Computer Standards & Interfaces 95 (2026) 104037

Contents lists available at ScienceDirect

Computer Standards & Interfaces

journal homepage: www.elsevier.com/locate/csi

# Towards a new standard for network access authentication: EAP-EDHOC

Francisco Lopez-Gomez [a,*], Rafael Marin-Lopez [a], Gabriel Lopez-Millan [a], Dan Garcia-Carrillo [b], John Preuß Mattsson [c], Göran Selander [c]

[a] Department of Information and Communications Engineering, University of Murcia, Spain
[b] Department for Computer Science, University of Oviedo, Spain
[c] Ericsson Research, Sweden

ARTICLE INFO

*Keywords:*
Network security
Wi-Fi networks
Authentication
EAP
EDHOC

ABSTRACT

The Extensible Authentication Protocol (EAP) has been a cornerstone of secure authentication in both wired and wireless networks, as well as enterprise systems, enabling integration with a wide range of authentication mechanisms. Recently, the IETF EAP Method Update (EMU) Working Group has adopted EAP-EDHOC, a method that combines EAP's extensibility with the recent standard Ephemeral Diffie–Hellman Over COSE (EDHOC). EDHOC is a lightweight authentication and key exchange protocol designed to be supported in resource-constrained environments. This enhances EAP-EDHOC as a high-performance authentication method for EAP-based networks. This paper presents a comprehensive analysis of the standardization efforts surrounding EAP-EDHOC, including a first proof-of-concept implementation and performance evaluation conducted over Wi-Fi networks. Additionally, a new design that optimizes the existing protocol by reversing the roles of the communication parties is proposed. The original and optimized versions are evaluated and compared with each other, as well as with EAP-TLS 1.3 and EAP-PSK. The results demonstrate that EAP-EDHOC achieves more efficient authentication than EAP-TLS 1.3 in terms of execution time, number of messages, and data transmitted. Meanwhile, EAP-PSK, which is based on symmetric cryptography, serves as a performance baseline.

## 1. Introduction

In the network security field, the need for robust, flexible, and efficient authentication mechanisms is more critical than ever, as they form the backbone of secure communications in a wide variety of environments, especially in wireless networks. These networks face unique challenges, including dynamic topologies, inconsistent signal quality, and increased exposure to external threats. Authentication protocols for wireless networks must not only ensure data integrity and confidentiality but also adapt to evolving security requirements.

The Extensible Authentication Protocol (EAP) [1] provides a standard mechanism supporting multiple authentication methods, known as EAP methods [2], ranging from password-based to certificate-based schemes (e.g., EAP-TLS [3]). EAP has served as a foundational protocol for flexible and secure authentication, and for establishing keying material between nodes in diverse environments. These environments include enterprise networks, Wi-Fi security—where EAP-TLS 1.3 [4] and EAP-PSK [5] play a significant role—cellular systems—such as EAP-AKA' [6] in 5G mobile networks—as well as Zigbee IP [7] and IEEE 802.15.9 [8]. In addition, EAP is used for authentication at the network level (e.g., PANA [9] and IKEv2 [10]), and at the application level (GGS-API [11]). Furthermore, EAP is also used for port-based network access control (PNAC) [12], which in turn is used to authenticate and key MACsec [13,14]. This adaptability has established EAP as a widely recognized standard for authentication. Moreover, its ability to integrate modern cryptographic protocols ensures its continuing relevance across both traditional and emerging network architectures, such as IoT networks.

To maintain EAP's effectiveness facing evolving security requirements, the IETF EAP Method Update (EMU) Working Group (WG) [15] focuses on designing, updating and expanding EAP methods, aligning the protocol with the latest security standards and requirements. In line with this objective, the integration with lightweight secure authentication protocols has become a key priority, as many applications increasingly require efficient and scalable authentication mechanisms.

Ephemeral Diffie–Hellman Over COSE (EDHOC) is a compact and secure authentication and key agreement protocol that uses ephemeral Diffie–Hellman keys, enabling a lightweight handshake protocol with robust security features (including mutual authentication, forward secrecy, and identity protection). It has been recently standardized in the

* Corresponding author.
*E-mail address:* francisco.lopezg@um.es (F. Lopez-Gomez).

https://doi.org/10.1016/j.csi.2025.104037
Received 3 February 2025; Received in revised form 2 May 2025; Accepted 16 June 2025
Available online 12 July 2025

IETF [16]. The protocol uses the Concise Binary Object Representation (CBOR) [17] for compact encoding, and the CBOR Object Signature and Encryption (COSE) [18] for secure encapsulation and identification of algorithms and credentials. EDHOC authentication can be based on raw public keys (RPKs) or public key certificates.

By integrating EDHOC into the EAP framework, the new EAP-EDHOC method combines the flexibility of EAP with the lightweight and secure features of EDHOC. This integration not only enables efficient network access authentication and key management in general network technologies, such as Wi-Fi [19], WiMAX [20], or 5G networks [21], but also opens the possibility of including EAP-EDHOC in constrained IoT networks that use EAP as authentication protocol, such as IEEE 802.15.9 or Zigbee IP. Notably, IEEE is already working on specifying EDHOC as a key management protocol (KMP) for IEEE 802.15.4 [22], and EAP-EDHOC is also applicable when the KMP is based on EAP. Furthermore, the EAP-EDHOC method aligns with the IETF EMU WG's mission to enhance EAP with the latest cryptographic protocols, to the point that the EAP-EDHOC draft has already been adopted for standardization.

This paper provides a comprehensive analysis and explanation of the ongoing standardization efforts around EAP-EDHOC. It presents the first proof-of-concept implementation and evaluates its performance. Additionally, it introduces a new design of EAP-EDHOC to further reduce the number of exchanges. Both versions are assessed in a Wi-Fi scenario and compared with EAP-TLS 1.3 – a reference method with similar characteristics based on asymmetric cryptography – and EAP-PSK, which serves as a performance benchmark due to its reliance on symmetric cryptography and inherently lower computational overhead.

While previous works have explored the performance of EDHOC or its potential in constrained environments, they have not addressed its integration into the EAP framework or provided practical evaluations of EAP-EDHOC itself. To the best of the authors' knowledge, this is the first attempt to analyze the IETF standardization effort aimed at integrating the features of EAP and EDHOC, bridging the gap between EAP-based authentication and lightweight key exchange protocols.

The remainder of this article is organized as follows. Section 2 reviews the background technologies related to EAP-EDHOC. Section 3 analyzes EAP-EDHOC and introduces a new design for it. Section 4 discusses the security properties of EAP-EDHOC. Section 5 describes the implementation, the deployed scenario, and the experimental results validating the proposal. Section 6 explores related work. Finally, Section 7 concludes the article and outlines directions for future research.

## 2. Background

### 2.1. The Extensible Authentication Protocol (EAP)

The Extensible Authentication Protocol (EAP) is an authentication framework that supports multiple authentication methods. It was designed for network access authentication where IP layer connectivity may not be available. It is a highly flexible and widely adopted technology, commonly used for authentication in IEEE 802.11 (Wi-Fi), IEEE 802.16 (WiMax) or protocols such as PANA, IKEv2. Its adaptability also enables EAP to support emerging technologies like IEEE 802.15.9 or Zigbee IP, where it plays a key role in secure authentication for IoT environments.

The key entities involved in the EAP exchange are the EAP Peer (Supplicant/Client), which is the device or user seeking network access, and communicates directly with the EAP Authenticator; the EAP Authenticator, which is a network device (such as a switch, access point, VPN server, etc.) that enforces access control and initiates the EAP authentication process; and the EAP Server, which is responsible for implementing the authentication methods and for authenticating the EAP Peer. For simplicity, the operational flow diagrams in this article depict only the EAP peer and the EAP server.

EAP can operate in two modes, namely *Standalone Mode* and *Pass-Through Mode*. In Standalone Mode, the EAP Authenticator and the EAP Server are co-located on the same device. In Pass-Through Mode, the EAP Authenticator relays EAP messages between the EAP Peer and the EAP Server, which is located on a remote authentication server. In this mode, authentication decisions are made by the EAP Server, not the EAP Authenticator. This setup is the most common, and its primary advantage is that the EAP Authenticator does not need to be updated to support new authentication methods or to store the EAP Peer's credentials.

The EAP authentication process begins with the EAP Authenticator sending a request to authenticate the Peer. This request specifies the information required from the Peer. Typically, the first request is an Identity Request. In response, the EAP Peer sends a response packet containing the information requested, in this case, its identifier. The message exchange continues between the EAP Server (either external to the EAP Authenticator or co-located with it) and the EAP Peer to perform the authentication of the client. Since EAP is a lock-step protocol, the EAP Server must wait for a valid response before sending the next request, retransmitting requests when needed. The process concludes when the EAP Server either determines that the Peer cannot be authenticated, sending an EAP Failure, or confirms successful authentication, sending an EAP Success.

In the context of wireless networks [23], EAP methods must generate key material for EAP entities after successful authentication, specifically the Master Session Key (MSK) and the Extended Master Session Key (EMSK), as defined by the EAP Key Management Framework (KMF) [24]. The MSK is used to derive Transient Session Keys (TSK) and then, to establish security associations, which allows securing communications between the EAP Peer and Authenticator. The MSK is transmitted by the EAP Server to the Authenticator via the AAA protocol in the pass-through mode, while the EMSK is kept secret between the EAP Peer and Server, not shared with other entities [25].

