ACE Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 16, 2021

R. Marin-Lopez
University of Murcia
D. Garcia-Carrillo
University of Oviedo

June 14, 2021

# EAP-based Authentication Service for CoAP
# draft-ietf-ace-wg-coap-eap-02

Abstract

This document describes an authentication service that uses EAP transported employing CoAP messages with following purposes: 1) Authenticate a CoAP-enabled device that enters a new security domain managed by a domain Controller, 2) Derive key material to protect CoAP messages exchanged between them, enabling the establishment of a security association between them, and 3) Optionally, to generate key material for other types of Security Associations.

Generally speaking, this document is specifying an EAP lower layer based on CoAP, to bring the benefits of EAP to IoT.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 16, 2021.

Table of Contents

1. Introduction . . . . . 3
1.1. Requirements Language . . . . . 3
2. General Architecture . . . . . 3
3. General Flow Operation . . . . . 4
3.1. EAP over CoAP flow of operation . . . . . 5
3.2. Message processing of EAP over CoAP . . . . . 8
3.3. EAP over CoAP operation casuistics . . . . . 8
4. Managing the State of the Service . . . . . 11
4.1. Deleting the state . . . . . 12
4.2. Renewing the state . . . . . 12
5. Key Derivation for protecting CoAP messages . . . . . 13
5.1. Deriving the OSCORE Security Context . . . . . 13
5.2. Deriving DTLS_PSK . . . . . 14
6. Examples of Use Case Scenario . . . . . 14
6.1. Example 1: CoAP-EAP in ACE . . . . . 15
6.2. Example 2: Multi-domain with AAA infrastructures . . . . 16
6.3. Example 3: Single domain with AAA infrastructure . . . . 17
6.4. Example 4: Single domain without AAA infrastructure . . . 17
6.5. Other use cases . . . . . 17
6.5.1. CoAP-EAP for network access control . . . . . 17
6.5.2. CoAP-EAP for service authentication . . . . . 17
7. Discussion . . . . . 18
7.1. CoAP as EAP lower layer . . . . . 18
7.2. Size of the EAP lower layer vs EAP method size . . . . . 19
7.3. Controller as the CoAP Client . . . . . 19
7.4. Possible Optimizations . . . . . 20
7.4.1. Empty Token . . . . . 20
7.4.2. Further re-authentication . . . . . 20
8. Security Considerations . . . . . 20
8.1. Authorization . . . . . 20
8.2. Cryptographic suite selection . . . . . 20
8.3. Freshness of the key material . . . . . 21
8.4. Additional Security Consideration . . . . . 21
9. IANA Considerations . . . . . 21
10. Acknowledgments . . . . . 21
11. References . . . . . 21
11.1. Normative References . . . . . 21
11.2. Informative References . . . . . 23
Authors' Addresses . . . . . 24

## 1. Introduction

The goal of this document is to describe an authentication service that uses the Extensible Authentication Protocol (EAP) [RFC3748]. The authentication service is built on top of the Constrained Application Protocol (CoAP) [RFC7252] and allows authenticating two CoAP endpoints by using EAP, to establish a security association between them.

In particular, this document describes how CoAP can be used as a constrained, link-layer independent, EAP lower layer [RFC3748] to transport EAP messages between a CoAP server (EAP peer) and a CoAP client (EAP authenticator) using CoAP messages. The CoAP client MAY contact with a backend AAA infrastructure to complete the EAP negotiation as described in the EAP specification [RFC3748].

The assumption is that the EAP method transported in CoAP MUST generate cryptographic material [RFC5247]. In this way, the CoAP messages can be protected after the authentication. The general flow of operation of CoAP-EAP establishes an OSCORE security association specifically for the service. In addition, using the key material derived from the authentication, we specify the establishment of other security associations depending on the security requirements of the services:

- OSCORE [RFC8613] security association can be established based on the cryptographic material generated from the EAP authentication.

- A DTLS security association can be established using the exported cryptographic material after a successful EAP authentication.

This document also indicates how to establish a security association for other types of technologies that rely on CoAP.

### 1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

## 2. General Architecture

Figure 1 shows the architecture defined in this document. Basically, a node acting as the EAP peer wants to be authenticated by using EAP. At the time of writing this document, we have considered a model where the entity acting as EAP peer will also act as a CoAP server for this service and the entity acting as EAP authenticator will act as a CoAP client and MAY interact with a backend AAA infrastructure,

which will place the EAP server and contain the information required to authenticate the CoAP client. The rationale behind this decision, as we will expand later, is that EAP requests go always from the EAP authenticator to the EAP peer. Accordingly, the EAP responses go from the EAP peer to the EAP authenticator.

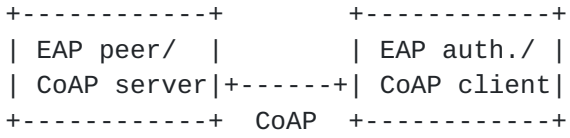



Figure 1: CoAP EAP Architecture

## 3. General Flow Operation

The authentication service uses CoAP as transport for EAP. In other words, CoAP becomes an EAP lower layer in EAP terminology. In general, it is assumed that, since the EAP authenticator MAY implement an AAA client to interact with the AAA infrastructure, this endpoint will have more resources or, at least, will not be so constrained device. We show the sequence flow in Figure 2 where we depict the usage of a generic EAP method that we call EAP-X as an authentication mechanism. (NOTE: any EAP method that can export cryptographic material is valid. For example, EAP-MD5 cannot be used since it does not export key material).

The first step to run CoAP-EAP is for the IoT device to discover the Controller, and that it implements the CoAP-EAP service. To do so, we rely on the discovery mechanism of CoAP. The URI of the CoAP-EAP service CAN be set to "/b" to save bytes over the air. The IoT device will always start the conversation sending the first message, acting as CoAP client only on this instance, to implement a trigger mechanism to let the Controller know it is ready for an authentication process to start.

