EAP Method Update
Internet-Draft
Intended status: Standards Track
Expires: 19 October 2026

D. Garcia-Carrillo
University of Oviedo
R. Marin-Lopez
University of Murcia
G. Selander
J. Preu Mattsson
Ericsson
F. Lopez-Gomez
University of Murcia

17 April 2026

Using the Extensible Authentication Protocol (EAP) with Ephemeral Diffie-Hellman over COSE (EDHOC)
draft-ietf-emu-eap-edhoc-09

Abstract

The Extensible Authentication Protocol (EAP), defined in RFC 3748, provides a standard mechanism for support of multiple authentication methods. This document specifies the EAP authentication method EAP-EDHOC, based on Ephemeral Diffie-Hellman Over COSE (EDHOC). EDHOC is a lightweight security handshake protocol, enabling authentication and establishment of shared secret keys suitable in constrained settings. This document also provides guidance on authentication and authorization for EAP-EDHOC.

About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-emu-eap-edhoc/.

Discussion of this document takes place on the EAP Method Update mailing list (mailto:emu@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/emu. Subscribe at https://www.ietf.org/mailman/listinfo/emu/.

Source for this draft and an issue tracker can be found at https://github.com/dangarciacarrillo/i-d-eap-edhoc.

Status of This Memo

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

Copyright Notice

Table of Contents

1. Introduction

The Extensible Authentication Protocol (EAP), defined in [RFC3748], provides a standard mechanism for support of multiple authentication methods. This document specifies the EAP authentication method EAP-EDHOC, which is based on the lightweight security handshake protocol Ephemeral Diffie-Hellman Over COSE (EDHOC) [RFC9528].

EAP-EDHOC is similar to EAP-TLS 1.3 [RFC9190], since EDHOC is based on a similar security protocol design as the TLS 1.3 handshake [RFC8446]. However, EDHOC has been optimized for highly constrained settings, for example involving wirelessly connected battery powered 'things' with embedded microcontrollers, sensors, and actuators. An overview of EDHOC is given in Section 1.1.

The EAP-EDHOC method enables the integration of EDHOC into different applications and use cases using the EAP framework.

1.1. EDHOC Overview

Ephemeral Diffie-Hellman Over COSE (EDHOC) is a lightweight authenticated ephemeral key exchange, including mutual authentication and establishment of shared secret keying material, see [RFC9528].

EDHOC provides state-of-the-art security design at very low message overhead, targeting low complexity implementations and allowing extensibility. The security of EDHOC has been thoroughly analyzed, some references are provided in Section 9.1 of [RFC9528].

The main features of EDHOC are:

* Support for different authentication methods and credentials. The authentication methods include (mixed) signatures and static Diffie-Hellman keys [RFC9528], and pre-shared keys [I-D.ietf-lake-edhoc-psk]. A wide and extensible range of authentication credentials is supported, including public key certificates such as X.509 and C509 [I-D.ietf-cose-cbor-encoded-cert], as well as CBOR Web Tokens (CWTs) and CWT Claims Sets (CCSs) [RFC8392].

* A standardized and extensible format for identification of credentials, using COSE header parameters [RFC9052], supporting credential transport by value or by reference, enabling very compact representations.

* Crypto agility and secure cipher suite negotiation, with predefined compactly represented cipher suites and support for extensibility using the COSE algorithms registry [RFC9053].

* Selection of connection identifiers identifying a session for which keys are agreed.

* Support for integration of external security applications into EDHOC by transporting External Authorization Data (EAD) included in and protected as EDHOC messages.

A necessary condition for a successful completion of an EDHOC session is that both peers support a common application profile including method, cipher suite, etc. More details are provided in [I-D.ietf-lake-app-profiles].

EDHOC messages make use of lightweight primitives, specifically CBOR [RFC8949] and COSE [RFC9052] [RFC9053] for efficient encoding and security services in constrained devices. EDHOC is optimized for use with CoAP [RFC7252] and OSCORE [RFC8613] to secure resource access in constrained IoT use cases, but it is not bound to a particular transport or communication security protocol.

2. Conventions and Definitions

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 defined in EAP [RFC3748] and EDHOC [RFC9528], in particular the following.

Acronyms and Terms:

* MSK - Master Session Key; defined in [RFC3748].

* EMSK - Extended Master Session Key; defined in [RFC3748].

* Initiator - In EAP-EDHOC, the EAP peer assumes the role of the EDHOC Initiator; therefore, this term is used interchangeably with EAP peer; defined in [RFC9528].

* Responder - In EAP-EDHOC, the EAP server assumes the role of the EDHOC Responder; therefore, this term is used interchangeably with EAP server; defined in [RFC9528].

3. Protocol Overview

3.1. Overview of the EAP-EDHOC Conversation

The EAP exchange involves three key entities: the EAP peer, the EAP authenticator, and the EAP server. The EAP authenticator is a network device that enforces access control and initiates the EAP authentication process. The EAP peer is the device seeking network access and communicates directly with the EAP authenticator. The EAP server is responsible for selecting and implementing the authentication methods and for authenticating the EAP peer. When the EAP server is not located on a separate backend authentication server, it is integrated into the EAP authenticator. For simplicity, the operational flow diagrams in this document depict only the EAP peer and the EAP server.

The EDHOC protocol running between an Initiator and a Responder consists of three mandatory messages (message_1, message_2, message_3), an optional message_4, and an error message. In an EDHOC session, EAP-EDHOC uses all messages including message_4, which is mandatory and acts as a protected success indication.

After receiving an EAP-Request packet with EAP-Type=EAP-EDHOC as described in this document, the conversation will continue with the EDHOC messages transported in the data fields of EAP-Response and EAP-Request packets. When EAP-EDHOC is used, the formatting and processing of EDHOC messages SHALL be done as specified in [RFC9528]. This document only lists additional and different requirements, restrictions, and processing compared to [RFC9528].

The message processing in Section 5 of [RFC9528] states that certain data (EAD items, connection identifiers, application algorithms, etc.) is made available to the application. Since EAP-EDHOC is now acting as the application of EDHOC, it may need to handle this data to complete the protocol execution. See also [I-D.ietf-lake-edhoc-impl-cons].

Resumption of EAP-EDHOC may be defined using the EDHOC-PSK authentication method [I-D.ietf-lake-edhoc-psk].

3.1.1. Successful EAP-EDHOC Message Flow without Fragmentation

EDHOC allows EAP-EDHOC to support authentication credentials of any type defined by COSE, which can be either transported or referenced during the protocol.

The optimization combining the execution of EDHOC with the first subsequent OSCORE transaction specified in [RFC9668] is not applicable to this EAP method.

Figure 1 shows an example message flow for a successful execution of EAP-EDHOC.