### 2.2. Ephemeral Diffie–Hellman Over COSE (EDHOC)

EDHOC is a lightweight authenticated key exchange protocol performed between two entities, the *Initiator* (I) and the *Responder* (R). It is based on the SIGn-and-MAc-I (SIGMA-I) variant protocol [26], which provides mutual authentication; identity protection of the Responder against passive attacks, and of the Initiator against active (and passive) attacks; and forward secrecy (all critical features in modern security frameworks). To put this in perspective, IKEv2 and TLS 1.3 are also based on SIGMA, which means EDHOC offers similar security characteristics to these widely used protocols. EDHOC authentication can be based on raw public keys (RPKs), or public key certificates. It supports the referencing of credentials in order to reduce message overhead, but credentials may alternatively be embedded in the messages. Furthermore, it offers compact cipher suite negotiation to enable crypto agility, flexibility and modularity.

EDHOC consists of three mandatory messages (message_1, message_2, message_3), an optional message_4, and an error message. These messages have a compact encoding based on CBOR sequences [27] with the secure encapsulation provided by COSE [17], minimizing message transmission and processing requirements. Additionally, EDHOC relies on elliptic curve cryptography, which is widely recognized for offering strong security with low computational overhead [28]. This makes EDHOC suitable for highly constrained networks [29], such as Cellular IoT, IPv6 over the TSCH mode of IEEE 802.15.4e (6TiSCH), and LoRaWAN. For instance, in scenarios with pre-provisioned public keys, the total messages size exchanged between I and R can be approximately 100 bytes [18]. To put this in perspective, this represents a significant improvement over traditional methods like TLS, where message sizes are typically larger. A detailed comparison of these protocols is provided in [30], which highlights the efficiency gains achievable with EDHOC without compromising security.

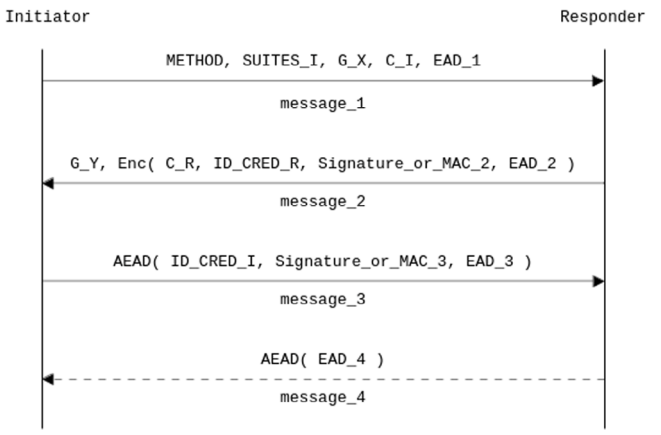



**Fig. 1.** EDHOC message flow.

EDHOC supports four authentication methods (Methods 0–3) within the ephemeral-ephemeral Diffie–Hellman key exchange. These methods rely on signature keys and static DH keys. Method 0 employs signature keys for both parties, whereas Method 3 uses static DH keys on both parties exclusively for authentication. Methods 1 and 2 combine signature keys and static DH keys, assigning them differently between the Initiator and Responder. It is worth noting that static DH keys are used for authentication together with ephemeral DH keys which provides good security properties, not to confuse with authentication based on static DH keys only, which is not used in EDHOC.

Fig. 1 illustrates the EDHOC message flow including the optional fourth message. The flow proceeds as follows:

- *Message_1*: The Initiator (I) starts the protocol by sending message_1, which includes the authentication method (METHOD) to be used, cipher suites (SUITES_I), the Initiator's Elliptic Curve Diffie–Hellman (ECDH) ephemeral public key (G_X), a connection identifier (C_I) for protocol state identification and additional external authorization data (EAD_1).
- *Message_2*: If the Responder (R) agrees on the method and cipher suites, it generates message_2. When the Responder uses Static Diffie–Hellman keys (methods 1 and 3), the authentication is achieved by a MAC function. Otherwise (methods 0 and 2), the Responder authenticates via a digital signature. Message_2 includes the Responder's ephemeral Diffie–Hellman public key G_Y, a connection identifier C_R, ID_CRED_R (identifying and optionally transporting R's credentials), the signature or MAC (Signature_or_MAC_2) and additional external authorization data EAD_2. Message_2 is encrypted (except for G_Y) and sent to the Initiator. Upon receiving this message, the Initiator authenticates the Responder's credentials by means of the signature or MAC. The keying material can be derived after message_2 has been sent or received.
- *Message_3*: The Initiator calculates MAC_3 and, depending on the authentication method used, the MAC is either sent in raw mode or embedded within a signature. Message_3, also containing ID_CRED_I and EAD_3, is protected using Authenticated Encryption with Associated Data (AEAD) and sent to R. The Responder receives the third message and decrypts it to authenticate the Initiator. At this point, both parties are mutually authenticated.
- *Message_4* (Optional): The Responder may then send an optional, encrypted fourth message containing external authorized data EAD_4 as acknowledgment. This message provides explicit Key Confirmation.

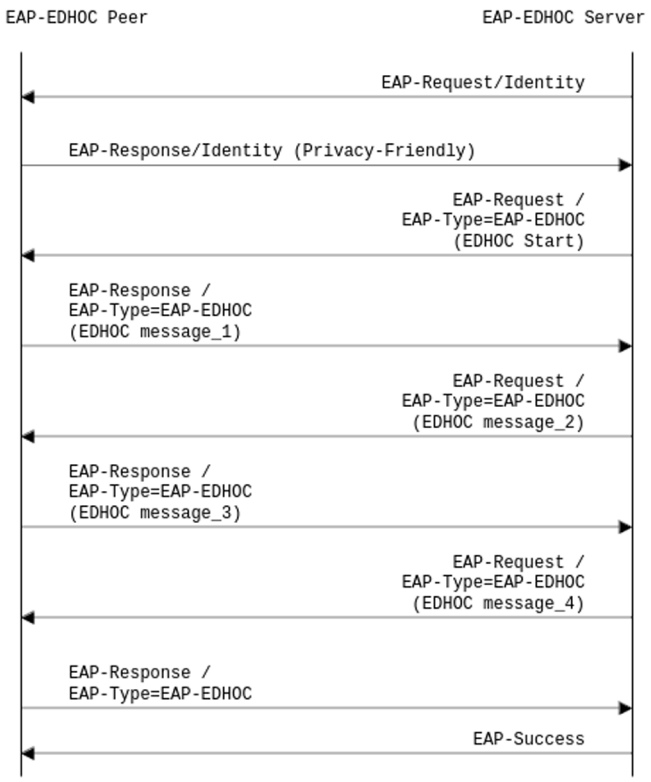



**Fig. 2.** EAP-EDHOC mutual authentication.

## 3. The EAP-EDHOC authentication method

### 3.1. EAP-EDHOC overview

EAP-EDHOC [31][1] is based on the authentication and security properties provided by EDHOC, using EAP as transport medium. It is important to mention that EDHOC specifies a list of transport requirements, detailed in Section 3.4 of RFC 9528 [16]. These requirements include handling the loss, reordering, duplication, correlation, and fragmentation of messages; demultiplexing EDHOC messages from other types of messages; and denial-of-service protection. The EAP-EDHOC method itself manages fragmentation and reassembly, while the rest of the requirements are handled by the EAP layer and lower layers.

Fig. 2 illustrates a typical message flow for successful EAP-EDHOC authentication. In this flow, the EAP-EDHOC Server, playing the role of the EAP Authenticator/Server, initiates communication by sending an EAP-Request/Identity message to request the peer's identity. The EAP-EDHOC Peer, playing the role of the EAP Peer, responds with a EAP-Response/Identity message, where the real username is replaced with a fixed or encrypted identifier (e.g., "@realm" or "anonymous@realm") to protect user privacy. This exchange is not part of EAP-EDHOC itself, as it serves to select the authentication method. Following this, the EAP-EDHOC Server sends an empty EAP-Request, specifying the type EAP-EDHOC. This message is also referred to as the EAP-EDHOC Start message. From this point, the communication proceeds with the EDHOC protocol messages encapsulated within the data fields of EAP-Response and EAP-Request packets. Once EDHOC messages have been

[1] It should be noted that at the time of writing this article, the EAP-EDHOC draft developed in the EMU working group is in version 00. Since this method is still in the standardization process, it may undergo changes over time. Updating the information presented in this paper to align with these changes is left as future work.

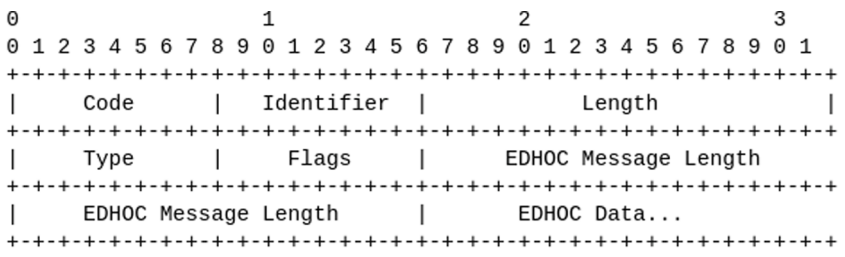



**Fig. 3.** EAP-EDHOC message format.

successfully exchanged, if everything is correct, the EAP-EDHOC Peer sends a final empty EAP-Request with EAP-Type=EAP-EDHOC and the EAP-EDHOC Server concludes the process by sending an EAP Success message, indicating the successful completion of the exchange.

It is important to note that this is a typical message flow, but EAP-Request/Identity and EAP-Response/Identity are optional, and EAP Success can be encoded in other network layer 2 specific messages.

