ACE Working Group
Internet-Draft
Intended status: Standards Track
Expires: 23 August 2025

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

19 February 2025

EAP-based Authentication Service for CoAP
draft-ietf-ace-wg-coap-eap-15

Abstract

This document specifies an authentication service that uses the Extensible Authentication Protocol (EAP) transported employing Constrained Application Protocol (CoAP) messages. As such, it defines an EAP lower layer based on CoAP called CoAP-EAP. One of the main goals is to authenticate a CoAP-enabled IoT device (EAP peer) that intends to join a security domain managed by a Controller (EAP authenticator). Secondly, it allows deriving key material to protect CoAP messages exchanged between them based on Object Security for Constrained RESTful Environments (OSCORE), enabling the establishment of a security association between them.

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 23 August 2025.

Copyright Notice

Table of Contents

1. Introduction

This document specifies an authentication service (application) that uses the Extensible Authentication Protocol (EAP) [RFC3748] and is built on top of the Constrained Application Protocol (CoAP)[RFC7252] called CoAP-EAP. CoAP-EAP is an application that allows authenticating two CoAP endpoints by using EAP and establishing an Object Security for Constrained RESTful Environments (OSCORE) security association between them. More specifically, this document specifies how CoAP can be used as a constrained, link-layer independent, reliable EAP lower layer [RFC3748] to transport EAP messages between a CoAP server (acting as EAP peer) and a CoAP client (acting as EAP authenticator) using CoAP messages. The CoAP client has the option of contacting a backend AAA infrastructure to complete the EAP negotiation, as described in the EAP specification [RFC3748].

The EAP methods that can be transported with CoAP-EAP MUST export cryptographic material [RFC5247] for this specification. Examples of such methods are EAP-GPSK [RFC5433], EAP-SIM [RFC4186], EAP-AKA' [RFC5448], EAP-TLS 1.3 [RFC9190], EAP-EDHOC [I-D.ietf-emu-eap-edhoc], etc. In general, any EAP method designed in EMU Working Group that exports the Master Session Key (MSK) can be used with CoAP-EAP. The Master Session Key (MSK) is used as the basis for further cryptographic derivations. This way, CoAP messages are protected after authentication. After CoAP-EAP's operation, an OSCORE security association is established between the endpoints of the service. Using the keying material from the authentication, other security associations could be generated. Appendix A shows how to establish a (D)TLS security association using the keying material from the EAP authentication.

One of the main applications of CoAP-EAP is Internet of Things (IoT) networks, where we can find very constrained links (e.g., limited bandwidth) and devices with limited capabilities. In these IoT scenarios, we identify the IoT device as the entity that wants to be authenticated by using EAP to join a domain that is managed by a Controller. The IoT device is in these cases the EAP peer and the Controller, the entity steering the authentication, the EAP authenticator. From now on, the IoT device is referred to as the EAP peer and the Controller as the EAP authenticator. In these cases, EAP methods with fewer exchanges, shorter messages, and cryptographic algorithms suitable for constrained devices are preferable. The benefits of the EAP framework in IoT are highlighted in [EAP-framework-IoT].

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Readers are expected to be familiar with the terms and concepts described in CoAP [RFC7252], EAP [RFC3748] [RFC5247] and OSCORE [RFC8613].

2. General Architecture

Figure 1 illustrates the architecture defined in this document. In this architecture, the Extensible Authentication Protocol (EAP) peer will act as a CoAP server for this service, and the domain EAP authenticator as a CoAP client. The rationale behind this decision is that EAP requests direction is always from the EAP server to the EAP peer. Accordingly, EAP responses direction is always from the EAP peer to the EAP server.

It is worth noting that the EAP authenticator MAY interact with a backend AAA infrastructure when EAP pass-through mode is used, which will place the EAP server in the AAA server that contains the information required to authenticate the EAP peer.

The protocol stack is described in Figure 2. CoAP-EAP is an application built on top of CoAP. On top of the application, there is an EAP state machine that can run any EAP method. For this specification, the EAP method MUST support key derivation and export, as specified in [RFC5247], a Master Session Key (MSK) of at least 64 octets, and an Extended Master Session Key (EMSK) of at least 64 octets. CoAP-EAP also relies on CoAP reliability mechanisms in CoAP to transport EAP: CoAP over UDP with Confirmable messages ([RFC7252]) or CoAP over TCP, TLS, or WebSockets [RFC8323].

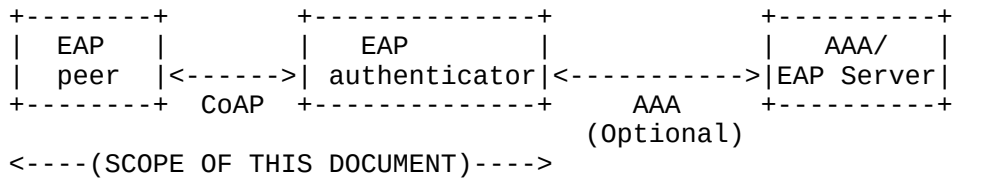

Figure 1: CoAP-EAP Architecture

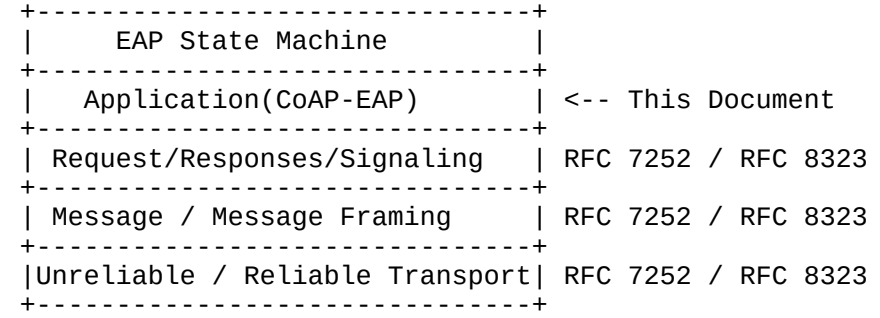

Figure 2: CoAP-EAP Stack

3. CoAP-EAP Operation

Because CoAP-EAP uses reliable delivery defined in CoAP ([RFC7252], [RFC8323]), EAP retransmission time is set to infinite as mentioned in [RFC3748]. To keep the ordering guarantee, CoAP-EAP uses Hypermedia as the Engine of Application State (HATEOAS). Each step during the EAP authentication accesses a new resource in the CoAP server (EAP peer). The previous resource is removed once the new resource is created, indicating the resource that will process the next step of the EAP authentication.

One of the benefits of using EAP is that we can choose from a large variety of authentication methods.

In CoAP-EAP, the EAP peer will only have one authentication session with a specific EAP authenticator, and it will not process any other EAP authentication in parallel (with the same EAP authenticator). That is, a single ongoing EAP authentication is permitted for the same EAP peer and the same EAP authenticator. It may be worth noting that the EAP authenticator may have parallel EAP sessions with multiple EAP peers.

To access the authentication service, this document defines the well-known URI "coap-eap" (to be assigned by IANA). The /.well-known/coap-eap URI is used with "coap", "coap+tcp" or "coap+ws".

3.1. Discovery

Before the CoAP-EAP exchange takes place, the EAP peer needs to discover the EAP authenticator or the entity that will enable the CoAP-EAP exchange (e.g., an intermediary proxy). The discovery process is out of the scope of this document.

The CoAP-EAP application can be accessed through the URI "coap-eap" for the trigger message (see Section 3.2, Step 0). The CoAP-EAP service can be discovered by asking directly about the services offered. This information can also be available in the resource directory [RFC9176].

Implementation Notes: There are different methods to discover the IPv6 address of the EAP authenticator or intermediary entity. For example, on a 6LoWPAN network, the Border Router will typically act as the EAP authenticator hence, after receiving the Router Advertisement (RA) messages from the Border Router, the EAP peer may engage on the CoAP-EAP exchange.

3.2. Flow of operation (OSCORE establishment)

Figure 3 shows the general flow of operation for CoAP-EAP to authenticate using EAP and establish an OSCORE security context. The flow does not show a specific EAP method. Instead, the chosen EAP method is represented by using a generic name (EAP-X). The flow assumes that the EAP peer knows the EAP authenticator implements the CoAP-EAP service. A CoAP-EAP message has a media type application/coap-eap, See Section 9.5.

