

ACE Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: August 26, 2021

R. Marin  
University of Murcia  
D. Garcia  
University of Oviedo  
February 22, 2021

**EAP-based Authentication Service for CoAP**  
**draft-ietf-ace-wg-coap-eap-00**

Abstract

This document describes an authentication service that uses EAP transported by means of CoAP messages with the following purposes:

- o Authenticate a CoAP-enabled device that enters a new security domain managed by a domain Controller.
- o Derive key material to protect CoAP messages exchanged between them, enabling the establishment of a security association between them.
- o 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 August 26, 2021.

## Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](https://trustee.ietf.org/license-info) and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

<a href="#">1.</a>	<a href="#">Introduction</a>	<a href="#">3</a>
<a href="#">1.1.</a>	<a href="#">Requirements Language</a>	<a href="#">3</a>
<a href="#">2.</a>	<a href="#">General Architecture</a>	<a href="#">4</a>
<a href="#">3.</a>	<a href="#">General Flow Operation</a>	<a href="#">4</a>
<a href="#">3.1.</a>	<a href="#">EAP over CoAP flow of operation</a>	<a href="#">5</a>
<a href="#">3.2.</a>	<a href="#">The SeqNum Option</a>	<a href="#">8</a>
<a href="#">4.</a>	<a href="#">Key Derivation for protecting CoAP messages</a>	<a href="#">9</a>
<a href="#">4.1.</a>	<a href="#">Deriving the OSCORE Security Context</a>	<a href="#">9</a>
<a href="#">4.2.</a>	<a href="#">Deriving DTLS_PSK</a>	<a href="#">11</a>
<a href="#">5.</a>	<a href="#">Examples of Use Case Scenario</a>	<a href="#">11</a>
<a href="#">5.1.</a>	<a href="#">Example 1: CoAP-EAP in ACE</a>	<a href="#">12</a>
<a href="#">5.2.</a>	<a href="#">Example 2: Multi-domain with AAA infrastructures</a>	<a href="#">13</a>
<a href="#">5.3.</a>	<a href="#">Example 3: Single domain with AAA infrastructure</a>	<a href="#">13</a>
<a href="#">5.4.</a>	<a href="#">Example 4: Single domain without AAA infrastructure</a>	<a href="#">13</a>
<a href="#">5.5.</a>	<a href="#">Other use cases</a>	<a href="#">13</a>
<a href="#">5.5.1.</a>	<a href="#">CoAP-EAP for network access control</a>	<a href="#">13</a>
<a href="#">5.5.2.</a>	<a href="#">CoAP-EAP for service authentication</a>	<a href="#">14</a>
<a href="#">6.</a>	<a href="#">Discussion</a>	<a href="#">14</a>
<a href="#">6.1.</a>	<a href="#">CoAP as EAP lower-layer</a>	<a href="#">14</a>
<a href="#">6.2.</a>	<a href="#">Need for SeqNum Option</a>	<a href="#">15</a>
<a href="#">6.3.</a>	<a href="#">Size of the EAP lower-layer vs EAP method size</a>	<a href="#">15</a>
<a href="#">6.4.</a>	<a href="#">Controller as the CoAP Client</a>	<a href="#">16</a>
<a href="#">6.5.</a>	<a href="#">Possible Optimizations</a>	<a href="#">16</a>
<a href="#">6.5.1.</a>	<a href="#">Empty Token</a>	<a href="#">16</a>
<a href="#">6.5.2.</a>	<a href="#">Removing SeqNum Option</a>	<a href="#">16</a>
<a href="#">6.5.3.</a>	<a href="#">Further re-authentication</a>	<a href="#">17</a>
<a href="#">7.</a>	<a href="#">Security Considerations</a>	<a href="#">17</a>
<a href="#">7.1.</a>	<a href="#">Authorization</a>	<a href="#">18</a>
<a href="#">7.2.</a>	<a href="#">Cryptographic suite selection</a>	<a href="#">18</a>
<a href="#">7.3.</a>	<a href="#">Freshness of the key material</a>	<a href="#">18</a>
<a href="#">7.4.</a>	<a href="#">Additional Security Consideration</a>	<a href="#">18</a>



