Architecture of security association establishment based on bootstrapping technologies for enabling secure IoT infrastructures

Salvador Pérez, Dan Garcia-Carrillo, Rafael Marín-López, José L. Hernández-Ramos, Rafael Marín-Pérez, Antonio F. Skarmeta

PII: S0167-739X(18)32557-3
DOI: https://doi.org/10.1016/j.future.2019.01.038
Reference: FUTURE 4731

To appear in: *Future Generation Computer Systems*

Received date : 19 October 2018
Revised date : 10 January 2019
Accepted date : 16 January 2019

Please cite this article as: S. Pérez, D. Garcia-Carrillo, R. Marín-López et al., Architecture of security association establishment based on bootstrapping technologies for enabling secure IoT infrastructures, *Future Generation Computer Systems* (2019), https://doi.org/10.1016/j.future.2019.01.038

Architecture of security association establishment based on bootstrapping technologies for enabling secure IoT infrastructures

Salvador Pérez[a,*], Dan Garcia-Carrillo[b], Rafael Marín López[a], José L. Hernández-Ramos[c], Rafael Marín-Pérez[b], Antonio F. Skarmeta[a]

[a] *Department of Information and Communication Engineering, Computer Science Faculty, University of Murcia, Spain*
[b] *Odin Solutions, Murcia, Spain*
[c] *European Commission, Joint Research Centre, Ispra 21027, Italy*

Abstract

The next generation of IoT scenarios must consider security aspects as a first class component. As a core aspect, key management is crucial for the establishment of security associations between endpoints. According to it, in this work we propose a novel architecture of security association establishment based on bootstrapping technologies in order to manage the life-cycle of cryptographic keys in IoT. Based on our previous work, we propose a key derivation process by using a lightweight bootstrapping mechanism specifically designed for IoT. Then, the derived cryptographic material is used as an authentication credential of the EDHOC protocol, which represents a standardisation effort for key agreement in IoT. EDHOC is an application layer alternative to the DTLS handshake, in order to provide end-to-end security properties even in the presence of intermediate entities, such as proxies. Evaluation results prove the feasibility of our approach, which represents one of the first efforts to consider application layer security approaches for the IoT.

Keywords: Internet of Things; Security Management; Bootstrapping; EDHOC

1. Introduction

Security aspects represent an extremely limiting factor for the deployment of IoT solutions [1]. As a core aspect, *key management* embraces the activities to handle the entire lifecycle of cryptographic keys including their generation, storage and establishment [2]. Like in the current Internet, these cryptographic keys need to be employed by IoT endpoints to establish security associations for data protection. Indeed, in some cases such endpoints could manage particularly

*Corresponding author
Email address: salvador.p.f@um.es (Salvador Pérez)

sensitive data (e.g., in eHealth scenarios [3]). Consequently, the lack of a suitable key management mechanism could harm users' privacy, specially if these data are intended to be further analysed or correlated [4] [5]. However, the realization of key management approaches for IoT must overcome scalability, flexibility and performance issues, specially in the case of resource-constrained devices. Furthermore, typical transport layer approaches (i.e., based on DTLS [6]) are no able to provide end-to-end security in the presence of intermediate entities, such as proxies and brokers.

To mitigate this issue, recent standardization efforts are focused on the application layer security into IoT constrained scenarios. In particular, the Ephemeral Diffie-Hellman Over COSE (EDHOC) [7] represents an authenticated and lightweight application-layer key management approach that provides perfect forward secrecy (PFS) [8]. EDHOC represents an ongoing initiative that is being evolved within the scope of the Authentication and Authorization for Constrained Environments (ACE) WG[1] of the IETF. The approach is based on the use of CBOR Object Signing and Encryption (COSE) [9], which is a compacted evolution of JSON Object Signing and Encryption (JOSE)[10], so that message overhead is reduced. EDHOC provides a high level of flexibility by enabling different authentication modes: pre-shared key (PSK), raw public key (RPK) and certificates. However, according to EDHOC specification, authentication credentials are previously established by out-of-band mechanisms, so it does not define any concrete approach to address this aspect.

This work aims to fill this gap through the integration with bootstrapping technologies, so that EDHOC authentication credentials are derived from the cryptographic material generated by the bootstrapping process. In the context of IoT, the bootstrapping is usually referred as the initial process by which a device securely joins a network [11]. In particular, we consider the integration with LO-CoAP-EAP[12], which provides a lightweight bootstrapping service that is specifically designed for IoT. LO-CoAP-EAP makes use of the Constrained Application Protocol (CoAP) [13] to transport Extensible Authentication Protocol (EAP) messages [14]. Consequently, it leverages the use of Authentication, Authorization and Accounting (AAA) infrastructures [15] for scalability reasons, while CoAP provides a more lightweight approach as a EAP lower layer protocol compared to well-known protocols, such as the Protocol for carrying Authentication for Network Access (PANA) [16]. Indeed, it should be noted that LO-CoAP-EAP is derived from CoAP-EAP [17], which represents an ongoing standardization effort. Based on the integration between LO-CoAP-EAP and EDHOC, the main contributions of this paper are:

- Design and implementation of an extension for the LO-CoAP-EAP protocol that enables to establish EDHOC credentials based on the use of PSK, RPK or certificates authentication modes.
- Deployment of the integration of LO-CoAP-EAP and EDHOC on real IoT

[1]https://datatracker.ietf.org/wg/ace/about/

devices.

- Performance evaluation of the proposed approach by considering different practical aspects, such as message size, runtime, number of hops between the endpoints and link loss ratio.
- Comparison of our proposal by considering another state-of-the-art bootstrapping protocol, specifically, PANA.
- Integration of the proposed approach's components into a Smart Building scenario.

The remainder of the paper is organized as follows. Section 2 describes existing proposals related to the establishment of security associations in IoT scenarios. Then, Section 3 presents EDHOC and LO-CoAP-EAP as the main building blocks of our work. Section 4 describes the proposed architecture, as well as the associated design considerations. Section 5 provides a detailed description of the proposed approach, which is instantiated in a smart building scenario in Section 6. Then, an evaluation of our proposal is given in Section 7. Finally, Section 8 concludes the paper with an outlook of our future work in this area.

2. Related Work

During last years, security issues have been widely considered as the main obstacle for the adoption of IoT-enabled services. Indeed, the development of such services still has to cope with security challenges related to authentication, authorization, access control, confidentiality or privacy aspects [18] [19]. Furthermore, the IoT ecosystem is being currently enhanced through the integration of emerging technologies, such as 5G, which is intended to address the demanding communication requirements of new IoT applications [20]. It means that security challenges can be aggravated due to the pervasiveness and ubiquity enabled by these technologies. Communication technologies can be leveraged by upper layer protocols for information exchange between IoT endpoints. In this direction, CoAP [13] is widely recognized as the main application-layer protocol for IoT due to its simplicity and very low overhead for constrained environments. To secure IoT communications, CoAP specifies a binding to the Datagram Transport Layer Security (DTLS) protocol [21]. According to it, the scientific literature reports works that propose different modifications of DTLS to improve its adaptation to constrained networks and devices. Specifically, [22] describes a two-way authentication security scheme for IoT, which is mainly based on the DTLS protocol. The implemented scheme uses a standard communication stack based on 6LoWPAN [23] that is tested on a real hardware platform. Similarly, [24] proposes a lightweight CoAP-DTLS scheme (*Lithe*) by using different 6LoWPAN header compression mechanisms. The aim is to reduce the message overhead to avoid 6LoWPAN fragmentation, while DTLS security properties are not compromised. Additionally, [25] presents a preliminary overhead estimation for the certificate-based DTLS handshake and details three design ideas, which are based on pre-validation, session resumption and handshake delegation, in order to reduce such overhead. Likewise, in [26], authors provide a communication architecture fully based on DTLS named *SecureSense* to secure communication in cloud-connected IoT scenarios, considering the different security modes of CoAP, that is, PSK, RPK and certificates. However, due to CoAP communications in IoT scenarios are usually performed through proxies for improving scalability and efficiency [27], DTLS cannot provide end-to-end security. The main reason is a DTLS connection has to be terminated at each proxy, so that this entity could get access to specific headers to provide its intended functionality. This issue can be mitigated through the approach proposed by [28], which is intended to make DTLS feasible in multi-hop scenarios. However, in addition to require new functionality to the intermediate entities to forward messages, such entities must be previously authenticated.

Unlike DTLS-based proposals, application layer security emerges as a solution to guarantee the end-to-end security, by providing a flexible alternative that is independent of the protocols conveying the application layer payloads. Based on it, there are two novel specifications defined by the IETF ACE WG that employ COSE [9], which, in turn, is based on the Concise Binary Object Representation (CBOR) [29]. On the one hand, the Object Security for Constrained RESTful Environments (OSCORE) [30] is a protocol intended to protect CoAP messages, that is, to ensure confidentiality and integrity of exchanged data. On the other hand, EDHOC [7] is a lightweight key exchange protocol that aims to establish shared symmetric keys between two endpoints. It should be noted that both approaches are complementary, and they can be considered as an application layer alternative to DTLS. In particular, the DTLS handshake and DTLS application data protocols can be mapped to EDHOC and OSCORE, respectively. Even though it is a recent proposal, EDHOC has attracted the interest due to its flexibility to be integrated in different scenarios (e.g.,t for the establishment of IPsec Security Associations (SA) [31]), and lightness, so it can be used on resource-constrained devices and networks through the use of CBOR and COSE standards. Indeed, in our previous work, we propose the use of EDHOC to derive and update LoRaWAN cryptographic material [32], in which EDHOC overhead is compared with DTLS handshake. Furthermore, [33] provides a new authorization and authentication framework for the IoT based on OAuth [34] and a EDHOC-based key agreement approach. It should be noted that the use of OAuth is widely considered in the scope of the ACE WG [35]. However, these proposals do not consider the establishment of credentials that are employed for authentication purposes during the EDHOC message exchange.

Based on it, there is a real need to consider additional approaches to complement security association protocols (such as EDHOC), in order to set the corresponding credentials that are employed to establish security associations. While recent proposals [36] partially address this issue, we focus on the integration with bootstrapping approaches to provide a more comprehensive solution to manage the lifecycle of cryptographic material. Towards this end, the device is authenticated in order to receive the required cryptographic material to become a trusted party in a security domain [11]. Different works about bootstrapping consider the use of pre-established shared key material and running a security association protocol such as DTLS [37, 38, 39, 40, 41]. Furthermore, [42] aims to make the use of PSK-based authentication scalable in IoT deployments. For this, they introduce a trusted third party, so that key material is generated to run the DTLS protocol. From the standardization point of view, the bootstrapping is typically performed by using protocols such as PANA, which is employed in the Zigbee IP standard [43] and proposed in works such as [44, 45, 46, 47]. In this case, PANA is used to transport EAP messages (i.e., it acts as an EAP lower layer). However, one of the main issues of PANA is that it was not designed with the constraints of IoT in mind, as demonstrated by [48]. Consequently, there is a need to design more lightweight bootstrapping approaches, while flexibility and scalability associated to EAP can be still provided.

Based on our previous work [12], in this paper we consider the use of Low Overhead CoAP-EAP (LO-CoAP-EAP) as a bootstrapping solution specifically designed for IoT. Then, the cryptographic material generated from this process is used to derive or exchange the authentication credentials that are employed during the EDHOC message exchange. Therefore, the resulting approach leverages the lightness of technologies, such as CoAP or COSE, as well as the end-to-end security properties that are provided by application layer security approaches.

3. Preliminaries

As already mentioned, our proposal is based on the integration of LO-CoAP-EAP and EDHOC. Consequently, this section aims to provide an overview of both technologies.

3.1. EDHOC protocol

EDHOC [7] is a lightweight authenticated key exchange protocol that enables to establish a cryptographic key between two entities. To this end, EDHOC implements the Elliptic Curve Diffie-Hellman algorithm with ephemeral keys (ECDHE) [49] by which both entities must generate a new ephemeral key pair every time they launch this protocol. Therefore, EDHOC also provides the perfect forward secrecy property [8]. Additionally, EDHOC supports the same authentication modes as DTLS (i.e., PSK, RPK, and certificates). Hence, the key generation process remains independent with respect of the selected authentication mode. Moreover, EDHOC defines a three-message exchange in order to negotiate certain security parameters to fulfill its functionality. These messages are encoded following the CBOR representation [29] and protected by the COSE standard [9]. This way, a minimum message size is ensured in contrast to other JSON-based representation formats (such as JWS [50] and JWE [51]) and, therefore, network overhead is reduced.