The steps of the operation are as follows:

* Step 0. The EAP peer MUST start the CoAP-EAP process by sending a "POST /.well-known/coap-eap" request (trigger message). This message carries the ’No-Response’ [RFC7967] CoAP option to avoid waiting for a response that is not needed. This is the only

message where the EAP authenticator acts as a CoAP server and the EAP peer as a CoAP client. The message also includes a URI in the payload of the message to indicate the resource where the EAP authenticator MUST send the next message. The name of the resource is selected by the CoAP server.

Implementation notes: When generating the URI for a resource of a step of the authentication, the resource could have the following format as an example "path/eap/counter", where:

* "path" is some local path on the device to make the path unique. This could be omitted if desired.

* "eap" is the name that indicates the URI is for the EAP peer. This has no meaning for the protocol but helps with debugging.

* "counter' which is an incrementing unique number for every new EAP request.

So, in Figure 3 for example, the URI for the first resource would be "a/eap/1"

* Step 1. The EAP authenticator sends a POST message to the resource indicated in Step 0 (e.g., '/a/eap/1'). The payload in this message contains the first EAP message (EAP Request/ Identity), the Recipient ID of the EAP authenticator (RID-C) for OSCORE, and MAY contain a CBOR array with a list of proposed cipher suites (CS-C) for OSCORE. If the cipher suite list is not included, the default cipher suite for OSCORE is used. The details of the cipher suite negotiation are discussed in Section 6.1.

* Step 2. The EAP peer processes the POST message sending the EAP request (EAP-Req/Id) to the EAP peer state machine, which returns an EAP response (EAP Resp/Id). Then, assigns a new resource to the ongoing authentication process (e.g., '/a/eap/2'), and deletes the previous one ('/a/eap/1'). Finally, sends a '2.01 Created' response with the new resource identifier in the Location-Path (and/or Location-Query) options for the next step. The EAP response, the Recipient ID of the EAP peer (RID-I) and the selected cipher suite for OSCORE (CS-I) are included in the payload. In this step, the EAP peer may create an OSCORE security context (see Section 6.2). The required Master Session Key (MSK) will be available once the EAP authentication is successful in step 7.

* Steps 3-6. From now on, the EAP authenticator and the EAP peer will exchange EAP packets related to the EAP method (EAP-X), transported in the CoAP message payload. The EAP authenticator will use the POST method to send EAP requests to the EAP peer. The EAP peer will use a response to carry the EAP response in the payload. EAP requests and responses are represented in Figure 3 using the nomenclature (EAP-X-Req and EAP-X-Resp, respectively). When a POST message arrives (e.g., '/a/eap/1') carrying an EAP request message, if processed correctly by the EAP peer state machine, returns an EAP Response. Along with each EAP Response, a new resource is created (e.g., '/a/eap/3') for processing the next EAP request and the ongoing resource (e.g., '/a/eap/2') is erased. This way, ordering guarantee is achieved. Finally, an EAP response is sent in the payload of a CoAP response that will also indicate the new resource in the Location-Path (and/or Location-Query) Options. In case there is an error processing a legitimate message, the server will return a (4.00 Bad Request). There is a discussion about error handling in Section 3.5.

* Step 7. When the EAP authentication ends successfully, the EAP authenticator obtains the Master Session Key (MSK) exported by the EAP method, an EAP Success message, and some authorization information (e.g., session lifetime) [RFC5247]. The EAP authenticator creates the OSCORE security context using the MSK and Recipient ID of both entities exchanged in Steps 1 and 2. The establishment of the OSCORE Security Context is defined in Section 6.2. Then, the EAP authenticator sends the POST message protected with OSCORE for key confirmation including the EAP Success. The EAP authenticator MAY also send a Session Lifetime, in seconds, which is represented with an unsigned integer in a CBOR object (see Section 5). If this Session Lifetime is not sent, the EAP peer assumes a default value of 8 hours, as RECOMMENDED in [RFC5247]. The reception of the OSCORE-protected POST message is considered by the EAP peer as an alternate indication of success ([RFC3748]). The EAP peer state machine in the EAP peer interprets the alternate indication of success (similarly to the arrival of an EAP Success) and returns the MSK, which is used to create the OSCORE security context in the EAP peer to process the protected POST message received from the EAP authenticator.

* Step 8. If the EAP authentication and the verification of the OSCORE-protected POST in Step 7 is successful, then the EAP peer answers with an OSCORE-protected '2.04 Changed'. From this point on, communication with the last resource (e.g., '/a/eap/(n)') MUST be protected with OSCORE. If allowed by application policy, the same OSCORE security context MAY be used to protect communication to other resources between the same endpoints.

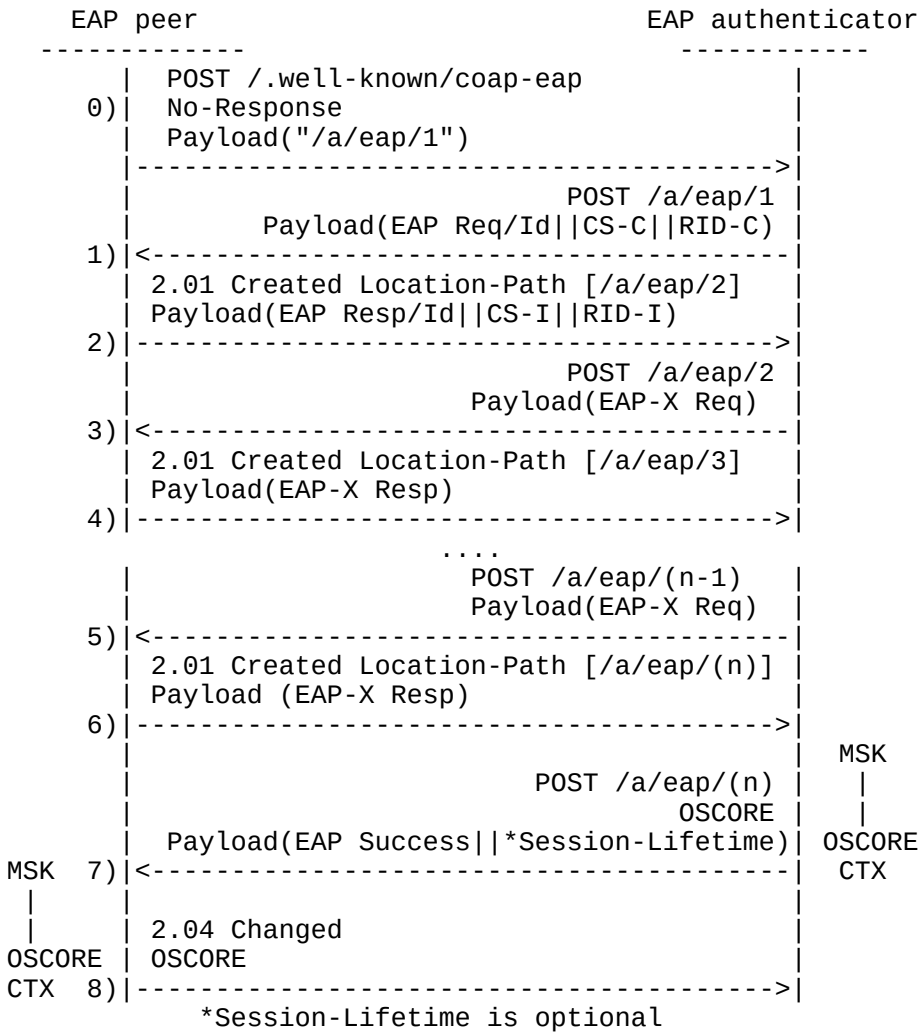

Figure 3: CoAP-EAP flow of operation with OSCORE

3.3. Reauthentication

When the CoAP-EAP state is close to expiring, the EAP peer may want to start a new authentication process (re-authentication) to renew the state. The main goal is to obtain new and fresh keying material (MSK/EMSK) that, in turn, allows deriving a new OSCORE security context, increasing the protection against key leakage. The keying material MUST be renewed before the expiration of the Session-Lifetime. By default, the EAP Key Management Framework establishes a default value of 8 hours to refresh the keying material. Certain EAP

methods such as EAP-NOOB [RFC9140] or EAP-AKA' [RFC5448] provide fast reconnect for quicker re-authentication. The EAP re-authentication protocol (ERP) [RFC6696] MAY also be used to avoid the repetition of the entire EAP exchange.

The re-authentication message flow will be the same as the one shown in Figure 3. Nevertheless, two different CoAP-EAP states will be active during the re-authentication: the current CoAP-EAP state and the new CoAP-EAP state, which will be created once the re-authentication has finished successfully. Once the re-authentication is completed successfully, the current CoAP-EAP state is deleted and replaced by the new CoAP-EAP state. If, for any reason, the re-authentication fails, the current CoAP-EAP state will be available until it expires, or it is renewed in another try of re-authentication.