<a href="#">8.</a>	<a href="#">IANA Considerations . . . . .</a>	<a href="#">19</a>
<a href="#">9.</a>	<a href="#">Acknowledgments . . . . .</a>	<a href="#">19</a>
<a href="#">10.</a>	<a href="#">References . . . . .</a>	<a href="#">19</a>
<a href="#">10.1.</a>	<a href="#">Normative References . . . . .</a>	<a href="#">19</a>
<a href="#">10.2.</a>	<a href="#">Informative References . . . . .</a>	<a href="#">21</a>
	<a href="#">Authors' Addresses . . . . .</a>	<a href="#">21</a>

## [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 without the need of additional protocols to establish a security association between them.

In particular, the document describes how CoAP can be used as a constrained, link-layer independent, EAP lower-layer [[RFC3748](#)] to transport EAP 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. 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:

- o OSCORE [[RFC8613](#)] security association can be established based on the cryptographic material generated from the EAP authentication.
- o A DTLS security association can be established using the exported cryptographic material after a successful EAP authentication.  
[[I-D.ohba-core-eap-based-bootstrapping](#)]

This document also provides comments on 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 EAP peer will act as CoAP server for this service and the EAP authenticator will act as 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. Nevertheless, a model where the EAP peer acts as CoAP client and the EAP authenticator as CoAP server can be also analyzed in the future.

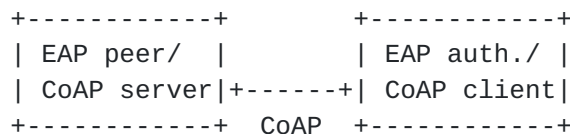


Figure 1: CoAP EAP Architecture

## 3. General Flow Operation

The authentication service uses CoAP as transport layer 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, be a not 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 authentication mechanism. (NOTE: any EAP method which is able to export cryptographic material is be 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, is set to "/b" to save bytes over the air. Alternatively, the if the Controller is aware of the presence of the IoT device (e.g., due to a previous authentication) this process can be avoided, and the Controller can directly start the authentication process.

The first message that is used to trigger the authentication process is sent by the IoT device, acting as 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 CoAP client and the IoT device acting as CoAP server. This is due to the fact that the IoT could be a constrained node, and following the recommendations of [\[I-D.ietf-lwig-coap\]](#) to simplify the implementation of the IoT device, having the Controller 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 from the work on [\[RFC7967\]](#) called no response. The rationale of this option is to avoid waiting for a if it is not needed. So the use of this option will allow signaling the intention the EAP peer to start the authentication process, as a mechanism. Immediately after that, the EAP authenticator will start authentication service. It is worth noting that the EAP authenticator MAY decide start the authentication without waiting for the trigger if it has knowledge about the presence of the peer, for instance, through a previous authentication.

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). This POST message contains a new option SeqNum that holds a sequence number randomly chosen by the CoAP client. This SeqNum is used to provide ordered and reliable delivery of messages involved during the whole authentication. In general, when a CoAP request with EAP message is received, the CoAP client considers a valid message if only if its sequence number is the expected value. The sequence number is monotonically incremented by 1 so that the CoAP server can know what it is the next expected sequence number. After receiving the first POST, the CoAP server assigns a resource and answers with an Acknowledgment with the piggy-backed resource identifier (Uri-Path) (Step 2). It is assumed that the CoAP server will only have an ongoing authentication and will not process simultaneous EAP authentications in parallel to save resources. In these two messages, the EAP Req/Id and Rep/ID are exchanged between the EAP authenticator and the EAP peer. The EAP Req/Id message is forwarded by the EAP authenticator, when EAP is in pass-through mode, to the local AAA server that 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





together to relate all the EAP conversation between both CoAP endpoints. Since, only an ongoing EAP authentication is permitted and EAP is a lock-step protocol a Token of a constant value and 1 byte can be used throughout the authentication process. This also allows to save bytes through the link.

