A YANG data model for SDN-based key management with EDHOC and OSCORE (draft-marin-yang-edhoc-oscore-00)

Rafa Marín-López (Presenter)
Gabriel López-Millán
(University of Murcia)
Laurent Toutain
Alex Fernández
(IMT Atlantique)

Introduction

- Software-Defined Networking (SDN) is an architecture that enables users to directly program, orchestrate, control and manage network resources through software.
- This model is being used in IoT networks.
- We have previous work to manage IKE and IPsec with SDN (RFC 9061)
- **Idea: SDN-based management of EDHOC and OSCORE.**
- Motivation: providing a centralized system for security associations using YANG and CORECONF

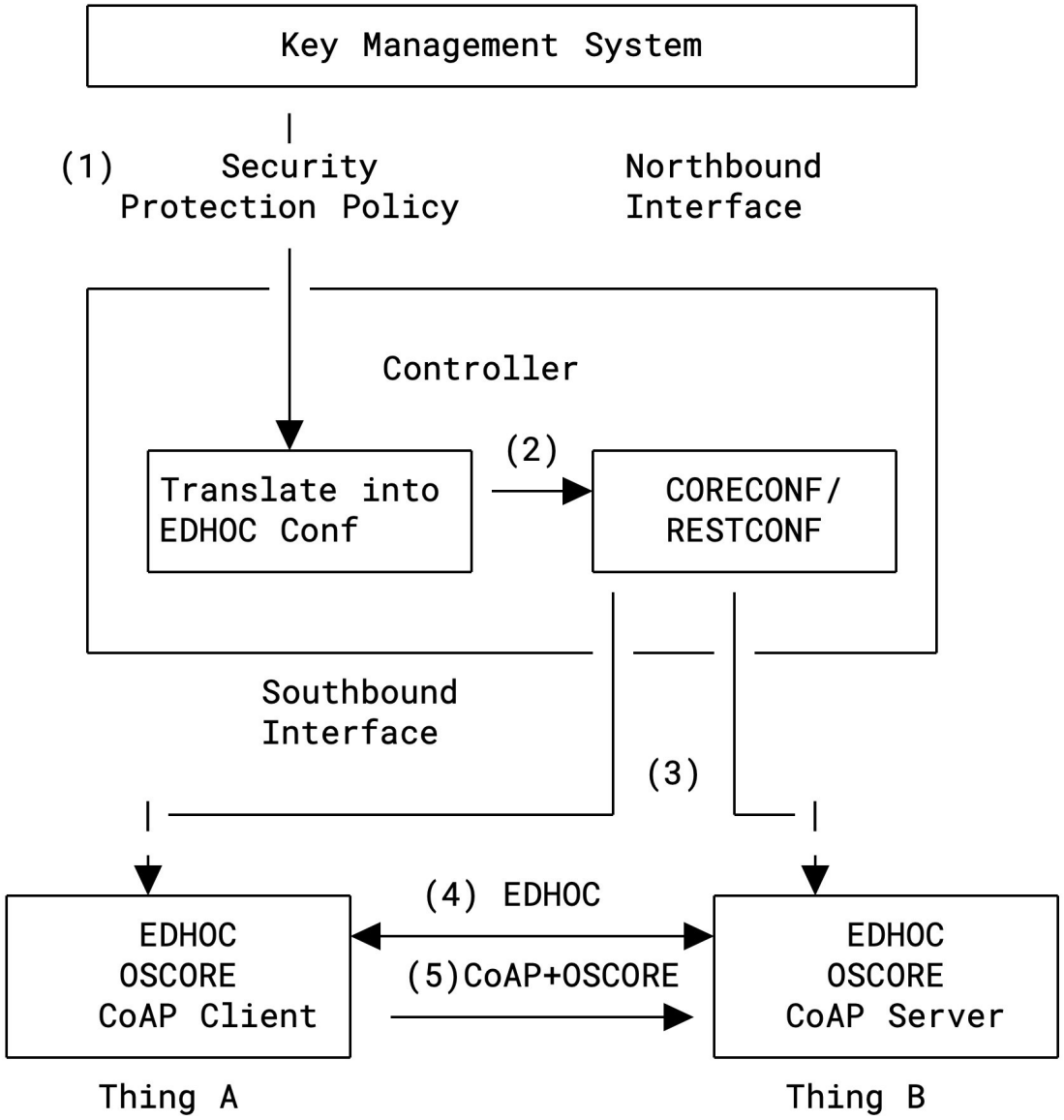
Steps

1. Thing registration/onboarding in the SDN controller
2. The SDN controller can send configuration information about EDHOC and OSCORE to the Things based on the YANG data model