If the re-authentication fails, it is up to the EAP peer to decide when to start a new re-authentication before the current EAP state expires.

3.4. Managing the State of the Service

The EAP peer and the EAP authenticator keep state during the CoAP-EAP negotiation. The CoAP-EAP state includes several important parts:

* A reference to an instance of the EAP (peer or authenticator/server) state machine.

* The resource for the next message in the negotiation (e.g., '/a/eap/2')

* The MSK is exported when the EAP authentication is successful. CoAP-EAP can access the different variables by the EAP state machine (i.e., [RFC4137]).

* A reference to the OSCORE context.

Once created, the EAP authenticator MAY choose to delete the state as described in Figure 4. Conversely, the EAP peer may need to renew the CoAP-EAP state because the key material is close to expiring, as mentioned in Section 3.3.

There are situations where the current CoAP-EAP state might need to be removed. For instance, due to its expiration or forced removal, the EAP peer has to be expelled from the security domain. This exchange is illustrated in Figure 4.

If the EAP authenticator deems it necessary to remove the CoAP-EAP state from the EAP peer before it expires, it can send a DELETE command in a request to the EAP peer, referencing the last CoAP-EAP state resource given by the CoAP server, whose identifier will be the last one received (e.g., '/a/eap/(n)' in Figure 3). This message is protected by the OSCORE security association to prevent forgery. Upon reception of this message, the CoAP server sends a response to the EAP authenticator with the Code '2.02 Deleted', which is also protected by the OSCORE security association. If a response from the EAP peer does not arrive after EXCHANGE_LIFETIME the EAP authenticator will remove the state.

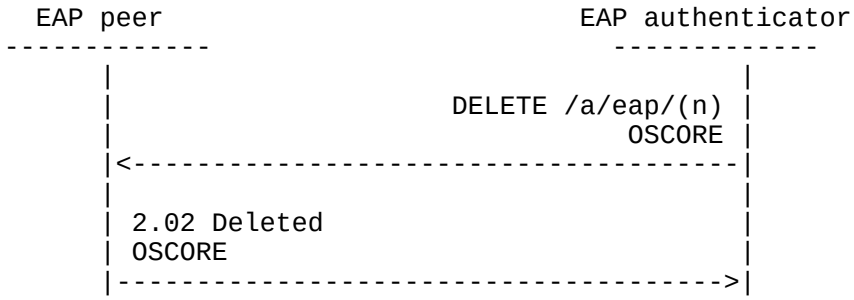

Figure 4: Deleting state

3.5. Error handling

This section elaborates on how different errors are handled. From EAP authentication failure, a non-responsive endpoint lost messages, or an initial POST message arriving out of place.

3.5.1. EAP authentication failure

The EAP authentication may fail in different situations (e.g., wrong credentials). The result is that the EAP authenticator will send an EAP Failure message because of the EAP authentication (Step 7 in Figure 3). In this case, the EAP peer MUST send a response '4.01 Unauthorized' in Step 8. Therefore, Step 7 and Step 8 are not protected in this case because no Master Session Key (MSK) is exported and the OSCORE security context is not yet generated.

If the EAP authentication fails during the re-authentication and the EAP authenticator sends an EAP failure, the current CoAP-EAP state will be still usable until it expires.

3.5.2. Non-responsive endpoint

If, for any reason, one of the entities becomes non-responsive, the CoAP-EAP state SHOULD be removed after a stipulated amount of time. The amount of time can be adjusted according to the policies established by the application or use case where CoAP-EAP is used. As a default value, the CoAP EXCHANGE_LIFETIME parameter, as defined in CoAP[RFC7252] will be used.

The removal of the CoAP-EAP state in the EAP authenticator assumes that the EAP peer will need to authenticate again.

According to CoAP, EXCHANGE_LIFETIME considers the time it takes until a client stops expecting a response to a request. A timer is reset every time a message is sent. By default, CoAP-EAP adopts the value of EXCHANGE_LIFETIME as a timer in the EAP peer and Authenticator to remove the CoAP-EAP state if the authentication process has not progressed in that time, in consequence, it has not been completed.

The EAP peer will remove the CoAP-EAP state either if the EXCHANGE_LIFETIME is triggered, or the EAP peer state machine returns an eapFail.

The EAP authenticator will remove the CoAP-EAP state either if the EXCHANGE_LIFETIME is triggered, or, when the EAP authenticator is acting in pass-through mode (i.e., the EAP authentication is performed against a AAA server), the EAP authenticator state machine returns an aaaTimemout.

3.5.3. Duplicated message with /.well-known/coap-eap

The reception of the trigger message in Step 0 containing the URI /coap-eap needs some additional considerations, as the resource is always available in the EAP authenticator.

If a trigger message (Step 0) arrives at the EAP authenticator during an ongoing authentication with the same EAP peer, the EAP authenticator MUST silently discard this trigger message.

If an old "POST /.well-known/coap-eap" (Step 0) arrives at the EAP authenticator and there is no authentication ongoing, the EAP authenticator may understand that a new authentication process is requested. Consequently, the EAP authenticator will start a new EAP authentication. However, if the EAP peer did not start any authentication and therefore, it did not select any resource for the EAP authentication. Thus, the EAP peer sends a ’4.04 Not found’ in the response (Figure 5).

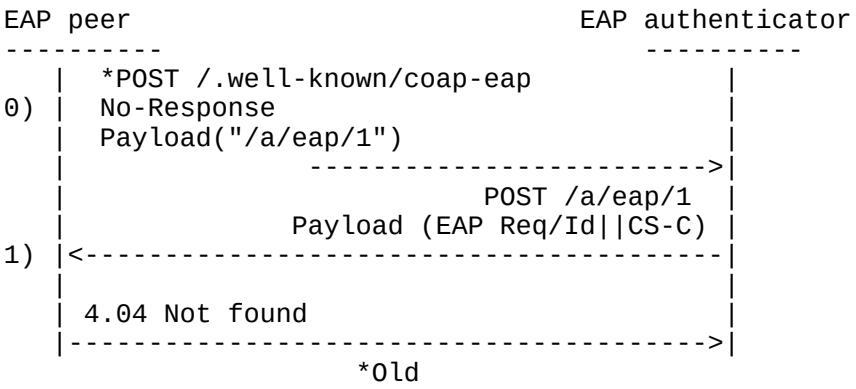

Figure 5: /.well-known/coap-eap with no ongoing authentication from the EAP authenticator

3.6. Proxy operation in CoAP-EAP

The CoAP-EAP operation is intended to be compatible with the use of intermediary entities between the EAP peer and the EAP authenticator when direct communication is not possible. In this context, CoAP proxies can be used as enablers of the CoAP-EAP exchange.

This specification is limited to using standard CoAP [RFC7252] as well as standardized CoAP options [RFC8613]. It does not specify any addition in the form of CoAP options. This is expected to ease the integration of CoAP intermediaries in the CoAP-EAP exchange.

When using proxies in the CoAP-EAP, it should be considered that the exchange contains a role-reversal process at the beginning of the exchange. In the first message, the EAP peer acts as a CoAP client and the EAP authenticator as the CoAP server. After that, in the remaining exchanges the roles are reversed, being the EAP peer, the CoAP server, and the EAP authenticator, the CoAP client. When using a proxy in the exchange, for message 0, the proxy will act as forward, and as reverse for the rest. Additionally, in the first exchange, the EAP peer, as a CoAP client, sends the URI for the next CoAP message in the payload of a request. This is not the typical behavior, as URIs referring to new services/resources appear in Location-Path and/or Location-Query Options in CoAP responses. Hence, the proxy will have to treat the payload of message 0, as if it were a Location-Path Option of a CoAP response.

4. CoAP-EAP Media type format

In the CoAP-EAP exchange, the following format will be used. This is the format is specified by application/coap-eap media type, see Section 9.5.

In CoAP-EAP there are two different elements that can be sent over the payload. The first one is a relative URI. This payload will be present for the first message (0) in Figure 3.

In all the other cases, an EAP message will be followed by the CBOR Object specified in Section 5. A visual example of the second case can be found in Figure 7.

5. CBOR Objects in CoAP-EAP

In the CoAP-EAP exchange, there is information that needs to be exchanged between the two entities. Examples of this information are the cipher suites that need to be negotiated or authorization information (Session-lifetime). There may also be a need to extend the information that has to be exchanged in the future. This section specifies the CBOR [RFC8949] data structure to exchange information between the EAP peer and the EAP authenticator in the CoAP payload.