In the default version of EAP-EDHOC, the entity acting as the EAP-EDHOC Peer (typically the client device requesting network access) assumes the role of the Initiator, sending EDHOC messages 1 and 3. Meanwhile, the EAP-EDHOC Server assumes the role of the Responder, sending EDHOC messages 2 and 4.

It is worth noting that EAP-EDHOC always uses message_4 as a protected success indication to ensure that the EAP Peer is securely informed of the success of the authentication process. This approach compensates for the lack of protection in the EAP Success message.

### 3.2. Message format

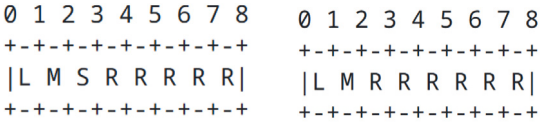
The EAP-EDHOC packet (Fig. 3) is composed of the following fields:

- On the one hand, a common portion to all EAP methods as it forms part of the EAP header:
  - Code: A byte field representing the type of the EAP packet. Options include Request (1), Response (2), Success (3), or Failure (4).
  - Identifier: A byte field used to match Responses with Requests.
  - Length: A two-byte field which indicates the total length in bytes of the EAP packet.
  - Type: A byte field present only in Request and Response EAP packets. It specifies the EAP method being used. A specific type value will be assigned to EAP-EDHOC during the standardization process.
- On the other hand, the EAP-EDHOC specific fields:
  - The Flags byte and the optional four-byte EDHOC Message Length field that serve to support fragmentation (see Section 3.3).
  - EDHOC Data: This field has a variable size and contains the actual EDHOC messages.

An empty (no payload) EAP-EDHOC message is 6 bytes in size (1 byte for the Code, 1 byte for the Identifier, 2 bytes for the Length, 1 byte for the Type, and 1 byte for the Flags). Finally, the EAP Success and Failure messages are only 4 bytes in size.

### 3.3. Fragmentation support

EAP-EDHOC supports fragmentation through the addition of the flags octet in the EAP-Request (Fig. 4(a)) and EAP-Response (Fig. 4(b)) packets, as well as a (conditional) 4-octet EAP-EDHOC Message Length field. Although, EAP-EDHOC itself generates very compact messages, there might be some network technologies with a reduced Maximum Transfer Unit (MTU) that can be too low for even small authentication messages containing cryptographic data or certificates, as it may happen with EDHOC authentication. For instance, IEEE 802.15.4 [32], commonly used in low-power IoT networks, has an MTU as low as 127 bytes. Without fragmentation, messages larger than the MTU would be unable to fit within a single packet. Therefore this fragmentation support open the door to extend the applicability of EAP-EDHOC to IoT networks.




(a) EAP-EDHOC Request flags. (b) EAP-EDHOC Response flags.

**Fig. 4.** EAP-EDHOC Flags octets.

The different flags used in the protocol (see Fig. 4) are the *Length included* (L), *More fragments* (M), and *EAP-EDHOC Start* (S) bits. The S flag simply marks the initial message of the EAP-EDHOC method's communication to distinguish it from the rest of EAP-EDHOC messages. Then, the S flag is set only in the EAP-EDHOC Start message sent from the EAP Server to the Peer. This is the reason why this bit is not included in the flags octet of EAP-Response. The M flag is used for fragmentation and is set on all but the last fragment (understand it as a "More fragments left"). The L flag is set to indicate the presence of the EDHOC Message Length field, and must be set for the first fragment of a fragmented EDHOC message. The EDHOC Message Length field provides the total length of the EDHOC message that is being fragmented, which simplifies buffer allocation. The 'R' in the flags octets means *Reserved*.

When an EAP-EDHOC entity receives an EAP-EDHOC packet with the M bit set, it must respond with an EAP packet with EAP-Type=EAP-EDHOC containing no data, serving as a fragment acknowledgment (ACK). The other entity must wait for this ACK before sending the next fragment.

### 3.4. Key derivation

Upon successful authentication using EDHOC, a symmetric shared session key, *PRK_out*, is generated. Additionally, another key, *PRK_exporter*, is derived from *PRK_out*. The *PRK_exporter* is crucial for deriving further symmetric keys, which are used to secure application-level data. More specifically, EAP-EDHOC uses the *PRK_exporter* key and the key derivation function named *EDHOC-Exporter* defined in [16] to derive the EAP Master Session Key (MSK) and the Extended Master Session Key (EMSK).

The MSK and EMSK are typical outputs of EAP methods and are essential for enabling lower-layer security mechanisms, such as establishing secure communication sessions. However, EAP-EDHOC itself does not define the specific usage of these keys at lower layers, leaving their application dependent on the underlying network implementation. For example, in Wi-Fi networks, the MSK is traditionally used

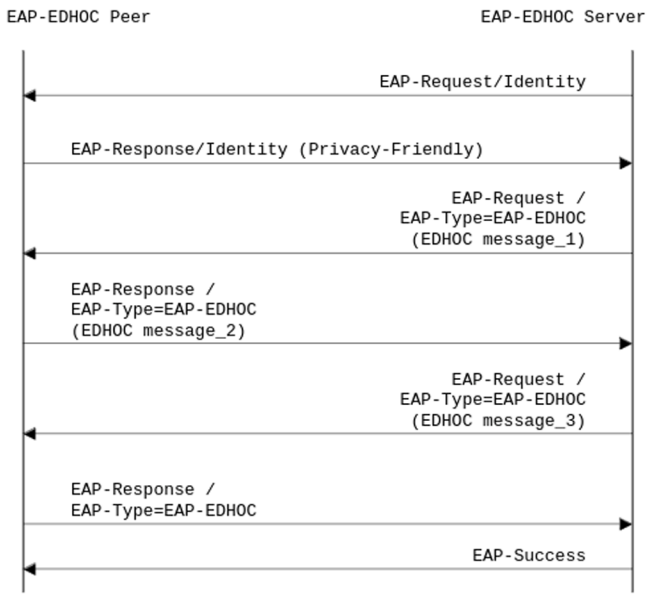



**Fig. 5.** EAP-EDHOC reverse-role flow.

by the 802.1X framework to derive transient session keys (TSKs) that secure communication between the supplicant and the access point.

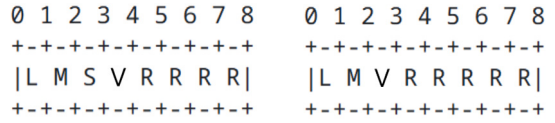
In addition to exporting key material, EAP-EDHOC generates other standard EAP-related attributes, such as the *Method-ID* and *Session-ID*. Notably, the Method-ID is derived using the *EDHOC-Exporter* function to ensure it is based on a fresh cryptographic key. The *Session-ID* is then constructed as *Type* || *Method-ID*. These attributes ensure compatibility with the EAP framework and facilitate integration with existing AAA infrastructures.

### 3.5. A new approach: reverse-role flow

Once presented the fundamentals of EAP-EDHOC this paper also presents an alternative design for EAP-EDHOC that involves switching the roles of Initiator and Responder (reverse-role flow). The main goal is to reduce the number of exchanged messages and bytes transmitted over the network, and hence the time for authentication.

In this new scheme, the EAP Server acts as the Initiator, sending EDHOC message_1 and message_3, while the EAP Peer assumes the role of the Responder. This enables the EAP-EDHOC Server to directly send EDHOC message_1 immediately after receiving the EAP-Response/Identity message, eliminating the need for the EAP-EDHOC Start message. Additionally, EDHOC message_4 is no longer necessary in this configuration. You can see the resulting scheme with reversed roles in Fig. 5. The rationale and security implications of this role reversal are discussed in detail in Section 4.

This proposal for the new scheme does not aim to replace the current one; instead, both schemes can coexist with a minor modification to the EAP-EDHOC method. The operation mode would be as follows: in the case of a server that supports this reverse-role flow, the server can directly send the EDHOC message_1 to the EAP-Peer after receiving the EAP-Response/Identity message, flagging it to indicate that the message follows the reverse-role flow. If the EAP-Peer does not support this scheme or is configured to use exclusively the default EAP-EDHOC's scheme, it will generate and send a new EDHOC message_1 (with the reverse-role flag disabled), allowing the exchange to proceed normally according to the standard EAP-EDHOC scheme. Conversely, if the EAP-Peer supports this new scheme, it will recognize the flag indicating the role inversion and it will start the processing of message_1, so the message exchange will continue as described in Fig. 5. It is worth noting




(a) EAP-EDHOC Request updated flags. (b) EAP-EDHOC Response updated flags.

**Fig. 6.** EAP-EDHOC's updated flags octets.

that the key derivation process remains unchanged in this new flow compared to the standard scheme.

A new flag is defined, present in every message which follows the reverse-role flow. In Fig. 6 the resulting EAP-EDHOC flag octets are shown, including the new reverse-role flag bit (V).

As observed, the EDHOC message_1 is included by the EAP Server in the first EAP Request of EAP-EDHOC. Thus, this message will have the S (Start) flag activated.

In the best-case scenario, where both the EAP Peer and EAP Server support the reverse-role flow, the EAP-EDHOC Start message (an empty EAP-EDHOC message) and EAP-EDHOC's message_4 (occupying 9 bytes plus the EAP-EDHOC header size) are omitted. Consequently, the number of exchanged messages is reduced by two. Considering that the EAP-EDHOC headers are 6 bytes in total, this results in an overall saving of 21 bytes (6 bytes from the Start message + 6 + 9 bytes from the message_4).