The first message is used to trigger the authentication process. This message is sent by the IoT device, acting as a CoAP client. This message uses the No-Response Option [RFC7967] to avoid the response from the Controller to this message. After this, the exchange continues with the Controller acting as a CoAP client and the IoT device acting as a CoAP server. This is because the IoT device could be a constrained node, and following the recommendations of [I-D.ietf-lwig-coap] to simplify the implementation of the IoT device, the Controller takes the responsibility of handling the retransmissions. In the next section, we refer to the IoT device as the EAP peer and the Controller as the EAP authenticator to elaborate the specifics of the flow of operation.

### 3.1. EAP over CoAP flow of operation

If the EAP peer discovers the presence of the EAP authenticator and wants to start the authentication, it can send a Non-Confirmable "POST /b" request to the node (Step 0). This message will carry an option developed in the work of [RFC7967] called 'No-Response'. The rationale of this option is to avoid waiting for a response if it is not needed. So the use of this option will allow signalling the intention the EAP peer to start the authentication process. Immediately after that, the EAP authenticator will start authentication service.

In any case, to perform the authentication service, the CoAP client (EAP authenticator) sends a Confirmable "POST /b" request to the CoAP Server (Step 1). After receiving the first POST, the CoAP server assigns a resource and answers with an Acknowledgment with the piggy-backed response containing the resource identifier (Location-Path) (Step 2). The name of the resource MAY be represented following the naming "/b/x". Where "b" is the general name of the bootstrapping service; "x" represents the resource established to process the following message of the authentication. This CoAP server can select this value as pleased, as long as, it serves to process the following message adequately. It is assumed that the CoAP server will only have an ongoing authentication with that particular CoAP client and will not process simultaneous EAP authentications in parallel (with the same EAP authenticator) to save resources.

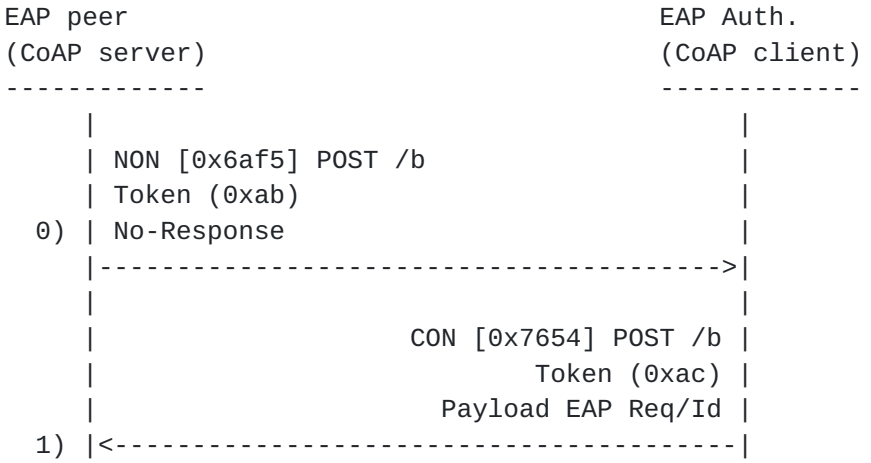
In this exchange (Step 1 and 2), the EAP Req/Id and Rep/Id messages are exchanged between the EAP authenticator and the EAP peer. Upon the reception of the EAP Req/Id message, the EAP authenticator forwards this message, when EAP is in pass-through mode, to the local AAA server. The AAA server is in charge of steering the conversation, choosing the EAP method to be used (e.g. EAP-X) if the user is local, or sending the EAP messages to the home AAA of the EAP peer. At this point, the CoAP server has created a resource for the EAP authentication. The resource identifier value will be used to relate the EAP conversation between both CoAP endpoints.

NOTE: Since only an ongoing EAP authentication is permitted per EAP authenticator/EAP client, and EAP is a lock-step protocol, a Token of a constant value can be used throughout the authentication process. An empty Token could be considered to reduce bytes.

From now on, the EAP authenticator and the EAP peer will exchange EAP packets related to the EAP method, transported in the CoAP message payload (Steps 3,4,5,6). The EAP authenticator will use the POST method to send EAP requests to the EAP peer. The EAP peer will use a Piggy-backed response in the Acknowledgment message to carry the EAP

response. When all the message exchange is completed, if everything has gone well, the EAP authenticator can send an EAP Success message, and both CoAP endpoints will share a Master Session Key (MSK) ([RFC5295])

To establish a security association that will confirm to the EAP peer that EAP authenticator received the MSK from the AAA server, as well as to the EAP authenticator that the EAP peer derived the MSK correctly, both entities engage in the establishment of a security association. In the context of constrained devices [RFC7228] and networks, we consider protocols that are designed for these cases. Concretely, we show here in the diagram the establishment of the OSCORE security association (Steps 7 and 8). From that point, any exchange between both CoAP endpoints is protected with OSCORE. Before sending the EAP success to the EAP peer, the EAP authenticator can derive the OSCORE Security Context, to confirm the establishment of the security association. The details of the establishment of the OSCORE Security Context are discussed in Section Section 5.1. The protection of the EAP Success is not a requirement. Here, we specify this exchange as protected by the lower layer with OSCORE. The purpose is double, we can avoid forgery of this message and we are using the exchange to perform the key confirmation through the establishment of the OSCORE security association. Adding to the previous consideration about the EAP Success, this message does not prevent the operation of the device from continuing as long as there is an alternate success indication that both the EAP peer and authentication can rely on to continue [RFC3748]. This indication can happen in two ways: 1) the reception of the CoAP message without EAP and with an OSCORE option (following the normal operational communication between both entities) is an indication that the controller considers the EAP authentication finished. 2) the IoT device knows that the EAP authentication went well if an MSK is available. Both entities need to prove the possession of the MSK as mentioned in the EAP KMF.