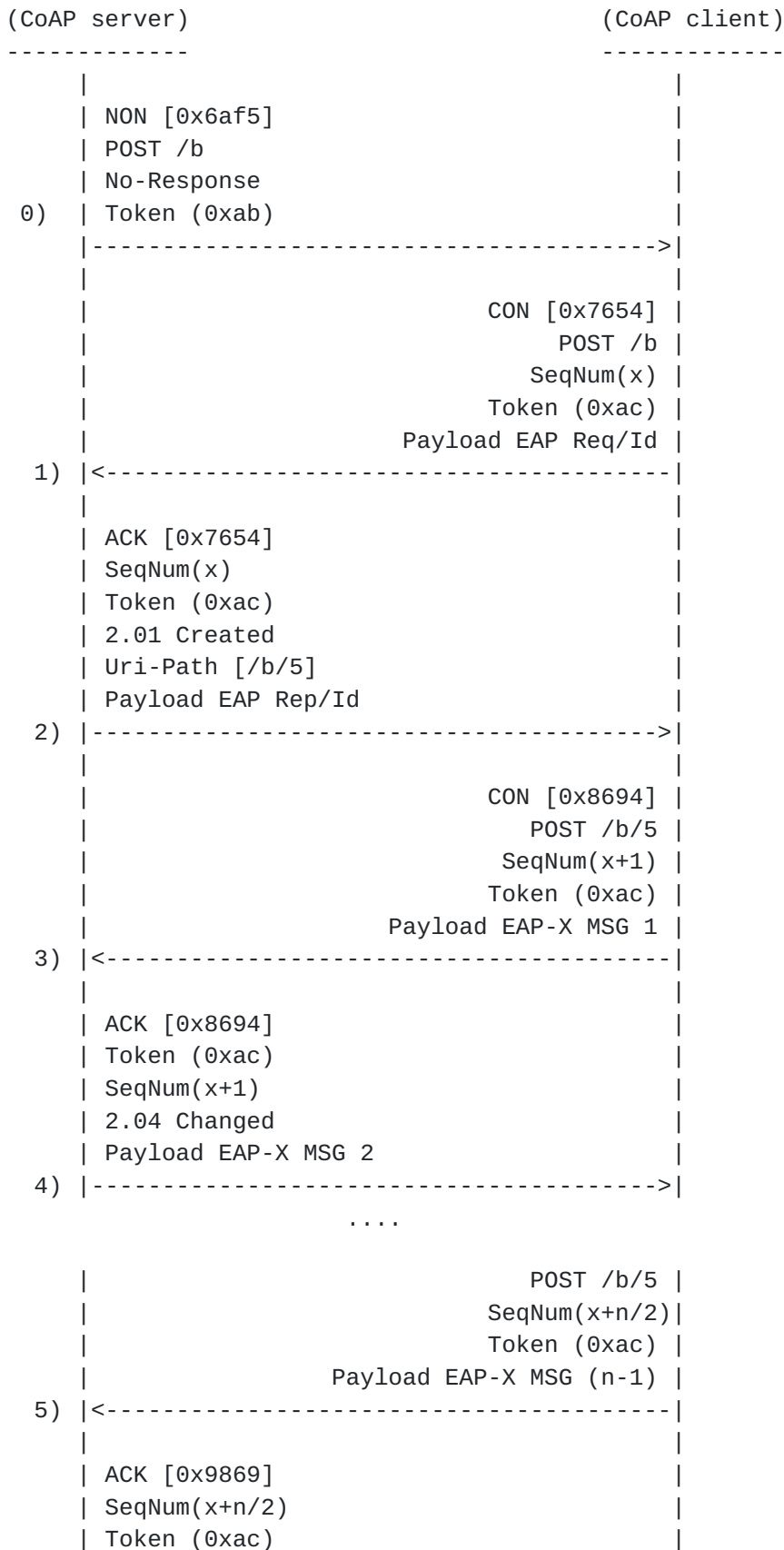
From now on, the EAP authenticator and the EAP peer will exchange EAP packets related to the EAP method, transported in the CoAP message payload (Steps 3,4,5,6). The EAP authenticator will use POST method to send EAP requests to the EAP peer. The EAP peer will use a Piggy-backed response in the Acknowledgment message to carry the EAP response. At the end of the message exchanges, if everything has gone well, the EAP authenticator is able to 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 sever, 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. This is shown in Steps 7 and 8. From that point any exchange between both CoAP endpoints are protected with OSCORE. Before sending the EAP success to the EAP peer, the EAP authenticator is able to 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 4.1](#) The protection of the EAP Success is not a requirement. In our case, we specify this exchange as protected by the lower layer in this scenario with OSCORE. The purpose is double, we can avoid forgery of this message and at the same time 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 preclude 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 a CoAP message without EAP and with an OSCORE option (following the normal operational communication between the both entities) is an indication that the controller considers the EAP authentication finished. 2) the IoT device is aware that the EAP authentication went well if an MSK is available. In any case, both entities need to prove the possession of the MSK as mentioned in the EAP KMF.