On the other hand, the worst-case scenario arises when the server supports the reverse-role scheme and directly sends message_1 to initiate the EDHOC exchange. However, if the EAP Peer does not support this scheme or demands greater privacy, it will respond by initiating the standard scheme, sending its own message_1. In this case, no additional EAP packets are introduced, but the EAP-EDHOC Start message sent by the server (which was previously empty in the standard scheme) now includes EDHOC message_1 (typically 37 bytes, excluding the EAD). Consequently, these 37 additional bytes (plus any EAD data) are transmitted over the network, but without requiring additional EAP packet exchanges.

## 4. Security considerations

This section discusses the key security properties of both EAP-EDHOC and the proposed reverse-role flow.

### 4.1. Default EAP-EDHOC

The security properties of EAP-EDHOC [31] are grounded in the strengths of both EAP and EDHOC, providing robust protections for network access authentication. As such, the security considerations outlined in EAP [1] and EDHOC [16] are directly applicable to EAP-EDHOC.

EAP is a flexible authentication framework designed to support a wide range of network technologies, including Wi-Fi and cellular networks. Its security guarantees, however, largely depend on the specific EAP method employed. By integrating EDHOC as an EAP method, EAP-EDHOC inherits the strong cryptographic properties of EDHOC—such as end-to-end authenticated key exchange, resistance to replay and man-in-the-middle (MITM) attacks, and resilience against key compromise—to authenticate the parties and derivate a shared secret (MSK).

The main security properties provided by EAP-EDHOC can be summarized as follows:

- *Mutual authentication*: EAP-EDHOC ensures mutual authentication between the client and the server by leveraging EDHOC's cryptographic handshake, which prevents impersonation and unauthorized access. By verifying both parties' identities, the protocol ensures only authenticated clients access the network, while clients

can trust they are communicating with a legitimate network server.

- *Ephemeral key exchange*: EDHOC performs an ephemeral DH exchange in the first two messages. This provides forward secrecy, ensuring that the compromise of a session key does not also compromise earlier sessions' keys and protecting future session keys from passive attackers (ephemeral key exchange forces attackers to be active if they want access to future session keys, even if they have access to a long-term key).
- *Identity protection*: EDHOC provides identity protection, which safeguards personal information and identity from theft or fraud and protects from tracking. This identity protection is asymmetrical due to the protocol's SIGMA-I design:
  - The Initiator receives protection against both passive and active attackers, ensuring that the EAP Peer/I identity is hidden from third parties attempting to eavesdrop or actively intervene.
  - The Responder receives identity protection only against passive attackers. An active attacker could potentially uncover the EAP Server/R identity by eavesdropping on the destination address used for transporting message_1 and then sending a crafted message_1 to that address. After that the Responder will answer with message_2, which can be deciphered by the attacker, hence discovering the Responder's identity.
- *Resistance to known attacks*: EDHOC's security has been thoroughly analyzed (e.g., in [33]) to ensure resilience against known cryptographic attacks, such as replay, man-in-the-middle, and reflection attacks. The handshake includes mitigation against key-compromise impersonation and guarantees session key freshness by integrating ephemeral keys.
- *Message integrity and confidentiality*: Each message in the EDHOC handshake is authenticated and encrypted to prevent tampering or interception by unauthorized parties. It makes EDHOC robust against interception and injection attacks during the authentication process.

### 4.2. Reverse-role EAP-EDHOC

As mentioned earlier, EAP-EDHOC provides asymmetrical identity protection. In this new scheme, which involves switching the roles of the communication, identity protection against active attackers shifts from the EAP Peer to the EAP Server, which now acts as the Initiator of the communication. Consequently, the EAP Peer's identity is protected only against passive attackers, while the EAP Server gains identity protection against both passive and active attackers.

This can actually be advantageous in certain scenarios, where safeguarding the server's identity can be more critical. For example, in public Wi-Fi deployments or enterprise networks, protecting the EAP Server's identity can help mitigate targeted attacks aimed at identifying, exploiting or denying the authentication infrastructure itself. Similarly, in situations where the EAP Server represents a centralized gateway or a high-value node, concealing its identity from active and passive attackers can reduce the risk of reconnaissance and targeted compromise.

The remaining security features discussed in Section 4.1 apply similarly to this new message flow.

On the other hand, message_4 is not required in this new design of EAP-EDHOC. Recall message_4 goals in the original EAP-EDHOC specification: first, it acts as a protected success indication and second, it provides explicit key confirmation by allowing the Initiator to confirm that Responder owns the agreed key material (the Responder already confirmed that the Initiator had the proper session key during verification of message_3).

With EAP-EDHOC with reversed roles, the correct verification of message_3 allows the Initiator to authenticate the Responder and to confirm that Responder had already authenticated it, hence acting as protected success indication.

Without message_4, the Responder delays the key confirmation of the Initiator's keys until some data communication happens using the MSK exported by the EAP-EDHOC method through the running of a secure association protocol, such as explained in the EAP Key Management Framework [24] (e.g., the 4-way handshake in Wi-Fi networks [34]).

### 4.3. Post-quantum considerations

At present, it remains uncertain when—or even if—a quantum computer powerful enough to break widely used public key cryptographic schemes will become a reality. Nonetheless, due to the long timelines involved in standardization and deployment, efforts to define Post-Quantum Cryptography (PQC) standards are already well underway. Authors in [35] discuss the emerging threats posed by quantum computing and review the current landscape and advancements in PQC research. In [36] it is provided an overview of the current state and recent advancements in PQC from the perspective of NIST. In addition, ongoing research continues to explore advanced approaches to PQC (e.g., [37]).

In the context of EAP-EDHOC, the underlying EDHOC protocol supports all signature algorithms defined by COSE, including post-quantum algorithms such as HSS-LMS. While EDHOC is currently specified for use with ECDH-based key exchange mechanisms, it is also compatible with alternative Key Encapsulation Mechanisms (KEMs), including those based on PQC, when used with method 0 [16]. However, it is important to note that current PQC algorithms generally exhibit limitations compared to traditional elliptic curve cryptography (ECC), particularly in terms of key and message sizes. These larger sizes can pose significant challenges in constrained IoT environments, which must be carefully considered when integrating PQC into lightweight authentication protocols like EDHOC [33]. The IETF LAKE working group is already exploring comprehensive solutions to enable post-quantum security in EDHOC. If post-quantum support is eventually integrated into EDHOC, the EAP-EDHOC method will automatically inherit these enhancements, extending its applicability to quantum-resistant authentication scenarios.

## 5. Implementation and experimental results

To the best of the authors' knowledge, this work presents the first proof-of-concept implementation of both EAP-EDHOC schemes (standard and reverse-role). It is worth highlighting that a first version of this implementation was presented and tested at the *Hackathon on Lightweight IoT Security* held in Paris in 2024 [38], providing an opportunity for early validation and feedback from experts in the field.

Additionally, performance results in a Wi-Fi scenario have been obtained to validate both EAP-EDHOC schemes, comparing them against two well-known and widely used EAP methods: EAP-TLS 1.3 and EAP-PSK. The comparison with EAP-TLS 1.3 is particularly relevant, as both TLS 1.3 and EDHOC are based on the SIGMA-I variant, making EAP-TLS 1.3 and EAP-EDHOC naturally comparable in terms of security features and structure. In contrast, the comparison with EAP-PSK serves as a performance benchmark, providing insight into how the proposed method compares with an authentication protocol based purely on symmetric cryptography.

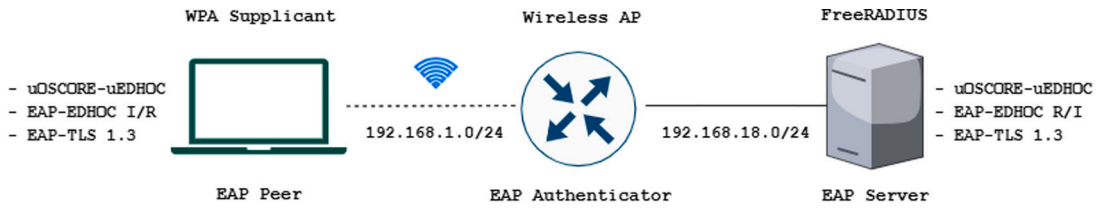



**Fig. 7.** Deployed scenario.

### 5.1. Implementation details

The EAP-EDHOC Peer implementation is based on WPA Supplicant 2.11_dev (development version) [39], while the EAP-EDHOC Server implementation has been developed in FreeRADIUS 3.2.3 [40]. These two well-established technologies, each with their own robust implementations of the EAP state machine, provide a solid foundation for integrating the new EAP method.

Both the WPA Supplicant and FreeRADIUS implementations of EAP-EDHOC rely on the uOSCORE-uEDHOC library (version 3.0.5_dev) [41]. This library uses *MbedTLS* [42] and *Compact25519* [43] as the underlying cryptographic engines. *Compact25519* supports EDHOC cryptographic suites 0 and 1, both based on Curve25519, while *MbedTLS* provides support for EDHOC suites 2 and 3, which use the NIST P-256 curve. A detailed description of these suites can be found in [44].

The implementation was entirely developed in the C programming language, both for performance reasons and because the aforementioned libraries are also written in C.

The implementation includes all major features of EAP-EDHOC as described in [31], including support for EAP-EDHOC fragmentation. Additionally, it supports EDHOC authentication methods 0 and 3. *Method 0* uses signature keys for both the Initiator and Responder, representing the worst case in terms of message size and transmitted data. In contrast, *Method 3* employs static Diffie–Hellman (DH) keys, representing the best case in terms of efficiency. Finally, the implementation incorporates an initial approach to the reverse-role flow.