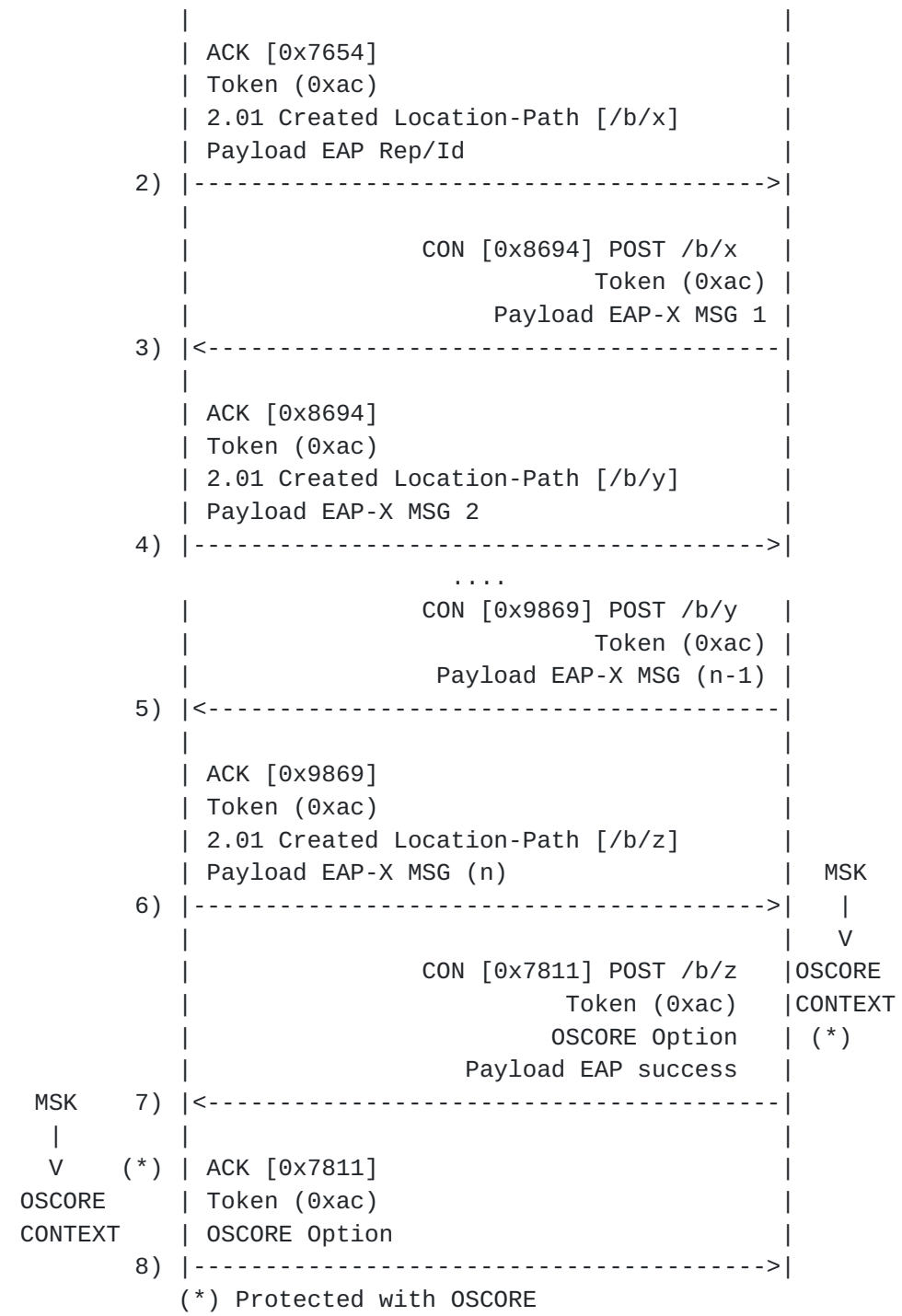



Figure 2: CoAP-EAP flow of operation

### 3.2. Message processing of EAP over CoAP

In this section, we introduce how the service is processed by the two entities involved in the exchange.

For the CoAP server, each time a new CoAP request arrives, containing the EAP message as payload, does the following:

1. Send the EAP message to the EAP state machine

2. If the EAP state machine processes the request correctly:

   A. A new resource is created. (e.g. b/y)

   B. The current resource is deleted. (e.g. b/x)

   C. A response is sent back to the client specifying the new resource

3. If the EAP state machine returns an error:

   A. The CoAP service will send an error message. (e.g: 4.00 Bad Request)

When the EAP authenticator (CoAP client) receives an EAP message from the EAP peer (CoAP server), it will send it to the EAP server, and vice-versa. In any case, the EAP exchange is initiated by the EAP authenticator, sending the EAP Request/Identity message to the EAP peer and the ongoing authentication will be tracked by a bootstrapping state where all the relevant information for the application is stored, such as the current URI for the exchange (or the expected URI for the next exchange). From that point on, the processing of the messages continues as follows:

1. If the ongoing message is an EAP request, is sent to the EAP peer. If it is an EAP response, it is sent to the EAP server.

2. If a CoAP response containing the new resource and the EAP response arrives, the new CoAP resource is updated.

3. If an error arrives, the CoAP client will rely on CoAP retransmission behavior.

### 3.3. EAP over CoAP operation casuistics

In this section, we introduce a couple of cases where a message is lost and retransmitted later, to show how the service will react. The first one shows what would happen if a piggybacked response with

a new resource identifier is lost. This case is illustrated in Figure 3. The second shows what would happen if an old request message arrives even though the process of authentication continued due to normal retransmission behaviour. This is illustrated in Figure 4.

For the first case, when a piggybacked response message containing the Location-Path of the new resource is lost, the CoAP client will retransmit. This will cause the CoAP server to recognize the message as retransmission due to the MSG-ID, and re-send the lost message.

