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

Rafa Marín-López (Presenter)
Gabriel López-Millán
Esther Salazar Saura
(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 the management of M2M security associations using YANG and CORECONF
Steps

1. Thing registration/onboarding in the SDN controller
2. The SDN controller sends configuration information about EDHOC and OSCORE to the Things through CORECONF/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

Key Management System

(1) Security Protection Policy
    Northbound Interface

Controller

(2) Translate into EDHOC Conf
    CORECONF/RESTCONF

Southbound Interface

(3) 

EDHOC OSCORE CoAP Client

(4) EDHOC
    CoAP+OSCORE

EDHOC OSCORE CoAP Server

Thing A

Thing B
Case 2: OSCORE in the Thing

Key Management System

(1) Security Protection Policy Northbound Interface

Controller

(2) Translate into OSCORE Contexts CORECONF/RESTCONF

Southbound Interface

(3)

OSCORE CoAP Client (4)CoAP+OSCORE OSCORE CoAP Server

Thing A Thing B
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-x>!
      <ead-a>MDE=</ead-a>!--01-->
      <ead-b>MDI=</ead-b>!--02-->
    </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

```xml
<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

- $N$ nodes receiving the EDHOC/OSCORE configurations from the SDN controller, either on demand or once they are all registered
  - NOTE: Registration implemented with a CoAP request from Thing to the controller
- Possibility of two topologies: *star* or *mesh*
- Southbound implementation (CORECONF) with AIOCOAP and YANG to CBOR library (pycoreconf) between the Controller (CoAP client) and the Things (CoAP server)
  - The southbound can be protected with OSCORE as well
- uedhoc-uoscore for EDHOC/OSCORE communication between nodes
Example Scenario (I)

• Case 1 : EDHOC (Mesh)
Example Scenario (II)

• Case 2: OSCORE (Mesh)
Next steps

• Extending YANG data models
  – To include different extensions to OSCORE (e.g. KUDOS, Group communications extensions, etc...)
  – Distributing a PSK from the controller if we have EDHOC PSK.
  – Add new protocols for securing communication between nodes, such as DTLS.

• Conduct proof-of-concept performance

• Transporting the virtualized scenario to a real scenario with distributed IoT devices and testing its performance

• YANG data models should be standarized