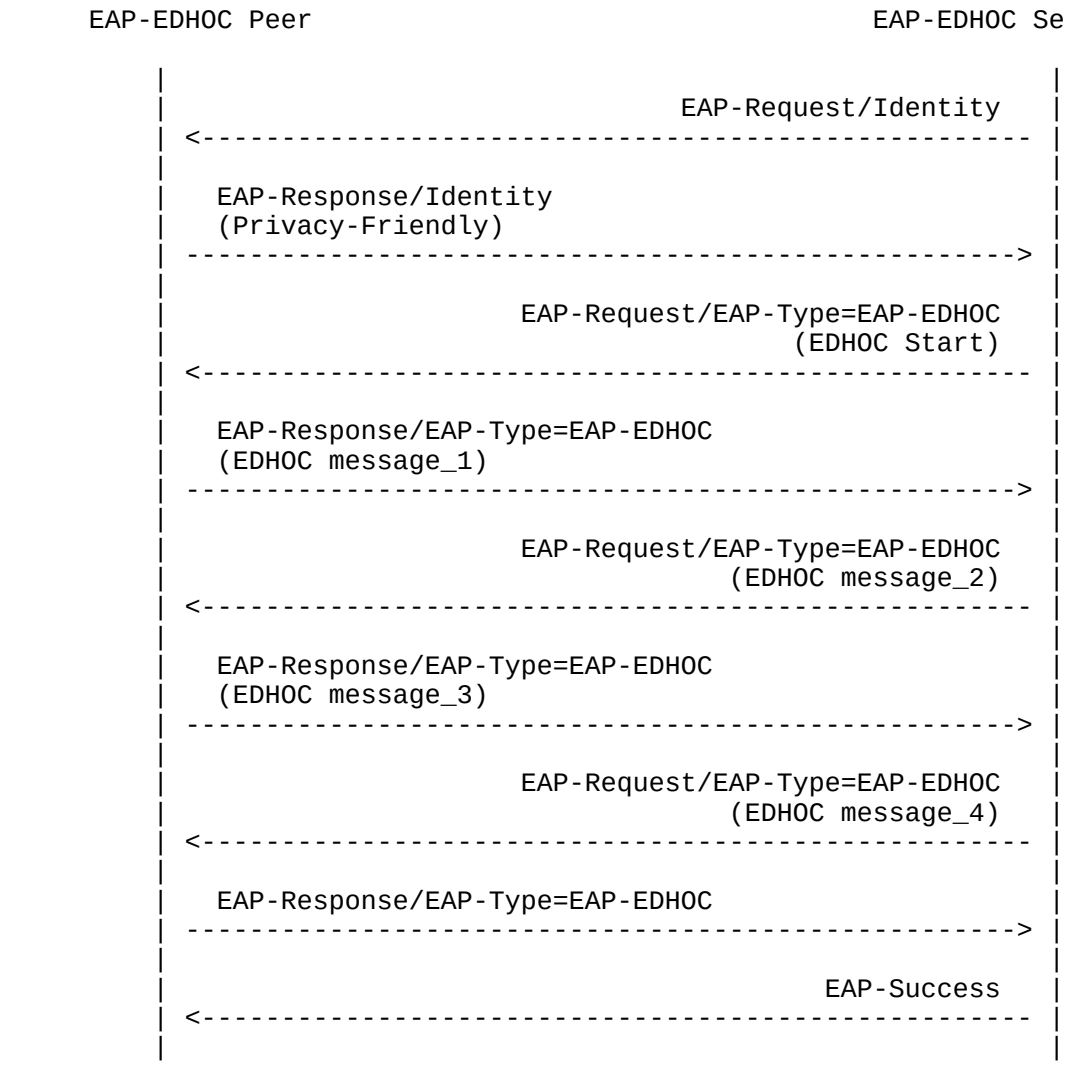


Figure 1: EAP-EDHOC Message Flow

If the EAP-EDHOC peer authenticates successfully, the EAP-EDHOC server MUST send an EAP-Request packet with EAP-Type=EAP-EDHOC containing message_4 as a protected success indication.

If the EAP-EDHOC server authenticates successfully, and the EAP-EDHOC peer achieves key confirmation by successfully verifying EDHOC message_4, then the EAP-EDHOC peer MUST send an EAP-Response message with EAP-Type=EAP-EDHOC containing no data. Finally, the EAP-EDHOC server sends an EAP-Success.

Note that the Identity request is optional [RFC3748] and might not be used in systems like 3GPP 5G [Sec5G] where the identity is transferred encrypted by other means before the EAP exchange. While the EAP-Response/EAP-Type=EAP-EDHOC and EAP-Success are mandatory [RFC3748] they do not contain any information and might be encoded into other system specific messages [Sec5G].

3.1.2. Transport and Message Correlation

EDHOC is not bound to a particular transport layer and can even be used in environments without IP. Nonetheless, [RFC9528] provides a set of requirements for a transport protocol to use with EDHOC. These include: handling the loss, reordering, duplication, correlation, and fragmentation of messages; demultiplexing EDHOC messages from other types of messages; and denial-of-service protection. These requirements can be met by the EAP framework, the EAP-EDHOC method specification, and properties of the EAP lower layer, as specified in [RFC3748].

For message loss, this can be either fulfilled by the EAP layer, the EAP lower layer, or both.

For message reordering, correct operation assumes that the EAP lower layer preserves packet ordering within an EAP conversation.

For duplication and message correlation, EAP has the Identifier field, which allows both the EAP peer and EAP authenticator to detect duplicates and match a request with a response.

Fragmentation is defined by this EAP method, see Section 3.1.6. The EAP framework [RFC3748], specifies that EAP methods need to provide fragmentation and reassembly if EAP packets can exceed the minimum MTU of 1020 octets.

To demultiplex EDHOC messages from other types of messages, EAP provides the Type field.

This method does not provide any additional mitigation against denial-of-service attacks beyond those specified in EAP [RFC3748].

3.1.3. Termination

Figure 2, Figure 3, Figure 4, and Figure 5 illustrate message flows in several cases where the EAP-EDHOC peer or EAP-EDHOC server sends an EDHOC error message.

Figure 2 shows an example message flow where the EAP-EDHOC server rejects message_1 with an EDHOC error message.

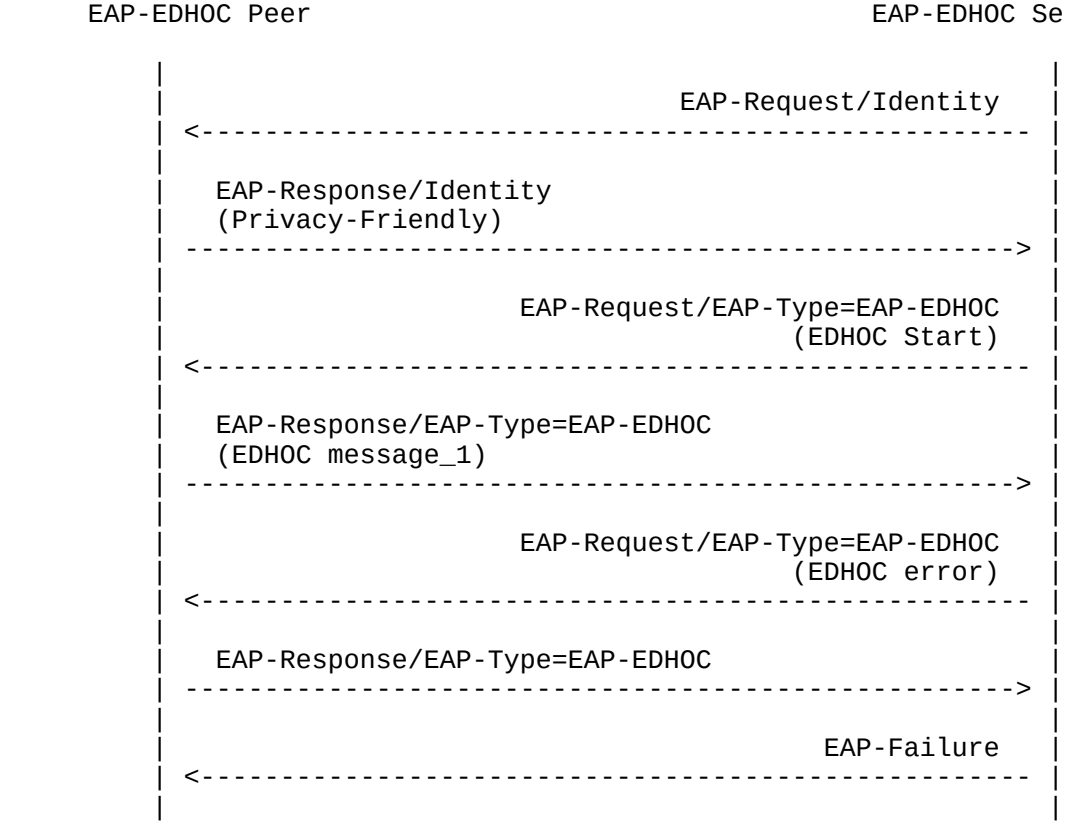


Figure 2: EAP-EDHOC Server Rejection of message_1

Figure 3 shows an example message flow where the EAP-EDHOC server authentication is unsuccessful and the EAP-EDHOC peer sends an EDHOC error message.