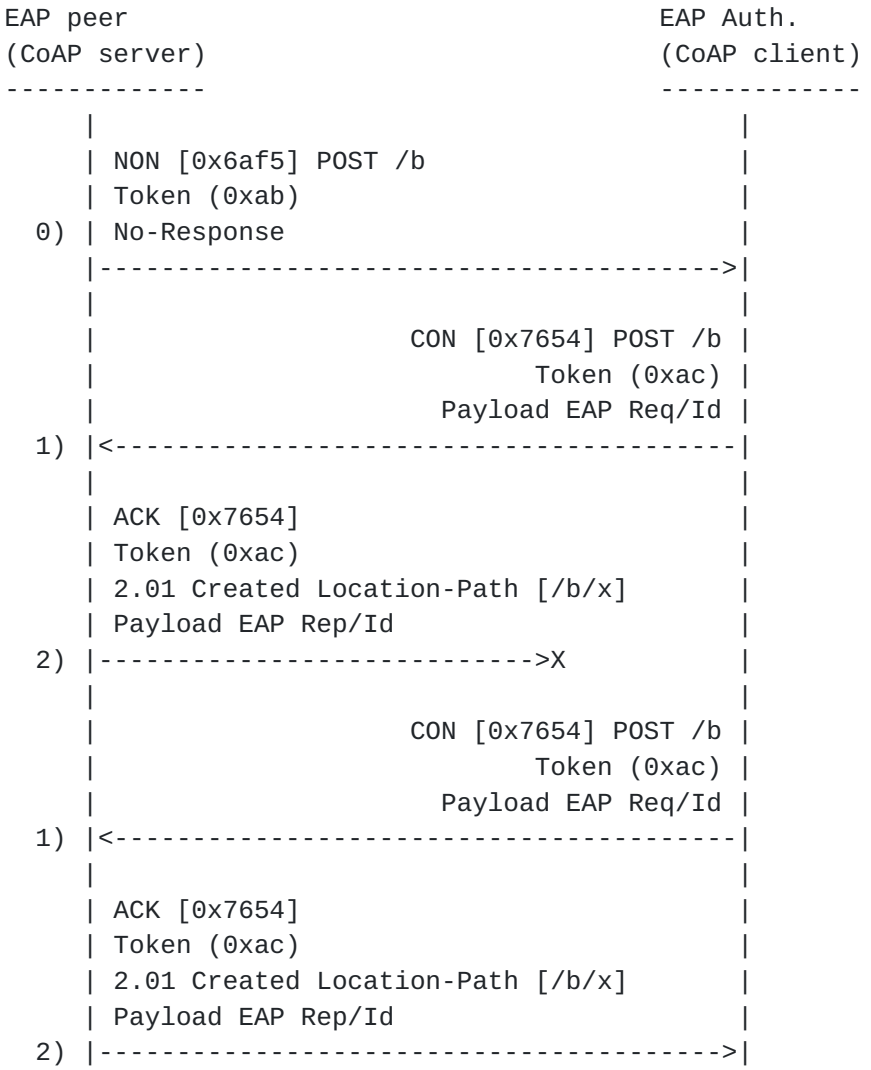



Figure 3: Casuistic - Response Lost

In the second case, when a message is lost, but due to the ongoing workings of CoAP retransmission, the flow of operation continues as

expected. If said lost message arrives later, how this message is handled will depend on which layer deals with it.

1. If the message is handled by the CoAP messaging layer, which means it will not go up to the service application:

   A. As the server recognizes the old message, due to internal tracking, it can send a stored copy of the response.

   B. Then the client would recognize the MSGID as old and that he got the response already, and simply dropping it.

2. If the messaging layer does not recognize the message as old, and takes care of it, it will try to send it to the service application:

