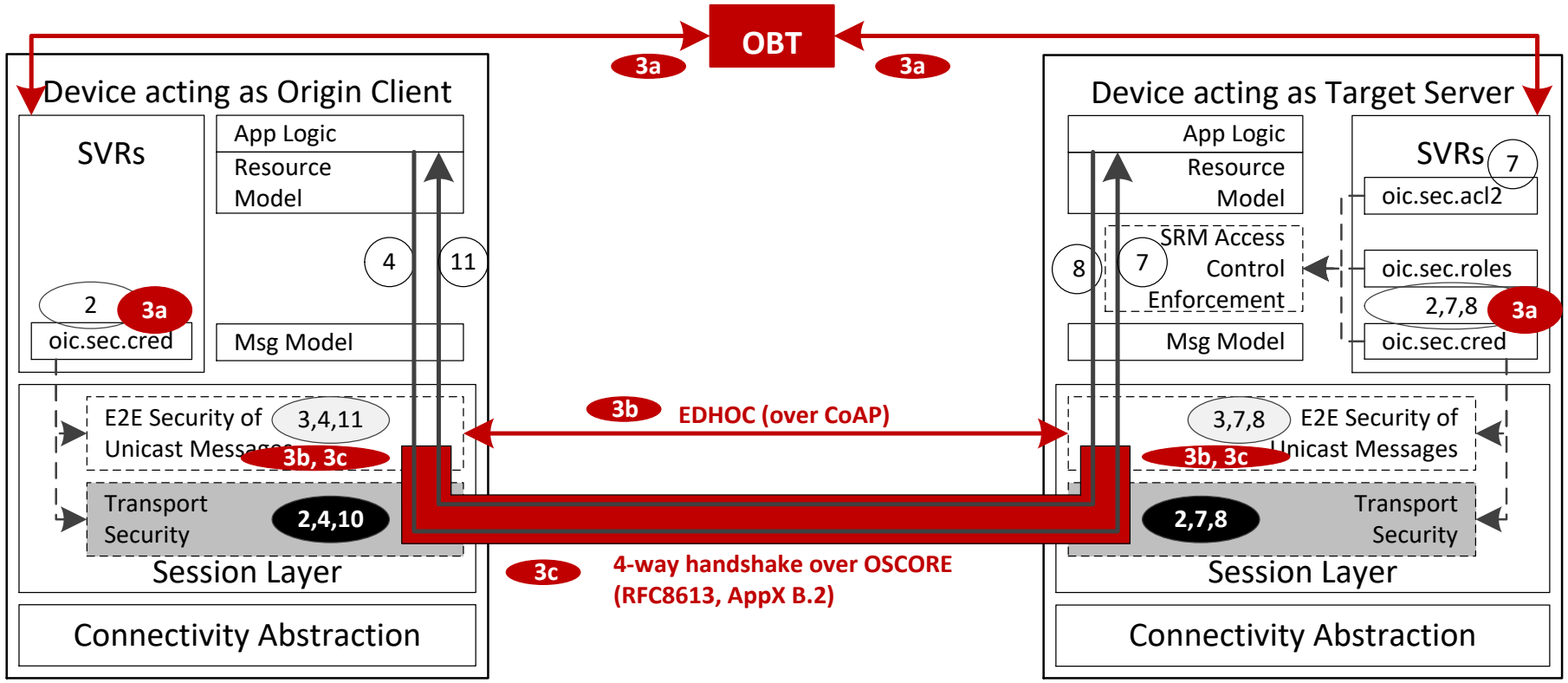
OSCORE using ACE in OCF



1. See slide 4
2. See slide 4
3. Orig Client and Tgt Svr establish E2E OSCORE session:
   a) OBT (Mediator) configures Orig Client and Tgt Svr with credentials for mutual authentication with Authorization Server (AS)
   b) Orig Client obtains token & Security Context info from AS
   c) Orig Client provides token & nonce N1 to Tgt Svr
   d) Tgt Svr gets authorization claims & OSCORE Security Context info & (deviceUUID, roles, etc.) Options:
      i. Self contained in token
      ii. Token introspection via AS
   e) Tgt Svr returns nonce N2 to Orig client
   f) Orig Svr & Tgt Svr compute session keys using N1 & N2. This is "initial" OSCORE security context
   g) Orig Client & Tgt Server establish "derived" OSCORE security context per RFC 8613 App'x B.2. Can be repeated for fresh security context.