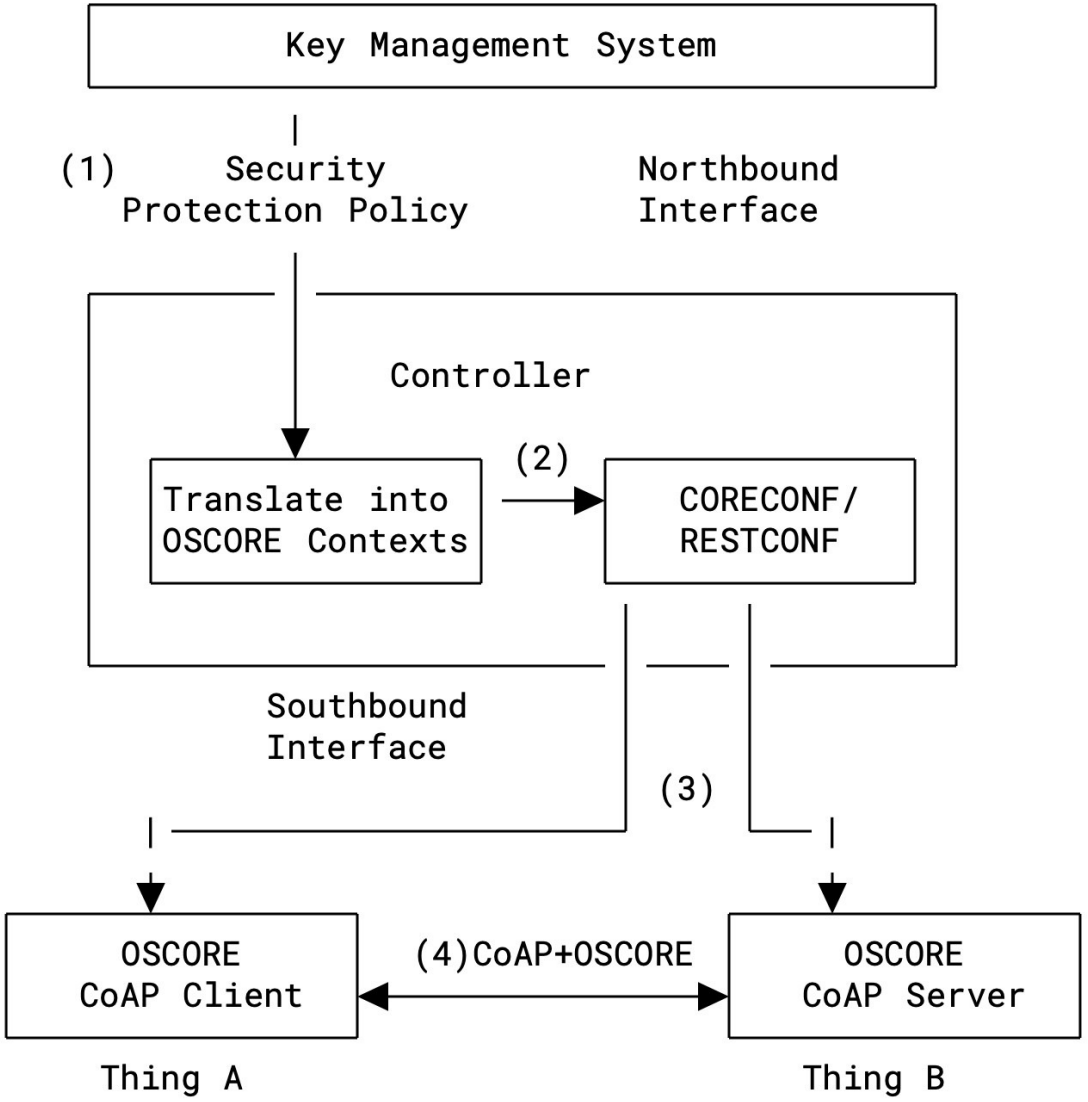
Thing registration/onboarding

- Before starting everything, the Thing needs to be authenticated under the Controller and to establish a security association between the Controller and the Thing to protect the exchanges.
- This is a preliminary step and it is assumed in the operation of SDN-based management for EDHOC and OSCORE.