   A. This will cause an error, since a resource of a previous step of the authentication is deleted

   B. The error code (e.g., 4.04 Not Found) with the same MSGID is sent back to the CoAP client.

   C. The CoAP client would recognize the MSGID as old and simply drop it.

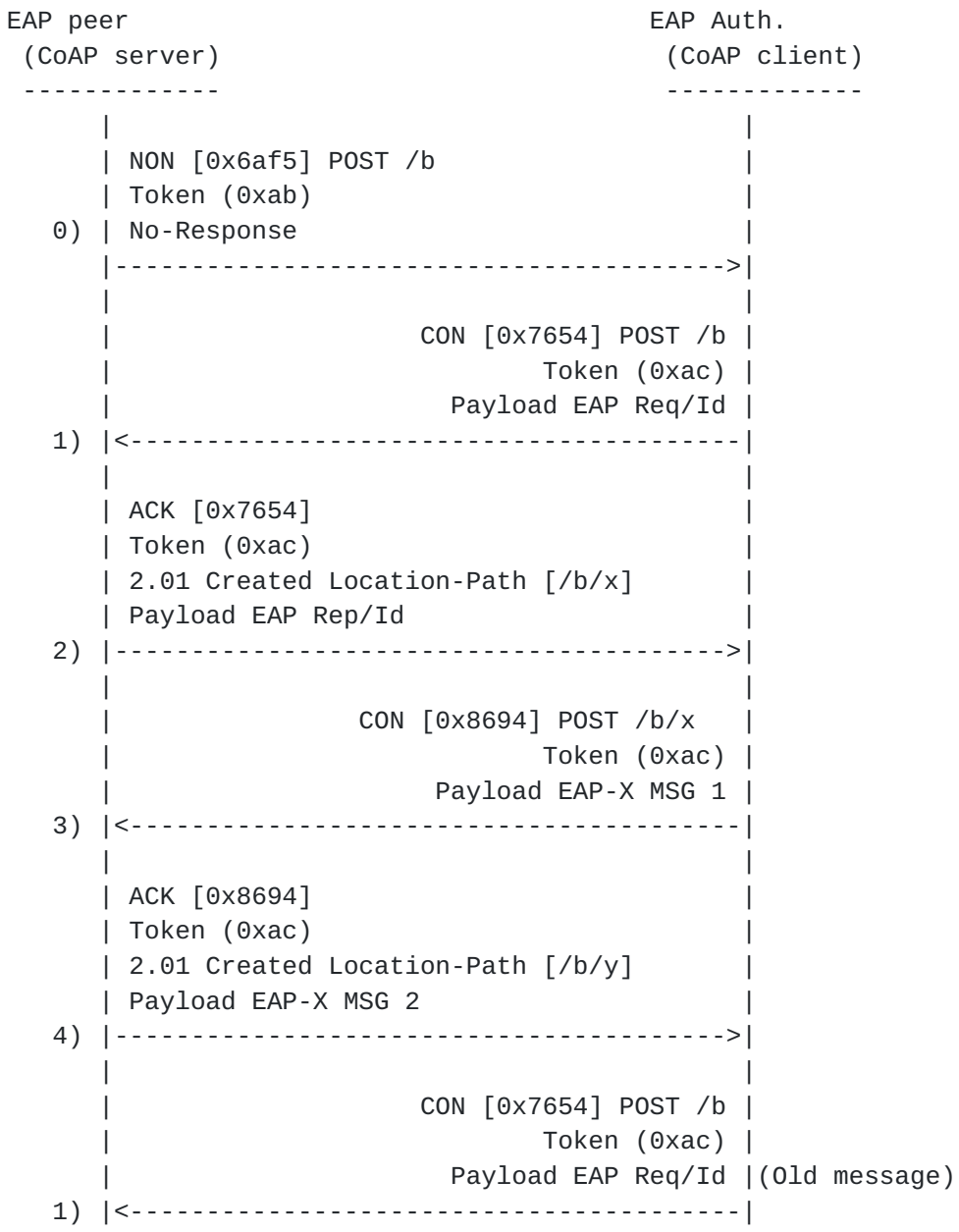



Figure 4: Casuistic - Old message

## 4. Managing the State of the Service

This document establishes the generation of a resource under the service identified by the URI '/b'. Once the authentication process is completed and the final representation of the bootstrapping state is returned to the CoAP client (e.g., /b/z in Figure 2) there are different interactions that may be of interest with regard to the bootstrapping state.

### 4.1. Deleting the state

There are situations where the current bootstrapping state might need to be removed. For instance due to its expiration or a forced removal if the device needs to be expelled from the security domain. If the Controller, which implements the CoAP client in this exchange, deems necessary the removal of the state, it can send a DELETE command to the CoAP server, referencing the boostrapping state resource. The identifier will be the last one received with the ACK of the EAP success message (/b/z in Figure 2) This message will be protected with the OSCORE security association to prevent forgery. Upon reception of this message, the CoAP server sends a piggybacked response to the client with the Code 2.02 Deleted. In the case, there is no ACK and response from the CoAP server, after the maximum retransmission attempts, the CoAP client will remove the state from its side. The repercussion in this case will translate in unauthorized communications from the IoT device towards the Controller within the security domain.

### 4.2. Renewing the state

When the state is close to expire the state might be renewed. This situation causes the possible duplication of states until the new state is generated and the previous one can be deleted. This re-authentication can be done using again the same EAP method, or a more lightweight (e.g., ERP [RFC6696]) could be used, if available. The exchange will be done in this case maintaining ongoing OSCORE security association as long as it is still valid. In other case, the exchange will be done as if it were new (see Figure 2). The re-authentication will be initiated by the IoT device (CoAP server), which is the interested party in maintaining a security association active. This is purposely done this way to avoid unnecessary or unwanted authentication messages, unless the IoT device is ready to start the process. This means, the IoT device will send the trigger message and will only process messages related to a re-authentication after it has sent the trigger message. The exchange will be very similar to the one in Figure 2. In fact, if everything goes well, there will be indistinguishable, up to the final exchange of the EAP success message and its response, which will be protected with the new OSCORE security association, using keys derived from the new EAP exchange. The difference will be in the case an EAP failure is generated. In the initial exchange, since no key material is derived due to a failed authentication, this message is not protected with OSCORE. In the reauthentication scenario, the EAP failure will be protected with the ongoing OSCORE security association. In case of failure by any reason, the old state will be valid until its expiration.

## 5. Key Derivation for protecting CoAP messages

As a result of a successful EAP authentication, both the CoAP server and CoAP client share a Master Key Session (MSK). The assumption is that MSK is a fresh key, so any key derived from the MSK will be also fresh. To complete the CoAP-EAP exchange, as part of the design, it is expected the establishment of an OSCORE security association specifically for the CoAP-EAP service. The security level for the CoAP-EAP exchanges with OSCORE is with integrity. Additionally, we considered the derivation of either the OSCORE Security Context or a pre-shared key that can be used for a DTLS negotiation (DTLS_PSK) for further communications depending on the security requirements of the services provided by the AS. The OSCORE security context generated for CoAP-EAP could be generalized to enable further OSCORE secured communications between the IoT device and the AS services that require the use of OSCORE.

### 5.1. Deriving the OSCORE Security Context

Key material needed to derive the OSCORE Security Context, from the MSK can be done as follows:

The Master Secret can be derived by using AES-CMAC-PRF-128 [RFC4615], which, in turn, uses AES-CMAC-128 [RFC4493]. The Master Secret can be derived as follows:

Master_Secret = KDF(MSK, "IETF_OSCORE_MASTER_SECRET", 64, length)

where:

- The AES-CMAC-PRF-128 is defined in [RFC4615]. This function uses AES-CMAC-128 as a building block.
- The MSK exported by the EAP method, which by design is a fresh key material. Discussions about the use of the MSK for the key derivation are done in Section Section 8.
- "IETF_OSCORE_MASTER_SECRET" is the ASCII code representation of the non-NULL terminated string (excluding the double quotes around it).
- 64 is the length of the MSK.
- length is the length of the label "IETF_OSCORE_MASTER_SECRET" (25 bytes).

The Master Salt, similarly to the Master Secret, can be derived as follows:

Master_Salt = KDF(MSK, "IETF_OSCORE_MASTER_SALT", 64, length)

where:

- The AES-CMAC-PRF-128 is defined in [RFC4615]. This function uses AES-CMAC-128 as a building block.
- The MSK exported by the EAP method, which by design is a fresh key material. Discussions about the use of the MSK for the key derivation are done in Section Section 8.
- "IETF_OSCORE_MASTER_SALT" is the ASCII code representation of the non-NULL terminated string (excluding the double quotes around it).
- 64 is the length of the MSK.
- length is the length of the label "IETF_OSCORE_MASTER_SALT" (23 bytes).

The ID Context can be set to the identity of the EAP peer.