Figure 6 shows the specification of the CBOR Object to exchange information in CoAP-EAP

```
CoAP-EAP_Info = {
   ?  1 : [+ int],     ; Cipher Suite (CS-C or CS-I)
   ?  2 : bstr,        ; RID-C
   ?  3 : bstr,        ; RID-I
   ?  4 : uint         ; Session-Lifetime
}
```

Figure 6: CBOR data structure for CoAP-EAP

The parameters contain the following information:

1. Cipher Suite: Is an array with the list of proposed, or selected, COSE algorithms for OSCORE. If the field is carried over a request, the meaning is the proposed cipher suite, if it is carried over a response, corresponds to the agreed-upon cipher suite.

2. RID-I: Is the Recipient ID of the EAP peer. The EAP authenticator uses this value as a Sender ID for its OSCORE Sender Context. The EAP peer uses this value as Recipient ID for its Recipient Context.

3. RID-C: Is the Recipient ID of the EAP authenticator. The EAP peer uses this value as a Sender ID for its OSCORE Sender Context. The EAP authenticator uses this value as Recipient ID for its Recipient Context.

4. Session-Lifetime: Is time the session is valid, in seconds.

Other indexes can be used to carry additional values as needed, like specific authorization parameters.

The indexes from 65001 to 65535 are reserved for experimentation.

6. Cipher suite negotiation and key derivation

6.1. Cipher suite negotiation

OSCORE runs after the EAP authentication, using the cipher suite selected in the cipher suite negotiation (Steps 1 and 2). To negotiate the cipher suite, CoAP-EAP follows a simple approach: the EAP authenticator sends a list, in decreasing order of preference, with the identifiers of the supported cipher suites (Step 1). In the response to that message (Step 2), the EAP peer sends a response with the choice.

This list is included in the payload after the EAP message through a CBOR array. An example of how the fields are arranged in the CoAP payload can be seen in Figure 7. An example of the exchange with the cipher suite negotiation is shown in Figure 8, where it can be appreciated the disposition of both EAP-Request/Identity and EAP-Response/Identity, followed by the CBOR object defined in Section 5, containing the cipher suite field for the cipher suite negotiation.

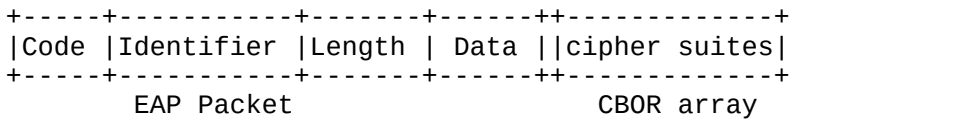

Figure 7: cipher suites are in the CoAP payload

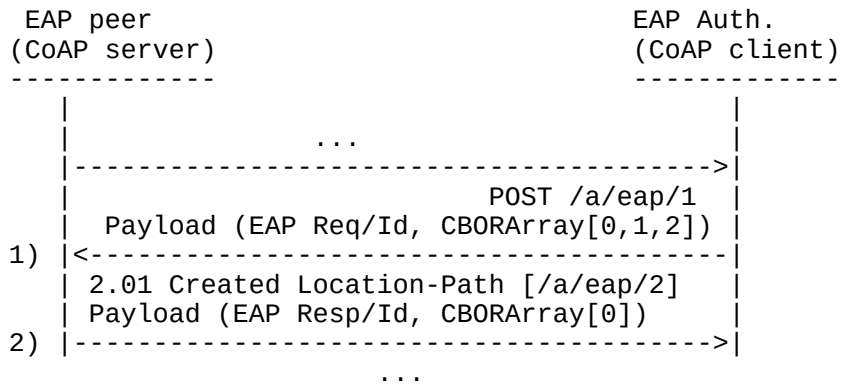

Figure 8: cipher suite negotiation

In case there is no CBOR array stating the cipher suites, the default cipher suites are applied. If the EAP authenticator sends a restricted list of cipher suites that are willing to accept, it MUST include the default value 0 since it is mandatory to implement. The EAP peer will have at least that option available.

The cipher suite requirements are inherited from the ones established by OSCORE [RFC8613], which are COSE algorithms [RFC9053]. By default, the HMAC-based Extract-and-Expand Key Derivation Function (HKDF) algorithm is SHA-256 and the AEAD algorithm is AES-CCM-16-64-128 [RFC9053]. Both are mandatory to implement. The other supported and negotiated cipher suites are the following:

* 0) AES-CCM-16-64-128, SHA-256 (default)

* 1) A128GCM, SHA-256

* 2) A256GCM, SHA-384

* 3) ChaCha20/Poly1305, SHA-256

* 4) ChaCha20/Poly1305, SHAKE256

This specification uses the HKDF defined in [RFC5869] to derive the necessary key material. Since the key derivation process uses the Master Session Key (MSK), which is considered fresh key material, the HKDF-Expand function will be used (shortened here as KDF). Why the use of this function is enough, and it is not necessary to use KDF-Extract is explained in Section 8.1.

6.2. Deriving the OSCORE Security Context

The derivation of the security context for OSCORE allows securing the communication between the EAP peer and the EAP authenticator once the MSK has been exported, providing confidentiality, integrity, key confirmation (Steps 7 and 8), and downgrading attack detection.

Once Master Secret and Master Salt are derived, they can be used to derive the rest of the OSCORE Security Context (see Section 3.2.1 of [RFC8613]). It should be noted that ID Context is not provided for the OSCORE Security Context derivation.

The Master Secret can be derived by using the chosen cipher suite and the KDF as follows:

```
Master Secret = KDF(MSK, CS | "COAP-EAP OSCORE MASTER SECRET", length)
```

where:

* The MSK exported by the EAP method. Discussion about the use of the MSK for key derivation is done in Section 8.

* CS is the concatenation of the content of the cipher suite negotiation, that is, the concatenation of two CBOR arrays CS-C and CS-I (with CBOR ints as elements), as defined in Section 5. If CS-C or CS-I were not sent, (i.e., default algorithms are used) the value used to generate CS will be the same as if the default algorithms were explicitly sent in CS-C or CS-I (i.e., a CBOR array with the cipher suite 0).

* "COAP-EAP OSCORE MASTER SECRET" is the ASCII code representation of the non-NULL terminated string (excluding the double quotes around it).

* CS and "COAP-EAP OSCORE MASTER SECRET" are concatenated.

* length is the size of the output key material.

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

```
Master Salt = KDF(MSK, CS | "COAP-EAP OSCORE MASTER SALT", length)
```

where:

* The MSK is exported by the EAP method. Discussion about the use of the MSK for the key derivation is done in Section 8.

* CS is the concatenation of the content of the cipher suite negotiation, that is, the concatenation of two CBOR arrays CS-C and CS-I (with CBOR ints as elements), as defined in Section 5. If CS-C or CS-I were not sent, (i.e., default algorithms are used) the value used to generate CS will be the same as if the default algorithms were explicitly sent in CS-C or CS-I (i.e., a CBOR array with the cipher suite 0).

* "COAP-EAP OSCORE MASTER SALT" is the ASCII code representation of the non-NULL-terminated string (excluding the double quotes around it).

* CS and "COAP-EAP OSCORE MASTER SALT" are concatenated.

* length is the size of the output key material.

Since the MSK is used to derive the Master Key, the correct verification of the OSCORE protected request (Step 7) and response (Step 8) confirms the EAP authenticator and the EAP peer have the same Master Secret, achieving key confirmation.

To prevent a downgrading attack, the content of the cipher suite negotiation (which is referred to here as CS) is embedded in the Master Secret derivation. If an attacker changes the value of the cipher suite negotiation, the result will be different OSCORE security contexts, which ends up with a failure in Steps 7 and 8.

The EAP authenticator will use the Recipient ID of the EAP peer (RID-I) as the Sender ID for its OSCORE Sender Context. The EAP peer will use this value as Recipient ID for its Recipient Context.

The EAP peer will use the Recipient ID of the EAP authenticator (RID-C) as the Sender ID for its OSCORE Sender Context. The EAP authenticator will use this value as Recipient ID for its Recipient Context.

7. Discussion

7.1. CoAP as EAP lower layer

This section discusses the suitability of the CoAP protocol as EAP lower layer and reviews the requisites imposed by EAP on any protocol transporting EAP. What EAP expects from its lower layers can be found in Section 3.1 of [RFC3748], which is elaborated next:

Unreliable transport. EAP does not assume that lower layers are reliable, but it can benefit from a reliable lower layer. In this sense, CoAP provides a reliability mechanism (e.g., using Confirmable messages).