EAP peer

EAP Auth.







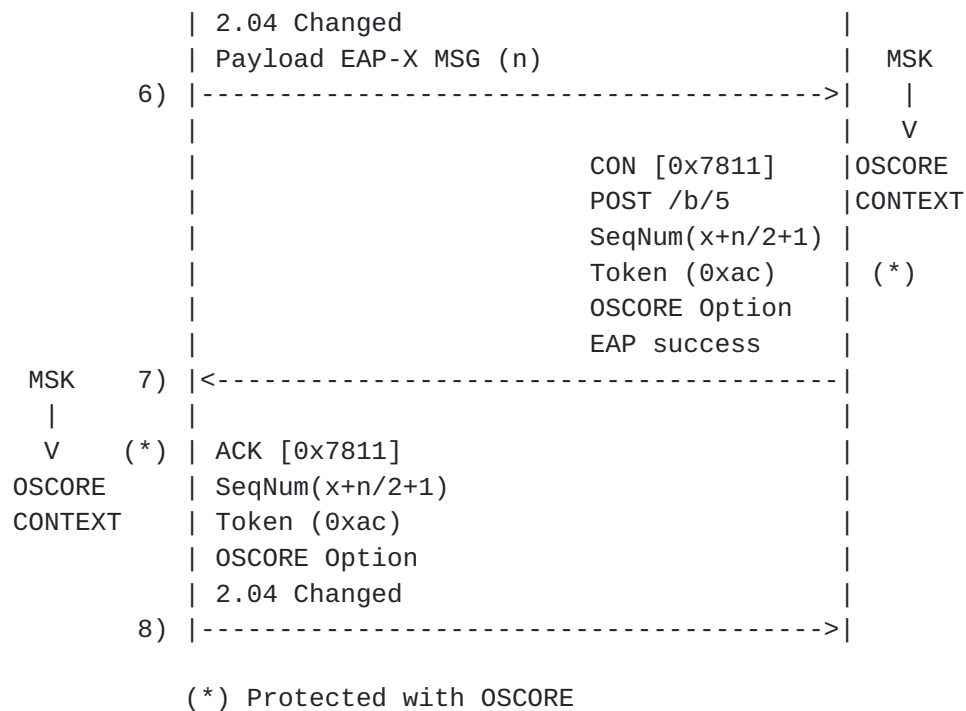


Figure 2: CoAP-EAP flow of operation

### 3.2. The SeqNum Option

A new SeqNum option is defined in this document for establishing the ordering guarantee of the EAP exchange. Following guidelines in [\[RFC7252\]](#) this option is:

1. Format opaque (sequence of bytes).
2. Critical
3. Safe to Forward
4. No cacheable and Not part of the Cache-Key
5. Not repeatable

The number of the option will be determined by this previous decisions.

1. Critical (C = 1)
2. Safe to Forward (1)



### 3. NoCacheKey (111)

The number of the SeqNum option will fit this pattern: xxx11111

```

      0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
|           | NoCacheKey | U | C |
+---+---+---+---+---+---+---+

```

Figure 3: SeqNum Option Number Mask

The option number is TBD.

The resultant SeqNum option is:

```

+-----+---+---+---+---+---+-----+-----+-----+-----+
| No.  | C | U | N | R | Name   | Format | Length | Default |
+-----+---+---+---+---+---+-----+-----+-----+-----+
| TBD  | x |   | x |   | SeqNum | uint   | 0-16   | (none)  |
+-----+---+---+---+---+---+-----+-----+-----+-----+

```

C = Critical,    U = Unsafe,    N = NoCacheKey,    R = Repeatable  
(\*) See below.