In spite of the advantages of EDHOC, nowadays, DTLS [21] is widely considered as the main alternative to protect communications between two endpoints in IoT constrained scenarios [13]. However, the common presence of intermediate entities, such as proxies, implies that DTLS can only provide hop-by-hop security [27], as previously pointed out. Unlike DTLS, EDHOC allows to selectively protect specific fields of the message through the use of COSE objects. This way, end-to-end security is ensured, while at the same time intermediate entities are still able to access information required to carry out their functionality. According to it, and taking into account the main aims of this work, we have included EDHOC as part of our proposal in order to establish security associations between two endpoints in IoT infrastructures. However, the EDHOC specification [7] does not define any mechanism to establish the authentication credentials that are required to perform this key protocol exchange. For this previous step, we propose the use of LO-CoAP-EAP, which is further described below.

3.2. LO-CoAP-EAP

LO-CoAP-EAP (Low Overhead CoAP-EAP) [illegible] is a bootstrapping service that is implemented by CoAP as EAP lower layer protocol. Indeed, LO-COAP-EAP is built on three pillars: CoAP [13], EAP [32] and AAA [15]. These technologies provide a unique set of properties to LO-COAP-EAP. On the one hand, because it is built on top of CoAP, LO-CoAP-EAP provides the integration of a smart object's bootstrapping process as a CoAP service. It also provides a link-layer independent solution since CoAP runs on top of UDP/IP. Furthermore, CoAP is the standard protocol for web transfer between IoT devices; consequently, unlike PANA-based proposals, LO-CoAP-EAP does not require to add any specific technology for performing the bootstrapping, so devices' burden can be alleviated. On the other hand, with EAP, LO-COAP-EAP provides a flexible approach by enabling a wide variety of authentication methods to be chosen according to the needs of the IoT deployment, or the organizations' policies involved on it.

Bootstrapping an IoT device with LO-CoAP-EAP also provides the necessary key material, so that the device can be integrated as a trustworthy entity in the domain. Such key material is derived according to the EAP Key Management Framework (EAP KMF)[14]. The EAP KMF allows the derivation of key material that is used for different purposes, depending on the protocol; for instance, IEEE 802.15.9 uses EAP carried over PANA to derive key material to protect the link-layer [53]. Furthermore, it can be also used to generate the key material needed to run different Security Association Protocols (SAP) [48, 12]. Finally, the use of an AAA infrastructure provides advanced features such as identity federation, in order to make multi-domain deployments more scalable.

LO-CoAP-EAP has been designed to cope with constrained devices and Low power and Lossy Networks (LLNs), providing a solution to bootstrap a wide variety of IoT devices. Based on it, we select this protocol as the bootstrapping mechanism to derive the authentication credentials required by EDHOC. Next sections provides a detailed description of the resulting architecture.

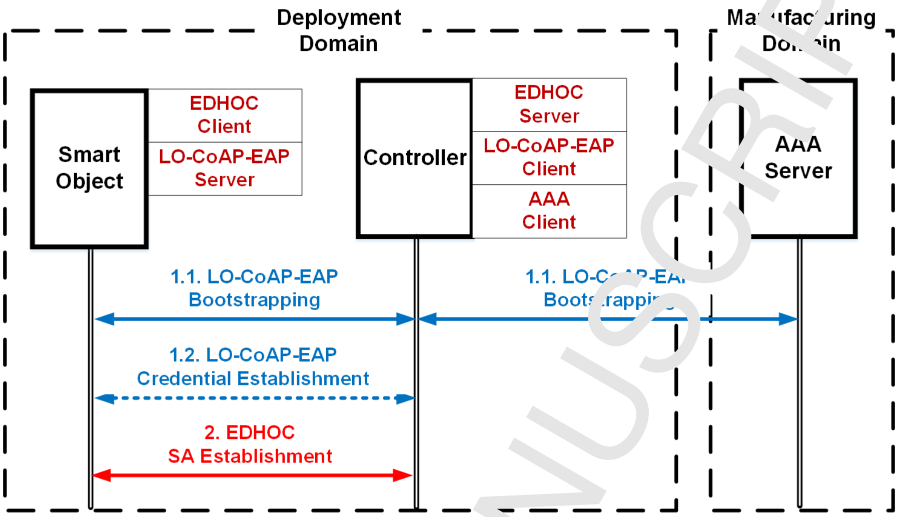


Figure 1: Proposed architecture and interactions overview

4. Security associations establishment

As already described, we consider EDHOC for establishment of end-to-end security associations (SAs) between two IoT endpoints. However, such protocol does not specify how these entities establish the required credentials for their authentication, that is, the pre-shared key, public keys or certificates. According to it, we propose the usage of LO-CoAP EAP as a bootstrapping protocol to establish such credentials.

4.1. Proposed architecture

Figure 1 shows an overview of the proposed architecture, which integrates the EDHOC protocol after performing LO-CoAP-EAP bootstrapping. For this, we consider three entities:

- *Smart Object.* It represents an IoT device that aims to join a certain security domain. Note that the Smart Object acts as a *LO-CoAP-EAP Server* to carry out the functionality of this protocol, except in the first message, where such entity plays the *LO-CoAP-EAP Client* role, as specified in [12]. Similarly, the Smart Object also acts an *EDHOC Client.*

- *Controller.* It is the entity in charge of managing a network security domain. The Controller can act as an *EDHOC Server* for the EDHOC protocol, as well as a *CoAP-EAP Server* and *AAA Client* for handling the bootstrapping process through LO-CoAP-EAP.

- *AAA Server.* It represents the AAA server of the Smart Object Identity Provider (e.g., the manufacturer organization).

In order to highlight the advantages provided by AAA, we consider two different domains; on the one hand, the *Manufacturing Domain* is supposed to represent the domain in which the Smart Object was manufactured. On the other hand, the *Deployment Domain* represents the domain in which the Smart Object will be operating after a successful bootstrapping process. This represents a common scenario in IoT deployments, in which the use of AAA enables a more scalable approach, since the *Deployment Domain* does not require any previous knowledge about new devices joining the domain. Furthermore, Figure 1 also depicts the two main phases identified in our proposal. The first phase is split into *LO-CoAP-EAP Bootstrapping*, which aims to provide certain cryptographic material to the Smart Object and Controller, and *LO-CoAP-EAP Credential Establishment* that is intended to set the EDHOC authentication credentials. Moreover, the second phase consists of the *EDHOC SA Establishment* in which the EDHOC protocol is carried out. Additionally, it should be noted that, while intermediate entities are not included between Smart Object and Controller for the sake of clarity, proxies are usually deployed in IoT scenarios for efficiency and scalability reasons [27].

During the *LO-CoAP-EAP Bootstrapping*, the Smart Object is authenticated and joins the *Deployment Domain* as a trustworthy entity. The Controller handles the authentication by interacting with the AAA Server. This last entity authenticates the Smart Object and notifies the Controller when the authentication has been correctly completed. The AAA Server also sends authorization information about the Smart Object to the Controller in the final step of the bootstrapping. As a consequence of a successful bootstrapping process, certain cryptographic material is derived and shared between the Smart Object and the Controller. The communication between the Smart Object and AAA Server is done through EAP. However, it should be noted that the transport of EAP messages between the AAA Server and the Controller is done through an AAA protocol (i.e., RADIUS in this case), and LO-CoAP-EAP in the case of the communication between the Smart Object and the Controller.

When the *LO-CoAP-EAP Bootstrapping* is successfully finished, both the Smart Object and the Controller initiate the *LO-CoAP-EAP Credential Establishment* process, in order to establish authentication credentials that will be used by EDHOC in subsequent interactions. Towards this end, such entities make use of cryptographic material previously derived during the bootstrapping process. As described in Section 5, depending on the authentication mode (i.e., PSK, RPK or certificates), the required operations and interactions to carry out this credential establishment will differ.

Once the corresponding credentials are established, the Smart Object (acting as the *EDHOC Client*) and the Controller (acting as the *EDHOC Server*) start the *EDHOC SA Establishment*. Then, these entities are able to compute a shared symmetric key, which could be used to enable other application-layer security solutions intended to protect future communications between the Smart Object and the Controller (e.g., OSCORE [30]).

4.2. Security associations updating and key management

According to the proposed architecture (Figure 1), we consider LO-CoAP-EAP as enabler of EDHOC. Particularly, after the LO-CoAP-EAP protocol has finalized with a successful EAP authentication, the Controller and the Smart Object establish a SA with EDHOC by using the cryptographic material derived from such authentication. However, it should be pointed out that such SA (and the associated cryptographic material) has a certain lifetime and, consequently, it should be updated from time to time [54, 55]. Under our proposal, such SA updating process only requires to relaunch the EDHOC protocol (fase 2), rather than starting the LO-CoAP-EAP protocol again. Accordingly, the bootstrapping and credential establishment processes (phase 1) are only performed once, specifically, when the Smart Object needs to join the *Deployment Domain*. This way, network overhead is reduced, as well as the PFS property is also assured due to the use of ephemeral cryptographic material in EDHOC. Note that employing EDHOC as security associations updating mechanism has been proposed in our previous work for LoRaWAN networks [56].

According to the different authentication modes, we consider two alternatives: EDHOC with symmetric keys (PSKs), or with asymmetric keys (RPKs and certificates). Particularly, when EDHOC authentication is based on a symmetric key pre-shared between the Smart Object and the Controller, both entities carry out a key derivation process that employs cryptographic material obtained from LO-CoAP-EAP to compute such pre-shared key. With this pre-shared key, EDHOC runs in PSK mode to authenticate both entities. In this case, it should be noted that this process does not required additional functionality, since the EAP-KMF already provides a way of deriving key material for different purposes [14]. In case of authentication based on asymmetric keys (i.e., RPK or certificates), both entities previously require to exchange their corresponding public keys or certificates. This exchange is an added functionality that is encompassed within the LO-CoAP-EAP service, and both entities make use of cryptographic material gained from the LO-CoAP-EAP bootstrapping to authenticate such exchange. It should be pointed out that, unlike PSK, in this case the Controller could release public keys or certificates associated to different Smart Objects that are within the same domain, so that they are able to directly establish security associations each other, without the Controller's participation. Such operation mode is out of scope of this work, although it represents part of our future work.

4.3. Transporting EDHOC messages

In order to transport EDHOC messages between the Smart Object and the Controller, a communication protocol is required. In this sense, we have selected the CoAP [15] protocol by considering different aspects:

- CoAP is proposed by the IETF as the standard application layer protocol for IoT scenarios.

- The EDHOC specification (version 8) [7] suggests the usage of CoAP to transport the EDHOC messages, which are embedded as payload of the corresponding CoAP request/response.
- The bootstrapping protocol of our proposal (i.e., LO-CoAP-EAP) employs CoAP to transport EAP messages. Therefore, by reusing this protocol also for transporting EDHOC messages instead of other communication protocols, we reduce memory consumption in Smart Objects.

According to it, Figure 2 shows an overview of the EDHOC message exchange over CoAP between the Smart Object (*EDHOC Client*) and the Controller (*EDHOC Server*). Particularly, the *Message-1* is included in a CoAP POST request, which is sent by the Smart Object to start EDHOC. It should be pointed out that this POST request is marked as *confirmable* (CON), so that the Smart Object expects the corresponding *acknowledgement* (ACK) from the Controller. Otherwise, the former will carry out a retransmission of such request. Regarding the *Message-2*, it is contained in a CoAP 2.04 Changed response, which is carried in the ACK that acknowledges the first POST request (*piggybacked* response). Similar to the *Message-1*, the *Message-3* is sent by the Smart Object to the Controller into a new *confirmable* CoAP POST request. At this point it should be noted that, while the EDHOC specification defines a three-message exchange, a new ACK is required to acknowledge this last request when CoAP is employed to transport the EDHOC messages. It aims to ensure the successful completion of the protocol. In case the whole exchange succeeds, the response will be empty; otherwise, it will contain the corresponding EDHOC error message. Once the protocol correctly finishes, both entities are able to compute a shared symmetric key by employing cryptographic material exchanged by the EDHOC messages, as detailed in next section.

5. Interactions description

By considering the proposed architecture, in this section we describe the interactions defined in our proposal between the Smart Object and the Controller in order to establish a security association.

5.1. LO-CoAP-EAP interactions

As already mentioned, during the *LO-CoAP-EAP Bootstrapping* (without handshake [16]), the Smart Object joins the security domain as a trusted entity. At this point, the Smart Object can begin its normal operation, which may involve establishing (or updating) security associations for specific services. In this section, we summarize the LO-CoAP-EAP message exchange.