Lower layer error detection. EAP relies on lower layer error detection (e.g., CRC, checksum, MIC, etc.). For simplicity, CoAP-EAP delegates error detection to the lower layers, such as the link layer or transport layer (e.g., UDP over IPv6 or TCP).

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. CoAP assumes an upper bound of 1024 octets for its payload, which covers the EAP requirements when in the CoAP payload only the EAP message is sent. In the case of Messages 1 and 2 in Figure 3, those messages have extra information apart from EAP. Nevertheless, the EAP Req/Id has a fixed length of 5 bytes. Message 2 with the EAP Resp/Id, would limit the length of the identity being used to the CoAP payload maximum size (1024) - len(CS-I || RID-I) - EAP Response header (5 bytes), which leaves enough space for sending even lengthy identities. Nevertheless, this limitation can be overcome by using CoAP block-wise transfer[RFC7959]. Note: When EAP is proxied over an AAA framework, the Access-Request packets in RADIUS MUST contain a Framed-MTU attribute with the value 1024, and the Framed-MTU AVP to 1024 in DIAMETER This attribute signals the AAA server that it should limit its responses to 1024 octets.

Ordering guarantees. EAP relies on lower layer ordering guarantees for correct operation. Regarding message ordering, every time a new message arrives at the authentication service hosted by the EAP peer, 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 options. This way, the application shows that its state has advanced.

Although the [RFC3748] states: "EAP provides its own support for duplicate elimination and retransmission", EAP is also reliant on lower layer ordering guarantees. In this regard, [RFC3748] talks about possible duplication and says: "Where the lower layer is reliable, it will provide the EAP layer with a non-duplicated stream of packets. However, while it is desirable that lower layers provide for non-duplication, this is not a requirement". CoAP provides a non-duplicated stream of packets and accomplishes the desirable non-duplication. In addition, [RFC3748] says that when EAP runs over a reliable lower layer "the authenticator retransmission timer SHOULD be set to an infinite value, so that retransmissions do not occur at the EAP layer."

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

Regarding the impact that an EAP lower layer will have on the number of bytes of the whole authentication exchange, there is a comparison with another network layer-based EAP lower layer, PANA [RFC5191], in [coap-eap].

The EAP method being transported will take most of the exchange, however, the impact of the EAP lower layer cannot be ignored, especially in very constrained communication technologies, such as the ones found in IoT, with limited capabilities.

Note: For constrained devices and network scenarios, the use of the latest versions of EAP methods (e.g., EAP-AKA' [RFC5448], EAP-TLS 1.3 [RFC9190]) is recommended in favor of older versions with the goal of economization, or EAP methods more adapted for IoT (e.g., EAP-NOOB [RFC9140], EAP-EDHOC [I-D.ietf-emu-eap-edhoc], etc.).

8. Security Considerations

There are some security aspects to be considered, such as how authorization is managed, the use of Master Session Key (MSK) as key material, and how trust in the EAP authenticator is established. Additional considerations such as EAP channel binding as per [RFC6677] are also discussed here.

8.1. Use of EAP Methods

This document limits the use of EAP methods to the ones compliant with [RFC4017] specification, yielding strong and fresh session keys such as the MSK. By this assumption, the HKDF-Expand function is used directly, as clarified in [RFC5869].

8.2. Authorization

Authorization is part of bootstrapping. It serves to establish whether the EAP peer can join and the set of conditions it must adhere to. The authorization data will be gathered from the organization that is responsible for the EAP peer and sent to the EAP authenticator in case AAA infrastructure is deployed.

In standalone mode, the authorization information will be in the EAP authenticator. If the pass-through mode is used, authorization data received from the AAA server can be delivered by the AAA protocol (e.g., RADIUS or Diameter). Providing more fine-grained authorization data can be with the transport of SAML in RADIUS [RFC7833]. After bootstrapping, additional authorization information may be needed to operate in the security domain. This can be taken care of by the solutions proposed in the ACE WG, such as the use of OAuth [RFC9200], among other solutions, to provide Authentication and Authorization for Constrained Environments.

8.3. Allowing CoAP-EAP traffic to perform authentication

Since CoAP is an application protocol, CoAP-EAP assumes certain IP connectivity in the link between the EAP peer and the EAP authenticator to run. This link MUST authorize exclusively unprotected IP traffic to be sent between the EAP peer and the EAP authenticator during the authentication with CoAP-EAP. Once the authentication is successful, the key material generated by the EAP authentication (MSK) and any other traffic can be authorized if it is protected. It is worth noting that if the EAP authenticator is not in the same link as the EAP peer and an intermediate entity helps with this process (i.e., CoAP proxy) and the same comment applies to the communication between the EAP peer and the intermediary.

Alternatively, the link-layer MAY provide support to transport CoAP-EAP without an IP address by using link-layer frames (e.g. by defining a new Key Management Protocol ID in IEEE 802.15.9 [ieee802159]).

8.4. Freshness of the key material

In CoAP-EAP there is no nonce exchange to provide freshness to the keys derived from the MSK. The MSK and Extended Master Session Key (EMSK) keys according to the EAP Key Management Framework [RFC5247] are fresh key material. Since only one authentication is established per EAP authenticator, there is no need to generate additional key material. In case a new MSK is required, a re-authentication can be done, by running the process again or using a more lightweight EAP method to derive additional key material as elaborated in

Section 3.3.

8.5. Channel Binding support

According to the [RFC6677], channel binding, related to EAP, is sent through the EAP method supporting it.

To satisfy the requirements of the document, the EAP lower layer identifier (To be assigned by IANA) needs to be sent, in the EAP Lower-Layer Attribute if RADIUS is used.

8.6. Additional Security Considerations

In the authentication process, there is a possibility of an entity forging messages to generate denial of service (DoS) attacks on any of the entities involved. For instance, an attacker can forge multiple initial messages to start an authentication (Step 0) with the EAP authenticator as if they were sent by different EAP peers. Consequently, the EAP authenticator will start an authentication process for each message received in Step 0, sending the EAP Request/ Id (Step 1).

To minimize the effects of this DoS attack, it is RECOMMENDED that the EAP authenticator limits the rate at which it processes incoming messages in Step 0 to provide robustness against denial of service (DoS) attacks. The details of rate limiting are outside the scope of this specification. Nevertheless, the rate of these messages is also limited by the bandwidth available between the EAP peer and the EAP authenticator. This bandwidth will be especially limited in constrained links (e.g., LPWAN). Lastly, it is also RECOMMENDED to reduce at a minimum the state in the EAP authenticator at least until the EAP Response/Id is received by the EAP authenticator.

Another security-related concern is how to ensure that the EAP peer joining the security domain can trust the EAP authenticator. This issue is elaborated in the EAP Key Management Framework [RFC5247]. In particular, the EAP peer knows it can trust the EAP authenticator because the key that is used to establish the security association is derived from the MSK. If the EAP authenticator has the MSK, it is because the AAA Server of the EAP peer trusted the EAP authenticator.

9. IANA Considerations

This section provides guidance to the Internet Assigned Numbers Authority (IANA) regarding the registration of values related to CoAP-EAP.

9.1. CoAP-EAP Cipher Suites

IANA has created a new registry titled "CoAP-EAP Cipher Suites" under the new group name "CoAP-EAP protocol". The registration procedures are "Specification Required", "Private Use", "Standards Action with Expert Review" and "Specification Required" following the indications in Table 1.

Range	Registration Procedures
-65536 to -25	Specification Required
-24 to -21	Private Use
-20 to 23	Standards Action with Expert Review
24 to 65535	Specification Required

Table 1: CoAP-EAP Cipher Suites Registration Procedures

The columns of the registry are Value, Algorithms, Description and Reference, where Value is an integer, and the other columns are text strings. The initial contents of the registry are shown in Table 2.

Value	Algorithms	Description	Reference
0	10, -16	AES-CCM-16-64-128, SHA-256	[[this document]]
1	1, -16	A128GCM, SHA-256	[[this document]]
2	3, -43	A256GCM, SHA-384	[[this document]]
3	24, -16	ChaCha20/Poly1305, SHA-256	[[this document]]
4	24, -45	ChaCha20/Poly1305, SHAKE256	[[this document]]

Table 2: CoAP-EAP Cipher Suites initial values

9.2. CDDL in CoAP-EAP Information elements

IANA has created a new registry titled "CoAP-EAP Information element" under the new group name "CoAP-EAP protocol". The registration procedure are "Specification Required", "Private Use", "Standards Action with Expert Review" and "Specification Required" following the indications in Table 3.