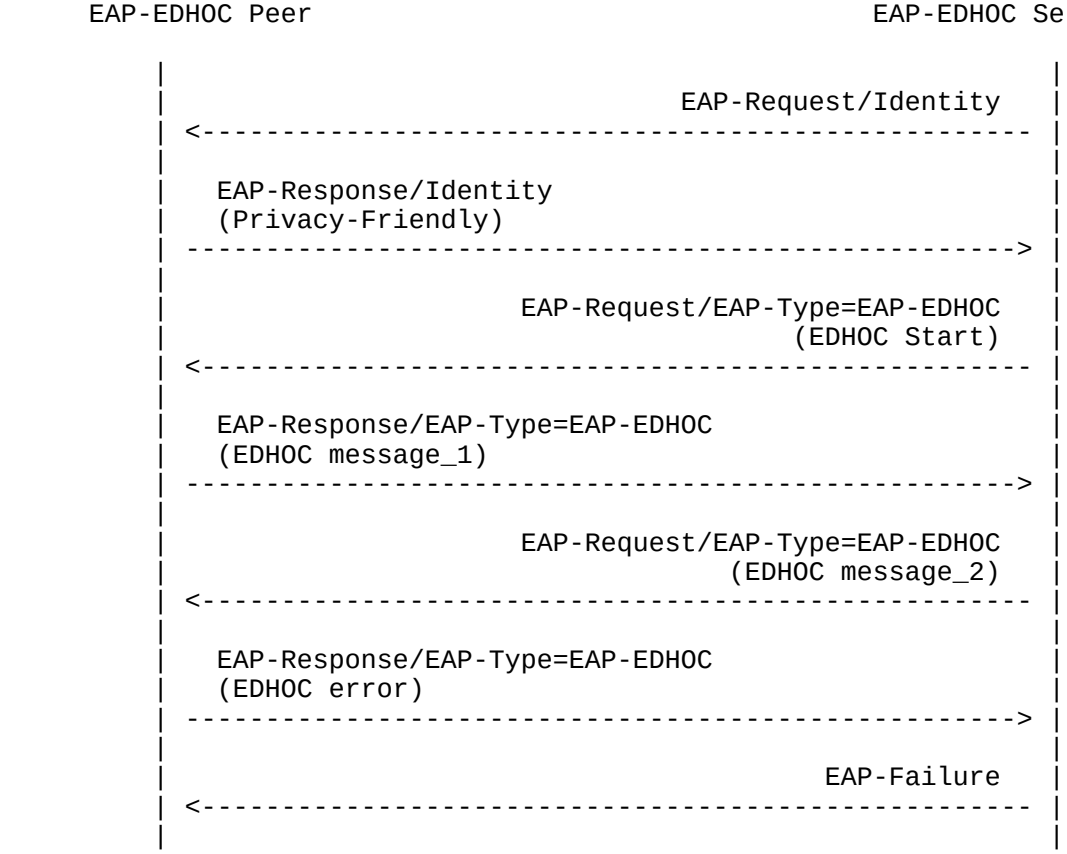


Figure 3: EAP-EDHOC Peer Rejection of message_2

Figure 4 shows an example message flow where the EAP-EDHOC server authenticates to the EAP-EDHOC peer successfully, but the EAP-EDHOC peer fails to authenticate to the EAP-EDHOC server, and the server sends an EDHOC error message.

Note that the EDHOC error message cannot be omitted. For example, with EDHOC ERR_CODE 3 "Unknown credential referenced", it is indicated that the EDHOC peer should, for the next EDHOC session, try another credential identifier supported according to the application profile.

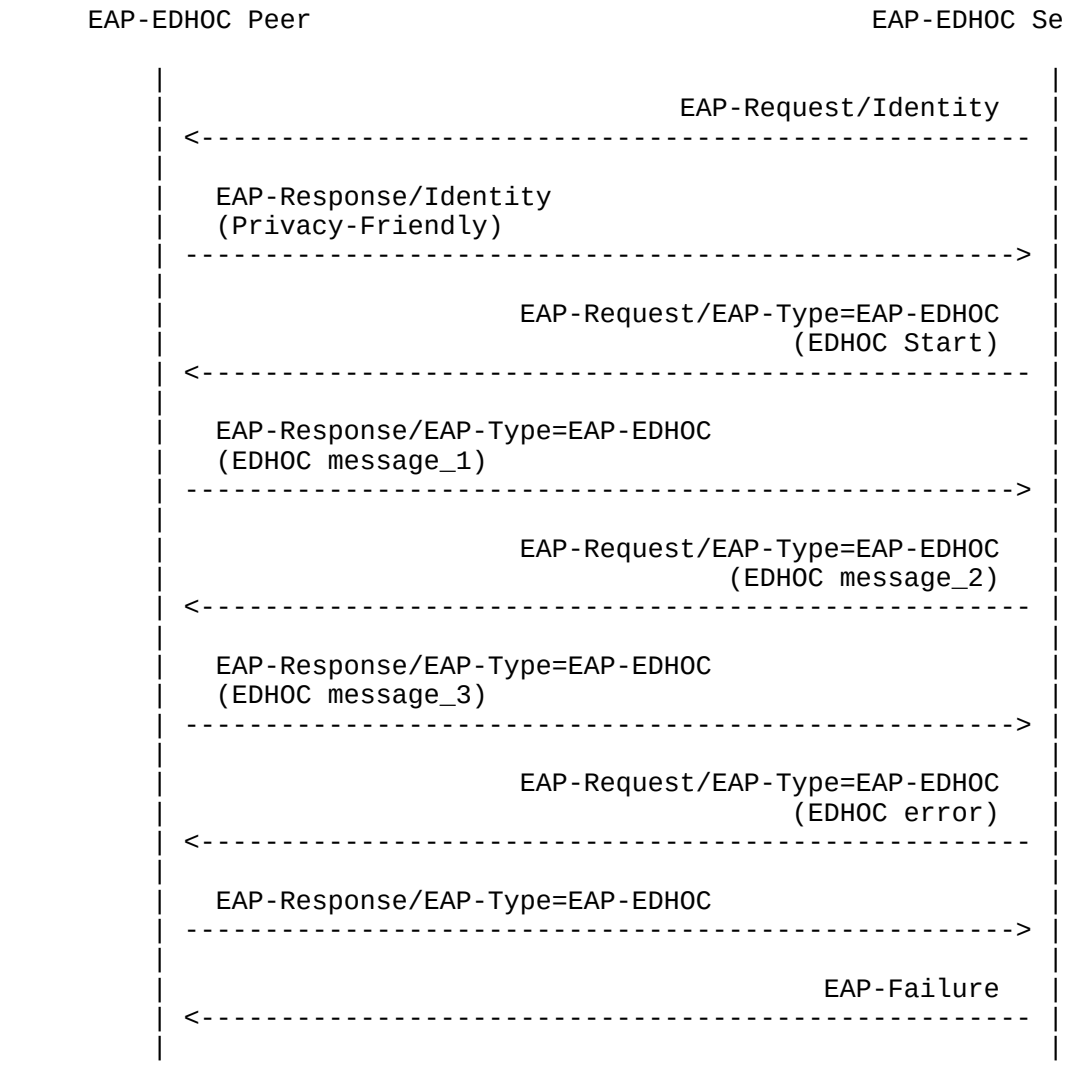


Figure 4: EAP-EDHOC Server Rejection of message_3

Figure 5 shows an example message flow where the EAP-EDHOC server sends the EDHOC message_4 to the EAP peer, but the protected success indication fails, and the peer sends an EDHOC error message.