In the first step, the Smart Object sends an initial message to the Controller to indicate that it aims to start the bootstrapping process. This message contains the identity of the Smart Object (e.g., smart-object@um.es in the example). Then, the Controller initiates the EAP exchange with the AAA Server by sending the Smart Object's identity that was received in the first message.

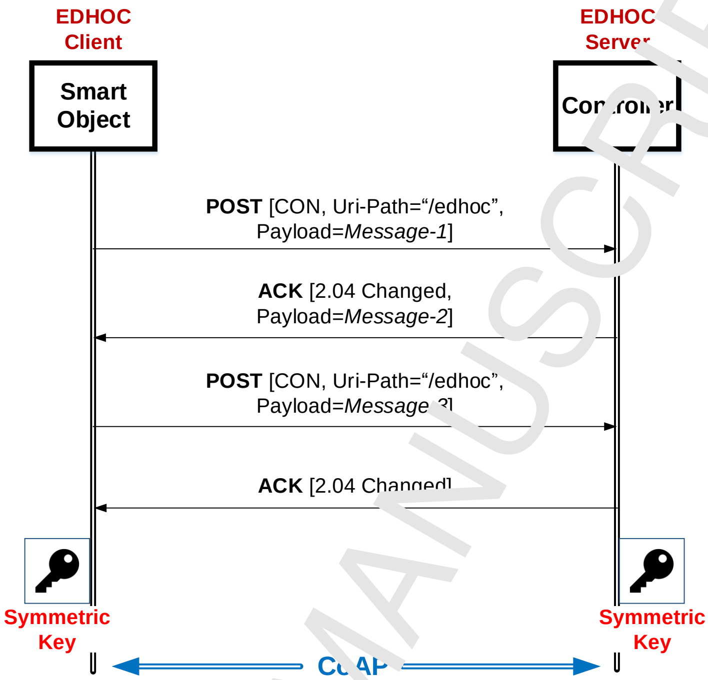


Figure 2: EDHOC message exchange over CoAP

Subsequently, the EAP exchange starts between the Smart Object and the AAA server, in which the Controller acts as a forwarder of these messages. When the EAP method has finished with a successful authentication, the AAA server sends the EAP-Success message to the Controller along with the Master Session Key (MSK) [14]. Then, the Smart Object and the Controller compute a Transient Session Key (TSK) [14], specifically, the $COAP\_PSK$ key. This key is derived from the MSK and the *nonces* exchanged during the LO-CoAP-EAP and it is used to obtain certain cryptographic material shared between both entities, as will be seen further on. The COAP_PSK is a 16-byte key that is computed with AES-CMAC PRF-128 [57] as Key Derivation Function (KDF), and it is generated as:

$$COAP\_PSK = KDF(MSK, \text{"IETF\_COAP\_PSK"} \,||\, Nonce_{SO} \,||\, Nonce_{C}, 64, length)$$

where

- MSK is the key derived from the EAP method.
- *"IETF_COAP_PSK"* is the non-NULL ASCII string without quotation marks.
- $Nonce_{SO}$ and $Nonce_{C}$ are the *nonces* exchanged in the LO-COAP-EAP protocol.

- *64* is the length of the *MSK*.
- *length* is the length of the nonces plus the ASCII string.

It should be pointed out that the last two messages of the LO-CoAP-EAP exchange include a CoAP option, called *AUTH*, which contains the CMAC of the entire message, providing integrity and authentication to the message itself. This *AUTH* option is generated by using the AES-CMAC-128 [58] as:

AUTH Option value = AES-CMAC-128(COAP_PSK, message, message_length)

where:

- *COAP_PSK* has been previously computed.
- *message* is the entire CoAP message to be protected, including the *AUTH* Option.
- *message_length* is the length of the message.

Additionally, note that the use of the *AUTH* option is restricted to LO-CoAP-EAP. Indeed, the original purpose of this option, according to our previous work [12, 48], is to confirm that the Smart Object and the Controller have properly established the *MSK*. While the use of other approaches (e.g., OSCORE) could have been also considered to this end, the *AUTH* option is chosen due to its simplicity (indeed, additional protocols/functionality are not required), and because it does not preclude the use of security associations protocols.

Once the LO-CoAP-EAP bootstrapping successfully finishes, the Smart Object and the Controller share certain cryptographic material, particularly, the *MSK* key and the *AUTH* option. This cryptographic material will allow them to establish the corresponding EDHOC authentication credentials in order to launch such key exchange protocol. Note that this credential establishment process depends on the selected EDHOC authentication mode, that is, authentication with symmetric keys or with asymmetric keys.

In case of EDHOC with authentication based on symmetric keys, the Smart Object and the Controller make use of a KDF for generating the corresponding pre-shared key (*PSK*) from the *MSK*, as shown in Figure 3 (*LO-CoAP-EAP Credential Establishment*). This *PSK* has an identifier associated (PSK_{ID}), which is generated by monotonically increasing its value. Particularly, we select the *prf+* function that is defined in [59] and recommended in [60] in order to derive such pre-shared key. It should be noted that this function employs the AES-CMAC-PRF-128 algorithm.

PSK = prf+(MSK, "IETF-EDHOC-PSK" | NULL | $Nonce_{SO}$ *|* $Nonce_{C}$*, 64, 128)*

where:

- *MSK* is the master session key from the LO-CoAP-EAP bootstrapping.

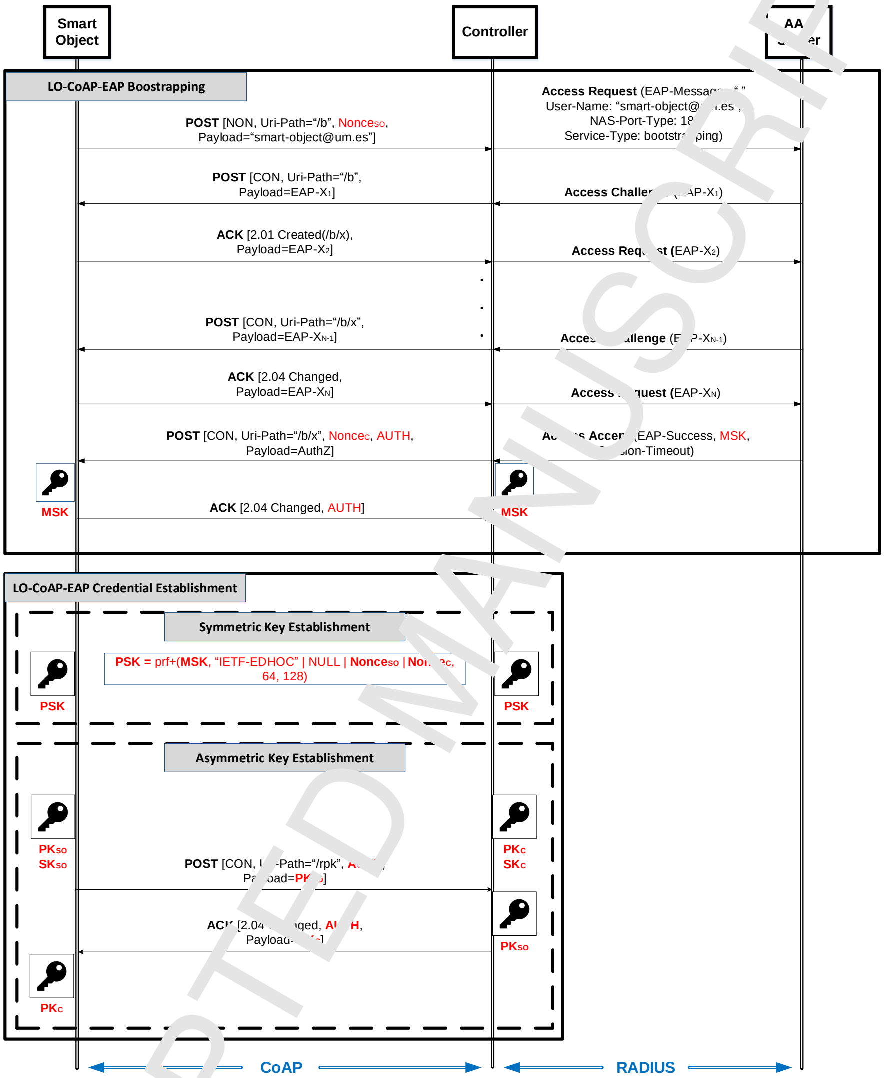


Figure 3: LO-COAP-EAP Bootstrapping and Credential Establishment message exchange

- An ASCII value formed by the concatenation of:
  - *“IETF-EDHOC-PSK”* represents a string identifying the protocol that will employ the derived key.
  - *NULL* is a null value.
  - $Nonce_{SO}$ and $Nonce_{C}$ are the *nonces* exchanged in the LO-COAP-EAP protocol.

- *64* indicates the MSK's length (in bytes).
- *16* indicates the PSK's length (in bytes).

In case of EDHOC with authentication based on asymmetric keys, both credentials (either RPK or certificates) must be securely exchanged between the Smart Object and the Controller during the *LO-CoAP-EAP Credential Establishment*. This exchange is done once the bootstrapping is completed. To exchange their RPKs or certificates, the Smart Object sends its credential to the Controller through a service called *ASYK Exchange*. Thus, the Smart Object sends a CoAP POST request to such service (Uri-Path CoAP header: */asyk*) by including the corresponding RPK or certificate. When the Controller receives this message, it knows the identity of the Smart Object, so it stores this credential associating it to the Smart Object. After that, the Controller sends its credential (RPK or certificate) to the Smart Object in the corresponding CoAP 2.04 Changed response, which is carried in the ACK that acknowledges the previous POST request (*piggybacked response*). It should be pointed out that both the request and the response are protected with the $AUTH$ option.

5.2. EDHOC interactions

Once the LO-CoAP-EAP protocol has successfully finished (*Bootstrapping* and *Credential Establishment*), the Smart Object and the Controller are enabled to initiate the EDHOC protocol in order to establish or update a security association between them. As already mentioned, the EDHOC message exchange depends on the selected authentication mode, which employs the previously established credentials, that is symmetric keys (PSKs) or asymmetric keys (RPKs or certificates). According to it, Figure 4 shows the EDHOC interactions by considering both authentication modes.

5.2.1. EDHOC message exchange authenticated with symmetric keys

When the selected authentication mode is based on symmetric keys, the Smart Object and the Controller share a PSK that has been previously derived by the *LO-CoAP-EAP Credential Establishment* process. Subsequently, in order to launch the EDHOC protocol, the Smart Object firstly generates its own ephemeral key pair (i.e., E_SK_{SO} and E_PK_{SO}) and then, it builds the *Message-1*. This message contains the following parameters:

- MSG_TYPE identifies the EDHOC *Message-1*.
- S_{SO} is a variable length session identifier for the Smart Object.
- N_{SO} is a 64-bit random nonce for the Smart Object.
- E_PK_{SO} represents the Smart Object's ephemeral public key.
- $ECDH-Curves$ indicates the set of elliptic curves for the Diffie-Hellman algorithm supported by the Smart Object.