4-11. See slide 4.

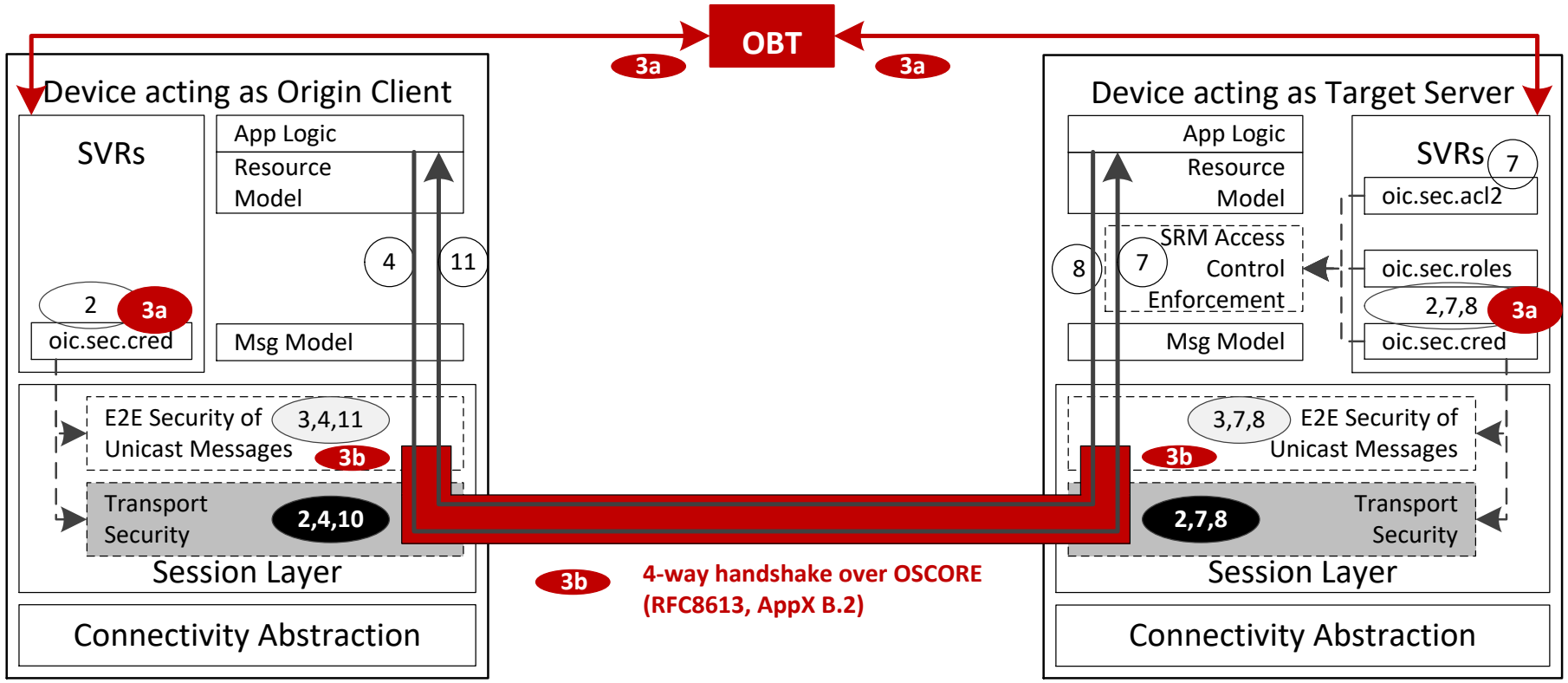
OSCORE using EDHOC



1. See slide 4
2. See slide 4
3. Orig Client and Tgt Svr establish OSCORE security context,
   a) OBT provisions OSCORE credentials to "/oic/sec/cred" of Orig Client & Tgt Svr
   b) Orig Client and Tgt Svr perform handshake using these cred's to establish "initial" OSCORE security context
   c) Orig Client & Tgt Server establish "derived" OSCORE security context per RFC 8613 App'x B.2. Can be repeated for fresh security context.