A virtualized testbed based on Docker has been developed to enable anyone to test the implemented authentication method and its variants. This environment is available in [45]. The repository includes comprehensive instructions for performing tests within the scenario, along with the complete implementation code.

### 5.2. Deployed testbed

The deployed testbed is illustrated in Fig. 7. The EAP-EDHOC Peer runs on a laptop equipped with an Intel® Core™ i7-13620H CPU and 16 GB of RAM. Additionally, the EAP-EDHOC Authenticator is a real AP (Linksys WRT54GL) which operates in pass-through mode, relaying authentication messages between the EAP Peer and an external authentication server. Finally, the Authentication Server (acting as EAP-EDHOC Server) is running on another laptop with an Intel® Core™ i5-9300H CPU and 8 GB of RAM. It is important to note that the link between the AP and the EAP Server is a wired Ethernet connection using 802.3 technology, whereas the link between the AP and the Supplicant is a wireless connection using 802.11 g, which has a theoretical maximum data rate of 54 Mbps. This serves as a representative wireless technology where EAP has been widely utilized for a long time.

Although EDHOC is primarily designed as a lightweight solution for constrained environments, we have chosen to deploy the test scenario in an unconstrained Wi-Fi environment (two laptops and an AP) as shown in Fig. 7. The goal of this decision is to simulate a common, real-world scenario where EAP has been frequently used for a long time. While testing in constrained environments is an important future step, the current focus is now on demonstrating the integration and functionality of EAP-EDHOC in environments where EAP is already prevalent, providing a practical and widely applicable test case. For that reason, in the following section we will also compare EAP-EDHOC's performance with two of the most known and widely used EAP methods: EAP-TLS 1.3 [4] and EAP-PSK [5].

### 5.3. Performance evaluation

Six tests have been designed to evaluate the implementation's performance:

- Test 1: *EAP-EDHOC-m3*: This first test assesses EAP-EDHOC's authentication method 3, based on the use of static DH keys for authentication in both parties.
- Test 2: *EAP-EDHOC-m3-REV*: This test evaluates EAP-EDHOC's authentication method 3 following the reverse-role flow.
- Test 3: *EAP-EDHOC-m0*: This test focuses on EAP-EDHOC's authentication method 0, based on the use of signature keys for authentication in both parties.
- Test 4: *EAP-EDHOC-m0-REV*: This test evaluates EAP-EDHOC's authentication method 0 following the reverse-role flow.
- Test 5: *EAP-TLS-1.3*: This test measures the performance of EAP-TLS 1.3, serving as a reference to compare the different EAP-EDHOC variants against a well-established and structurally similar protocol.
- Test 6: *EAP-PSK*: This test measures the performance of EAP-PSK, providing a reference point for comparing EAP-EDHOC with a method based solely on symmetric cryptography.

For each of the tests described, 30 runs were performed. In each run, a supplicant is authenticated using the EAP method corresponding to the specific test. The cryptographic suites employed are those detailed in Table 1. For the tests based on asymmetric cryptography (EAP-EDHOC and EAP-TLS), the suites were carefully selected to ensure a comparable workload during the authentication process. In contrast, the EAP-PSK test uses the fixed profile defined in the method's specification [5].

#### 5.3.1. Number of messages

The number of messages exchanged over the network in each test, including the EAP and method overheads, can be seen in Table 2. As can be observed, in EAP-TLS 1.3, a total of 13 messages are exchanged. This is due to the large size of its messages, which necessitates fragmentation even in a resource-rich scenario like the one under consideration. On the other hand, in EAP-PSK a total of 7 messages are exchanged.

In EAP-EDHOC, the standard flow requires the exchange of 9 messages, whereas the reverse-role flow reduces this to 7 messages by eliminating the EAP-EDHOC Start and EAP-EDHOC message_4.

#### 5.3.2. Messages size

Fig. 8 shows the size in bytes (Y-axis) of the messages exchanged over the network for each test (X-axis). Three different values are shown: the size in bytes of the EAP headers (*EAP overhead*); the size in bytes of the EAP method itself, i.e., the payload of the EAP packets (*Method overhead*); and the total size in bytes, which is the sum of the previous two values.

The results show that the reverse-role flow proposed in this paper reduces the total size in 21 bytes. This represents a 2.4% reduction in

**Table 1**
Cryptographic suites used in the different tests.

| | EAP-EDHOC (Tests 1 to 4) | EAP-TLS 1.3 | EAP-PSK |
|---|---|---|---|
| Suite name | EDHOC Suite 2 | TLS_AES_128 _CCM_SHA256 | Fixed profile (AES-128, CMAC/EAX) |
| Key exchange alg. | ECDHE using the P-256 curve | ECDHE using the P-256 curve | – |
| Signature alg. | ES256 (ECDSA using P-256) | ES256 (ECDSA using P-256) | – |
| Hash alg. | SHA-256 | SHA-256 | CMAC (AES-128-based) |
| AEAD | AES-CCM-16-64-128 | AES_128_CCM | AES-128 in EAX mode |
| Authentication type | X.509 with P-256 key | X.509 with P-256 key | PSK (16-byte symmetric key) |

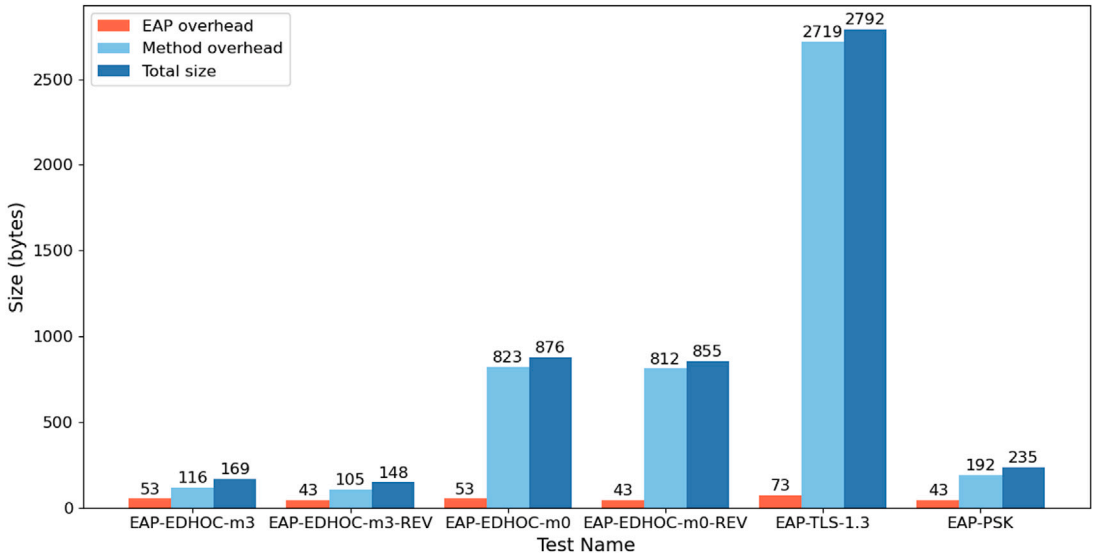



**Fig. 8.** Message sizes.

**Table 2**
Number of messages exchanged.

| | Number of messages |
|---|---|
| EAP-EDHOC-m3 | 9 |
| EAP-EDHOC-m3-REV | 7 |
| EAP-EDHOC-m0 | 9 |
| EAP-EDHOC-m0-REV | 7 |
| EAP-TLS-1.3 | 13 |
| EAP-PSK | 7 |

transmitted bytes for EAP-EDHOC-m0-REV compared to the standard EAP-EDHOC-m0 and a 12.4% reduction for EAP-EDHOC-m3-REV compared to EAP-EDHOC-m3. On the other hand, EAP-EDHOC method 3 significantly reduces message size compared to EAP-EDHOC method 0, dropping from 876 bytes transmitted in method 0 to just 169 bytes in method 3. It is worth noting that in tests 1 and 2, X.509 certificates are sent by reference, whereas in tests 3, 4 and 5, they are sent by value. Sending certificates by reference results in a saving of 293 bytes. Additionally, a notable difference can be seen in the number of bytes transmitted between EAP-TLS 1.3 and EAP-EDHOC, where EAP-EDHOC, in any of its variants, results in a substantial reduction in transmitted bytes over the network. However, EAP-EDHOC on its method 3 variants, achieves a result even better, introducing less bytes on the network. This shows the potential of EAP-EDHOC for authentication in innumerable types of networks, where it can complete an authentication using even fewer bytes than symmetric cryptography methods.

Finally, it can be observed that an EAP-PSK authentication introduces an overhead of 235 bytes, which is quite low due to its reliance on symmetric cryptography. However, EAP-EDHOC on its method 3 variants achieves even better results, introducing fewer bytes on the network. This highlights the potential of EAP-EDHOC for authentication in a wide range of networks, including constrained links, where it can complete the authentication process using even fewer bytes than symmetric cryptography methods.

### 5.3.3. *Time measurements*

Fig. 9 presents the time required to complete a user authentication (Y-axis) for each test (X-axis). Two specific time measurements are included: First, *Total Time* measures the overall time required to complete the authentication using the corresponding method, including both the processing time at each part of the communication and the network transmission time; Second, *Internal Method Processing Time* measures the cumulative processing time taken by the EAP method on both client and server sides to complete the authentication.

It is worth noting that this plot shows the average time values (in milliseconds) of 30 runs performed for each case. The vertical lines at the top of the bars indicate the confidence intervals, providing a measure of the variability and reliability of the results.