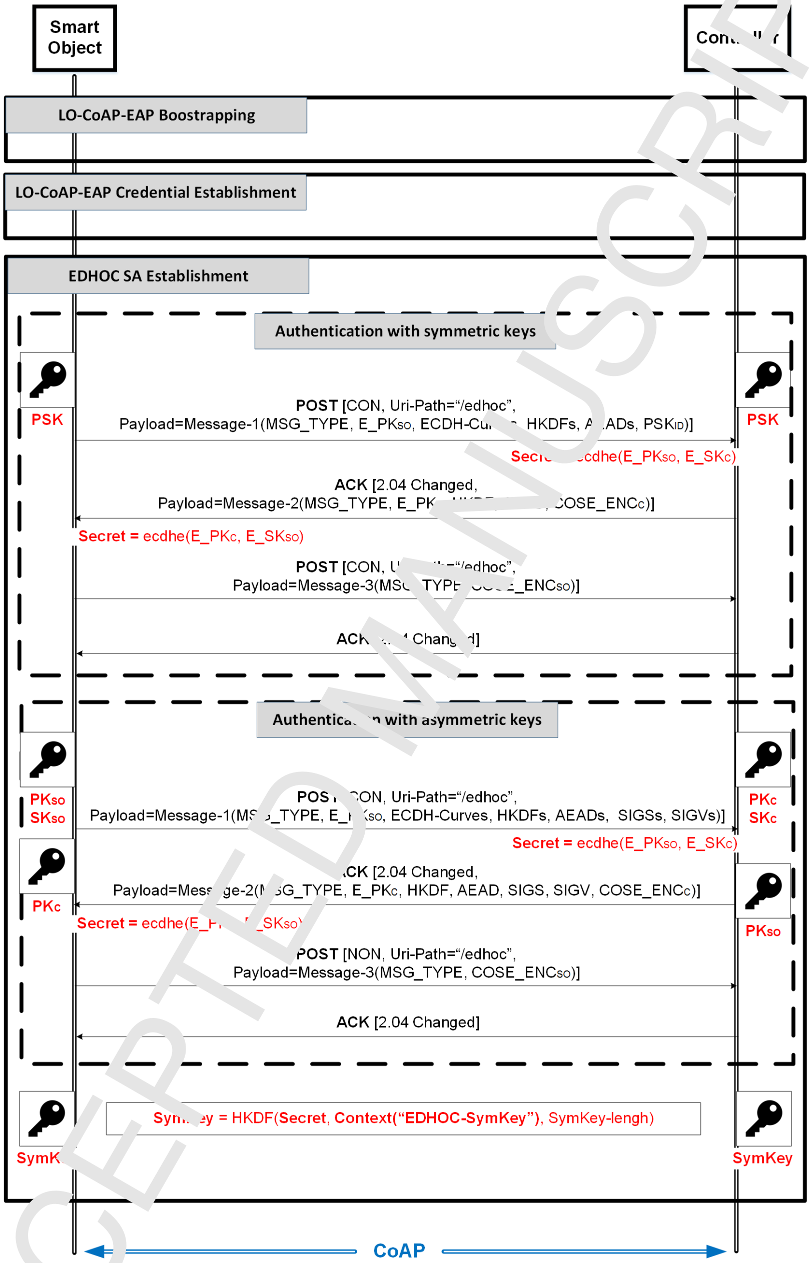


Figure 4: EDHOC interactions between the Smart Object and the Controller for SA establishment and updating (authentication with symmetric keys and asymmetric keys)

- $HKDFs$ states the set of key derivation functions supported by the Smart Object.
- $AEADs$ indicates the set of algorithms for authenticated encryption with associated data supported by the smart object.
- PSK_ID is a unique identifier generated during the *LO-CoAP-EAP Credential Establishment*, which is associated to the PSK.

Then, the Smart Object embeds the *Message-1* in a CoAP POST request and sends it to the Controller.

When the Controller receives the request, this entity extracts *Message-1* and verifies that it supports, at least, one of each set of algorithms supported by the Smart Object. If so, the Controller generates its own ephemeral key pair (i.e., E_SK_C and E_PK_C) and computes the *Secret* with the Diffie-Hellman algorithm as:

$$Secret = ecdhe(E\_SK_C,\ E\_PK_{SO})$$

Additionally, the Controller builds the *Message-2* by including the next parameters:

- MSG_TYPE identifies the EDHOC *Message-2*.
- S_{SO} is the Smart Object's session identifier.
- S_C is a variable length session identifier for the Controller.
- N_C is a 64-bit random nonce for the Controller.
- E_PK_C represents the Controller's ephemeral public key.
- $HKDF$ states the selected key derivation function.
- $AEAD$ indicates the selected algorithm for authenticated encryption.
- $COSE\_ENC_C$ is a *COSE_Encrypt0* object encrypted by the $AEAD$ algorithm, the PSK and the *Secret*. Note that, as mentioned in the COSE and EDHOC specifications [9, 7], the $COSE\_ENC_C$ allows to authenticate the Controller, as well as to protect the *Message-1* and *Message-2* integrity.

Then, the Controller sends this message in a CoAP 2.04 Changed response to the requesting Smart Object.

Upon receiving the *Message-2*, the Smart Object is able to compute the *Secret* similarly to the Controller as:

$$Secret = ecdhe(E\_SK_{SO},\ E\_PK_C)$$

Subsequently, it decodes this second message and tries to decrypt the $COSE\_ENC_C$ object by using the $AEAD$ algorithm, the PSK and the just-computed *Secret*. If this decryption operation is successful, the Smart Object builds and sends the *Message-3* in a new CoAP POST request to the corresponding Controller. Such message contains these parameters:

- *MSG_TYPE* identifies the EDHOC *Message-3*.
- S_C is the Controller's session identifier.
- $COSE\_ENC_{SO}$ is a new *COSE_Encrypt0* object encrypted similarly to the $COSE\_ENC_C$, so that the $COSE\_ENC_{SO}$ allows to authenticate the Smart Object, as well as to protect integrity of all exchanged messages.

Finally, once the Controller obtains the third message, it tries to decrypt the *COSE_ENC_SO* by employing the *AEAD*, the *PSK* and the *Secret* and, if such operation succeed, the EDHOC message exchange finishes. Note that, in order to confirm this successful completion of the EDHOC protocol, the Controller sends to the Smart Object the corresponding ACK with an empty response, as already pointed out.

At this point, both entities are able to compute a shared symmetric key (*SymKey*) by using the *HKDF* with the following parameters:

- *Secret* includes the results of the Diffie-Hellman algorithm.
- *Context("EDHOC-SymKey", exchange_hash, 128)* is a *COSE_KDF_CONTEXT* structure ([9, 7]) defined as follows:
  - *"EDHOC-SymKey"* specifies the algorithm for which the *SymKey* will be used.
  - *exchange_hash* includes a hash of all exchanged messages:

    exchange_hash = hash(hash(Message-1 | Message-2) | Message-3))
  - *128* indicates the length of the *SymKey* (in bits). By considering the NIST recommendation [35], we establish this value to 128-bit length in order to ensure a proper security level to subsequent communications.

It should be pointed out that the Smart Object and the Controller could employ this *SymKey* at any layer to enable a security association for other protocols (i.e., OSCORE [30] or IPsec [31]), thus allowing them to protect their future communications.

5.2.2. EDHOC message exchange authenticated with asymmetric keys

In case of authentication is based on RPKs or certificates, the Smart Object must be in possession of the Controller's public part (PK_C), while the Controller must have the Smart Object's public part (PK_{SO}). Note that the exchange of these public parts between both entities has been performed during the *Credential Establishment* process. Afterwards, the Smart Object and the Controller are able to start the EDHOC protocol, which is authenticated with such asymmetric keys. It should be pointed out that the corresponding message exchange is similar to that employed when authentication is based on symmetric keys. Nevertheless, there are differences with respect to some parameters included in the exchanged messages. Particularly, the *Message-1* does not contain the PSK_{ID} due to the usage of asymmetric keys for authentication. Instead, such message includes the following parameters:

- $SIGSs$ represents the set of algorithms for signing supported by the Smart Object.
- $SIGVs$ states the set of algorithms for signature verification supported by the Smart Object.

Regarding the *Message-2*, it further contains two new parameters:

- $SIGS$ states the selected algorithm for the Smart Object's signature.
- $SIGV$ indicates the selected algorithm for the signature verification by the Smart Object.

Additionally, the $COSE\_ENC_C$ is now encrypted by using only the $AEAD$ algorithm and the *Secret*. This way, while this object protects the *Message-1* and *Message-2* integrity, the Controller is authenticated through the $COSE\_SIG_C$. It is a *COSE_Sign1* object signed by using the $SIGV$ and the corresponding SK_C. In addition, the $COSE\_SIG_C$ is included as part of the $COSE\_ENC_C$ object, as indicated in the EDHOC specification.

Finally, concerning the *Message-3*, note that the $COSE\_ENC_{SO}$ is also encrypted by employing only the $AEAD$ and the *Secret*, so the Smart Object's authentication is provided through the $COSE\_SIG_{SO}$. In this case, such COSE_Sign1 object is signed by using the $SIGS$ and the corresponding SK_{SO}. In addition, it is contained in the the $COSE\_ENC_{SO}$, similar to the *Message-2*.

At this point, as in case of authentication with symmetric keys, once the EDHOC message exchange successfully finishes, both entities may compute the shared symmetric key ($SymKey$) by using the $HKDF$ function with the parameters previously described.

This section focuses on describing the required interactions between the Smart Object and the Controller to establish and update a security association that allows them to protect their subsequent communications. Towards this end, they make use of the EDHOC protocol with authentication based on either symmetric or asymmetric keys, where the corresponding credentials are derived by using certain cryptographic material obtained by the LO-CoAP-EAP protocol. According to it, next section describes a real use case, particularly, a smart building scenario in which our proposal has been deployed.

6. IoT use case: Building Automation

Building Automation (BA) is a useful environment to show the importance of the proposed security architecture. In this environment, Smart Objects are deployed to collect critical building information that must be transmitted to allow data-driven applications to perform automatic operations for energy efficiency, security alarms or access control aspects. With the integration of IoT technologies, BA is achieving a broader dimension through the so-called Industrial IoT

Figure 5: View of the ground floor of the testbed building

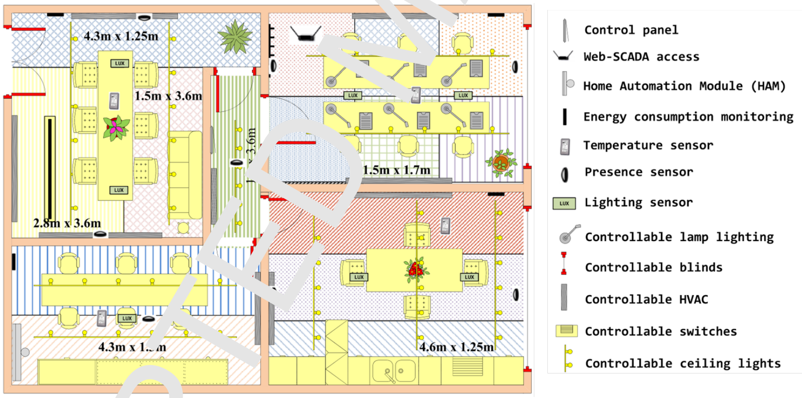


Figure 6: Equipment in the laboratories of the building automation testbed

(IIoT) [61], in which security aspects must be properly addressed taking into account the use of heterogeneous devices, including legacy components [62].

In particular, the considered scenario is a real building called Technology Transfer Center that is located in University of Murcia (Spain). The ground floor of this building is shown in Figure 5, where several laboratories are presented on the lower part of the map. This screenshot has been obtained from the SCADA-web platform for the BA system of the building. The SCADA-web enables to establish automatic operations according to collected data from the building equipment.

To collect data, Smart Objects have been deployed to control different aspects of the building, such as temperature, lighting, presence and power consumption. The huge variety of equipment deployed is shown in Figure 6, which demonstrates the heterogeneity of devices and components that can be potentially used in a certain building. The Smart Objects act as data sources providing information related to the building. Indeed, they transmit the corresponding data to the BA system by using a standardized Internet protocol (i.e., COAP). All the information from Smart Objects is finally provided to users/administrators through a SCADA-web platform deployed in the cloud.

However, such data can be compromised by cyberattacks if security technologies are not implemented. For this reason, the BA scenario is considered for the application of the proposed architecture incorporating a set of components to enable the secure data exchange between Smart Objects and the SCADA-web platform. To achieve that, the proposed use case implements our proposal for the establishment and update of security associations by integrating LO-CoAP-EAP and EDHOC protocols. Accordingly, we describe the secure process by which a Smart Object joins the security domain of the network through the bootstrapping process, when it is deployed. This process allows to obtain the required authentication credentials to launch EDHOC, by which a security association is created between the Smart Object and the IoT Controller. Thus, Figure 7 shows such process for a BA system, where different Smart Objects with a wireless communication through a *6LoWPAN Border Router* are deployed. As depicted in this figure, data from Smart Objects goes to the IoT Controller. This entity is outside the 6LoWPAN network, and it is in charge of sending the Smart Objects' data to the SCADA-web server through a secure channel. In this scenario, it should be noted that the LO-CoAP-EAP Controller [12] is instantiated by the IoT Controller. In particular, the process is as follows: firstly, each Smart Object has to be authenticated to join the network before sending any data. To do that, a Smart Object starts the LO-CoAP-EAP bootstrapping with the IoT Controller (step 1.a.1). If the Smart Object is from the local organization, it will only contact the local AAA server (step 1.a.2) (for which we used a RADIUS implementation, as already mentioned). In case it is from an external organization, it will have to go to the global AAA server of Smart Object's organization (step 1.a.3), as explained in our previous works [48, 12]. After the bootstrapping is completed, the Smart Object and the IoT Controller obtain the necessary key material, as explained in Section 5, in order to establish security associations by using EDHOC (step 2). Note that such security associations could be updated in the future by relaunching only this key exchange protocol. Once the EDHOC protocol finishes, the Smart Object is able to perform a secure data exchange with the IoT Controller by using the cryptographic material associated to the corresponding SA. Additionally, it should be pointed out that the data exchange between the IoT Controller and the SCADA-web is also secured by HTTP over TLS (HTTPS) since these components have high computing resources to manage HTTPS data communications.