### 5.2. Deriving DTLS_PSK

In the second alternative, a DTLS_PSK is derived from the MSK between both CoAP endpoints. The length of the DTLS_PSK will depend on the cipher-suite. For AES-128, the DTLS_PSK will have a 16-byte length and it will be derived as follows:

DTLS_PSK = KDF(MSK, "IETF_DTLS_PSK" , 64, length). This value is concatenated with the value of the Token Option value.

where:

- MSK is exported by the EAP method.
- "IETF_DTLS_PSK" is the ASCII code representation of the non-NULL terminated string (excluding the double quotes around it).
- 64 is the length of the MSK.
- length is the length of the label "IETF_DTLS_PSK" (13 bytes).

## 6. Examples of Use Case Scenario

For a device to act as a trustworthy entity within a security domain, certain key material is needed to be shared between the IoT device and AS. In ACE, the process of Client registration and provisioning

of credentials to the client is not specified. The process of Client registration and provisioning can be achieved by using CoAP-EAP. Once the process of authentication with EAP is completed, a fresh key material is shared between the IoT device and the AS.

Next, we elaborate on examples of different use case scenarios about the usage of CoAP-EAP. Generally, we are dealing with 4 entities:

- 2 nodes (A and B), which are constrained devices. They are the EAP peers.
- 1 controller (C). The controller manages a domain where nodes can be deployed. It can be considered a more powerful machine than the nodes. In this scenario, the Controller (and EAP Authenticator), can be co-located with the AS.
- 1 AAA server (AAA) - Optional. The AAA is an Authentication, Authorization and Accounting Server, which is not constrained.

Generally, any node wanting to join the domain managed by the controller MUST perform a CoAP-EAP authentication with the controller C. This authentication MAY involve an external AAA server. This means that A and B, once deployed, will perform this CoAP-EAP once as a bootstrapping phase to establish a security association with controller C. Moreover, any other entity, which wants to join and establish communications with nodes under controller C's domain must also do the same. By using EAP, we can have the flexibility of having different types of credentials. For instance, if we have a device that is not battery dependent, and not very constrained, we could use a heavier authentication method. With very constrained devices and networks we might need to resort to more lightweight authentication methods (e.g., EAP-PSK, EAP-EDHOC, etc.) being able to adapt to different types of devices according to policies or devices capabilities.

### 6.1. Example 1: CoAP-EAP in ACE

Next, we exemplify how CoAP-EAP can be used to perform the Client registration in a general way, to allow two IoT devices (A and B) to communicate and interact after a successful client registration.

Node A wants to communicate with node B (e.g. to activate a light switch). The overall process is divided into three phases. Let's start with node A. In the first phase, the node A (EAP peer) does not yet belong to controller C's domain. Then, it communicates with controller C (EAP authenticator) and authenticates with CoAP-EAP, which, optionally, communicates with the AAA server to complete the authentication process. If the authentication is successful, key

material is distributed to controller C and derived by node A. This key material allows node A to establish a security association with the controller (C). Some authorization information may be also provided in this step. If authentication and authorization are correct, node A is enrolled in controller C's domain for a period of time. In particular, [RFC5247] recommends 8 hours, though the AAA server can establish this lifetime. In the same manner, B needs to perform the same process with CoAP-EAP to be part of the controller C's domain.

In the second phase, when node A wants to talk with node B, it contacts controller C for authorization to access node B and obtain all the required information to do that securely (e.g. keys, tokens, authorization information, etc.). This phase does NOT require the usage of CoAP-EAP. The details of this phase are out of the scope of this document, and the ACE framework is used for this purpose [I-D.ietf-ace-oauth-authz].

In the third phase, the node A can access node B with the credentials and information obtained from the controller C in the second phase. This access can be repeated without contacting the controller, while the credentials given to A are still valid. The details of this phase are out of scope of this document.

It is worth noting that first phase with CoAP-EAP is ONLY required to join the controller C's domain. Once it is performed with success, the communications are local to the controller C's domain so there is no need to contact the external AAA server nor performing EAP authentication.

### 6.2. Example 2: Multi-domain with AAA infrastructures

We assume we have a device (A) of the domain acme.org, which uses a specific kind of credential (e.g., AKA) and intends to join the um.es domain. This user does not belong to this domain, for which first it performs a client registration using CoAP-EAP. For this, it interacts with the Domain Controller acting as EAP authenticator, which in turn communicates with a AAA infrastructure (acting as AAA client). Through the local AAA server to communicate with the home AAA server to complete the authentication and integrate the device as a trustworthy entity into the domain of controller C. In this scenario, the AS under the role of the Controller receives the key material from the AAA infrastructure

### 6.3. Example 3: Single domain with AAA infrastructure

A University Campus, we have several Faculty buildings and each one has its own criteria or policies in place to manage IoT devices under an AS. All buildings belong to the same domain (e.g., um.es). All these buildings are managed with a AAA infrastructure. A new device (A) with credentials from the domain (e.g., um.es) will be able to perform the device registration with a Controller (C) of any building as long as they are managed by the same general domain.

### 6.4. Example 4: Single domain without AAA infrastructure

In another case, without a AAA infrastructure, we have a Controller that has co-located the EAP server and using EAP standalone mode we can manage all the devices within the same domain locally. Client registration of a node (A) with Controller (C) can also be performed in the same manner, transparent to the IoT device. In this scenario, the communication with a AAA server is not used, nevertheless, we have the capacity of adapting to more complex scenarios such as the ones previously described.

### 6.5. Other use cases

#### 6.5.1. CoAP-EAP for network access control

One of the first steps for an IoT device life-cycle is to perform the authentication to gain access to the network. To do so, the device first has to be authenticated and granted authorization to gain access to the network. Additionally, security parameters such as credentials can be derived from the authentication process allowing the trustworthy operation of the IoT device in a particular network by joining the security domain. By using EAP, we are able to achieve this with flexibility and scalability, because of the different EAP methods available and the ability to rely on AAA infrastructures if needed to support multi-domain scenarios, which is a key feature when the IoT devices deployed under the same security domain, belong to different organizations. Given that EAP is also used for network access control, we can adapt this service for other technologies. For instance, to provide network access control to very constrained technologies (e.g., LoRa network). In this specific case, we could leverage the compression by SCHC for CoAP.