It is important to highlight that *Internal Method Processing Time* excludes the time spent on processing EAP Success and EAP Request/Response Identity messages, since it is common to all tests. Instead, it measures the time taken to process and generate only the messages specific to the EAP method being used (i.e., those with the EAP-Type field enabled). It allows us to focus on comparing the processing time of EAP-EDHOC on its different variants with the processing time of EAP-TLS 1.3 and EAP-PSK.

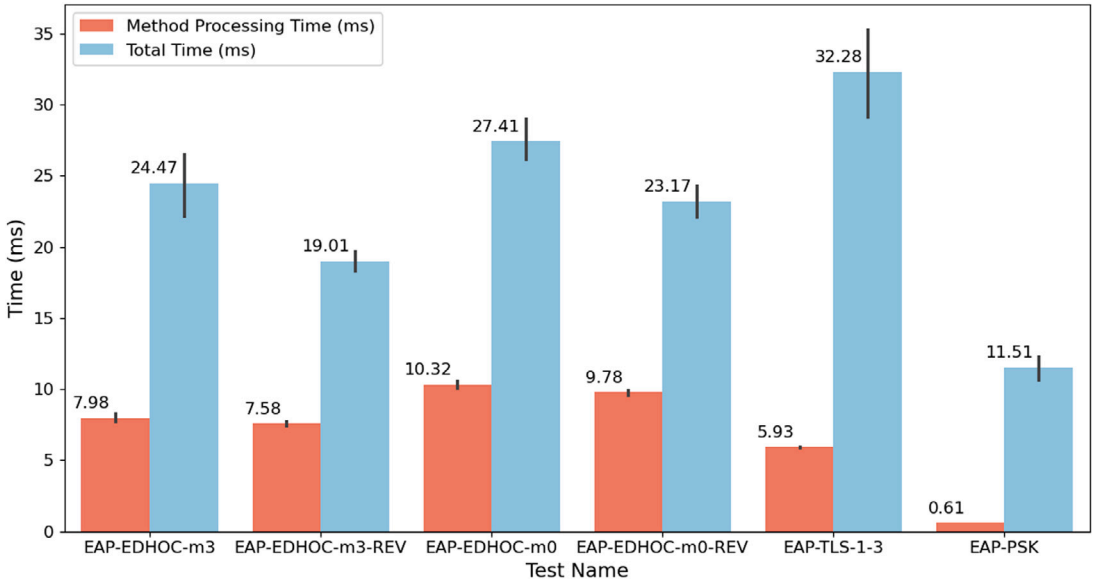



**Fig. 9.** Authentication time.

It can now be concluded that the results shown in Fig. 9 align with the expected outcomes. *EAP-EDHOC method 3* outperforms *EAP-EDHOC method 0*, particularly in method processing time. This is because both methods transmit the same number of messages, but the messages in method 3 are significantly shorter and lighter, leading to a slight reduction in network transmission time and a substantial decrease in processing time.

Furthermore, the *reverse-role variant* also improves upon the default EAP-EDHOC scheme for both method 3 and method 0, mainly in network time (which is reflected in the total authentication time), as fewer messages are exchanged.

EAP-EDHOC achieves results that are close to those of EAP-PSK, which is particularly relevant given that EAP-PSK relies on symmetric cryptographic operations, inherently faster than the asymmetric operations used in EAP-EDHOC. As such, EAP-PSK serves as a performance reference point, which is not far from the current implementation, and which will be attempted to be achieved as part of future work.

Finally, even EAP-EDHOC variant using method 0 with the standard message flow, which represents the EAP-EDHOC worst-case scenario, is faster than EAP-TLS 1.3 in terms of total time. This occurs because the number of messages (and their size) is much higher in EAP-TLS, which also needs to fragment the messages due to their length. This affects the network time and, therefore, the total authentication time. Furthermore, this is particularly notable considering that EAP-TLS has a much lower internal processing time, as its implementations (which use *OpenSSL*) are significantly faster than those of EAP-EDHOC, which rely on *uOSCORE-uEDHOC* (and therefore are optimized to microcontrollers).

In scenarios more favorable to constrained libraries (e.g., IoT environments, constrained radio networks [46]), EAP-EDHOC is a promising alternative and it is expected to perform even better in contrast with EAP-TLS.

### 5.4. Discussion

In summary, the performance evaluations demonstrate the clear advantages of EAP-EDHOC over one of the most widely-used EAP methods, EAP-TLS 1.3. The results show that EAP-EDHOC is faster and transmits significantly fewer bytes and messages, while providing similar security characteristics. These improvements make the protocol highly scalable and applicable to a variety of real-world deployments, including Wi-Fi environments and other potential scenarios (e.g., IoT deployments).

On the other hand, EAP-EDHOC has achieved results close to those of EAP-PSK, despite EAP-PSK relying on symmetric cryptography. This comparison presents a promising direction for further development of EAP-EDHOC, as an EDHOC authentication method based on pre-shared keys is currently being standardized [47]. Future work will focus on integrating this EDHOC-PSK variant into EAP-EDHOC, enabling symmetric key-based authentication and potentially achieving performance levels comparable to those of EAP-PSK.

Additionally, these performance evaluations have also demonstrated that the reverse-role flow proposed for EAP-EDHOC in this document results in a two-message reduction in the overall exchange, a slight reduction in the sizes transmitted over the network, and a significant decrease in total authentication time. The decision between using the standard EAP-EDHOC flow or the reverse-role flow will depend on the specific needs of the domain. Some domains may prioritize protecting the identity of their nodes against active attackers, in which case the current draft of the EAP-EDHOC flow would be preferred. On the other hand, other domains might focus on reducing the number of bytes transmitted over the network and minimizing authentication time, making the reverse-role flow more suitable.

## 6. Related work

Several efforts have laid the foundation for the development and analysis of EAP-EDHOC, EDHOC, and related protocols in constrained environments. Among these, the following key works are particularly relevant to our research.

In [2], the authors present an overview of EAP and its application in wireless technologies, discussing widely used EAP methods and their suitability in this field. This work supports our research by establishing a context for evaluating EAP-EDHOC within wireless network scenarios.

Authors in [48] compare EDHOC's and DTLS 1.3's performances in IoT environments. This paper underscores EDHOC's advantages over DTLS 1.3 regarding resource consumption, demonstrating its suitability for constrained networks. However, our research advances beyond this comparison by focusing on EAP-EDHOC, extending the evaluation to contexts where EAP plays a central role in network access authentication.

Authors in [49] present performance results for EDHOC in IoT environments. These results illustrate EDHOC's efficiency, underscoring

its suitability for developing a more compact and efficient EAP authentication method that could be applied across diverse networks where EAP is utilized.

In [50], the authors present the design and performance evaluation of an embedded EDHOC module developed for constrained environments. This work investigates the runtime and memory footprint overheads associated with EDHOC, demonstrating its good performance and efficiency in constrained settings. These characteristics make EAP-EDHOC a promising candidate for usage in constrained environments.

The work in [33] provides a comprehensive security analysis of EDHOC, outlining its strengths, vulnerabilities, and possible mitigations. This analysis is particularly relevant to our research, as EAP-EDHOC inherits EDHOC's security properties. Thus, this work serves as a preliminary security evaluation for the proposed method.

Finally, authors in [51] explore the adaptation of the EAP-NOOB method for LoRaWAN, marking an early attempt to adapt EAP for authenticating IoT devices in constrained environments. This effort highlights EAP's flexibility, which supports the development of more streamlined EAP methods, such as the one based on EDHOC presented in this paper. However, EAP-NOOB is designed for a different objective. Unlike most EAP methods, including the one proposed here, EAP-NOOB does not perform direct authentication. Instead, it relies on an out-of-band channel requiring user interaction. Additionally, it is used for authenticating non-preconfigured devices.

In summary, this paper advances the state of the art as it represents the first attempt to analyze the ongoing standardization of EAP-EDHOC. To this end, an implementation of this method is provided, and an alternative design is explored to further improve its efficiency, reducing the number of exchanged messages and the bytes processed by devices and transmitted over the network.

## 7. Conclusions and future work

This paper presents a comprehensive analysis and detailed explanation of the ongoing standardization efforts surrounding EAP-EDHOC, an authentication method that integrates the lightweight EDHOC key exchange protocol within the well-established Extensible Authentication Protocol (EAP) framework. By merging the flexibility of EAP with the efficiency of EDHOC, EAP-EDHOC enables secure authentication and key management in a wide range of network infrastructures, including Wi-Fi settings, where high performance and low overhead are essential.

Additionally, the design of a novel feature that allows the roles of the communicating parties to be reversed for the authentication process has been proposed, optimizing the EAP-EDHOC authentication procedure by reducing the number of messages and bytes needed to complete it and, consequently, the authentication time.

Following this, a first proof-of-concept implementation of EAP-EDHOC is presented, alongside performance evaluations that highlight clear advantages of EAP-EDHOC over one of the most widely-used EAP methods, EAP-TLS 1.3. The results indicate that EAP-EDHOC operates faster and transmits considerably fewer messages and bytes, making the protocol highly scalable and adaptable to a range of real-world applications, including Wi-Fi networks. This efficiency also suggests its potential applicability in more constrained environments, such as IoT networks.

In contrast to previous works, which primarily focused on evaluating EDHOC or comparing its performance with other lightweight protocols such as DTLS, this study goes further by implementing and evaluating EAP-EDHOC as a complete EAP method in a real-world setting. The proposed reverse-role optimization and the direct comparison with EAP-TLS 1.3 and EAP-PSK provide new insights into how EAP-EDHOC can improve performance in practical deployments.