Case 1: EDHOC+OSCORE in the Thing



Case 2: OSCORE in the Thing



YANG Data Model - EDHOC

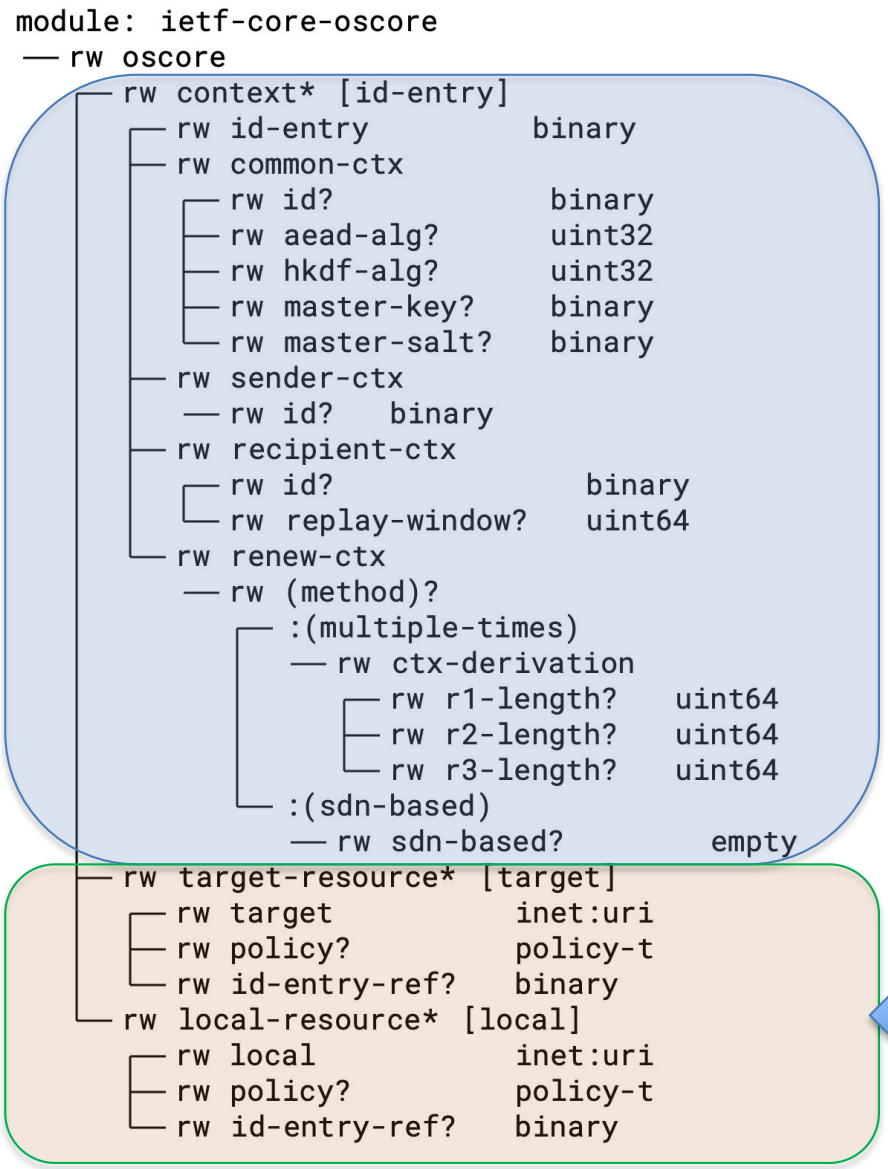
```
module: ietf-core-edhoc
 ── rw edhoc
    ├─ rw auth-entry* [name]
    │  ├─ rw name            string
    │  ├─ rw id-cred-x       binary
    │  ├─ rw auth-method?    auth-method-t
    │  ├─ rw cred-x          binary
    │  └─ rw private-key     binary
    ├─ rw connection* [name]
    │  ├─ rw name                    string
    │  ├─ rw local
    │  │  ├─ rw autostartup?      boolean
    │  │  ├─ rw auth-cred-ref     string
    │  │  ├─ rw c-x?              binary
    │  │  ├─ rw suites-x?         binary
    │  │  └─ rw ead-x
    │  │     ├─ rw ead-a?    binary
    │  │     └─ rw ead-b?    binary
    │  ├─ rw remote
    │  │  ├─ rw id-cred-x       binary
    │  │  ├─ rw auth-method?    auth-method-t
    │  │  └─ rw cred-x          binary
    │  ├─ rw key-confirmation?      boolean
    │  ├─ rw set-oscore?            boolean
    │  ├─ rw key-update-context?    binary
    │  └─ rw reauth-time
    │     ├─ rw soft?    uint32
    │     └─ rw hard?    uint32
    ├─ rw target-resource* [target]
    │  ├─ rw target       inet:uri
    │  ├─ rw policy?      policy-t
    │  └─ rw conn-ref?    string
    └─ rw local-resource* [local]
       ├─ rw local        inet:uri
       ├─ rw policy?      policy-t
       └─ rw conn-ref?    string
```