4-11. See slide 4.

OSCORE using RFC8613 Appendix B.2



1. See slide 4
2. See slide 4
3. Orig Client and Tgt Svr establish E2E OSCORE session,
   a) OBT provisions "initial" OSCORE Security Context to "/oic/sec/cred" of Orig Client & Tgt Svr
   b) Orig Client & Tgt Server establish "derived" OSCORE security context per RFC 8613 App'x B.2. Can be repeated for fresh security context.

4-11. See slide 4.

Comments on RFC8613 [1/3]

A. Is Appendix B.2 supported by any implementations?

B. In my opinion, Appendix B.2 is mostly not normative (except for some MUST NOT statements)
   - *Concern: Implementers need normative text. Testing needs normative text*
   - *Interim Solution: Say "shall perform Appendix B.2" . Probably won't do detailed testing of Appendix B.2.*
   - *Long Term Solution: OCF shares issue w/ IRTF T2TRG.*

C. Lengths of random values are not specified in Appendix B.2
   - *Concern: Interoperability. Providing minimum security levels. Difficult for the Server to determine whether receiving request #1 or request #2.*
   - *Interim Solution: Specify length in Core Security spec.*
   - *Long term solution: OCF shares issue w/ IRTF T2TRG.*

Comments on RFC8613 [2/3]

D. RFC 8613 Appendix B.2 suggests that Server can initiate the process at step 2. However, "ID Context" at step 2 includes ID1 from step 1, so how can step 1 be skipped?
   - *Concern: Inconsistent?*
   - *OCF's Interim Solution: Mandate starting at step 1.*
   - *Long term solution: OCF shares w/ IRTF T2TRG.*

E. Difficult to avoid collisions between identifiers
   - *Concern: Recipient ID Collisions produce inefficiencies. Sender ID collisions and can lead to security problems by deriving identical Sender Keys. ID Collisions are best avoided*
     - *Each protocol has mechanism to avoid colliding IDs*
       - ***In EDHOC**: endpoints assign IDs to each other .*
       - ***In ACE**: Authorization Server (AS) assigns ID of OSCORE Server; AS or OSCORE Server assigns ID of OSCORE Client*
       - ***In our "Directly Configured" initial Security Context solution**: OBT assigns IDs*
     - *How do we avoid collisions identifiers assigned by different protocols?*
   - *Interim Solution: Using random values for our "Directly Configured" solution*
   - *Long term solution: OCF shares w/ IRTF T2TRG. Will likely encounter this problem re EDHOC vs ACE*

Comments on RFC8613 [3/3]

F. Clause 3.3 says

```
If an endpoint has the same Recipient ID with different Recipient Contexts, i.e.,
the Recipient Contexts are derived from different Common Contexts, then the
endpoint may need to try multiple times before verifying the right security
context associated to the Recipient ID.
```

Clause 8.2 makes no mention of this. This is important because at step 6, the server should not send an error message if there are other Recipient Contexts that server should try. Details should be added at step 2 and 6.

- *Concern: Inconsistent?*
- *OCF's Interim Solution: Add details to OCF Spec.*
- *Long term solution: OCF shares w/ IRTF T2TRG.*

G. When using OSCORE with ACE, can exchange of nonces be used to derive multiple contexts, or should RFC 8613 Appendix B.2 be used?

- *Concern: We currently assume only RFC 8613 Appendix B.2 is mechanism is used for deriving multiple contexts and are writing spec according.*
- *OCF's Interim Solution: Not applicable at this stage. Might impact spec in future.*
- *Long term solution: OCF shares w/ IRTF T2TRG.*

BACKUP SLIDES

Why is OSCORE important for OCF?