Additionally, the results show that the reverse-role flow proposed in this paper reduces the overall exchange by two messages and decreases network transmission by 21 bytes. This represents a 2.4% reduction in transmitted bytes for EAP-EDHOC's method 0 with reverse roles compared to the standard EAP-EDHOC's method 0, and a 12.4% reduction for method 3 with reverse-roles compared to the standard EAP-EDHOC's method 3. Furthermore, this reverse-role flow significantly improves overall authentication time. This reduction is especially valuable in constrained environments with limited resources or power, where network links often have a very restricted Maximum Transfer Unit (MTU).

Looking ahead, future work will focus on further validating the efficiency of EAP-EDHOC by deploying the testbed in more constrained environments. This will allow us to assess its performance under stricter resource limitations, such as in low-power IoT networks, and to explore its potential for EAP-based authentication in IoT contexts like IEEE 802.15.9 or Zigbee IP. By refining and adapting EAP-EDHOC, the goal is to support the development of secure, lightweight authentication mechanisms for next-generation IoT systems and other emerging technologies. Additionally, implementing any updates introduced to the EAP-EDHOC draft as it moves towards standardization is also part of the planned future work. During this process, it will be also necessary to perform a formal security analysis of EAP-EDHOC.

Finally, it has been shown that EAP-PSK remains faster than EAP-EDHOC, opening another line of work consisting in incorporating a new authentication method based on PSK to EAP-EDHOC, which will enable symmetric key-based authentication and will potentially achieve even better results, comparable to those of EAP-PSK.

## CRediT authorship contribution statement

**Francisco Lopez-Gomez:** Writing – original draft, Validation, Software, Resources. **Rafael Marin-Lopez:** Writing – review & editing, Validation, Software, Methodology, Conceptualization. **Gabriel Lopez-Millan:** Writing – review & editing, Validation, Methodology, Conceptualization. **Dan Garcia-Carrillo:** Writing – review & editing, Software, Conceptualization. **John Preuß Mattsson:** Writing – review & editing, Conceptualization. **Göran Selander:** Writing – review & editing, Conceptualization.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgments

The authors would like to thank Stefan Hristozov for his invaluable assistance with the implementation. His constant support and readiness to help were crucial to its successful completion. The article has been partially funded by Project PID2023-148104OB-C43 from MCIN/AEI, partially funded by the FPI Grant PREP2023-002169, also from MCIN/AEI and partially funded from the European Union's HE Research and Innovation Programme under Grant Agreement No. 101069471.

## Data availability

Data will be made available on request.

## References


[1] J. Vollbrecht, J.D. Carlson, L. Blunk, D.B.D. Aboba, H. Levkowetz, Extensible Authentication Protocol (EAP), in: Request for Comments, (3748) RFC, 2004, http://dx.doi.org/10.17487/RFC3748, RFC 3748, URL https://www.rfc-editor.org/info/rfc3748.

[2] R. Dantu, G. Clothier, A. Atri, EAP methods for wireless networks, Comput. Stand. Interfaces 29 (3) (2007) 289–301, http://dx.doi.org/10.1016/j.csi.2006.04.001, URL https://www.sciencedirect.com/science/article/pii/S092054890600050X.

[3] D. Simon, R. Hurst, D.B.D. Aboba, The EAP-TLS Authentication Protocol, in: Request for Comments, RFC, 2008, http://dx.doi.org/10.17487/RFC5216, URL https://www.rfc-editor.org/info/rfc5216, RFC 5216.

[4] J.P. Mattsson, M. Sethi, EAP-TLS 1.3: Using the Extensible Authentication Protocol with TLS 1.3, in: Request for Comments, RFC Editor, 2022, http://dx.doi.org/10.17487/RFC9190, URL https://www.rfc-editor.org/info/rfc9190, RFC 9190.

[5] F. Bersani, H. Tschofenig, The EAP-PSK Protocol: A Pre-Shared Key Extensible Authentication Protocol (EAP) Method, in: Request for Comments, RFC Editor, 2007, http://dx.doi.org/10.17487/RFC4764, URL https://www.rfc-editor.org/info/rfc4764, RFC 4764.

[6] J. Arkko, V. Lehtovirta, V. Torvinen, P. Eronen, Improved Extensible Authentication Protocol Method for 3GPP Mobile Network Authentication and Key Agreement (EAP-AKA'), in: Request for Comments, RFC, 2021, http://dx.doi.org/10.17487/RFC9048, URL https://www.rfc-editor.org/info/rfc9048, RFC 9048.

[7] C.S. Alliance, Zigbee IP Specification, Connectivity Standards Alliance, 2013, https://csa-iot.org/all-solutions/zigbee/.

[8] IEEE, IEEE standard for transport of key management protocol (KMP) datagrams, 2022, http://dx.doi.org/10.1109/IEEESTD.2022.9690134, IEEE Std 802.15.9-2021 (Revision of IEEE Std 802.15-9-2016).

[9] D. Forsberg, B. Patil, A.E. Yegin, Y. Ohba, H. Tschofenig, Protocol for Carrying Authentication for Network Access (PANA), in: Request for Comments, RFC, 2008, http://dx.doi.org/10.17487/RFC5191, URL https://www.rfc-editor.org/info/rfc5191, RFC 5191.

[10] C. Kaufman, P.E. Hoffman, Y. Nir, P. Eronen, T. Kivinen, Internet Key Exchange Protocol Version 2 (IKEv2), in: Request for Comments, RFC, 2014, http://dx.doi.org/10.17487/RFC7296, URL https://www.rfc-editor.org/info/rfc7296, RFC 7296.

[11] S. Hartman, J. Howlett, A GSS-API Mechanism for the Extensible Authentication Protocol, in: Request for Comments, RFC, 2013, http://dx.doi.org/10.17487/RFC7055, URL https://www.rfc-editor.org/info/rfc7055, RFC 7055.

[12] IEEE, IEEE standard for local and metropolitan area networks–port-based network access control, 2020, http://dx.doi.org/10.1109/IEEESTD.2020.9018454, IEEE Std 802.1X-2020 (Revision of IEEE Std 802.1X-2010 Incorporating IEEE Std 802.1Xbx-2014 and IEEE Std 802.1Xck-2018).

[13] IEEE, IEEE standard for local and metropolitan area networks-media access control (MAC) security, 2018, http://dx.doi.org/10.1109/IEEESTD.2018.8585421, IEEE Std 802.1AE-2018 (Revision of IEEE Std 802.1AE-2006).

[14] Cisco Systems, MACsec Using EAP-TLS Authentication, Cisco Systems, 2022, URL https://www.cisco.com/c/en/us/td/docs/iosxr/ncs5500/security/76x/b-system-security-cg-ncs5500-76x/macsec-using-eap-tls-authentication.pdf, (Accessed 30 December 2024).

[15] IETF EMU Working Group, Extensible Authentication Protocol Method Update (EMU) Working Group, Internet Engineering Task Force (IETF), 2006, Accessed 11 2024, https://datatracker.ietf.org/wg/emu/about/.

[16] G. Selander, J.P. Mattsson, F. Palombini, Ephemeral diffie-hellman over COSE (EDHOC), in: Request for Comments, RFC, 2024, http://dx.doi.org/10.17487/RFC9528, URL https://www.rfc-editor.org/info/rfc9528, RFC 9528.

[17] J. Schaad, CBOR object signing and encryption (COSE): Structures and process, Request for Comments, RFC, 2022, http://dx.doi.org/10.17487/RFC9052, URL https://www.rfc-editor.org/info/rfc9052, RFC 9052.

[18] G. Selander, J.P. Mattsson, M. Serafin, M. Tiloca, M. Vučinić, Traces of ephemeral diffie-hellman over COSE (EDHOC), in: Request for Comments, RFC, 2024, http://dx.doi.org/10.17487/RFC9529, URL https://www.rfc-editor.org/info/rfc9529, RFC 9529.

[19] IEEE, IEEE standard for information technology–telecommunications and information exchange between systems - local and metropolitan area networks–specific requirements - part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications, 2021, http://dx.doi.org/10.1109/IEEESTD.2021.9363693, IEEE Std 802.11-2020 (Revision of IEEE Std 802.11-2016).

[20] IEEE, IEEE standard for air interface for broadband wireless access systems, 2018, http://dx.doi.org/10.1109/IEEESTD.2018.8303870, IEEE Std 802.16-2017 (Revision of IEEE Std 802.16-2012).

[21] 3rd Generation Partnership Project, Technical Specification Group Services and System Aspects; 5G System; Technical Specification, Tech. Rep. TS 33.501, (TS 33.501) 3GPP, 2017, URL https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3169, (Accessed January 2025).

[22] T. Kivinen, TG 802.15.9a list of changes to the IEEE std 802.15.9, Tech. Rep., IEEE P802.15-24-0500-02-009a, 2024, URL https://mentor.ieee.org/802.15/dcn/24/15-24-0500-02-009a-list-of-changes-to-ieee-std-802-15-9.docx, (Accessed 10 January 2025).

[23] D. Stanley, J.R. Walker, D.B.D. Aboba, Extensible Authentication Protocol (EAP) Method Requirements for Wireless LANs, in: Request for Comments, RFC, 2005, http://dx.doi.org/10.17487/RFC4017, URL https://www.rfc-editor.org/info/rfc4017, RFC 4017.