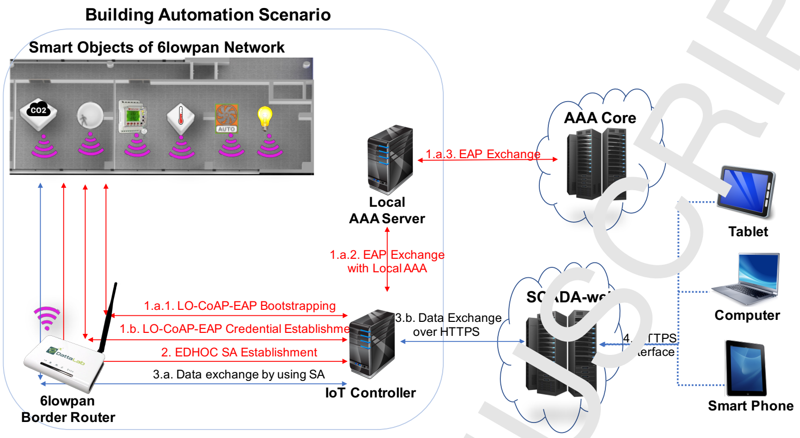


Figure 7: General overview of the use case

Once this information is available in the SCADA-web, the building administrator is able to analyze and monitor this information, such as the energy use, to establish the proper automatic operations in the SCADA-web. Note that, in this scenario, the IoT controller represents the main enabler of secure data communications within the building, while the SCADA-web is the brain of the data processing and automatic decision generation according to the collected data and end-user configurations. For instance, this component can optimize the power consumption by making decisions such as turning on/off the lights or HVAC depending on the temperature and users presence in a certain room. In case of fire detectors' data, the building administrator is responsible for managing a potential critical situation based on data received from the SCADA-web application through secure HTTPS interface for mobile devices, such as laptops, tablets or smartphones.

Next section provides a detailed evaluation of the proposed architecture and its smart technologies using real devices of this building scenario.

7. Evaluation results

In this section, we provide a performance analysis of our proposal by comparing different configurations for each phase, that is, the *Bootstrapping*, the *Credential Establishment* and the *SA Establishment*. Furthermore, we also analyze certain security properties related to the protocols integrating our solution, that is, LO-CoAP-EAP and EDHOC.

7.1. Performance analysis with real devices

In order to demonstrate the advantages of the proposed architecture on a real IoT deployment, we carry out a performance analysis of our proposal taking into account relevant practical aspects, such as message size and runtime.

Towards this end, we compare the integration of LO-CoAP-EAP and EDHOC with different configurations against other potential alternatives. Particularly, LO-CoAP-EAP and PANA[2] are evaluated for the *Bootstrapping* phase. Note that, for these tests and without loss of generality, the EAP method employed by LO-CoAP-EAP is EAP-PSK. This should not be confused with the PSK derived in the *Credential Establishment* phase, which is used in EDHOC. In addition, it should be pointed out that we have also considered PANA in such phase due to it is widely proposed as bootstrapping mechanism in IoT deployments ([53, 43]). Similarly, these protocols are also employed and compared for the *Credential Establishment* phase. Finally, EDHOC with authentication based on both symmetric and asymmetric keys is considered for the *SA Establishment* phase. Specifically, we test EDHOC with PSK-based and RPK-based authentication. Note that we do not consider the certificates-based authentication in this work since its corresponding EDHOC message exchange is similar to that employed with authentication is based on RPKs. In this case, our EDHOC implementation employs the GitHub project pointed out in the COSE specification[3] to manage the corresponding COSE objects, which, in turn, specifies a particular implementation of CBOR representation[4].

7.1.1. *Testbed*

We have employed real devices to deploy the Smart Object and Controller entities for the different tests. Specifically, the Controller has been deployed on an Intel Core i5 with 2.7 GHz and 4 GB RAM. In addition, it is enabled to accept IPv6 connections. Similarly, we have deployed the Smart Object on a device that includes two hardware components, specifically, a MSP430F5419A-EP mote and a PIC32MX795F512L. The former is employed to enable 6LoWPAN connections and manage LO-CoAP-EAP messages, while the latter is in charge of performing public key operations specified by EDHOC, such as the Diffie-Hellman algorithm. It should be pointed out that the mote and PIC32 communicate each other through a USART serial port in order to support all functionality of the Smart Object, that is, LO-CoAP-EAP client and EDHOC client. Regarding the main features of these components, the MSP430F5419A-EP has a frequency of 25 MHz, 128 KB ROM and 16 KB RAM, and the PIC32MX795F512L presents a frequency of 80 MHz, 512 KB ROM and 128 KB RAM. Additionally, we have also employed an intermediate entity acting as a *6LoWPAN Border Router* (6LBR). Its aim is only to route packets between the 6LoWPAN network and the IPv6 network, so the 6LBR is agnostic to messages exchanged between the Smart Object and the Controller. This entity has been deployed on another MSP430F5419A-EP mote.

[2] `https://sourceforge.net/projects/panatiki/`
[3] `https://github.com/cose-wg/COSE-C`
[4] `https://github.com/cabo/cn-cbor`

Table 1: Length of messages exchanged in each phase

Phase	Protocol	Message	Length
Bootstrapping	LO-CoAP-EAP	POST	29
		POST(EAP-PSK1)	36
		ACK(EAP-PSK2)	69
		POST(EAP-PSK3)	68
		ACK(EAP-PSK4)	48
		POST(EAP-Success)	38
		ACK	23
		Total	**311**
	PANA	PCI	16
		PAR	40
		PAN	40
		PAR Req(Id)	48
		PAN Rep(Id)	60
		PAR(EAP-PSK1)	56
		PAN(EAP-PSK2)	84
		PAR(EAP-PSK3)	84
		PAN(EAP-PSK4)	68
		PAR(EAP-Success)	88
		PAN	52
		Total	**636**
Credential Establishment	LO-CoAP-EAP credential provisioning extension	POST(RPK)	91
		ACK(RPK)	89
		Total	**180**
	Dynamic credential provisioning with PANA [63]	PNR(RPK)	112
		PNA(RPK)	112
		Total	**224**
SA Establishment	CoAP (EDHOC-PSK)	POST(Message-1)	84
		ACK(Message-2)	99
		POST(Message-3)	42
		ACK	4
		Total	**229**
	CoAP (EDHOC-RPK)	POST(Message-1)	86
		ACK(Message-2)	211
		POST(Message-3)	152
		ACK	4
		Total	**453**

7.1.2. Message size

Message size is a crucial aspect to be considered in IoT deployments due to the typical limitations related to network bandwidth and resources of involved devices in this type of scenarios. In this sense, Table 1 details the size of each message for a specific protocol, which is considered as potential alternative to fulfill with the corresponding functionality of certain phase. According to the results, the total size of all messages required by LO-CoAP-EAP (311 bytes) is 51% lower than by PANA (636 bytes) when these protocols are employed in the *Bootstrapping* phase. In addition, PANA requires 4 more messages to be exchanged in comparison with LO-CoAP-EAP. Similarly, when LO-CoAP-EAP and PANA are considered in the *Credential Establishment* phase, the total size of all messages is 20% lower with the former (180 and 224 bytes, respectively). It should be pointed out that the PRN and PNA exchanged in PANA in this phase are used as defined in [63], to exchange the public key credentials (RPK or certificates). Regarding the last phase (*SA Establishment*), the total messages size is 50% lower when EDHOC authentication is based on PSK than when it is based on RPKs (229 and 453 bytes, respectively).

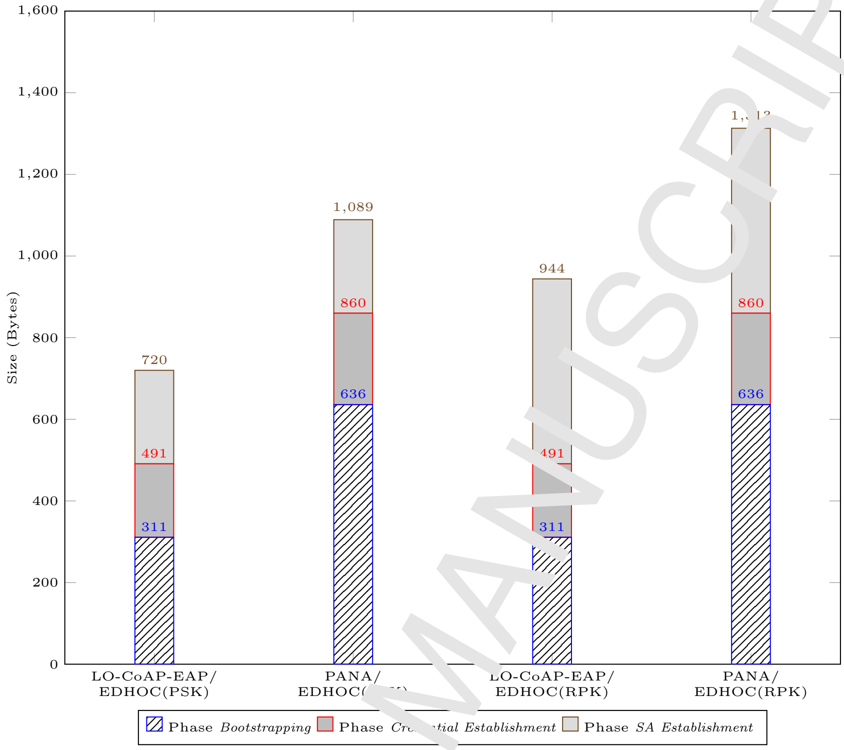


Figure 8: Total length of all exchanged messages to establish a security association with each configuration

Furthermore, Figure 8 shows a message size comparison between the different alternatives; that is, our proposal (LO-CoAP-EAP and EDHOC with authentication based on PSK or RPKs) against PANA and EDHOC with PSK and RPK based authentication. Results show that the total size of all message exchanged is 34% lower with LO-CoAP-EAP than with PANA when EDHOC authentication is performed with PSK (720 and 1089 bytes, respectively), while it is 28% less in case of employing RPKs (944 and 1313 bytes, respectively). According to it, it should be pointed out that this reduction of the total messages size that is achieved by our proposal is specially relevant in IoT scenarios where networks present limited bandwidth or are made up by resource-constrained devices.

7.1.3. Runtime

Another aspect to be considered for evaluation purposes is the runtime spent by the different configurations, which is shown in Figure 9. It should be pointed out that we have performed 10 executions of each configuration and runtime results for each particular case were similar. Therefore, we have not considered

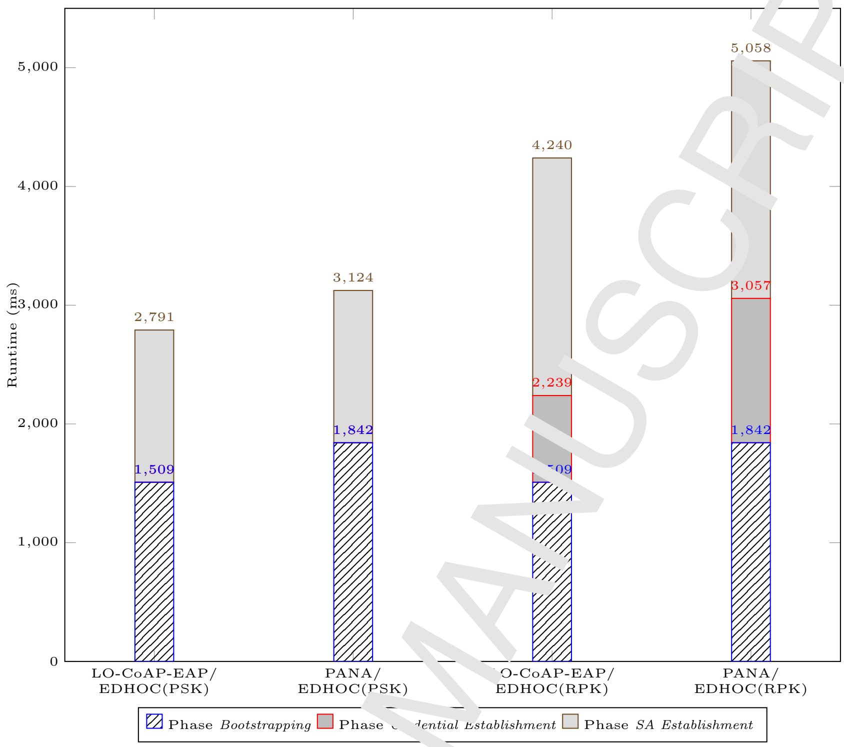


Figure 9: Runtime to establish a security association with each configuration (Smart Object side)

to carry out more executions. For the same reason, we have not included any confidence intervals due to runtime variations were marginal.