#### 6.5.2. CoAP-EAP for service authentication

It is not uncommon that the infrastructure where the device is deployed and the services of the IoT device are managed by different organizations. Therefore, in addition to the authentication for network access control, we have to consider the possibility of a

secondary authentication to access different services. This process of authentication, for example, will provide with the necessary key material to establish a secure channel and interact with the entity in charge of granting access to different services.

# 7. Discussion

## 7.1. CoAP as EAP lower layer

In this section, we discuss the suitability of the CoAP protocol as EAP lower layer, and review the requisites imposed by the EAP protocol to any protocol that transports EAP. The assumptions EAP makes about its lower layers can be found in section 3.1 of [RFC3748], which are enumerated next:

- Unreliable transport. EAP does not assume that lower layers are reliable.
- Lower layer error detection. EAP relies on lower layer error detection (e.g., CRC, Checksum, MIC, etc.)
- Lower layer security. EAP does not require security services from the lower layers.
- Minimum MTU. Lower layers need to provide an EAP MTU size of 1020 octets or greater.
- Possible duplication. EAP stipulates that, while desirable, it does not require for the lower layers to provide non-duplication.
- Ordering guarantees. EAP relies on lower layer ordering guarantees for correct operation.

Regarding unreliable transport, although EAP assumes a non-reliable transport, CoAP does provide a reliability mechanism through the use of Confirmable messages. For the error detection, CoAP goes on top of UDP which provides a checksum mechanism over its payload. Lower layer security services are not required. About the minimum MTU of 1020 octets, CoAP assumes an upper bound of 1024 for its payload which covers the requirements of EAP. Regarding message ordering, every time a new message arrives at the bootstrapping service hosted by the IoT device, a new resource is created and this is indicated in a 2.01 Created response code along with the name of the new resource via Location-Path or Location-Query. This way the application indicates that its state has advanced. The name of the resource MAY be represented following the naming "/b/x". Where "b" is the general name of the bootstrapping service; "x" represents the resource established to refer to the ongoing authentication exchange.

NOTE: This document does not assume any specific naming schema. The only requisite that both CoAP client and server MUST agree, is the establishment of a nomenclature indicates that the next URI used to refer to a resource, univocally points to the next expected EAP exchange.

Regarding the Token, we consider the use of a constant value. This is because the EAP server will not send a new EAP request until it has processed the expected EAP response. Additionally, we are under the assumption that there will a single EAP authentication between the constrained device and the same Controller. This would also enable the possibility of using an Empty Token to reduce the number of bytes.

As we can see, CoAP can fulfil the requirements of EAP to be considered suitable as lower layer.

### 7.2. Size of the EAP lower layer vs EAP method size

Regarding the impact an EAP lower layer will have to the total byte size of the whole exchange, there is a comparison with another network layer based EAP lower layer, PANA [RFC5191] in [coap-eap]. Authors compared focusing EAP lower layer (alone) and taking into account EAP. On the one hand, at the EAP lower layer level, the usage of CoAP gives important benefits. On the other hand, when taking into account the EAP method overload, this reduction is less but still significant if the EAP method is lightweight (we used EAP-PSK as a representative example of a lightweight EAP method). If the EAP method is very taxing the improvement achieved in the EAP lower layer is less significant. This leads to the conclusion that possible next steps in this field could be also improving or designing new EAP methods that can be better adapted to the requirements of constrained devices and networks. However, we cannot ignore the impact of the EAP lower layer itself and try to propose something lightweight as CoAP. We consider that may be other EAP methods such as EAP-AKA or new lightweight EAP methods such as EAP-EDHOC [I-D.ingles-eap-edhoc] that can benefit from a CoAP-based EAP lower layer, as well as new ones that may be proposed in the future with IoT constraints in mind.

### 7.3. Controller as the CoAP Client

Due to the constrained capacities of the devices, to relieve them of the retransmission tasks, we set the Controller as the CoAP client, for the main exchange following the recommendations of the [I-D.ietf-lwig-coap] document to simplify the constrained device implementation.

### 7.4. Possible Optimizations

#### 7.4.1. Empty Token

Assuming that the bootstrapping service runs before any other service, and that no other service will run concurrently until it has finished, we could use an Empty Token value to save resources, since there will be no other endpoint or CoAP exchange.

#### 7.4.2. Further re-authentication

Since the initial bootstrapping is usually taxing, it is assumed to be done only once over a long period of time. If further re-authentications for refreshing the key material are necessary, there are other methods that can be used to perform these re-authentications. For example, the EAP re-authentication (ERP) [RFC6696] can be used to avoid repeating the entire EAP exchange in few exchanges.

## 8. Security Considerations

There are some aspects to be considered such as how authorization is managed, how the cryptographic suite is selected and how the trust in the Controller is established.

### 8.1. Authorization

Authorization is part of bootstrapping. It serves to establish whether the node can join and the set of conditions it has to adhere. The authorization data received from the AAA server can be delivered by the AAA protocol (e.g. Diameter). Providing more fine-grained authorization data can be with the transport of SAML in RADIUS [RFC7833]. After bootstrapping, additional authorization to operate in the security domain, e.g., access services offered by other nodes, can be taken care of by the solutions proposed in the ACE WG.

### 8.2. Cryptographic suite selection

How the cryptographic suite is selected is also important. To reduce the overhead of the protocol we use a default cryptographic suite. As OSCORE is assumed to run after the EAP authentication, the same default crypto-suite is used in this case as explained in the Key Derivation Section Section 5 The cryptographic suite is not negotiated. If the cryptographic suite to be used by the node is different from the default, the AAA server will send the specific parameters to the Authenticator. If the cryptographic suite is not supported, the key derivation process would result in a security association failure.