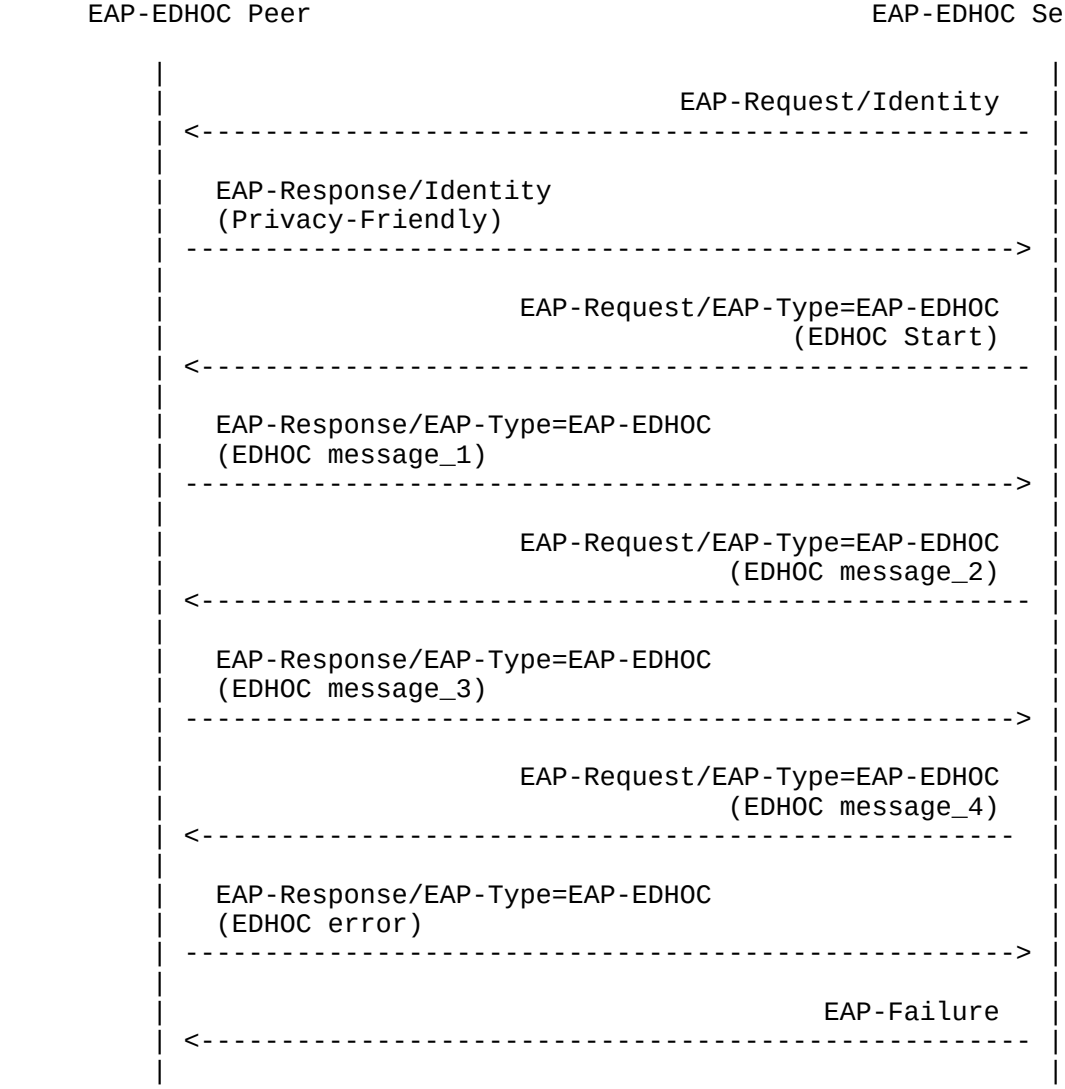


Figure 5: EAP-EDHOC Peer Rejection of message_4

3.1.4. Identity

It is RECOMMENDED to use anonymous Network Access Identifiers (NAIs) [RFC7542] in the Identity Response, as such identities are routable and privacy-friendly.

While opaque blobs are allowed by [RFC3748], such identities are NOT RECOMMENDED as they are not routable and should only be considered in local deployments where the EAP-EDHOC peer, EAP authenticator, and EAP-EDHOC server all belong to the same network.

Many client certificates contain an identity such as an email address, which is already in NAI format. When the certificate contains an NAI as subject name or alternative subject name, an anonymous NAI SHOULD be derived from the NAI in the certificate. See Section 3.1.5.

3.1.5. Privacy

EAP-EDHOC peer and server implementations supporting EAP-EDHOC MUST support anonymous NAIs (Section 2.4 of [RFC7542]). A node supporting EAP-EDHOC MUST NOT send its username (or any other permanent identifiers) in cleartext in the Identity Response (or any message used instead of the Identity Response). Following [RFC7542], it is RECOMMENDED to omit the username (i.e., the NAI is @realm), but other constructions such as a fixed username (e.g., anonymous@realm) or an encrypted username (e.g., xCZINCPTK5+7y81CrSYbPg+RKPE3OTrYLn4AQc4AC2U=@realm) are allowed. Note that the NAI MUST be a UTF-8 string as defined by the grammar in Section 2.2 of [RFC7542].

3.1.6. Fragmentation

EDHOC is designed to perform well in constrained networks where message sizes are restricted for performance reasons. When credentials are transferred by reference, EAP-EDHOC messages are typically so small that fragmentation is not needed. However, as EAP-EDHOC also supports large X.509 certificate chains, EAP-EDHOC implementations MUST provide support for fragmentation and reassembly. Some EAP implementations and access networks impose limits on the number of EAP packet exchanges that can be processed. To minimize fragmentation, it is RECOMMENDED to use compact EAP-EDHOC peer, EAP-EDHOC server, and trust anchor authentication credentials, as well as to limit the length of certificate chains. Additionally, mechanisms that reduce the size of Certificate messages are RECOMMENDED.

Since EAP is a lock-step protocol, fragmentation support can be easily added. To do so, the EAP-Response and EAP-Request packets of EAP-EDHOC have a set of information fields that allow for the specification of the fragmentation process (see Section 4 for the detailed description). As a summary, EAP-EDHOC fragmentation support is provided through the addition of flag bits (M and L) within the EAP-Response and EAP-Request packets, as well as a (conditional) EAP-EDHOC Message Length field that can be zero to four octets.

The EDHOC Message Length field conveys the total length of the EDHOC message being fragmented, which facilitates buffer allocation. The L flag consists of three bits that determine the length of the EDHOC Message Length field. This L flag and the EAP-EDHOC Message Length field MUST be present only in the first fragment of a fragmented EDHOC message, thereby signaling that the EDHOC message is fragmented. Implementations MUST NOT set any of the L flag bits to 1 in unfragmented messages.

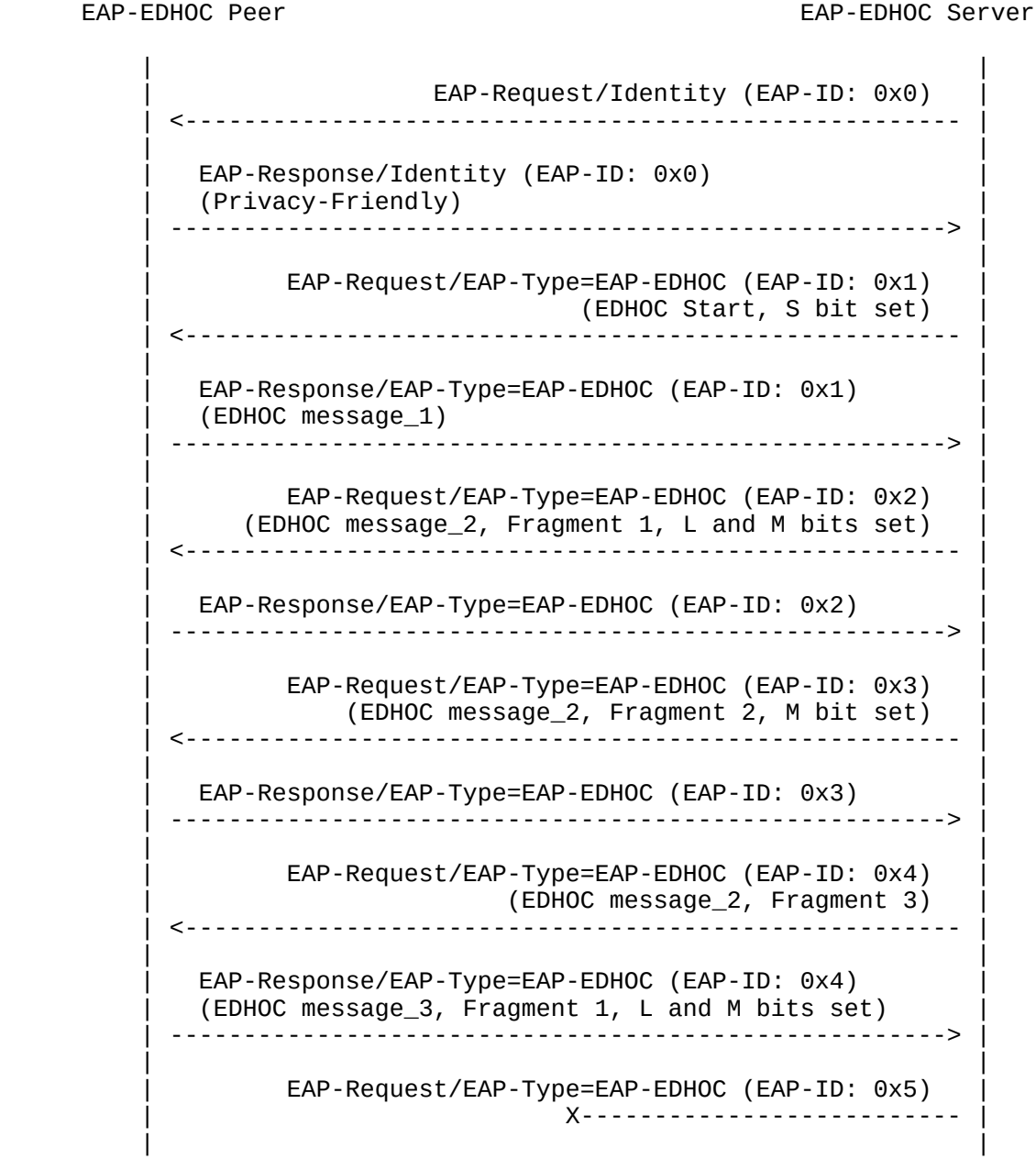
The S flag bit SHALL be set in the EAP-EDHOC Start message sent by the EAP server to the peer. The S flag bit SHALL NOT be set in any other EAP-EDHOC messages. The M flag bit SHALL be set in all fragments except the last one.