Range	Registration Procedures
0 to 23	Standards Action with Expert Review
24 to 49	Private Use
50 to 65000	Specification Required
65001 to 65535	Experimental Use

Table 3: CoAP-EAP Information Elements Registration Procedures

The columns of the registry are Name, Label, CBOR Type, Description and Reference, where Value is an integer, and the other columns are text strings. The initial contents of the registry are described in Table 4.

Name	Label	CBOR Type	Description	Reference
Cipher Suite	1	CBOR Array	List of the proposed or selected COSE algorithms for OSCORE	[[this document]]
RID-C	2	Byte String	It contains the Recipient ID of the EAP authenticator	[[this document]]
RID-I	3	Byte String	It contains the Recipient ID of the EAP peer	[[this document]]
Session-Lifetime	4	uint	Contains the time the session is valid in seconds	[[this document]]

Table 4: CoAP-EAP Information Elements initial content

9.3. The Well-Known URI Registry

IANA has added the well-known URI "coap-eap" to the "Well-Known URIs" registry under the group name "Well-Known URIs" defined by [RFC8615].

* URI suffix: coap-eap
* Change controller: IETF
* Specification document(s): [[this document]]
* Related information: None
* Status: permanent

9.4. The EAP lower layer identifier registry

IANA has added the identifier of CoAP-EAP to the registry "EAP Lower Layer" under the Extensible Authentication Protocol (EAP) Registry defined by [RFC6677].

* Value: TBD.
* Lower Layer: CoAP-EAP
* Specification document(s): [[this document]]

9.5. Media Types Registry

IANA has added the media types "application/coap-eap" to the "Media Types" registry. Section 4 defines the format.

* Type name: application
* Subtype name: coap-eap
* Required parameters: N/A
* Optional parameters: N/A
* Encoding considerations: binary
* Security considerations: See Section 8 of [[this document]].
* Interoperability considerations: N/A
* Published specification: [[this document]]
* Applications that use this media type: To be identified
* Fragment identifier considerations: N/A
* Additional information:
  - Magic number(s): N/A
  - File extension(s): N/A
  - Macintosh file type code(s): N/A
* Person and email address to contact for further information: ace@ietf.org

* Intended usage: COMMON

* Restrictions on usage: N/A

* Author: See "Authors' Addresses" section of [[this document]].

* Change Controller: IETF

9.6. CoAP Content-Formats Registry

IANA has added the media types "application/coap-eap" to the "CoAP Content-Formats" registry under the group name "Constrained RESTful Environments (CoRE) Parameters" following the specification in Section 12.3 of [RFC7252].

Media Type	Content Encoding	ID	Reference
application/coap-eap	-	TBD	[[this document]]

Table 5: CoAP Content-Formats Registry

9.7. Resource Type (rt=) Link Target Attribute Values Registry

IANA has added the resource type "core.coap-eap" to the "Resource Type (rt=) Link Target Attribute Values" registry under the group name "Constrained RESTful Environments (CoRE) Parameters".

* Value: "core.coap-eap"
  - Description: CoAP-EAP resource.
  - Reference: [[this document]]

9.8. Expert Review Instructions

The IANA registries established in this document are defined as "Specification Required", "Private Use", "Standards Action with Expert Review", "Experimental Use" and "Specification Required". This section provides general guidelines for what experts should focus on, but as they are designated experts for a reason, they should be granted flexibility.

* When defining the use of CoAP-EAP Information Elements: Experts are expected to evaluate how the values are defined, their scope, and whether they align with CoAP-EAP's functionality and

constraints. They are expected to assess if the values are clear, well-structured, and follow CoAP and CoAP-EAP conventions, such as concise encoding for constrained environments. They should ensure these IEs can seamlessly integrate with existing CoAP implementations and extensions. It is also expected that they verify if IE values are protected from unauthorized modification or misuse during transmission.

* When adding new cipher suites: Experts must ensure that algorithm values are sourced from the appropriate registry when required. They should also consider seeking input from relevant IETF working groups regarding the accuracy of registered parameters.

10. References

10.1. Normative References

[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>.

[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>.

[RFC6677] Hartman, S., Ed., Clancy, T., and K. Hoeper, "Channel-Binding Support for Extensible Authentication Protocol (EAP) Methods", RFC 6677, DOI 10.17487/RFC6677, July 2012, <https://www.rfc-editor.org/info/rfc6677>.

[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>.

[RFC7959] Bormann, C. and Z. Shelby, Ed., "Block-Wise Transfers in the Constrained Application Protocol (CoAP)", RFC 7959, DOI 10.17487/RFC7959, August 2016, <https://www.rfc-editor.org/info/rfc7959>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8323] Bormann, C., Lemay, S., Tschofenig, H., Hartke, K., Silverajan, B., and B. Raymor, Ed., "CoAP (Constrained Application Protocol) over TCP, TLS, and WebSockets", RFC 8323, DOI 10.17487/RFC8323, February 2018, <https://www.rfc-editor.org/info/rfc8323>.

[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>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/info/rfc8949>.

10.2. Informative References

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

[EAP-framework-IoT] Sethi, M., "Secure Network Access Authentication for IoT Devices: EAP Framework vs. Individual Protocols", 2021, <https://ieeexplore.ieee.org/document/9579387>.

[I-D.ietf-emu-eap-edhoc] Garcia-Carrillo, D., Marin-Lopez, R., Selander, G., and J. P. Mattsson, "Using the Extensible Authentication Protocol (EAP) with Ephemeral Diffie-Hellman over COSE (EDHOC)", Work in Progress, Internet-Draft, draft-ietf-emu-eap-edhoc-02, 21 October 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-emu-eap-edhoc-02>.

[ieee802159] IEEE, "IEEE Standard for Transport of Key Management Protocol (KMP) Datagrams", 2021.

[lo-coap-eap]
Garcia-Carrillo, D., Marin-Lopez, R., Kandasamy, A., and A. Pelov, "A CoAP-Based Network Access Authentication Service for Low-Power Wide Area Networks: LO-CoAP-EAP", 2017, <https://www.mdpi.com/1424-8220/17/11/2646>.

[RFC4017] Stanley, D., Walker, J., and B. Aboba, "Extensible Authentication Protocol (EAP) Method Requirements for Wireless LANs", RFC 4017, DOI 10.17487/RFC4017, March 2005, <https://www.rfc-editor.org/info/rfc4017>.

[RFC4137] Vollbrecht, J., Eronen, P., Petroni, N., and Y. Ohba, "State Machines for Extensible Authentication Protocol (EAP) Peer and Authenticator", RFC 4137, DOI 10.17487/RFC4137, August 2005, <https://www.rfc-editor.org/info/rfc4137>.

[RFC4186] Haverinen, H., Ed. and J. Salowey, Ed., "Extensible Authentication Protocol Method for Global System for Mobile Communications (GSM) Subscriber Identity Modules (EAP-SIM)", RFC 4186, DOI 10.17487/RFC4186, January 2006, <https://www.rfc-editor.org/info/rfc4186>.

[RFC4764] Bersani, F. and H. Tschofenig, "The EAP-PSK Protocol: A Pre-Shared Key Extensible Authentication Protocol (EAP) Method", RFC 4764, DOI 10.17487/RFC4764, January 2007, <https://www.rfc-editor.org/info/rfc4764>.

[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>.

[RFC5433] Clancy, T. and H. Tschofenig, "Extensible Authentication Protocol - Generalized Pre-Shared Key (EAP-GPSK) Method", RFC 5433, DOI 10.17487/RFC5433, February 2009, <https://www.rfc-editor.org/info/rfc5433>.

[RFC5448] Arkko, J., Lehtovirta, V., and P. Eronen, "Improved Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA')", RFC 5448, DOI 10.17487/RFC5448, May 2009, <https://www.rfc-editor.org/info/rfc5448>.

[RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010, <https://www.rfc-editor.org/info/rfc5869>.

[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>.

[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>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8615] Nottingham, M., "Well-Known Uniform Resource Identifiers (URIs)", RFC 8615, DOI 10.17487/RFC8615, May 2019, <https://www.rfc-editor.org/info/rfc8615>.

[RFC8824] Minaburo, A., Toutain, L., and R. Andreasen, "Static Context Header Compression (SCHC) for the Constrained Application Protocol (CoAP)", RFC 8824, DOI 10.17487/RFC8824, June 2021, <https://www.rfc-editor.org/info/rfc8824>.

[RFC9031] Vuini, M., Ed., Simon, J., Pister, K., and M. Richardson, "Constrained Join Protocol (CoJP) for 6TiSCH", RFC 9031, DOI 10.17487/RFC9031, May 2021, <https://www.rfc-editor.org/info/rfc9031>.