Credentials

Connection information between two Things (local and remote)

Policies (BYPASS, PROTECT, DISCARD)

YANG Data Model - OSCORE



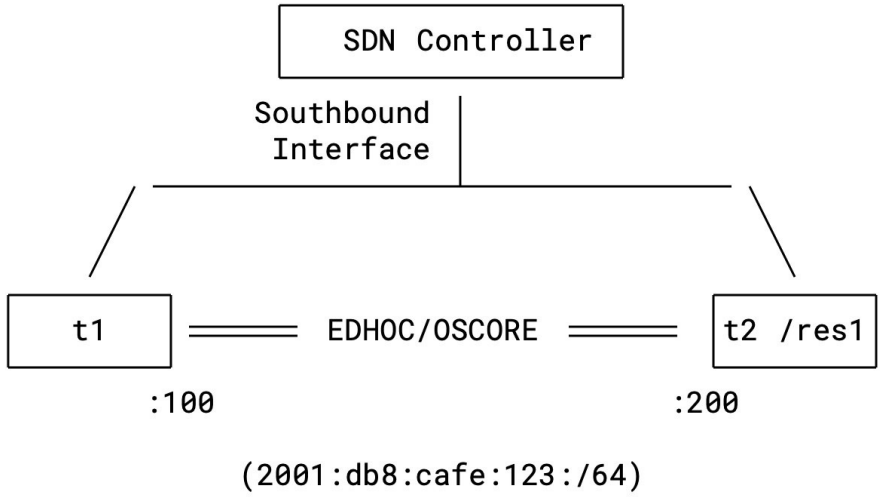
Credentials

OSCORE contexts (common, sender, recipient)

Policies (BYPASS, PROTECT, DISCARD)

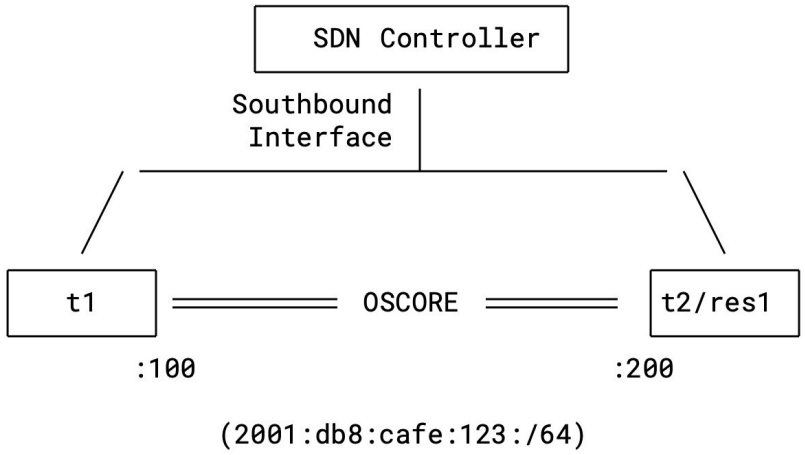
Example : EDHOC

```
<edhoc xmlns="urn:ietf:params:xml:ns:yang:ietf-core-edhoc"
    xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
    <auth-entry>
        <name>auth_entry_t1</name>
      <id-cred-x>base64encodedvalue==</id-cred-x>
      <private-key>base64encodedvalue==</private-key>
      <auth-method>signature-key</auth-method>
      <cred-x>base64encodedvalue==</cred-x>
    </auth-entry>
    <connection>
        <name>edhoc_conn_t1_t2</name>
        <local>
            <autostartup>true</autostartup>
          <auth-cred-ref>auth_entry_t1</auth-cred-ref>
            <c-x>Mzc=</c-x><!--37-->
            <suites-x>MDI=</suites-x><!--02-->
            <ead-x>\
                 <ead-a>MDE=</ead-a><!--01-->
                 <ead-b>MDI=</ead-b><!--02-->
            </ead-x>
        </local>
        <remote>
            <id-cred-x>base64encodedvalue==</id-cred-x>
            <cred-x>base64encodedvalue==</cred-x>
        </remote>
        <key-confirmation>true</key-confirmation>
        <set-oscore>true</set-oscore>
        <key-update-context/>
        <reauth-time/>
    </connection>
    <target-resource>
        <target>coap://2001:db8:cafe:123::200/res1</target>
        <policy>protect</policy>
        <conn-ref>edhoc_conn_t1_t2</conn-ref>
    </target-resource>
</edhoc>
```