When an EAP-EDHOC peer receives an EAP-Request packet with the M bit set, it MUST respond with an EAP-Response packet with EAP-Type=EAP-EDHOC, including the flags octect with S=0, M=0, L=0 and an empty EDHOC payload. This message serves as a fragment ACK. The EAP server MUST wait until it receives the EAP-Response before sending another fragment. To prevent errors in the processing of fragments, the EAP server MUST increment the Identifier field for each fragment contained within an EAP-Request, and the peer MUST include this Identifier value in the fragment ACK contained within the EAP-Response. Retransmitted fragments will contain the same Identifier value.

Similarly, when the EAP-EDHOC server receives an EAP-Response with the M bit set, it MUST respond with an EAP-Request packet with EAP-Type=EAP-EDHOC, including the flags octect with S=0, M=0, L=0 and an empty EDHOC payload. This message serves as a fragment ACK. The EAP peer MUST wait until it receives the EAP-Request before sending another fragment. To prevent errors in the processing of fragments, the EAP server MUST increment the Identifier value for each fragment ACK contained within an EAP-Request, and the peer MUST include this Identifier value in the subsequent fragment contained within an EAP-Response.

Explanations and considerations regarding retransmission timers are provided in [RFC4137].

Figure 6 illustrates the exchange between the endpoints in the case where the EAP-EDHOC mutual authentication is successful and fragmentation is required. EAP Identifiers are also shown to illustrate their progression. A retransmission is also included.



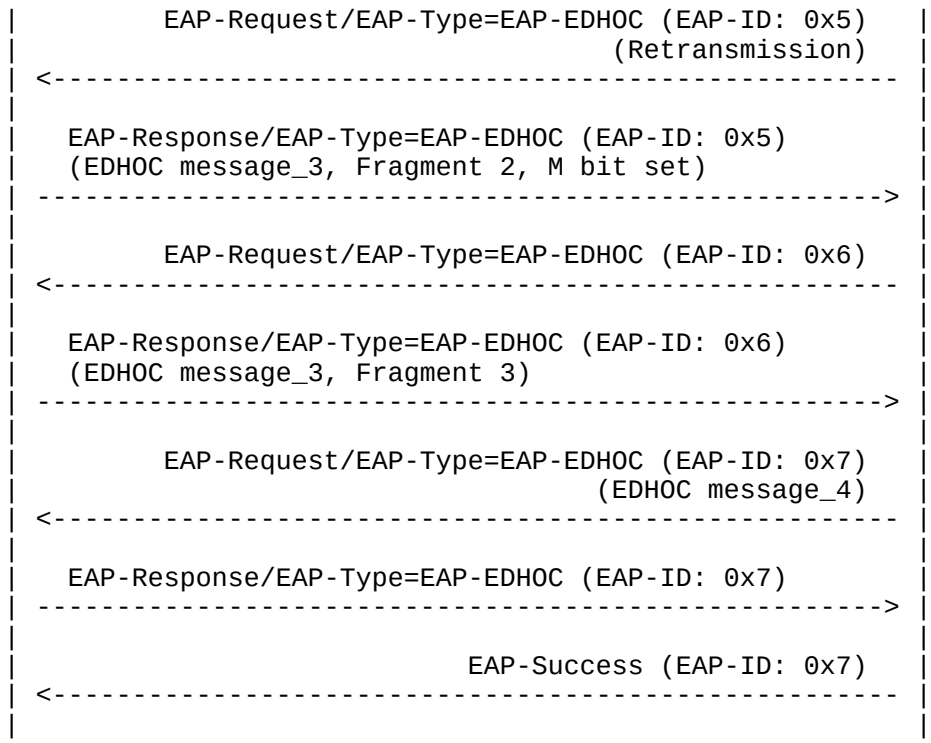


Figure 6: EAP-EDHOC Fragmentation Example

3.2. Identity Verification

The EAP peer identity provided in the EAP-Response/Identity is not authenticated by EAP-EDHOC. Unauthenticated information MUST NOT be used for accounting purposes or to give authorization. The EAP authenticator and the EAP server MAY examine the identity presented in EAP-Response/Identity for purposes such as routing and EAP method selection. EAP-EDHOC servers MAY reject conversations if the identity does not match their policy.

The EAP server identity in the EDHOC server certificate is typically a fully qualified domain name (FQDN) in the SubjectAltName (SAN) extension. Since EAP-EDHOC deployments may use more than one EAP server, each with a different certificate, EAP peer implementations SHOULD allow for the configuration of one or more trusted root certificates (CA certificate) to authenticate the server certificate and one or more server names to match against the SubjectAltName (SAN) extension in the server certificate. If any of the configured names match any of the names in the SAN extension, then the name check passes. To simplify name matching, an EAP-EDHOC deployment can

assign a designated name to represent an authorized EAP server. This name can then be included in the SANs list of each certificate used by this EAP-EDHOC server. If server name matching is not used, the EAP peer has reduced assurance that the EAP server it is interacting with is authoritative for the given network. If name matching is not used with a trusted root CA, then effectively any server can obtain a certificate that will be trusted for EAP authentication by the peer.

The process of configuring a root CA certificate and a server name is non-trivial; therefore, automated methods of provisioning are RECOMMENDED. For example, the eduroam federation [RFC7593] provides a Configuration Assistant Tool (CAT) to automate the configuration process. In the absence of a trusted root CA certificate (user-configured or system-wide), EAP peers MAY implement a Trust On First Use (TOFU) mechanism where the peer trusts and stores the server certificate during the first connection attempt. The EAP peer ensures that the server presents the same stored certificate on subsequent interactions. The use of a TOFU mechanism does not allow for the server certificate to change without out-of-band validation of the certificate and is therefore not suitable for many deployments including ones where multiple EAP servers are deployed for high availability. TOFU mechanisms increase the susceptibility to traffic interception attacks and should only be used if there are adequate controls in place to mitigate this risk.

As an alternative to standard certificate validation, EDHOC extensions such as the Lightweight Authorization mechanism defined in [I-D.ietf-lake-authz] can provide additional means of verifying server credentials. Such mechanisms may be suitable in deployments where no prior trust relationship exists or where managing trusted root CAs is impractical.

3.3. Key Hierarchy

The key derivation for EDHOC is described in Section 4 of [RFC9528]. The key material and Method-Id SHALL be derived from the PRK_exporter using the EDHOC_Exporter interface, see Section 4.2.1 of [RFC9528].