Figure 4: SeqNum option

## 4. Key Derivation for protecting CoAP messages

As a result of a successful EAP authentication, both CoAP server and CoAP client share a Master Key Session (MSK). The assumption is that MSK is a fresh key so any derived key from the MSK will be also fresh. To complete the CoAP-EAP exchange, as part of the design, the establishment of an OSCORE security association specifically for the CoAP-EAP service is expected. 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 of the security requirements of the services provided by the AS. The CoAP-EAP OSCORE security context could be generalized to enable further OSCORE secured communications between the IoT device and the AS services that require the use of OSCORE.

### 4.1. Deriving the OSCORE Security Context

Key material needed to derive the OSCORE Security Context, from the MSK can be done as follows. In this case, rest of CoAP exchanges between both entities can be protected with OSCORE.





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:

- o The AES-CMAC-PRF-128 is defined in [[RFC4615](#)]. This function uses AES-CMAC-128 as building block.
- o 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 7](#).
- o "IETF\_OSCORE\_MASTER\_SECRET" is the ASCII code representation of the non-NULL terminated string (excluding the double quotes around it).
- o 64 is the length of the MSK.
- o length is the length of the label "IETF\_OSCORE\_MASTER\_SECRET" (25 bytes).

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

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

where:

- o The AES-CMAC-PRF-128 is defined in [[RFC4615](#)]. This function uses AES-CMAC-128 as building block.
- o 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 7](#).
- o "IETF\_OSCORE\_MASTER\_SALT" is the ASCII code representation of the non-NULL terminated string (excluding the double quotes around it).
- o 64 is the length of the MSK.
- o 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.



#### **4.2. Deriving DTLS\_PSK**

In the second alternative, a DTLS\_PSK is derived from the MSK between both CoAP endpoints. So far, DTLS\_PSK will have also 16 byte length and it will be derived as follows:

$\text{DTLS\_PSK} = \text{KDF}(\text{MSK}, \text{"IETF\_DTLS\_PSK"}, 64, \text{length})$ . This value is concatenated with the value of the Token Option value.

where:

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

#### **5. 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, fresh key material is shared between the IoT device and the AS.

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

- o 2 nodes (A and B), which are constrained devices. They are the EAP peers.
- o 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.
- o 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 the controller C. Moreover, any other entity, which wants to join and establish communications with nodes under the 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 a we could be using a heavier authentication method. With very constrained devices we might need to go to other authentication methods (e.g., EAP-PSK, EAP-EDHOC, etc.) being able to adapt to different types of devices according to policies or devices capabilities.

### **5.1. 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 active a light switch). The overall process is divided in three phases. Let's start with node A. In the first phase, the node A (EAP peer) does not yet belong to the 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 the 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 the controller C's domain during 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 the controller C for authorization to access node B and obtain all the required information to do that in a secure manner (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 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.

## **5.2. 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 the controller C. In this scenario the AS under the role of the Controller, receives the key material from the AAA infrastructure

## **5.3. 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.

## **5.4. Example 4: Single domain without AAA infrastructure**

Another case, without a AAA infrastructure, we have a Controller that has co-located the AAA server and using EAP standalone mode we are able to 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 AAA server is co-located within the Controller (C)-

## **5.5. Other use cases**

### **5.5.1. 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 relying in 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, it is possible that this service can be used to provide network access control service (e.g., LoRa network). In this specific case, we could leverage the compression by SCHC for CoAP.

### **5.5.2. CoAP-EAP for service authentication**

It is not uncommon that the infrastructure where the device is deployed and the services 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.

## **6. Discussion**

### **6.1. 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:

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



- o Ordering guarantees. EAP relies on lower layer ordering guarantees for correct operation.

Regarding the 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, we propose the use of a new CoAP option, the SeqNum option described in Section ([Section 3.2](#)), which will allow us to determine the order in which the different messages are exchanged. Regarding the Token, we consider the use of a constant value using a small 1 byte Token. In fact, 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 be a single EAP authentication between the constrained device and the same Controller.