[24] D. Simon, D.B.D. Aboba, P. Eronen, Extensible Authentication Protocol (EAP) Key Management Framework, in: Request for Comments, RFC, 2008, http://dx.doi.org/10.17487/RFC5247, URL https://www.rfc-editor.org/info/rfc5247, RFC 5247.

[25] D. Garcia-Carrillo, R. Marin-Lopez, A. Kandasamy, A. Pelov, A CoAP-based network access authentication service for low-Power Wide Area networks: LO-CoAP-EAP, Sensors 17 (11) (2017) http://dx.doi.org/10.3390/s17112646, URL https://www.mdpi.com/1424-8220/17/11/2646.

[26] H. Krawczyk, SIGMA: the 'SIGn-and-MAc' approach to authenticated diffie-hellman and its use in the IKE-protocols, in: CRYPTO 2003, Proceedings of the 23rd Annual International Cryptology Conference, 2003, URL https://www.iacr.org/cryptodb/archive/2003/CRYPTO/1495/1495.pdf.

[27] C. Bormann, Concise Binary Object Representation (CBOR) Sequences, in: Request for Comments, RFC, 2020, http://dx.doi.org/10.17487/RFC8742, URL https://www.rfc-editor.org/info/rfc8742, RFC 8742.

[28] R. Azarderakhsh, K.U. Järvinen, M. Mozaffari-Kermani, Efficient algorithm and architecture for elliptic curve cryptography for extremely constrained secure applications, IEEE Trans. Circuits Syst. I. Regul. Pap. 61 (4) (2014) 1144–1155, http://dx.doi.org/10.1109/TCSI.2013.2283691.

[29] S. Farrell, Low-Power Wide Area Network (LPWAN) Overview, in: Request for Comments, RFC, 2018, http://dx.doi.org/10.17487/RFC8376, URL https://www.rfc-editor.org/info/rfc8376, RFC 8376.

[30] J.P. Mattsson, F. Palombini, M. Vučinić, Comparison of CoAP Security Protocols, IETF, 2024, URL https://datatracker.ietf.org/doc/draft-ietf-iotops-security-protocol-comparison/, Work in Progress, Internet-Draft draft-ietf-iotops-security-protocol-comparison-07.

[31] D. Garcia-Carrillo, R. Marin-Lopez, G. Selander, J.P. Mattsson, Using the Extensible Authentication Protocol with Ephemeral Diffie-Hellman over COSE (EDHOC), Internet Engineering Task Force, 2024, Internet-Draft draft-ietf-emu-eap-edhoc-00 URL https://datatracker.ietf.org/doc/draft-ietf-emu-eap-edhoc/00/, Work in Progress.

[32] IEEE, IEEE standard for low-rate wireless networks, 2020, http://dx.doi.org/10.1109/IEEESTD.2020.9144691, IEEE Std 802.15.4-2020 (Revision of IEEE Std 802.15.4-2015).

[33] E. López Pérez, G. Selander, J.P. Mattsson, T. Watteyne, M. Vučinić, EDHOC is a new security handshake standard: An overview of security analysis, Comput. (2024) URL https://inria.hal.science/hal-04635881.

[34] S.A. Vanstone, A.J. Menezes, J. Katz, P.C.V. Oorschot, Handbook of Applied Cryptography, CRC Press, Boca Raton, FL, USA, 1996.

[35] A.C. Canto, J. Kaur, M.M. Kermani, R. Azarderakhsh, Algorithmic security is insufficient: A comprehensive survey on implementation attacks haunting post-quantum security, 2023, arXiv:2305.13544.

[36] National Institute of Standards and Technology (NIST), Post-quantum cryptography FAQs, 2023, https://csrc.nist.gov/Projects/post-quantum-cryptography/faqs, (Accessed 14 April 2025).

[37] A. Cintas-Canto, M. Mozaffari-Kermani, R. Azarderakhsh, Reliable code-based post-quantum cryptographic algorithms through fault detection on FPGA, in: 2023 IEEE Nordic Circuits and Systems Conference, NorCAS, 2023, pp. 1–5, http://dx.doi.org/10.1109/NorCAS58970.2023.10305475.

[38] IETF LAKE Working Group, Hackathon on lightweight IoT security, 2024, https://parishackathon.lakewg.org/.

[39] Hostapd Project, Hostap/WPA supplicant website, 2009, https://w1.fi/, (Accessed November 2024).

[40] FreeRADIUS Project, FreeRADIUS website, 1999, https://freeradius.org/, (Accessed November 2024).

[41] Eriptic, uOSCORE-uEDHOC official GitHub, 2021, https://github.com/eriptic/uoscore-uedhoc, (Accessed November 2024).

[42] MbedTLS Project, MbedTLS official GitHub, 2014, https://github.com/Mbed-TLS/mbedtls, (Accessed 2024 November).

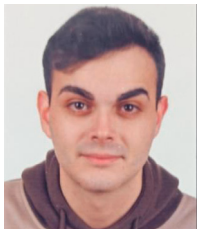
[43] Davy Landman, Compact25519 official GitHub, 2014, https://github.com/DavyLandman/compact25519, (Accessed November 2024).

[44] Internet Assigned Numbers Authority (IANA), Ephemeral diffie-hellman over COSE (EDHOC) parameters: Cipher suites, 2024, https://www.iana.org/assignments/edhoc/edhoc.xhtml#edhoc-cipher-suites, (Accessed November 2024).

[45] F. Lopez-Gomez, G. Lopez-Millan, R. Marin-Perez, EAP-EDHOC testbed, 2024, URL https://gitlab.com/franciscolopezg/eap-edhoc-testbed.

[46] J.P. Mattsson, G. Selander, B. Smeets, E. Thormarker, Constrained radio networks, small ciphertexts, signatures, and non-interactive key exchange, in: Fourth PQC Standardization Conference, 2022, URL https://csrc.nist.gov/csrc/media/Events/2022/fourth-pqc-standardization-conference/documents/papers/constrained-radio-networks-pqc2022.pdf.

[47] E. Lopez-Perez, G. Selander, J.P. Mattsson, R. Marin-Lopez, EDHOC Authenticated with Pre-Shared Keys (PSK), in: Internet-Draft draft-ietf-lake-edhoc-psk-03, Internet Engineering Task Force, 2025, URL https://datatracker.ietf.org/doc/draft-ietf-lake-edhoc-psk/03/, Work in Progress.
[48] G. Fedrecheski, M. Vučinić, T. Watteyne, Performance comparison of EDHOC and DTLS 1.3 in internet-of-things environments, in: IEEE Wireless Communications and Networking Conference, Dubai, United Arab Emirates, 2024, URL https://hal.science/hal-04382397.
[49] S. Hristozov, M. Huber, L. Xu, J. Fietz, M. Liess, G. Sigl, The cost of OSCORE and EDHOC for constrained devices, 2021, CoRR abs/2103.13832, arXiv:2103.13832, URL https://arxiv.org/abs/2103.13832.
[50] L.P. Fraile, A. Fournaris, C. Koulamas, Design and performance evaluation of an embedded EDHOC module, in: 2021 10th Mediterranean Conference on Embedded Computing, MECO, 2021, pp. 1–6, http://dx.doi.org/10.1109/MECO52532.2021.9459724.
[51] E. Ingles-Sanchez, D. Garcia-Carrillo, G.Z. Papadopoulos, N. Montavont, A.F. Skarmeta Gómez, Adaptation of EAP-NOOB method for LoRaWAN with LO-CoAP-EAP and CBOR, in: 2020 Global Internet of Things Summit, GIoTS, 2020, pp. 1–6, http://dx.doi.org/10.1109/GIOTS49054.2020.9119629.



**Francisco Lopez-Gomez**: Francisco Lopez-Gomez holds a bachelor's degree in Informatics Engineering and a Master's Degree in Cybersecurity from the University of Murcia (Spain), where he is currently conducting research. His research interests include network security, Software-Defined Networking (SDN), and the Internet of Things (IoT), among others.

**Rafael Marin-Lopez**: Rafael Marin-Lopez is a full -time assistant lecturer in the Department Information and Communications Engineering at the University of Murcia (Spain). Additionally, he is collaborating actively in IETF, especially in I2NSF and ACE Working Groups. He is co-author of RFC 9061, RFC 7499 and other standards. His main research interests include network access authentication, key distribution and security in different type of networks such as SDN, IoT and mobile networks.

**Gabriel Lopez-Millan**: Gabriel Lopez-Millan is a full -time assistant professor in the Department of Information and Communications Engineering of the University of Murcia. He has collaborated actively in IETF WG such as ABFAB and I2NSF. He is co-author of RFCs 9061 and 7499. His research interests include network security, SDN/NFV, PKI, identity management, authentication and authorization. He received his Ph.D. in computer science from the University of Murcia.

**Dan Garcia-Carrillo**: Dan Garcia-Carrillo received the Ph.D. degree in Computer Science from the University of Murcia, in 2018, under an Industrial Doctorate Grant. He is involved in the IETF in several standardization efforts regarding bootstrapping and security in the context of the Internet of Things (IoT). He is currently an Associate Professor in the Department of Computer Science of the University of Oviedo, Spain. He continues his research on new protocols and proposals to secure IoT in constrained networks such as 6LoWPAN, 6TiSCH, LP-WAN and, 5G. He has collaborated in EU projects, such as Sociotal, SMARTIE, ANASTACIA, and Plug-N-Harvest.

**John Preuß Mattsson**: John Preuß Mattsson is a senior security expert in cryptographic algorithms and security protocols at Ericsson Research, Stockholm, Sweden.

**Göran Selander**: Göran Selander is a principal security researcher at Ericsson Research Stockholm, Sweden