Type is the value of the EAP Type field defined in Section 2 of [RFC3748]. For EAP-EDHOC, Type has the value TBD1. The << >> notation defined in Section G.3 of [RFC8610] means that the CBOR-encoded integer Type value is embedded in a CBOR byte string. The use of Type as context enables the reuse of exporter labels across other future EAP methods based on EDHOC.

```
Type        =  TBD1
MSK         =  EDHOC_Exporter(TBD2, << Type >>, 64)
EMSK        =  EDHOC_Exporter(TBD3, << Type >>, 64)
Method-Id   =  EDHOC_Exporter(TBD4, << Type >>, 64)
Session-Id  =  Type || Method-Id
Peer-Id     =  ID_CRED_I
Server-Id   =  ID_CRED_R
```

EAP-EDHOC exports the MSK and the EMSK and does not specify how it is used by lower layers.

3.4. Parameter Negotiation and Compliance Requirements

The EAP-EDHOC peers and EAP-EDHOC servers MUST comply with the requirements defined in Section 8 of [RFC9528], including mandatory-to-implement cipher suites, signature algorithms, key exchange algorithms, and extensions.

3.5. EAP State Machines

It is worth remembering that the EAP state machine is defined in [RFC4137]. However, the following considerations apply to EAP-EDHOC.

The EAP-EDHOC server sends message_4 in an EAP-Request as a protected success result indication.

Because EDHOC error messages are unauthenticated, they MUST NOT be relied upon to determine the cause of failure; they only indicate that the exchange did not complete and are vulnerable to injection by an attacker (DoS). However, EDHOC error messages SHOULD be considered failure result indication, as defined in [RFC3748]. After sending or receiving an EDHOC error message, the EAP-EDHOC server may only send an EAP-Failure.

The keying material can be derived by the Initiator upon receiving EDHOC message_2, and by the Responder upon receiving EDHOC message_3. Implementations following [RFC4137] can then set the eapKeyData and aaaEapKeyData variables.

The keying material can be made available to lower layers and the EAP authenticator after the protected success indication (message_4) has been sent or received. Implementations following [RFC4137] can set the eapKeyAvailable and aaaEapKeyAvailable variables.

3.6. EAP Channel Binding

EAP-EDHOC allows the secure exchange of information between the endpoints of the authentication process (i.e., the EAP peer and the EAP server) using protected data fields. These fields can be used to exchange EAP channel binding information, as defined in [RFC6677].

Section 6 in [RFC6677] outlines requirements for components implementing channel binding information, all of which are satisfied by EAP-EDHOC, including confidentiality and integrity protection. Additionally, EAP-EDHOC supports fragmentation, allowing the inclusion of additional information at the method level without issues.

While the EAD_1 and EAD_2 fields (carried in EDHOC message_1 and EDHOC message_2, respectively) are integrity protected through the transcript hash, the channel binding protocol defined in [RFC6677] must be transported after keying material has been derived between the endpoints in the EAP communication and before the peer is exposed to potential adverse effects from joining an adversarial network. Therefore, compliance with [RFC6677] requires use of the EAD_3 and EAD_4 fields, transmitted in EDHOC message_3 and EDHOC message_4, respectively.

It is important to note that EAD fields in EDHOC are optional; consequently, the inclusion of EAP Channel Binding information in an authentication exchange is also optional.

Accordingly, this document specifies a new EAD item, with ead_label = TBD5, to incorporate EAP channel binding information into the EAD fields of the EAP-EDHOC messages. See the definition in Section 5.3. This new EAD item is intended only for EAD_3 and EAD_4. Then, it MUST be ignored if included in other EAD fields. Multiple occurrences of this new EAD item in one EAD field are NOT allowed.

> *Implementation Note:* This document defines only the container for carrying EAP Channel Binding information within EAP-EDHOC messages, using the EAD_3 and EAD_4 fields. The format and semantics of the channel binding content are application-specific and are determined by the authentication domain in which the protocol is deployed.

If the server detects a consistency error in the channel binding information contained in EAD_3, it MUST send an EDHOC error message, as specified in [RFC9528], since the new EAD item defined to carry EAP Channel Binding information is critical. In this case, the exchange proceeds according to Figure 4.

Similarly, if the Initiator detects an error in the channel binding information contained in EAD_4, it MUST send an EDHOC error message, and the exchange proceeds according to Figure 5.

4. Detailed Description of the EAP-EDHOC Request and Response Protocol

The EAP-EDHOC packet format for Requests and Responses is summarized in Figure 7. Fields are transmitted from left to right, following a structure inspired by the EAP-TLS packet format [RFC5216]. As specified in Section 4.1 of [RFC3748], EAP Request and Response packets consist of Code, Identifier, Length, Type, and Type-Data fields. The functions of the Code, Identifier, Length, and Type fields are reiterated here for convenience. The EAP Type-Data field consists of the R, S, M, L, EDHOC Message Length, and EDHOC Data fields.

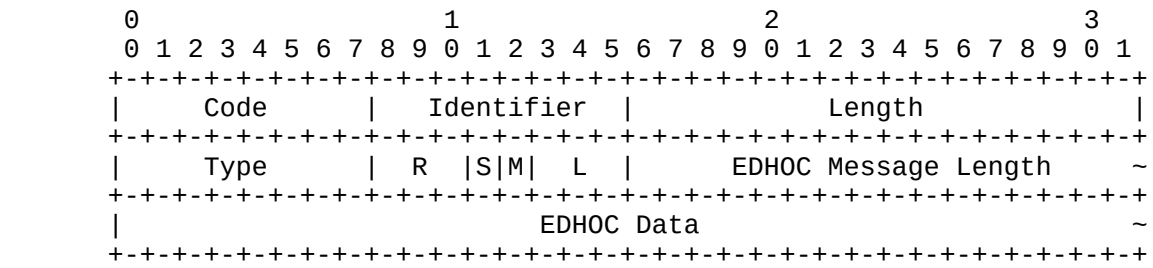


Figure 7: EAP-EDHOC Request and Response Packet Format

4.1. EAP-EDHOC Request Packet

Code: 1 (Request)

Identifier: The Identifier field is one octet and aids in matching responses with requests. The Identifier field MUST be changed on each new (non-retransmission) Request packet, and MUST be the same if a Request packet is retransmitted due to a timeout while waiting for a Response. In the case of fragmented messages, the Identifier will follow the indications of Section 3.1.6.

Length: The Length field is two octets and indicates the length of the EAP packet including the Code, Identifier, Length, Type, and Data fields. Octets outside the range of the Length field should be treated as Data Link Layer padding and MUST be ignored on reception.

Type: TBD1 (EAP-EDHOC)

R: Implementations of this specification MUST set the R bits

(reserved) to zero and MUST ignore them on reception.

S: The S bit (EAP-EDHOC start) is set in an EAP-EDHOC Start message. This differentiates the EAP-EDHOC Start message from a fragment acknowledgement.

M: The M bit (more fragments) is set on all but the last fragment. I.e., when there is no fragmentation, it is set to zero.

L: The L flag bits represent the binary encoding of the size of the EDHOC Message Length, which can range from 0 to 4 bytes. When all three bits are set to 0, the EDHOC Message Length field is not present. If the first two bits of the L field are set to 0 and the last bit is set to 1, then the size of the EDHOC Message Length field is 1 byte, and so on. Values from 5 to 7 are not used in this specification.

EDHOC Message Length: The EDHOC Message Length field, when present, has a size of one to four octets, as determined by the L flag bits. It is included only if the L flag bits indicate a value greater than zero and specifies the total length of the EDHOC message being fragmented. When fragmentation is not used, this field is omitted.

EDHOC Data: The EDHOC data consists of the whole or a fragment of the transported EDHOC message.

4.2. EAP-EDHOC Response Packet

Code: 2 (Response)

Identifier: The Identifier field is one octet and MUST match the Identifier field from the corresponding request.

Length: The Length field is two octets and indicates the length of the EAP packet including the Code, Identifier, Length, Type, and Data fields. Octets outside the range of the Length field should be treated as Data Link Layer padding and MUST be ignored on reception.

Type: TBD1 (EAP-EDHOC)

R: Implementations of this specification MUST set the R bits (reserved) to zero and MUST ignore them on reception.