[RFC9053] Schaad, J., "CBOR Object Signing and Encryption (COSE): Initial Algorithms", RFC 9053, DOI 10.17487/RFC9053, August 2022, <https://www.rfc-editor.org/info/rfc9053>.

[RFC9140] Aura, T., Sethi, M., and A. Peltonen, "Nimble Out-of-Band Authentication for EAP (EAP-NOOB)", RFC 9140, DOI 10.17487/RFC9140, December 2021, <https://www.rfc-editor.org/info/rfc9140>.

[RFC9147] Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", RFC 9147, DOI 10.17487/RFC9147, April 2022, <https://www.rfc-editor.org/info/rfc9147>.

[RFC9176] Amsss, C., Ed., Shelby, Z., Koster, M., Bormann, C., and P. van der Stok, "Constrained RESTful Environments (CoRE) Resource Directory", RFC 9176, DOI 10.17487/RFC9176, April 2022, <https://www.rfc-editor.org/info/rfc9176>.

[RFC9190] Preu Mattsson, J. and M. Sethi, "EAP-TLS 1.3: Using the Extensible Authentication Protocol with TLS 1.3", RFC 9190, DOI 10.17487/RFC9190, February 2022, <https://www.rfc-editor.org/info/rfc9190>.

[RFC9200] Seitz, L., Selander, G., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "Authentication and Authorization for Constrained Environments Using the OAuth 2.0 Framework (ACE-OAuth)", RFC 9200, DOI 10.17487/RFC9200, August 2022, <https://www.rfc-editor.org/info/rfc9200>.

[THREAD] Thread Group, "Thread specification 1.3", 2023.

[TS133.501] ETSI, "5G; Security architecture and procedures for 5G System - TS 133 501 V15.2.0 (2018-10)", 2018.

[ZigbeeIP] Zigbee Alliance, "ZigBee IP Specification (Zigbee Document 095023r34)", 2014.

Appendix A. Flow of operation (DTLS establishment)

CoAP-EAP makes it possible to derive a PSK from the MSK to allow (D)TLS PSK-based authentication between the EAP peer and the EAP authenticator instead of using OSCORE. In the case of using (D)TLS to establish a security association, there is a limitation on the use of intermediaries between the EAP peer and the EAP authenticator, as (D)TLS breaks the end-to-end communications when using intermediaries such as proxies.

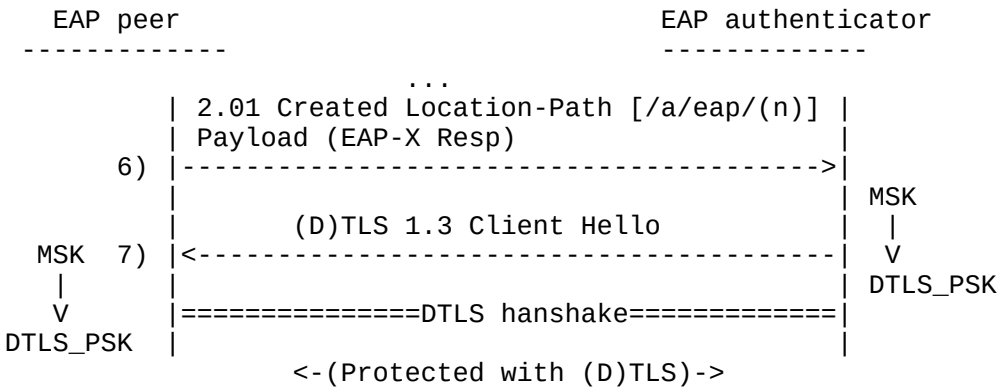

Figure 9: CoAP-EAP flow of operation with DTLS

Figure 9 shows the last steps of the operation for CoAP-EAP when (D)TLS is used to protect the communication between the EAP peer and the EAP authenticator using the keying material exported by the EAP authentication. The general flow is essentially the same as in the case of OSCORE, except that DTLS negotiation is established in Step 7). Once DTLS negotiation has finished successfully, the EAP peer is granted access to the domain. Step 7 MUST be interpreted by the EAP peer as an alternate success indication, which will end up with the MSK and the DTLS_PSK derivation for the (D)TLS authentication based on PSK.

According to [RFC8446] the provision of the PSK out-of-band also requires the provision of the KDF hash algorithm and the PSK identity. To simplify the design in CoAP-EAP, the KDF hash algorithm can be included in the list of cipher suites exchanged in Step 1 and Step 2 if DTLS wants to be used instead of OSCORE. For the same reason, the PSK identity is derived from (RID-C) (RID-I) as defined in Appendix A.1.

Analogous to how the cipher suite is negotiated for OSCORE Section 6.1, the EAP authenticator sends a list, in decreasing order of preference, with the identifiers of the hash algorithms supported (Step 1). In the response, the EAP peer sends the choice.

This list is included in the payload after the EAP message with a CBOR array that contains the cipher suites. This CBOR array is enclosed as one of the elements of the CBOR Object used for transporting information in CoAP-EAP (See Section 5). An example of how the fields are arranged in the CoAP payload can be seen in Figure 7.

In case there is no CBOR array stating the cipher suites, the default cipher suites are applied. If the EAP authenticator sends a restricted list of cipher suites that is willing to accept, it MUST include the default value 0 since it is mandatory to implement. The EAP peer will have at least that option available.

For DTLS, the negotiated cipher suite is used to determine the hash function to be used to derive the "DTLS PSK" from the MSK:

The hash algorithms considered are the following:

* 5) TLS_SHA256

* 6) TLS_SHA384

* 7) TLS_SHA512

The inclusion of these values, apart from indicating the hash algorithm, indicates if the EAP authenticator intends to establish an OSCORE security association or a DTLS security association after the authentication is completed. If both options appear in the cipher suite negotiation, the OSCORE security association will be preferred over DTLS.

A.1. Deriving DTLS PSK and identity

To enable DTLS after an EAP authentication, the Identity and the PSK for DTLS is defined. The Identity in this case is generated by concatenating the exchanged Sender ID and the Recipient ID.

CoAP-EAP PSK Identity = RID-C || RID-I

It is also possible to derive a pre-shared key for DTLS [RFC9147], referred to here as "DTLS PSK", from the MSK between both the EAP peer and EAP authenticator if required. The length of the DTLS PSK will depend on the cipher suite. To have keying material with sufficient length, a key of 32 bytes is derived that can be later truncated if needed:

DTLS PSK = KDF(MSK, "CoAP-EAP DTLS PSK", length).

where:

* MSK is exported by the EAP method.

* "CoAP-EAP DTLS PSK" is the ASCII code representation of the non-NULL terminated string (excluding the double quotes around it).

* length is the size of the output key material.

Appendix B. Using CoAP-EAP for distributing key material for IoT networks

Similarly, to the example of Appendix A.1, where a shared key PSK for DTLS is derived, it is possible to provide key material to different link-layers after the CoAP-EAP authentication is complete.

One example is that CoAP-EAP could be used to derive the required PSK required to run the 6TiSCH Constrained Join Protocol (CoJP) [RFC9031].

Another example is when a shared Network Key is required by the devices that join a network. An example of this Network Key can be found in ZigBee IP [ZigbeeIP] or THREAD protocol [THREAD]. After CoAP-EAP execution, a security association based on OSCORE protects any exchange between the EAP peer and the EAP authenticator. This security association can be used for distributing the Network Key securely and other required parameters. How the Network Key is distributed after a successful CoAP-EAP authentication is out of the scope of this document.

How a particular link-layer technology uses the MSK to derive further key material for protecting the link-layer or use the OSCORE protection to distribute key material is out of the scope of this document.

Appendix C. Examples of Use Case Scenario

In IoT, for an EAP peer to act as a trustworthy entity within a security domain, certain key material needs to be shared between the EAP peer and the EAP authenticator.

Next, examples of different use case scenarios will be elaborated on, about the usage of CoAP-EAP.

Generally, four entities are involved:

* 2 EAP peers (A and B), which are EAP peers. They are the EAP peers.

* 1 EAP authenticator (C). The EAP authenticator manages a domain where EAP peers can be deployed. In IoT, it can be considered a more powerful machine than the EAP peers.

* 1 AAA server (AAA) - Optional. The AAA is an Authentication, Authorization, and Accounting Server, which is not constrained. Here, the EAP authenticator acts as an EAP authenticator in pass-through mode.