By considering the obtained results, we find that the runtime taken by our proposal to establish a security association with PSK-based authentication (1509 ms (*Phase 1.1*) + 6 ms (*Phase 1.2*) + 1282 ms (*Phase 2*) = 2791 ms) is 11% lower than by PANA and EDHOC with such authentication mode (1842 ms (*Phase 1.1*) + 6 ms (*Phase 1.2*) + 1282 ms (*Phase 2*) = 3124 ms). Similarly, when RPKs are employed, LO-CoAP-EAP and EDHOC (1509 ms (*Phase 1.1*) + 730 ms (*Phase 1.2*) + 2001 ms (*Phase 2*) = 4240 ms) take 16% lower than PANA and EDHOC (1842 ms (phase 1.1) + 1215 ms (phase 1.2) + 2001 ms (phase 2) = 5058 ms). These results are a direct consequence of the shorter size of LO-CoAP-EAP messages compared to PANA messages (see Table 1).

7.2. Performance analysis with Cooja

Once we have evaluated our proposal on real devices, we make use of the network simulator *Cooja* [64] in order also to consider more realistic IoT scenarios where there is no direct communication between Smart Objects and the Controller (several hops) or network conditions are not ideal (loss of packets).

As our previous works [12, 48] already provided a performance analysis for LO-CoAP-EAP, in this case we study the EDHOC key exchange protocol with the aim of analyzing its behavior under such restrictions. Specifically, we evaluate EDHOC, in terms of runtime, when it is employed for establishing or updating security associations between two endpoints. It should be pointed out that, as a first approximation and taking into account the network limitations introduced in this performance analysis with *Cooja*, we have considered the lightest EDHOC authentication mode, that is, authentication based on a pre-shared key. This authentication mechanism has been also employed by the LO-CoAP-EAP protocol, as already mentioned.

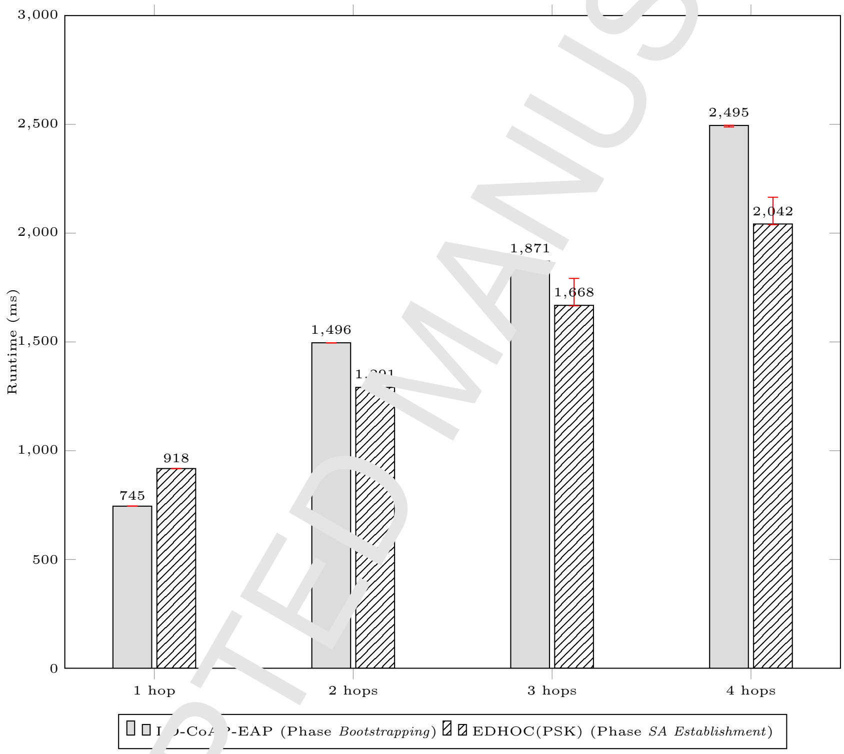
7.2.1. Testbed

In the *Cooja* environment, we have simulated the Smart Object on a Exp5384 mote since it is similar to the MSP430F5419A-EP mote employed in the testbed with real devices. Regarding the PIC32, note that *Cooja* does not include any type of device to simulate this hardware component. Because of that, we have included different time delays in the Exp5384 mote, which correspond to the runtime employed by the PIC32 to perform each public key operation specified by EDHOC. In addition, the 6LBR has been simulated by using a Zolertia Z1 mote, which has a frequency of 8 MHz, 92 KB ROM and 8 KB RAM. Finally, we have also simulated the different intermediate entities in charge of forwarding packets between the Smart Object and the 6LBR (hops) by employing this type of motes, that is, Zolertia Z1.

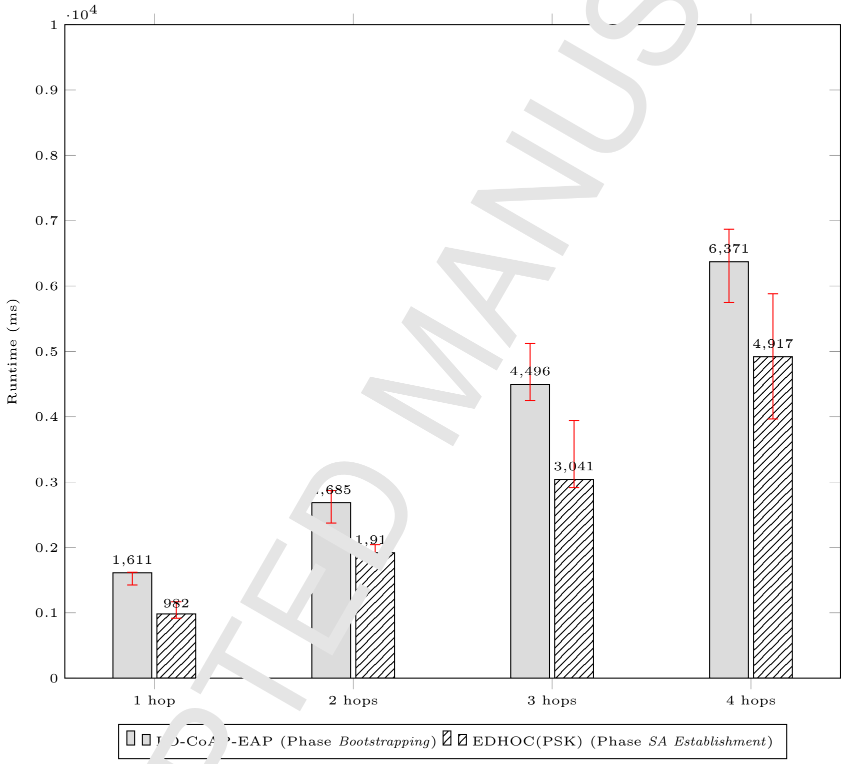
7.2.2. Runtime

In oder to evaluate the performance of EDHOC for establishing and updating security associations in scenarios with different networks limitations, we have focused on analyzing the runtime required by this protocol to complete the corresponding phase, specifically, the *EDHOC SA Establishment*. It should be noted that, in this case, we have performed 70 executions of each experiment due mainly to the introduction of different packet loss ratios. Therefore, we have obtained each runtime result as the median of such executions, and confidence intervals have been calculated by considering a 95% confidence level. According to it, Figure 10 shows the runtime required by EDHOC with packet loss ratios of 0% (10a), 10% (10b) and 20% (10c), and different number of hops between the Smart Object and the Controller for each loss ratio, specifically, from 1 to 4 hops. In addition, note that such figure also displays the LO-CoAP-EAP runtime with the aim of presenting a more detailed analysis of our whole integration proposal.

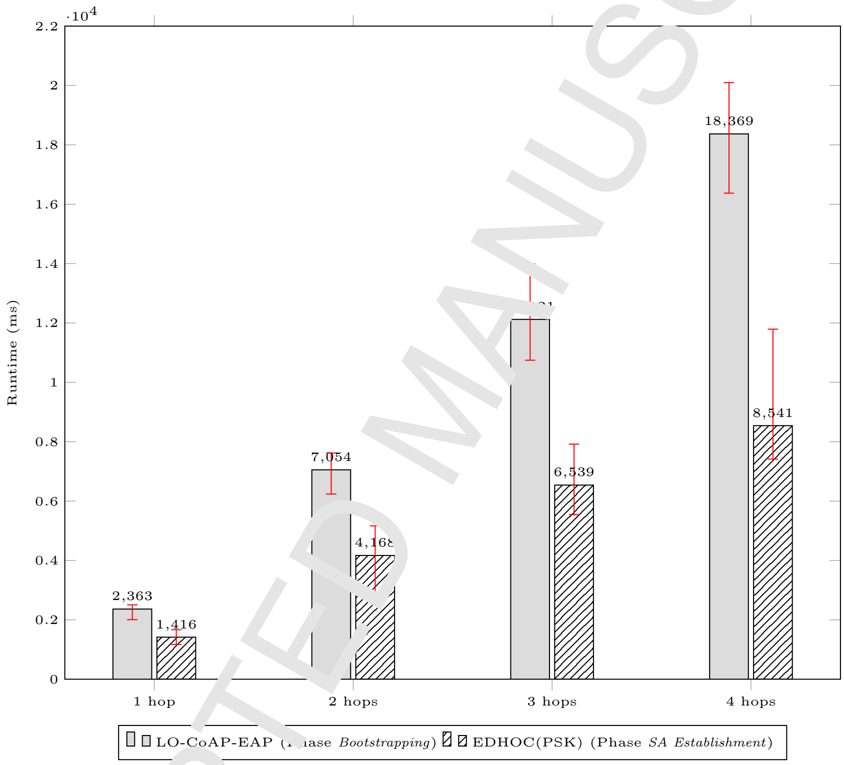
In the case of a loss ratio of 0%, obtained results show that the SA establishment phase (EDHOC protocol) usually requires a lower runtime than the bootstrapping phase (LO-CoAP-EAP protocol) during the establishment of a security association between two endpoints. However, it should be pointed out that, with 1 hop, the runtime spent by the latter is lower than that spent by the former. This fact is mainly because the runtime required by public key operations in EDHOC has a significant influence on the total runtime when



(a) Packet loss ratio of 0%



(b) Packet loss ratio of 10%



(c) Packet loss ratio of 20%

Figure 10: Runtime to establish a security association with our integration proposal and by considering different numbers of hops and packet loss ratios (Smart Object side)

network conditions are optimum (there is no loss of packets) and the Smart Object and Controller are close (1 hop). Consequently, as the number of hops is incremented, the influence of the public key operations runtime decreases, while the number of bytes to be transmitted has a higher impact on the runtime of each protocol. For this reason and as shown in 10a, the SA establishment phase runtime is 14% lower than the bootstrapping phase runtime with 2 hops (1291 ms and 1496 ms, respectively); it is 11% less with 3 hops (1668 ms and 1871 ms); and 18% less with 4 hops (2042 ms and 2495 ms).

Similarly, Figure 10b depicts the runtime required by both phases when the packet loss ratio is established at 10%. In this case, the second phase requires a runtime 39% less than the first phase with 1 hop (982 ms and 1611 ms, respectively); it is 29% less with 2 hops (1918 ms and 268[illegible] ms); with 3 hops, the SA establishment runtime is 32% lower than bootstrapping runtime (3041 ms and 4496 ms); and it is 23% less with 4 hops (4917 ms and 6371 ms).

Finally, we have also considered a high packet loss ratio of 20%, as shown in Figure 10c. Accordingly, the second phase runtime is 40% lower than the first phase runtime with 1 hop (1416 ms and 2[illegible]63 ms, respectively); it is 41% less with 2 hops (4168 ms and 7054 ms); it is 46% less with 3 hops (6539 ms and 12121 ms); and it is 54% less with 4 hops (854[illegible] ms and 18369 ms). These results demonstrate that, as the packet loss ratio is incremented, the runtime difference between these phases raises. This is a direct consequence of the lower number of bytes to be transmitted with EDHOC in comparison with LO-CoAP-EAP (see Table 1).