S: The S bit (EAP-EDHOC start) is set to zero.

M: The M bit (more fragments) is set on all but the last fragment.

I.e., when there is no fragmentation, it is set to zero.

L: The L flag bits represent the binary encoding of the size of the EDHOC Message Length, which can range from 0 to 4 bytes. When all three bits are set to 0, the EDHOC Message Length field is not present. If the first two bits of the L field are set to 0 and the last bit is set to 1, then the size of the EDHOC Message Length field is 1 byte, and so on. Values from 5 to 7 are not used in this specification.

EDHOC Message Length: The EDHOC Message Length field, when present, has a size of one to four octets, as determined by the L flag bits. It is included only if the L flag bits indicate a value greater than zero and specifies the total length of the EDHOC message being fragmented. When fragmentation is not used, this field is omitted.

EDHOC Data: The EDHOC data consists of the whole or a fragment of the transported EDHOC message.

5. IANA Considerations

5.1. EAP Type

IANA has registered the following new type in the "Method Types" registry under the group name "Extensible Authentication Protocol (EAP) Registry":

Value: TBD1
Description: EAP-EDHOC
Reference: [this document]

NOTE: Suggested value: TBD1 = 57. RFC Editor: Remove this note.

5.2. EDHOC Exporter Label Registry

IANA has registered the following new labels in the "EDHOC Exporter Label" registry under the group name "Ephemeral Diffie-Hellman Over COSE (EDHOC)":

Label: TBD2
Description: MSK of EAP method EAP-EDHOC
Change Controller: IETF
Reference: [this document]

Label: TBD3
Description: EMSK of EAP method EAP-EDHOC
Change Controller: IETF
Reference: [this document]

Label: TBD4
Description: Method-Id of EAP method EAP-EDHOC
Change Controller: IETF
Reference: [this document]

NOTE: Suggested values: TBD2 = 26, TBD3 = 27, TBD4 = 28. RFC Editor: Remove this note.

The allocations have been updated to reference this document.

5.3. EDHOC External Authorization Data Registry

IANA has registered the following new label in the "EDHOC External Authorization Data" registry under the group name "Ephemeral Diffie-Hellman Over COSE (EDHOC)":

Name: EAPChannelBinding
Label: TBD5
Description: EAP channel binding information
Reference: [this document]

NOTE: Suggested value: TBD5 = 1. RFC Editor: Remove this note.

6. Security Considerations

The security considerations of EAP [RFC3748] [RFC5247] and EDHOC [RFC9528] apply to this document. Since the design of EAP-EDHOC closely follows EAP-TLS 1.3 [RFC9190], many of its security considerations are also relevant.

6.1. Security Claims

6.1.1. EAP Security Claims

EAP security claims are defined in Section 7.2.1 of [RFC3748]. EAP-EDHOC security claims are described next and summarized in Table 1.

Claim	
Auth. principle:	Certificates, CWTs, and all credential types for which COSE header parameters are defined (1)
Cipher suite negotiation:	Yes (2)
Mutual authentication:	Yes (3)
Integrity protection:	Yes (4)
Replay protection:	Yes (5)
Confidentiality:	Yes (6)
Key derivation:	Yes (7)
Key strength:	The specified cipher suites provide key strength of at least 128 bits.
Dictionary attack protection:	Yes (8)
Fast reconnect:	No
Crypt. binding:	N/A
Session independence:	Yes (9)
Fragmentation:	Yes (Section 3.1.6)
Channel binding:	Yes (Section 3.6: EAD_3 and EAD_4 can be used to convey integrity-protected channel properties, such as network SSID or peer MAC address.)

Table 1: EAP-EDHOC security claims

* (1) Authentication principle: EAP-EDHOC establishes a shared secret based on an authenticated ECDH key exchange. The key exchange is authenticated using different kinds of credentials. EAP-EDHOC supports EDHOC credential types. EDHOC supports all credential types for which COSE header parameters are defined. These include X.509 certificates [RFC9360], C509 certificates, CWTs ([RFC9528] Section 3.5.3.1), and CCSs ([RFC8392] Section 7.1).

* (2) Cipher suite negotiation: The Initiator’s list of supported cipher suites and order of preference is fixed, and the selected cipher suite is the cipher suite that is most preferred by the Initiator and that is supported by both the Initiator and the Responder. EDHOC supports all signature algorithms defined by COSE.

* (3) Mutual authentication: The initiator and responder authenticate each other through the EDHOC exchange.

* (4) Integrity protection: EDHOC integrity protects all message content using transcript hashes for key derivation and as additional authenticated data, including, e.g., method type, cipher suites, and external authorization data.

* (5) Replay protection. EDHOC broadens the message authentication coverage to include algorithms, external authorization data, and prior plaintext messages, as well as adding an explicit method type. By doing this, an attacker cannot replay or inject messages from a different EDHOC session.

* (6) Confidentiality. EDHOC protects the method payload transported within the EAP-Request and EAP-Response messages; however, the EAP-Success and EAP-Failure packets themselves are not encrypted or protected by EDHOC. EDHOC message_1 is not confidential, but it does not convey any private information. EDHOC message_2 provides confidentiality against passive attackers, while message_3 and message_4 provide confidentiality against active attackers.

* (7) Key derivation. Except for MSK and EMSK, derived keys are not exported. Key derivation is discussed in Section 3.3.

* (8) Dictionary attack protection. EAP-EDHOC provides Dictionary attack protection.

* (9) Session independence. EDHOC generates computationally independent keys derived from the ECDH shared secret.

6.1.2. Additional Security Claims

* (10) Cryptographic strength and Forward secrecy: Only ephemeral key exchange methods are supported by EDHOC, which ensures that the compromise of a session key does not also compromise earlier sessions' keys.

* (11) Identity protection: EDHOC secures the Responder's credential identifier against passive attacks and the Initiator's credential identifier against active attacks. An active attacker can get the credential identifier of the Responder by eavesdropping on the destination address used for transporting message_1 and then sending their own message_1.

6.2. Peer and Server Identities

The Peer-Id represents the identity to be used for access control and accounting purposes. The Server-Id represents the identity of the EAP server. The Peer-Id and Server-Id are determined from the information provided in the credentials used.

ID_CRED_I and ID_CRED_R are used to identify the credentials of the Initiator (EAP peer) and Responder (EAP server). Therefore, for Server-Id the ID_CRED_R is used, and for Peer-Id the ID_CRED_I is used.

6.3. Certificate Validation

Same considerations as in EAP-TLS 1.3 Section 5.3 [RFC9190] apply here in relation to the use of certificates.

When other types of credentials are used such as CWT/CCS, the application needs to have a clear trust-establishment mechanism and identify the pertinent trust anchors [RFC9528].

6.4. Certificate Revocation

Same considerations as in EAP-TLS 1.3 Section 5.4 [RFC9190] apply here in relation to certificates.

When other types of credentials are used such as CWT/CCS, the endpoints are in charge of handling revocation and confirming the validity and integrity of CWT/CCS [RFC9528].

6.5. Packet Modification Attacks

EAP-EDHOC relies on EDHOC, which is designed to encrypt and integrity protect as much information as possible. Any change in any message is detected by means of the transcript hashes integrity verification.

6.6. Authorization

Following the considerations of EDHOC in appendix D.5 Unauthenticated Operation [RFC9528], EDHOC can be used without authentication by allowing the Initiator or Responder to communicate with any identity except its own.

When peer authentication is not used, EAP-EDHOC server implementations MUST take care to limit network access appropriately for authenticated peers. Authorization and accounting MUST be based on authenticated information such as information in the certificate. The requirements for Network Access Identifiers (NAIs) specified in Section 4 of [RFC7542] apply and MUST be followed.

6.7. Privacy Considerations

Considerations in Section 9.6 of [RFC9528] against tracking of users and eavesdropping on Identity Responses or certificates apply here. Also, the considerations of Section 5.8 of [RFC9190] regarding anonymous NAIs also applies.

6.8. Pervasive Monitoring

Considerations in Section 9.1 of [RFC9528] about pervasive monitoring apply here.

6.9. Cross-Protocol Attacks

The cross-protocol attack of [RFC9190] does not apply here, as no resumption mechanism has been defined for EAP-EDHOC.