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

## **6.2. Need for SeqNum Option**

We consider the use of the SeqNum Option due to the independence of how the CoAP engine is implemented. Since we do not know before hand if the implementation will allow us to pre-establish the MSG-ID or the Token from the application perspective, we need to be sure we are able to provide order delivery. If the implementation of CoAP allows us to pre-establish the MSG-ID and Token, we could avoid using this option, due to the characteristics of the CoAP-EAP exchange, i.e., the EAP exchange is done in lock-step and only one session is considered at a time. Another consideration regarding the workings of the SeqOption, is that since the initial number from which is monotonically increased by 1, could cause the overflow of the number. To manage this scenario, the SeqNum Option performs rounding, going to zero and continue from there.

## **6.3. 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 EAP lower-layer level, the usage of CoAP gives us an 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 light as we do proposing CoAP. We consider that may be others 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.

#### **[6.4.](#) 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.

#### **[6.5.](#) Possible Optimizations**

##### **[6.5.1.](#) 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.

##### **[6.5.2.](#) Removing SeqNum Option**

An alternative to consider would be to try to rely on the Message ID values as a way of achieving the order delivery throughout the authentication exchange. Here we have two approximations: 1) Removing the option from the ACKs and 2) removing the option completely.

1. Since the ACKs are piggybacked by design, there is only 1 ongoing authentication process and the EAP exchange is done in a lockstep fashion, when we get a response we will get the same Message ID of the request and we can confirm the SeqNum of the Request.
2. An alternative to consider would be to try to solely rely on the Message ID values as a way of achieving the order delivery throughout the authentication exchange. Here we also have two approaches: A) To expect randomly generated Message IDs and B) set the Message ID to increase monotonically by 1.



- A. Regarding the use of the Message ID, their values in the requests sent by the Controller are generated randomly, as suggested by CoAP. The Controller selects a new Message ID value each time a new request is sent to the CoAP server, until the bootstrapping service finishes. Moreover, the Controller stores the last Message ID sent until correctly receiving the corresponding ACK. The CoAP server keeps track of the last received Message ID to identify retransmissions, and the previous Message IDs during the current bootstrapping to identify old messages. In general, a request is considered valid in terms of the Message ID if either this value matches the last value received, which means a retransmission of the last response is required, or the arrival of a new Message ID, which therefore represents a new message. If these rules do not apply (i.e., an old Message ID has been received), the CoAP server silently discards the request. This is possible because the bootstrapping service is designed as lockstep, i.e. the Controller will not send a new request until it has received the expected response. When the bootstrapping exchange finishes successfully, the CoAP server can free the tracked Message IDs, except for the last received Message ID at the end of the bootstrapping, just in case a retransmission is required.
- B. This case would avoid having to keep track of the already used Message IDs, monotonically increasing by 1 the message ID value once the first is randomly picked by the Controller.

### **6.5.3. 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 (EAP-ERP) [[RFC6696](#)] can be used to avoid repeating the entire EAP exchange in few exchanges.

## **7. 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.





### **7.1. Authorization**

Authorization is part of the 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.

### **7.2. Cryptographic suite selection**

How the cryptographic suit 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 4](#). The cryptographic suite is not negotiated. If the cryptographic suite to be used by the node is different from 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.

### **7.3. 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 process again, or using a more lightweight EAP method to derive additional key material such as EAP-ERP.

### **7.4. 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](#)]. 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.



## **8. IANA Considerations**

This document has no actions for IANA.

## **9. Acknowledgments**

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

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

As well as the reviewers of this work, Alexandre Petrescu, Pedro Moreno-Sanchez, Eduardo Ingles-Sanchez and Benjamin Kaduk.

## **10. References**

### **10.1. 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.

[I-D.ohba-core-eap-based-bootstrapping]

Das, S. and Y. Ohba, "Provisioning Credentials for CoAP Applications using EAP", [draft-ohba-core-eap-based-bootstrapping-01](#) (work in progress), March 2012.

[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>>.

## **10.2. 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