Example: OSCORE

```
<oscore
    xmlns="urn:ietf:params:xml:ns:yang:ietf-core-oscore"
    xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
    <context>
        <name>ctx-t1_t2</name>
        <common-ctx>
            <id>Mzc6Y2I6ZjM6MjE6MDA6MTc6YTI6ZDM=</id>
            <aead-alg>10</aead-alg>
              <hkdf-alg>1</hkdf-alg>
              <master-key>base64encodedvalue==</master-key>
              <master-salt>base64encodedvalue==</master-salt>
        </common-ctx>
        <sender-ctx>
            <id>MEY=</id><!-- 0F -->
          </sender-ctx>
        <recipient-ctx>
            <id>MDE=</id>
        </recipient-ctx>
    </context>
    <target-resource>
        <target>coap://2001:db8:cafe:123::200/res1</target>
        <policy>protect</policy>
        <name-ref>ctx-t1_t2</name-ref>
    </target-resource>
</oscore>
```



Proof-of-concept

- Case 2 : OSCORE
  - YANG to CBOR library (pycoreconf)
  - AIOCOAP implementation for the Controller (CoAP client) and Things (CoAP server)
  - uedhoc-uoscore (modified to accept a config file with the oscore context generated from CBOR outcome)

Next steps

- Extending YANG data models
  - To include different extensions to OSCORE (e.g. KUDOS)
- Improving implementation
- Should we standardize these YANG data models?

Backup

Procedure (Python implementation)

1. XML or JSON to Python Dictionary
   - `yanglint` converts XML to JSON
   - Parse with `xmltodict` or `json` Python modules.

```
{ 'ietf-core-oscore:oscore': { 'context': [ {
'common-ctx': { 'aead-alg': 10,
                'hkdf-alg': 1,
                'id': 'Mzc6Y2I6ZjM6M...',
                'master-key': 'MDE6MDI...',
                'master-salt': 'OWU6N...'},
            'name': 'ctx-v3',
            'recipient-ctx': {'id': 'MDE='},
            'sender-ctx': {'id': 'MEY='}}],
'target-resource': [ { 'name-ref': 'ctx-v3',
                       'policy': 'protect',
                     'target': 'coap://192.168.123.200/tv1'}]}}
```

2. Get identifier : SID pairs (and data types) from model's SID file.
   - pyang --sid-generate-file $START:$NUM --sid-list --sid-extention $YANG -p $MODULES

Procedure (Python implementation)

3. Match identifiers & SIDs.

```
{ 60001: { 1: [ { 1: { 1: 10,
                       2: 1,
                       3: b'Mzc6Y2I6ZjM6MjE6MDA6MTc6YTI6ZDM=',
                       4: b'MDE6MDI6MDM6MDQ6MDU6MDY6MDc6MDg6MDk6MGE6MGI6MGM6MGQ6MGU6MGY6MTA=',
                       5: b'OWU6N2M6YTk6MjI6MjM6Nzg6NjM6NDA='},
                  7: 'ctx-v3',
                  8: {1: b'MDE='},
                  11: {1: b'MEY='}}],
            18: [{1: 'ctx-v3', 2: 0, 3: 'coap://192.168.123.200/tv1'}]}}
```

4. Encode in CBOR.

```
A1                                  # map(1)
   19 EA61                          # unsigned(60001)
   A2                               # map(2)
      01                            # unsigned(1)
      81                            # array(1)
         A4                         # map(4)
            01                      # unsigned(1)
...
```