1. OSCORE protects confidentiality, integrity and ordering of messages going through OCF Proxies.
   - Consequence: Security Domains remain secure even if OCF Proxies snoop or get compromised or misbehave.
2. Target Servers which receive request via OCF Proxies can perform access control based on the identity and roles of the Origin Client which sent the OSCORE message
   - Whereas, currently the Target Server performs access control based on the identity and roles associated with the DTLS/TLS connection (to the OCF Proxy) over which the request was received

Notes

- With OSCORE
  - OCF Proxies can perform CoAP-to HTTP translation without affecting E2E Security
  - Connectivity and transport security sessions can be established and torn down without impacting OSCORE security contexts

Observation

- The main changes are
  - Might need to add a new credential type(s) "credtype"
  - New session protection (establishing E2E secured sessions, protecting CRUDN messages). Can be used in addition to DTLS/TLS.
  - Routing at Proxies.
- Otherwise everything stays "as is". E.g. no changes to
  - CRUDN messaging, & Resource model
  - Credential provisioning (use "/oic/sec/cred") & access control (use "/oic/sec/acl2")
- What we could add in future
  - Onboarding using OSCORE in the place of DTLS … but don't worry about that now ☺

OSCORE (RFC 8613)

- Format and processing of OSCORE request and response msgs
  - RFC describes how to transport & proxy using CoAP(S) and/or HTTP(S)
  - Payload is a CBOR Object Signature and Encryption (COSE) object [RFC8152]
  - All elements of the CoAP message are encrypted, integrity protected & replay protected
    - Except for parts needed for delivering the message to the receiving OSCORE endpoint,
- Assumes OSCORE Endpoints have already established initial Security Context,
  - Including algorithms, identifiers for identifying Security Context, & Master Secret
  - Analogous to cryptographic parameters established by DTLS/TLS handshake protocol
- But…

OSCORE (RFC 8613) continued

- !! RFC8163 Does not specify how initial Security Context is established !!
  - That is, currently no equivalent to DTLS/TLS handshake protocol
- IETF ACE WG (Authentication and Authorization for Constrained Environments) also working on Authorization Server/Token based approach
  - OSCORE profile of the Authentication and Authorization for Constrained Environments Framework https://datatracker.ietf.org/doc/draft-ietf-ace-oscore-profile/
    - This is submitted for publication. Updated in last day or so.
    - OCF doesn't support an Authorization Server/Token based framework yet.
- IETF LAKE WG (Lightweight Authenticated Key Exchange) working on a solution similar to TLS 1.3 handshake!
  - Requirements for a Lightweight AKE for OSCORE
    - https://datatracker.ietf.org/doc/draft-ietf-lake-reqs/
  - Ephemeral Diffie-Hellman Over COSE (EDHOC)
    - https://datatracker.ietf.org/doc/draft-selander-ace-cose-ecdhe/
    - A draft handshake protocol for OSCORE …but this isn't likely to be ready in near future.

OSCORE and other IoT standards

- OMA Lightweight M2M v1.1 supports OSCORE
  - See clause 5.5 of http://www.openmobilealliance.org/release/LightweightM2M/V1_1_1-20190617-A/OMA-TS-LightweightM2M_Transport-V1_1_1-20190617-A.pdf
  - Regarding establishing a session, clause 5.5.3 says only

    An implementation SHALL follow the procedure described in Appendix B2 of [OSCORE] when the OSCORE security context is used for the first time.
  - Note: OMA LWM2Mv1.1 does not address the concerns on the previous slide.

OSCORE open source implementations

- https://gitlab.informatik.uni-bremen.de/obergman/libcoap/-/tree/oscore
- https://github.com/chrysn/liboscore

Status

	Location	Group	Status
Overview: Security Model for E2E Security	Core Sec, 5.1	Core Sec	Draft
Intro to using OSCORE for E2E Security of Unicast Messages	Core Sec, 5.x	Core Sec	Draft
OSCORE Credential Type(s)	Core Sec, 9.3.x	Core Sec	Draft
OSCORE: Introduction	Core Sec, New	Core Sec	Draft
OSCORE: General requirements	Core Sec, New	Core Sec	Draft
Establishing OSCORE Security Context	Core Sec, New	Core Sec	Draft
OBT Spec changes	OBT, New	Core Sec	**Phil to do**
Proxy URI and proxying	Core framework	CTWG	CTWG to do
Group Comm	Core Sec, 5.1	Core Sec. CTWG	**Phil to show overlap**

OPEN CONNECTIVITY FOUNDATION™