### 8.3. Freshness of the key material

In this design, we do not exchange nonces to provide freshness to the keys derived from the MSK. This is done under the assumption that the MSK and EMSK keys derived following the EAP KMF [RFC5247] are fresh key material by the specifications of the EAP KMF. Since only one session key is derived from the MSK we do not have to concern ourselves with the generation of additional key material. In case another session has to be established, a re-authentication can be done, by running the process again, or using a more lightweight EAP method to derive additional key material such as ERP [RFC6696].

### 8.4. Additional Security Consideration

Other security-related concerns can be how to ensure that the node joining the security domain can in fact trust the Controller. This issue is elaborated in the EAP KMF [RFC5247]. To summarizing, the node knows it can trust the Controller because the key that is used to establish the security association is derived from the MSK. If the Controller has the MSK, it is clear the AAA Server of the node trusts the Controller, which confirms it is a trusted party.

## 9. IANA Considerations

TBD.

## 10. Acknowledgments

We would like to thank as the reviewers of this work: Carsten Bormann, Benjamin Kaduk, Alexandre Petrescu, Pedro Moreno-Sanchez and Eduardo Ingles-Sanchez.

We would also like to thank Gabriel Lopez-Millan for the first review of this document and we would like to thank Ivan Jimenez-Sanchez for the first proof-of-concept implementation of this idea.

And thank for their valuables comments to Alexander Pelov and Laurent Toutain, especially for the potential optimizations of CoAP-EAP.

## 11. References

### 11.1. Normative References

[I-D.ietf-ace-oauth-authz]
Seitz, L., Selander, G., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "Authentication and Authorization for Constrained Environments (ACE) using the OAuth 2.0 Framework (ACE-OAuth)", draft-ietf-ace-oauth-authz-36 (work in progress), November 2020.

[I-D.ietf-lwig-coap]
Kovatsch, M., Bergmann, O., and C. Bormann, "CoAP Implementation Guidance", draft-ietf-lwig-coap-06 (work in progress), July 2018.

[I-D.ingles-eap-edhoc]
Sanchez, E., Garcia-Carrillo, D., and R. Marin-Lopez, "EAP method based on EDHOC Authentication", draft-ingles-eap-edhoc-01 (work in progress), November 2020.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, Ed., "Extensible Authentication Protocol (EAP)", RFC 3748, DOI 10.17487/RFC3748, June 2004, <https://www.rfc-editor.org/info/rfc3748>.

[RFC4493] Song, JH., Poovendran, R., Lee, J., and T. Iwata, "The AES-CMAC Algorithm", RFC 4493, DOI 10.17487/RFC4493, June 2006, <https://www.rfc-editor.org/info/rfc4493>.

[RFC4615] Song, J., Poovendran, R., Lee, J., and T. Iwata, "The Advanced Encryption Standard-Cipher-based Message Authentication Code-Pseudo-Random Function-128 (AES-CMAC-PRF-128) Algorithm for the Internet Key Exchange Protocol (IKE)", RFC 4615, DOI 10.17487/RFC4615, August 2006, <https://www.rfc-editor.org/info/rfc4615>.

[RFC5191] Forsberg, D., Ohba, Y., Ed., Patil, B., Tschofenig, H., and A. Yegin, "Protocol for Carrying Authentication for Network Access (PANA)", RFC 5191, DOI 10.17487/RFC5191, May 2008, <https://www.rfc-editor.org/info/rfc5191>.

[RFC5247] Aboba, B., Simon, D., and P. Eronen, "Extensible Authentication Protocol (EAP) Key Management Framework", RFC 5247, DOI 10.17487/RFC5247, August 2008, <https://www.rfc-editor.org/info/rfc5247>.

[RFC5295] Salowey, J., Dondeti, L., Narayanan, V., and M. Nakhjiri, "Specification for the Derivation of Root Keys from an Extended Master Session Key (EMSK)", RFC 5295, DOI 10.17487/RFC5295, August 2008, <https://www.rfc-editor.org/info/rfc5295>.

[RFC6696] Cao, Z., He, B., Shi, Y., Wu, Q., Ed., and G. Zorn, Ed., "EAP Extensions for the EAP Re-authentication Protocol (ERP)", RFC 6696, DOI 10.17487/RFC6696, July 2012, <https://www.rfc-editor.org/info/rfc6696>.

[RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, DOI 10.17487/RFC7228, May 2014, <https://www.rfc-editor.org/info/rfc7228>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <https://www.rfc-editor.org/info/rfc7252>.

[RFC7833] Howlett, J., Hartman, S., and A. Perez-Mendez, Ed., "A RADIUS Attribute, Binding, Profiles, Name Identifier Format, and Confirmation Methods for the Security Assertion Markup Language (SAML)", RFC 7833, DOI 10.17487/RFC7833, May 2016, <https://www.rfc-editor.org/info/rfc7833>.

[RFC7967] Bhattacharyya, A., Bandyopadhyay, S., Pal, A., and T. Bose, "Constrained Application Protocol (CoAP) Option for No Server Response", RFC 7967, DOI 10.17487/RFC7967, August 2016, <https://www.rfc-editor.org/info/rfc7967>.

[RFC8613] Selander, G., Mattsson, J., Palombini, F., and L. Seitz, "Object Security for Constrained RESTful Environments (OSCORE)", RFC 8613, DOI 10.17487/RFC8613, July 2019, <https://www.rfc-editor.org/info/rfc8613>.

## 11.2. Informative References

[coap-eap]
Garcia-Carrillo, D. and R. Marin-Lopez, "Lightweight CoAP-Based Bootstrapping Service for the Internet of Things - https://www.mdpi.com/1424-8220/16/3/358", March 2016.

Authors' Addresses

Rafa Marin-Lopez
University of Murcia
Campus de Espinardo S/N, Faculty of Computer Science
Murcia 30100
Spain

Phone: +34 868 88 85 01
Email: rafa@um.es

Dan Garcia-Carrillo
University of Oviedo
Calle Luis Ortiz Berrocal S/N, Edificio Polivalente
Gijon, Asturias 33203
Spain

Email: garciadan@uniovi.es