Generally, any EAP peer wanting to join the domain managed by the EAP authenticator MUST perform a CoAP-EAP authentication with the EAP authenticator (C). This authentication MAY involve an external AAA server. This means that A and B, once deployed, will run CoAP-EAP once, as a bootstrapping phase, to establish a security association with C. Moreover, any other entity, which wants to join and establish communications with EAP peers under C's domain must also do the same.

By using EAP, the flexibility of having different types of credentials can be achieved. For instance, if a device that is not battery-dependent and not very constrained is available, a heavier authentication method could be used. With varied EAP peers and networks, more lightweight authentication methods might need to be used (e.g., EAP-NOOB[RFC9140], EAP-AKA'[RFC5448], EAP-PSK[RFC4764], EAP-EDHOC[I-D.ietf-emu-eap-edhoc], etc.) being able to adapt to different types of devices according to organization policies or devices capabilities.

C.1. Example 1: CoAP-EAP in ACE

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 using CoAP-EAP. Once the process of authentication with EAP is completed, the fresh key material is shared between the EAP peer and the EAP authenticator. In this instance, the EAP authenticator and the Authorization Server (AS) of ACE can be co-located.

Next, a general way to exemplify how Client registration can be performed using CoAP-EAP is presented, to allow two EAP peers (A and B) to communicate and interact after a successful client registration.

EAP peer A wants to communicate with EAP peer B (e.g., to activate a light switch). The overall process is divided into three phases. Let's start with EAP peer A. In the first phase, EAP peer A does not yet belong to EAP authenticator C's domain. Then, it communicates with C and authenticates with CoAP-EAP, which, optionally, communicates with the AAA server to complete the authentication process. If the authentication is successful, a fresh MSK is shared between C and EAP peer A. This key material allows EAP peer A to establish a security association with the C. Some authorization

information may also be provided in this step. In case EAP is used in standalone mode, the AS itself having information about the devices can be the entity providing said authorization information.

If authentication and authorization are correct, EAP peer A has been enrolled in the EAP authenticator C's domain for some time. In particular, [RFC5247] recommends 8 hours, though the entity providing the authorization information can establish this lifetime. In the same manner, B needs to perform the same process with CoAP-EAP to be part of EAP authenticator C's domain.

In the second phase, when EAP peer A wants to talk to EAP peer B, it contacts EAP authenticator C for authorization to access EAP peer 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 [RFC9200].

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

It is worth noting that the first phase with CoAP-EAP is required to join the EAP authenticator C's domain. Once it is performed successfully, the communications are local to the EAP authenticator C's domain and there is no need to perform a new EAP authentication as long as the key material is still valid. When the keys are about to expire, the EAP peer can engage in a re-authentication as explained in Section 3.3, to renew the key material.

C.2. Example 2: Multi-domain with AAA infrastructures

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 EAP authenticator's domain, which in turn communicates with an AAA infrastructure (acting as AAA client). Through the local AAA server communicate with the home AAA server to complete the authentication and integrate the device as a trustworthy entity into the domain of EAP authenticator C. In this scenario, the AS under the role of the EAP authenticator receives the key material from the AAA infrastructure

C.3. Example 3: Single domain with AAA infrastructure

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

C.4. Example 4: Single domain without AAA infrastructure

In another case, without a AAA infrastructure, with an EAP authenticator that has co-located the EAP server, and using EAP standalone mode, all the devices can be managed within the same domain locally. Client registration of an EAP peer (A) with Controller (C) can also be performed in the same manner.

C.5. Other use cases

C.5.1. CoAP-EAP for network access authentication

One of the first steps for an EAP peer is to perform the authentication to gain access to the network. To do so, the device first must 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 EAP peer in a particular network by joining the security domain. By using EAP, this can be achieved 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 EAP peers deployed under the same security domain belong, for example, to different organizations.

In the process of joining a network, there are two cases: 1) the node does not have an IPv6 address; 2) the node does have an IPv6 address (e.g., link-local IPv6 or IPv6 global address).

In networks where the device is placed, and no IP support is available until the EAP peer is authenticated, specific support for this EAP lower layer has to be defined to allow CoAP-EAP messages to be exchanged between the EAP peer and the EAP authenticator. For example, in IEEE 802.15.4 networks, a new KMP ID can be defined to add such support in the case of IEEE 802.15.9 [ieee802159]. Where can be assumed that the device has at least a link-layer IPv6 address.

When the EAP peer intends to be admitted into the network, it would search for an entity that offers the CoAP-EAP service, be it the EAP authenticator directly, or through the intermediary (i.e., proxy). See Section 3.1.

CoAP-EAP will run between the EAP peer and the EAP authenticator or through an intermediary entity such as a proxy, as happens in a mesh network, where the EAP authenticator could be placed to 1 or more hops from the EAP peer. In the case a proxy participates in CoAP-EAP, it will be because it is already a trustworthy entity within the domain, which communicates through a secure channel with the EAP authenticator, as illustrated by Figure 10.

Thus, the EAP peer follows the same process described in Appendix C.5.1 to perform the authentication. As mentioned, either with a direct link to the EAP authenticator, or through an intermediary entity (proxy) that is already part of the network (already shares key material and communicates through a secure channel with the authenticator) and can aid in running CoAP-EAP.

When CoAP-EAP is completed, and the OSCORE security association is established with the EAP authenticator, the EAP peer receives the local configuration parameters for the network (e.g. a network key) and can configure a global IPv6 address. Moreover, there is no need of a CoAP proxy after a successful authentication.

For removal, if the EAP authenticator decides to remove a particular EAP peer from the network or the peer itself intends to leave, either EAP peer or EAP authenticator can directly send a DELETE command to explicitly express that the network access state is removed, and the device will no longer belong to the network. Thus, any state related to the EAP peer is removed in the EAP authenticator. Forced removal can be done by sending new specific key material to the devices that still belong to the network, excluding the removed device, following a similar model as 6TiSCH Join Protocol [RFC9031] or Zigbee IP[ZigbeeIP]. The specifics on how this process is to be done, is out of the scope of this document.

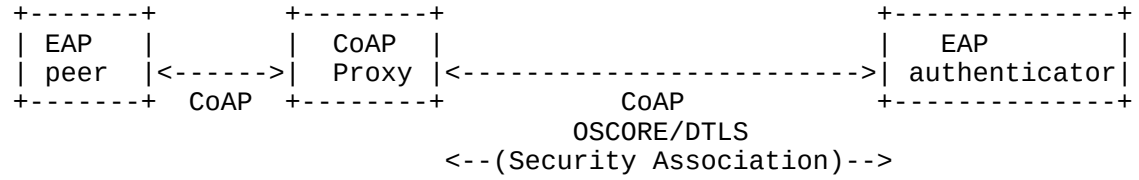

Figure 10: CoAP-EAP through CoAP proxy

Given that EAP is also used for network access authentication, this service can be adapted to other technologies. For instance, to provide network access control to very constrained technologies (e.g., LoRa network). Authors in [lo-coap-eap] provide a study of a minimal version of CoAP-EAP for LPWAN networks with interesting results. In this specific case, the compression by SCHC for CoAP [RFC8824] can be leveraged.

C.5.2. CoAP-EAP for service authentication

It is not uncommon that the infrastructure where the device is deployed and the services of the EAP peer are managed by different organizations. Therefore, in addition to the authentication for network access control, the possibility of a secondary authentication to access different services has to be considered. This process of authentication, for example, will provide the necessary key material to establish a secure channel and interact with the entity in charge of granting access to different services.

In 5G, for example, consider primary and secondary authentication using EAP [TS133.501].

Acknowledgments

We would like to thank the reviewers of this work: Paul Wouters, Heikki Vatiainen, Josh Howlett, Deb Cooley, Eliot Lear, Alan DeKok, Carsten Bormann, Mohit Sethi, Benjamin Kaduk, Christian Amsuss, John Mattsson, Goran Selander, 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, Julian Niklas Schimmelpfennig for the implementation of the Erbium-based IoT device implementation, and Daniel Menendez Gonzalez for the Python implementation.

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

This work was supported in part by Grant PID2020-112675RB-C44 funded by MCIN/AEI/10.13039/5011000011033 (ONOFRE-3-UMU) and in part by the H2020 EU project IoTCrawler under contract 779852.

Authors’ Addresses

Rafa Marin-Lopez
University of Murcia
Campus de Espinardo S/N, Faculty of Computer Science
30100 Murcia
Spain
Email: rafa@um.es

Dan Garcia-Carrillo
University of Oviedo
Calle Luis Ortiz Berrocal S/N, Edificio Polivalente
33203 Gijon Asturias
Spain
Email: garciadan@uniovi.es