7. References

7.1. 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/rfc/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/rfc/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/rfc/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/rfc/rfc6677>.

[RFC7542] DeKok, A., "The Network Access Identifier", RFC 7542, DOI 10.17487/RFC7542, May 2015, <https://www.rfc-editor.org/rfc/rfc7542>.

[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/rfc/rfc8174>.

[RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <https://www.rfc-editor.org/rfc/rfc8610>.

[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/rfc/rfc9190>.

[RFC9528] Selander, G., Preu Mattsson, J., and F. Palombini, "Ephemeral Diffie-Hellman Over COSE (EDHOC)", RFC 9528, DOI 10.17487/RFC9528, March 2024, <https://www.rfc-editor.org/rfc/rfc9528>.

7.2. Informative References

[I-D.ietf-cose-cbor-encoded-cert] Mattsson, J. P., Selander, G., Raza, S., Hglund, J., and M. Furuhed, "CBOR Encoded X.509 Certificates (C509 Certificates)", Work in Progress, Internet-Draft, draft-ietf-cose-cbor-encoded-cert-17, 2 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-cose-cbor-encoded-cert-17>.

[I-D.ietf-lake-app-profiles]
Tiloca, M. and R. Hglund, "Coordinating the Use of Application Profiles for Ephemeral Diffie-Hellman Over COSE (EDHOC)", Work in Progress, Internet-Draft, draft-ietf-lake-app-profiles-04, 2 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-lake-app-profiles-04>.

[I-D.ietf-lake-authz]
Selander, G., Mattsson, J. P., Vuini, M., Fedrecheski, G., and M. Richardson, "Lightweight Authorization using Ephemeral Diffie-Hellman Over COSE (ELA)", Work in Progress, Internet-Draft, draft-ietf-lake-authz-07, 2 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-lake-authz-07>.

[I-D.ietf-lake-edhoc-impl-cons]
Tiloca, M., "Implementation Considerations for Ephemeral Diffie-Hellman Over COSE (EDHOC)", Work in Progress, Internet-Draft, draft-ietf-lake-edhoc-impl-cons-06, 2 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-lake-edhoc-impl-cons-06>.

[I-D.ietf-lake-edhoc-psk]
Lopez-Perez, Selander, G., Mattsson, J. P., Marin-Lopez, R., and F. Lopez-Gomez, "EDHOC Authenticated with Pre-Shared Keys (PSK)", Work in Progress, Internet-Draft, draft-ietf-lake-edhoc-psk-07, 2 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-lake-edhoc-psk-07>.

[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/rfc/rfc4137>.

[RFC5216] Simon, D., Aboba, B., and R. Hurst, "The EAP-TLS Authentication Protocol", RFC 5216, DOI 10.17487/RFC5216, March 2008, <https://www.rfc-editor.org/rfc/rfc5216>.

[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/rfc/rfc7252>.

[RFC7593] Wierenga, K., Winter, S., and T. Wolniewicz, "The eduroam Architecture for Network Roaming", RFC 7593, DOI 10.17487/RFC7593, September 2015, <https://www.rfc-editor.org/rfc/rfc7593>.

[RFC8392] Jones, M., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "CBOR Web Token (CWT)", RFC 8392, DOI 10.17487/RFC8392, May 2018, <https://www.rfc-editor.org/rfc/rfc8392>.

[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/rfc/rfc8446>.

[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/rfc/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/rfc/rfc8949>.

[RFC9052] Schaad, J., "CBOR Object Signing and Encryption (COSE): Structures and Process", STD 96, RFC 9052, DOI 10.17487/RFC9052, August 2022, <https://www.rfc-editor.org/rfc/rfc9052>.

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

[RFC9360] Schaad, J., "CBOR Object Signing and Encryption (COSE): Header Parameters for Carrying and Referencing X.509 Certificates", RFC 9360, DOI 10.17487/RFC9360, February 2023, <https://www.rfc-editor.org/rfc/rfc9360>.

[RFC9668] Palombini, F., Tiloca, M., Hglund, R., Hristozov, S., and G. Selander, "Using Ephemeral Diffie-Hellman Over COSE (EDHOC) with the Constrained Application Protocol (CoAP) and Object Security for Constrained RESTful Environments (OSCORE)", RFC 9668, DOI 10.17487/RFC9668, November 2024, <https://www.rfc-editor.org/rfc/rfc9668>.

[Sec5G] 3GPP TS 33 501, "Security architecture and procedures for 5G System", March 2025, <https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3169>.

Appendix A. Fragmented Message Example

This section provides an example of a fragmented message, including all header field values, to improve clarity and avoid ambiguity.

In this example, the total size of the EDHOC message is 128 octets. The MTU allows 32 EAP bytes to be transmitted per packet. No retransmissions are considered in this example. Furthermore, this does not correspond to the EAP-EDHOC Start message. The packets sent by the server are shown below:

```
+-----
Code = 1 | ID = 1 | Length = 32 |
Type = TBD1 | R = 0 | S = 0 | M = 1 | L = 0b001 | EDHOC Msg Length = 128 |
EDHOC Data Bytes 1-25
+-----

+-----
Code = 1 | ID = 2 | Length = 32 |
Type = TBD1 | R = 0 | S = 0 | M = 1 | L = 0b000 |
EDHOC Data Bytes 26-51
+-----

+-----
Code = 1 | ID = 3 | Length = 32 |
Type = TBD1 | R = 0 | S = 0 | M = 1 | L = 0b000 |
EDHOC Data Bytes 52-77
+-----

+-----
Code = 1 | ID = 4 | Length = 32 |
Type = TBD1 | R = 0 | S = 0 | M = 1 | L = 0b000 |
EDHOC Data Bytes 78-103
+-----

+-----
Code = 1 | ID = 5 | Length = 32 |
Type = TBD1 | R = 0 | S = 0 | M = 0 | L = 0b000 |
EDHOC Data Bytes 104-128
+-----
```

As shown, the L flags have a value of 0b001 in the first fragment, indicating that the EDHOC Msg Length field is 1 byte in size. The EDHOC Msg Length field indicates the total size of the EDHOC message being fragmented, which is 128 bytes. The L flags and the EDHOC Msg Length field are only present in the first fragment.

Another relevant detail is that the first fragment includes 7 bytes of header (1 byte Code, 1 byte Identifier, 2 bytes Length, 1 byte Type, 1 byte Flags, and 1 byte EDHOC Msg Length), as illustrated in Figure 7. Subsequent fragments include 6 bytes of header, since they do not carry the EDHOC Msg Length field. Consequently, with an MTU of 32 bytes, the first fragment carries 25 bytes of payload, while each subsequent fragment carries up to 26 bytes of payload.

Acknowledgments

The authors sincerely thank Eduardo Ingles-Sanchez for his contribution in the initial phase of this work. We also want to thank Marco Tiloca, Christian Amsss, Gabriel Lopez-Millan, Renzo Navas, Paul Wouters, Rich Salz, Ines Robles and Christopher Inacio for their reviews.

This work was supported partially by Project PID2023-148104OB-C43 (ONOFRE4-UMU) from MCIN/AEI; the Formacion del Profesorado Universitario predoctoral grant FPU24/03871, also from MCIN/AEI; and the European Union’s NextGenerationEU, as part of the Recovery, Transformation, and Resilience Plan, supported by Spanish INCIBE (6G-SOC project).

This work was supported partially by Vinnova - the Swedish Agency for Innovation Systems - through the EUREKA CELTIC-NEXT project CYPRESS.

Authors’ Addresses

Dan Garcia-Carrillo
University of Oviedo
Gijon, Asturias 33203
Spain
Email: garciadan@uniovi.es

Rafael Marin-Lopez
University of Murcia
Murcia 30100
Spain
Email: rafa@um.es

Gran Selander
Ericsson
SE-164 80 Stockholm
Sweden
Email: goran.selander@ericsson.com

John Preu Mattsson
Ericsson
SE-164 80 Stockholm
Sweden
Email: john.mattsson@ericsson.com

Francisco Lopez-Gomez
University of Murcia
Murcia 30100
Spain
Email: francisco.lopezg@um.es