7.3. Security considerations

As already mentioned, the proposed architecture is based on the integration of LO-CoAP-EAP and EDHOC to establish and update security associations between Smart Objects and the Controller. In this sense, it should be noted that the use of these protocols gives rise certain security properties to be considered.

In the case of LO-CoAP-EAP, it relies on the CoAP and EAP protocols, as well as AAA infrastructures to carry out the bootstrapping process. Therefore, LO-CoAP-EAP inherits its security properties from such underlying mechanisms, and more concretely, from the employed EAP method. Thus, and taking into account that the selected EAP method in this work is EAP-PSK [65], we focus on security properties provided by such method. According to it, LO-CoAP-EAP provides mutual authentication between the endpoints based on a valid message authentication code (MAC). Such MAC is computed by the CMAC algorithm with AES-128, which is recommended by the NIST [66]. Furthermore, LO-CoAP-EAP also guarantees integrity protection by the *Tag* of the protected channel provided by the EAP-PSK method. Similarly, this bootstrapping protocol affords replay protection by the use of 128-bit random numbers and a 32-bit nonce, which is monotonically incremented by involved endpoints in every exchanged message. In addition, LO-CoAP-EAP offers protection against dictionary attacks due to it is not a password protocol. Finally, it should be pointed out that more details about these security properties may be found in the corresponding RFC of EAP-PSK [65].

Regarding the EDHOC protocol, we consider security properties detailed in its specification [7], in particular, mutual authentication, perfect forward secrecy, identity protection and integrity protection. According to this document, the EDHOC message exchange is authenticated by using symmetric key (PSKs) or asymmetric keys (RPKs or certificates), as already mentioned. In addition, EDHOC provides perfect forward secrecy since this protocol makes use of the ephemeral version of the ECDH algorithm (ECDHE) [49]. Furthermore, EDHOC guarantees identity protection regardless of selected authentication mode. Particularly, when authentication is based on asymmetric keys, endpoints signatures includes in the messages 2 and 3 are hidden by employing an authenticated encryption with associated data (AEAD) algorithm. At this point, it should be pointed out that, following the EDHOC specification, we have employed the AES-CCM [67] with 128-bit symmetric keys, 64-bit tag and 7-bit nonce as AEAD algorithm, while ECDSA has been selected as signature algorithm instead of EdDSA [68] to avoid potential interoperability issues with other IoT devices. Similarly, EDHOC also provides integrity protection by using this AEAD algorithm and including a hash of the corresponding EDHOC messages as authenticated data during the encryption process, which ensures integrity of such messages. Finally, note that there is a recent work [69] where authors perform an exhaustive security analysis of EDHOC with the aim of ensuring that such protocol preserves the security properties previously mentioned. Towards this end, they develop a formal model of EDHOC by considering both authentication modes, that is, based on symmetric and asymmetric keys, and carry out its verification with the tool *ProVerif* [70].

In this section, we have compared different configurations in order to establish and update security associations by considering certain relevant aspects, specifically, message size and runtime. To this end, we have employed real resource-constrained devices to evaluate their performance in typical IoT scenarios. Additionally, we have also tested another widely employed bootstrapping protocol, PANA, as potential enabler of EDHOC. Furthermore, we have made use of the *Cooja* simulator to analyze the behaviour of EDHOC when network presents different restrictions, such as several hops between the endpoints or different packet loss ratios. Finally, we have introduced certain relevant security properties related to our solution that should be considered for its deployment in real scenarios. The obtained results demonstrate that the proposed architecture is a feasible solution to be applied in IoT deployments, with the aim to establish and refresh security associations between Smart Objects and the Controller.

8. Conclusions and future work

Key management represents a crucial aspect to build more secure IoT-enabled scenarios. According to it, this work proposed an integrative approach to manage the life-cycle of cryptographic material, which is employed to establish security associations between a Smart Object and the Controller that manages the access to a certain IoT security domain. In particular, we proposed the integration of the LO-CoAP-EAP bootstrapping protocol as an enabler of the EDHOC protocol, by considering different authentication modes. To accomplish this, we extended LO-CoAP-EAP to derive cryptographic material that is employed by EDHOC to establish and update a security association between two entities. The resulting approach is intended to leverage the advantages provided by recent standards and technologies, in terms of lightness and flexibility. Indeed, it should be noted that this approach represents the integration of two standardization efforts in the IETF. Furthermore, our solution was deployed and evaluated on real hardware devices as part of the proposed *Building Automation* use case. Finally, we provide a performance and security evaluation of the proposed architecture by employing different configurations, and considering other protocols widely used in IoT scenarios, such as PANA. Results show that our proposal is a feasible solution to be applied in IoT scenarios, in order to establish and refresh security associations between two endpoints. As future work, we will focus on the integration with different compression mechanisms at different layer to further reduce the message overhead, as well as the use of OSCORE, in order to leverage the advantages provided by an application-layer security approach in the IoT. Moreover, this work can be applied to other scenarios, such as Industrial IoT (IIoT), where security measures implemented in efficient solutions are of paramount importance.

Acknowledgements

This work has been partially funded by the H2020 EU ANASTACIA project under Grant Agreement 731558, the H2020 SerIoT project under Grant Agreement 780139, and the H2020 EU Plug-n-Harvest project under Grant Agreement 768735, also in part by the ODIN Solutions, S.L. under Doctorado Industrial Grant DI-16-08432, CHIST-ERA PCIN-2016-010, and PEANA UNMU13-2E-2536 that is partially funded by FEDER funds.

References

[1] V. Chang, G. Sun, J. Li, Guest editorial: Security and privacy for multimedia in the internet of things (iot), Multimedia Tools and Applications (2018) 1–2.

[2] E. Barker, W. Barker, W. Burr, W. Polk, M. Smid, Recommendation for key management part 1: General (revision 3), NIST special publication 800 (57) (2012) 1–147.

[3] Y. Yang, X. Zheng, W. Guo, X. Liu, V. Chang, Privacy-preserving fusion of iot and big data for e-health, Future Generation Computer Systems 86 (2018) 1437–1455.

[4] G. Sun, V. Chang, M. Ramachandran, Z. Sun, G. Li, H. Yu, D. Liao, Efficient location privacy algorithm for internet of things (iot) services and

applications, Journal of Network and Computer Applications 89 (2017) 3–13.

[5] G. Sun, D. Liao, H. Li, H. Yu, V. Chang, L2p2: A location-label based approach for privacy preserving in lbs, Future Generation Computer Systems 74 (2017) 375–384.

[6] T. Dierks, E. Rescorla, The transport layer security (TLS) protocol version 1.2, Tech. rep. (aug 2008).
URL https://tools.ietf.org/html/rfc5246

[7] G. Selander, J. Mattsson, F. Palombini, Ephemeral diffie-hellman over cose (edhoc), Tech. rep. (March 2018).
URL http://www.ietf.org/internet-drafts/draft-selander-ace-cose-ecdhe-08.txt

[8] H. Krawczyk, Perfect forward secrecy, in: Encyclopedia of Cryptography and Security, Springer, 2011, pp. 921–922.

[9] J. Schaad, Cbor object signing and encryption (cose), Tech. rep. (July 2017).
URL https://tools.ietf.org/html/rfc8152

[10] R. Barnes, Use cases and requirements for JSON object signing and encryption (JOSE), Tech. rep. (apr 2014). doi:10.17487/rfc7165.
URL https://doi.org/10.17487%2Frfc7165

[11] O. Garcia-Morchon, S. Kumar, M. Sethi, State-of-the-Art and Challenges for the Internet of Things Security, Internet-Draft draft-irtf-t2trg-iot-seccons-15, Internet Engineering Task Force, work in Progress (May 2018).
URL https://datatracker.ietf.org/doc/html/draft-irtf-t2trg-iot-seccons-15

[12] 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). doi:10.3390/s17112646.
URL http://www.mdpi.com/1424-8220/17/11/2646

[13] Z. Shelby, K. Hartke, C. Bormann, The constrained application protocol (coap), Tech. rep. (June 2014).
URL http://www.rfc-editor.org/rfc/rfc7252.txt

[14] D. Simon, D. B. D. A. Ph.D., P. Eronen, Extensible Authentication Protocol (EAP) Key Management Framework, RFC 5247 (Aug. 2008). doi:10.17487/RFC5247.
URL https://rfc-editor.org/rfc/rfc5247.txt

[15] G. Gross, C. de Laat, D. Spence, L. H. Gommans, J. Vollbrecht, Generic AAA Architecture, RFC 2903 (Aug. 2000).
URL https://rfc-editor.org/rfc/rfc2903.txt

[16] D. Forsberg, Y. Ohba, B. Patil, H. Tschofenig, A. Yegin, Protocol for carrying authentication for network access (pana), Tech. rep. (2008).

[17] D. Garcia, , R. Lopez, Eap-based authentication service for coap, Tech. rep., IETF, Internet-Draft, Apr. 2016, work in Progress.[Online]. Available: https://tools. ietf. org/html/draft-marin-ace-wg-coap-eap-0[illegible] (2017).

[18] S. Li, L. Da Xu, Securing the internet of things, Syngress, 2017.

[19] Y. Lu, L. Da Xu, Internet of things (iot) cybersecurity research: a review of current research topics, IEEE Internet of Things Journal.

[20] S. Li, L. Da Xu, S. Zhao, 5g internet of things: A survey, Journal of Industrial Information Integration.

[21] E. Rescorla, N. Modadugu, Datagram transport layer security version 1.2, Tech. rep. (jan 2012).
URL https://tools.ietf.org/html/rfc6347

[22] T. Kothmayr, C. Schmitt, W. Hu, M. Brünig, G. Carle, Dtls based security and two-way authentication for the internet of things, Ad Hoc Networks 11 (8) (2013) 2710–2723.

[23] N. Kushalnagar, G. Montenegro, C. Schumacher, Ipv6 over low-power wireless personal area networks (6lowpans): Overview, assumptions, problem statement, and goals, Tech. rep. (August 2007).
URL http://www.rfc-editor.org/rfc/rfc4919.txt

[24] S. Raza, H. Shafagh, K. Hewage, R. Hummen, T. Voigt, et al., Lithe: Lightweight secure coap for the internet of things, IEEE Sensors Journal 13 (10) (2013) 3711–3720.

[25] R. Hummen, J. H. Ziegeldorf, H. Shafagh, S. Raza, K. Wehrle, Towards viable certificate-based authentication for the internet of things, in: Proceedings of the 2nd ACM workshop on Hot topics on wireless network security and privacy, ACM, 2013, pp. 37–42.

[26] S. Raza, T. Helgason, P. Papadimitratos, T. Voigt, Securesense: End-to-end secure communication architecture for the cloud-connected internet of things, Future Generation Computer Systems 77 (2017) 40–51.

[27] G. Selander, F. Palombini, K. Hartke, Requirements for coap end-to-end security, Tech. rep., IETF Secretariat (July 2017).
URL http://www.ietf.org/internet-drafts/draft-hartke-core-e2e-security-reqs-03.txt

[28] O. G.-M. S. Kumar, S. Keoh, Dtls relay for constrained environments, Internet-Draft draft-kumar-dice-dtls-relay-02, IETF Secretariat, https://datatracker.ietf.org/doc/draft-kumar-dice-dtls-relay/ (March 2014 - work in progress).

URL https://datatracker.ietf.org/doc/draft-kumar-ace-dtls-relay/

[29] C. Bormann, P. Hoffman, Concise binary object representation (cbor), Tech. rep. (October 2013).
URL https://tools.ietf.org/html/rfc7049

[30] G. Selander, J. Mattsson, F. Palombini, L. Seitz, Object security for constrained restful environments (oscore), Tech. rep., IETF Secretariat (March 2018).
URL http://www.ietf.org/internet-drafts/draft-ietf-core-object-security-12.txt

[31] S. Aragon, M. Tiloca, S.Raza, Ipsec profile of ace, Internet-Draft draft-aragon-ace-ipsec-profile-01, IETF Secretariat, https://tools.ietf.org/html/draft-aragon-ace-ipsec-profile-01 (March 2017 - work in progress).
URL https://tools.ietf.org/html/draft-aragon-ace-ipsec-profile-01

[32] L. Alliance, Lorawan specification version 1.0, LoRa Alliance.

[33] T. Claeys, F. Rousseau, B. Tourancheau, Securing complex iot platforms with token based access control and authenticated key establishment, in: International Workshop on Secure Internet of Things (SIoT), 2017.

[34] D. Hardt, The oauth 2.0 authorization framework, Tech. rep. (October 2012).
URL http://www.rfc-editor.org/rfc/rfc6749.txt

[35] L. Seitz, G. Selander, E. Wahlstroem, S. Erdtman, H. Tschofenig, Authentication and authorization for constrained environments (ace) using the oauth 2.0 framework (ace-oauth), Tech. rep., IETF Secretariat (July 2018).
URL http://www.ietf.org/internet-drafts/draft-ietf-ace-oauth-authz-13.txt

[36] M. E. S. Saeed, Q.-Y. Liu, G. Tian, B. Gao, F. Li, Akaiots: authenticated key agreement for internet of things, Wireless Networks 1–21.

[37] Z. Shelby, C. Chauvenet, The ipso application framework draft-ipso-app-framework-04, IPSO Alliance, Interop Committee.

[38] S. Rao, D. Chendanda, C. Deshpande, V. Lakkundi, Implementing lwm2m in constrained iot devices, in: Wireless Sensors (ICWiSe), 2015 IEEE Conference on, 2015, pp. 52–57. doi:10.1109/ICWISE.2015.7380353.

[39] J. Korhonen, Applying generic bootstrapping architecture for use with constrained devices - iab workshop on 'smart object security', 2012.

[40] O. Bergmann, S. Gerdes, S. Schäfer, F. Junge, C. Bormann, Secure bootstrapping of nodes in a coap network, in: Wireless Communications and Networking Conference Workshops (WCNCW), 2012 IEEE, IEEE, 2012, pp. 220–225.

[41] O. Garcia-Morchon, S. L. Keoh, S. Kumar, P. Moreno-Sanchez, F. Vidal-Meca, J. H. Ziegeldorf, Securing the ip-based internet of things with hip and dtls, in: Proceedings of the sixth ACM conference on Security and privacy in wireless and mobile networks, WiSec '13, ACM, New York, NY, USA, 2013, pp. 119–124, http://doi.acm.org/10.1145/2462096.2462117. doi:10.1145/2462096.2462117.
URL http://doi.acm.org/10.1145/2462096.2462117

[42] S. Raza, L. Seitz, D. Sitenkov, G. Selander, S3k: scalable security with symmetric keys—dtls key establishment for the internet of things, IEEE Transactions on Automation Science and Engineering 13 (3) (2016) 1270–1280.

[43] Z. Alliance, Zigbee ip specification, ZigBee 095023r10, Work in Progress, July.

[44] B. Sarikaya, Y. Ohba, R. Moskowitz, Z. Cao, R. Cragie, Security bootstrapping solution for resource-constrained devices, Internet-Draft draft-sarikaya-core-sbootstrapping-05, IETF Secretariat, https://tools.ietf.org/html/draft-sarikaya-core-sbootstrapping-05.txt (July 2012 - work in progress).
URL https://tools.ietf.org/html/draft-sarikaya-core-sbootstrapping-05.txt

[45] B. Sarikaya, Secure bootstrapping solution for resource-constrained devices, Internet-Draft draft-sarikaya-6lo-bootstrapping-solution-00, IETF Secretariat, https://tools.ietf.org/html/draft-sarikaya-6lo-bootstrapping-solution-00.txt (June 2013 - work in progress).
URL https://tools.ietf.org/html/draft-sarikaya-6lo-bootstrapping-solution-00.txt

[46] C. P. O'Flynn, B. Sarikaya, Y. Ohba, Z. Cao, R. Cragie, Security bootstrapping of resource-constrained devices, Internet-Draft draft-oflynn-core-bootstrapping-03, IETF Secretariat, https://tools.ietf.org/html/draft-oflynn-core-bootstrapping-03 (November 2010 - work in progress).
URL https://tools.ietf.org/html/draft-oflynn-core-bootstrapping-03

[47] S. Das, Y. Ohba, Provisioning credentials for coap applications using eap, Internet-Draft draft-ohba-core-eap-based-bootstrapping-01, IETF Secretariat, https://tools.ietf.org/html/draft-ohba-core-eap-based-bootstrapping-01 (March 2012 - work in progress).

URL https://tools.ietf.org/html/draft-ohba-core-eap-based-bootstrapping-01

[48] D. Garcia-Carrillo, R. Marin-Lopez, Lightweight coap-based bootstrapping service for the internet of things, Sensors 16 (3) (2016) 358.

[49] D. McGrew, K. Igoe, M. Salter, Fundamental elliptic curve cryptography algorithms (February 2011).
URL http://www.rfc-editor.org/rfc/rfc6090.txt

[50] M. Jones, J. Bradley, N. Sakimura, Json web signature (jws), Tech. rep. (May 2015).
URL http://www.rfc-editor.org/rfc/rfc7515.txt

[51] M. Jones, J. Hildebrand, Json web encryption (jwe), Tech. rep. (May 2015).
URL http://www.rfc-editor.org/rfc/rfc7516.txt

[52] J. Vollbrecht, J. D. Carlson, L. Blunk, D. B. D. A. Ph.D., H. Levkowetz, Extensible Authentication Protocol (EAP), RFC 3748 (Jun. 2004). doi:10.17487/RFC3748.
URL https://rfc-editor.org/rfc/rfc3748.txt

[53] Ieee recommended practice for transport of key management protocol (kmp) datagrams, IEEE Std 802.15.9-2016 (2016) 1–74doi:10.1109/IEEESTD.2016.7544442.

[54] S. Bellovin, R. Housley, Guidelines for Cryptographic Key Management, RFC 4107 (Jun. 2005). doi:10.17487/RFC4107.
URL https://rfc-editor.org/rfc/rfc4107.txt

[55] E. Barker, Recommendation for key management part 1: General (revision 4), Tech. Rep. 800-57, National Institute of Standards and Technology (1 2016).

[56] R. Sanchez-Iborra, J. Sánchez-Gómez, S. Pérez, P. J. Fernández, J. Santa, J. L. Hernández-Ramos, A. F. Skarmeta, Enhancing lorawan security through a lightweight and authenticated key management approach, Sensors (Basel, Switzerland) 18 (6).

[57] J. Song, R. Poovendran, J. Lee, T. Iwata, The advanced encryption standard-cipher-based message authentication code-pseudo-random function-128 (aes-cmac-prf-128) algorithm for the internet key exchange protocol (ike), RFC 4615, RFC Editor (August 2006).

[58] J. Song, R. Poovendran, J. Lee, T. Iwata, The aes-cmac algorithm, RFC 4493, RFC Editor (June 2006).

[59] C. Kaufman, P. Hoffman, Y. Nir, P. Eronen, T. Kivinen, Internet key exchange protocol version 2 (ikev2), STD 79, RFC Editor, http://www.rfc-editor.org/rfc/rfc7296.txt (October 2014).
URL http://www.rfc-editor.org/rfc/rfc7296.txt

[60] J. Salowey, L. Dondeti, V. Narayanan, M. Nakhjiri, Specification for the derivation of root keys from an extended master session key (emsk), RFC 5295, RFC Editor (August 2008).

[61] L. Da Xu, W. He, S. Li, Internet of things in industries: A survey, IEEE Transactions on industrial informatics 10 (4) (2014) 2233–2243.

[62] J. Cheng, W. Chen, F. Tao, C.-L. Lin, Industrial iot in 5g environment towards smart manufacturing, Journal of Industrial Information Integration 10 (2018) 10–19.

[63] J. L. Hernandez-Ramos, D. G. Carrillo, R. Marín-López, A. F. Skarmeta, Dynamic security credentials pana-based provisioning for iot smart objects, in: 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT), 2015, pp. 783–788. doi:10.1109/WF-IoT.2015.7389153.

[64] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, T. Voigt, Cross-level sensor network simulation with cooja, in: Local computer networks, proceedings 2006 31st IEEE conference on, IEEE, 2006, pp. 641–648.

[65] F. Bersani, H. Tschofenig, The eap-psk protocol: A pre-shared key extensible authentication protocol (eap) method, RFC 4764, RFC Editor (January 2007).

[66] M. J. Dworkin, Recommendation for block cipher modes of operation: The cmac mode for authentication, Tech. rep. (2016).

[67] D. McGrew, An interface and algorithms for authenticated encryption, Tech. rep. (January 2008).
URL http://www.rfc-editor.org/rfc/rfc5116.txt

[68] S. Josefsson, I. Liusvaara, RFC 8032: Edwards-curve digital signature algorithm (EdDSA), Request for Comments, IETF.

[69] A. Bruni, T. S. Jørgensen, T. G. Petersen, C. Schürmann, Formal verification of ephemeral diffie-hellman over cose (edhoc), in: International Conference on Research in Security Standardisation, Springer, 2018, pp. 21–36.

[70] B. Blanchet, B. Smyth, V. Cheval, M. Sylvestre, Proverif 2.00: Automatic cryptographic protocol verifier, user manual and tutorial.

Dan Garcia-Carrillo (dgarcia@odins.es) received the B.Eng. degree in technologies in computer science (with a specialization in networks and telecommunications) from the University of Murcia, Murcia, Spain, in and 2014, respectively. He is is currently pursuing the Ph.D. degree with a specialization in security related to authentication for Internet of Things under an industrial Ph.D. grant to ODIN Solutions, S.L. at the University of Murcia.

José L. Hernandez-Ramos (Jose-Luis.HERNANDEZ-RAMOS@ec.europa.eu) received the M.Sc. and Ph.D. degrees in computer science from the University of Murcia, Spain. He was a research fellow in the Department of Information and Communications Engineering at the University of Murcia, before joining the Joint Research Centre of the European Commission in Ispra, Italy, in 2018 as a project officer. He has participated in different European research projects, such as SocIoTal and SMARTIE. His research interests are mainly related to the application of security and privacy mechanisms for the Internet of Things.

Rafael Marín-López (rafa@um.es) received the B.E., M.E., and Ph.D. degrees in computer sciences from the University of Murcia, Murcia, Spain, in 1998, 2000, and 2008, respectively. He is a full-time Associate Lecturer with the Department Information and Communications Engineering, University of Murcia. He did a pre-doctoral internship with Toshiba America Research, Piscataway, NJ, USA, from 2005 to 2006. Since 2003, he has been collaborating actively in standardization. In particular, he has participated in the IETF in diverse working groups (WGs) and is currently active in LPWAN WG and I2NSF WG. He has co-authored RFC 5193, RFC 5609, RFC 5637, the standard IEEE 802.21a, and several Internet-Drafts. His current research interests include authentication, authorization, access control, and key distribution in different types of networks and services. He is currently exploring security aspects in Internet of Things and software-defined networks.

Rafael Marín-Pérez (rmarin@odins.es) is a Technology Manager of OdinS. He received his Ph.D. in Computer Science, at University of Murcia in 2012 in the research field of Wireless Sensor Networks. Since 2006, he worked as full-time researcher on EU projects like ANASTACIA, Smartie, IoT6 and GEN6, as well in national projects such as SAVIA, HospiSegur, MCiudad and MARTA. He gained his expertise on the innovation areas of low-power wireless technologies (Zigbee, Sigfox, LORA and LTE-M) and IoT communication protocols (6lowpan, MQTT, COAP, etc). His main interests are the research and development of monitoring and tele-control solutions, especially focused on Smart Cities and Industry 4.0.

Salvador Pérez (salvador.p.f@um.es) received the B.Sc degree in Computer Science in 2013 and the Ms.C degree in New Technologies in Computer Science in 2015 from the University of Murcia, Spain. He is currently working towards the PhD degree and as a researcher at the same university in the Department of Information and Communications Engineering. His main research interests are focused on defining data-centric security approaches to be deployed in IoT scenarios.

Antonio F. Skarmeta (skarmeta@um.es) received the M.S. degree in computer science from the University of Granada, and B.S. (Hons.) and the Ph.D. degrees in computer science from the University of Murcia, Spain. Since 2009 he has been a full professor at the same department and University. His main interests are the integration of security services, identity, IoT, and smart cities. He has published more than 200 international papers, and he has been a member of several program committees.

- We design an extension of LO-CoAP-EAP to derive cryptographic material, which is employed at different layers to establish a security association between two endpoints
- We design and implement a process to establish the EDHOC credentials based on the use of PSK, RPK and certificates authentication modes
- We deploy our integration proposal based on LO-CoAP-EAP and EDHOC on real hardware devices, and compare with state-of-the-art protocols, such as PANA
- The evaluation results demonstrate the resulting approach is a feasible solution to be applied in IoT scenarios, in order to establish and refresh security associations between two endpoints