Architecture of security association establishment based on bootstrapping technologies for enabling critical 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 an standardization effort for key agreement in IoT. EDHOC is an application layer alternative to 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, Critical Infrastructure

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

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

data protection. 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 TLS [3]) 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, within the IETF, the Authentication and Authorization for Constrained Environments (ACE) WG[1] proposes the Ephemeral Diffie-Hellman Over COSE (EDHOC) [4]. EDHOC represents an authenticated and lightweight application-layer key management approach that provides Perfect Forward Secrecy (PFS) [5]. Furthermore, it is based on the use of CBOR Object Signing and Encryption (COSE) [6], which is a compacted evolution of JSON Object Signing and Encryption (JOSE)[7], 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 particular, we consider the integration with LO-CoAP-EAP[8], which provides a lightweight bootstrapping service that is specifically designed for IoT. LO-CoAP-EAP makes use of the Constrained Application Protocol (CoAP) [9] to transport Extensible Authentication Protocol (EAP) messages [10]. Consequently, it leverages the use of Authentication, Authorization and Accounting (AAA) infrastructures [11] 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) [12]. Indeed, it should be noted that LO-CoAP-EAP is derived from CoAP-EAP [13], which represents an ongoing standardization effort in the ACE WG. Based on the integration between LO-CoAP-EAP and EDHOC, the main contributions of this paper are:

- Extension of LO-CoAP-EAP to derive cryptographic material that is employed at different layers to establish a security association between two endpoints
- Design and implementation of a process to derive the EDHOC authentication credentials based on the use of PSK and RPK authentication modes.
- Deployment of the integration of LO-CoAP-EAP and EDHOC on real hardware devices.

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

- Performance evaluation of the proposed approach and comparison with state-of-the-art protocols, such as PANA.

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, and Section 6 describe a use case in which our proposal is considered. Then, the performance evaluation is given in Section 7. Finally, Section 8 concludes the paper with an outlook of our future work in this area.

2. Related Work

To secure IoT communications, CoAP [9] specifies a binding to the Datagram Transport Layer Security (DTLS) protocol [14]. According to it, the scientific literature reports works that propose different modifications of DTLS to improve its adaptation to constrained networks and devices. Specifically, [15] 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 [16] that is tested on a real hardware platform. Similarly, [17] 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, [18] 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 [19], 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 [20], the usage of DTLS to protect such communications requires the establishment of different security associations, in such a way that *end-to-end* security cannot be provided.

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 being used on lower layers. Based on it, there are two novel specifications defined by the IETF ACE WG that employ COSE [6], which, in turn, is based on the Concise Binary Object Representation (CBOR) [21]. On the one hand, the Object Security for Constrained RESTful Environments (OSCORE) [22] is a protocol intended to protect CoAP messages, that is, ensuring confidentiality and integrity of exchanged data. On the other hand, EDHOC [4] is a lightweight key exchange protocol that aims to establish shared symmetric keys between two endpoints. Even though it is a

recent proposal, EDHOC has attracted the interest due to its flexibility to be integrated with different protocols, and lightness, so it can be used on resource-constrained devices and networks. Indeed, in our previous work, we propose the use of EDHOC to derive and update LoRaWAN cryptographic material [23], in which EDHOC overhead is compared with DTLS handshake. Furthermore, [24] provides a new authorization and authentication framework for the IoT based on OAuth [25] and a EDHOC-based key agreement approach. However, these proposals do not provide evaluation results to demonstrate the feasibility of EDHOC. In addition, they do not consider the establishment of credentials that are employed for authentication purposes during the EDHOC three-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 manage the lifecycle of the cryptographic material associated to an IoT endpoint. While recent proposals [26] partially address this issue, we focus on the integration with bootstrapping approaches to provide a more comprehensive solution. Indeed, in the context of IoT, the bootstrapping is usually referred as the initial process by which a device securely joins a network. Towards this end, the device is authenticated in order to receive the required cryptographic material to become a trusted party in a security domain [27]. Different works about bootstrapping consider the use of pre-established shared key material and running a security association protocol such as DTLS [28, 29, 30, 31, 32]. Furthermore, [33] 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 [34] and proposed in works such as [35, 36, 37, 38]. 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 [39]. 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 [8], 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 the EDHOC authentication credentials (based on PSK and RPK authentication modes) that are employed during the three-message exchange. Therefore, the resulting approach leverages the lightness of technologies, such as CoAP or COSE, as well as the flexibility associated with the use of application layer security. Furthermore, unlike previous proposals, our proposal is intended to provide a more comprehensive approach to key management aspects in IoT scenarios.

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 [4] 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) [40], 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 (PFS) property [5]. 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 [21] and protected by the COSE standard [6]. This way, a minimum message size is ensured in contrast to other JSON-based representation formats (such as JWS [41] and JWE [42]) and, therefore, network overhead is reduced.

In spite of the advantages of EDHOC, nowadays, DTLS [14] is widely considered as the main alternative to protect communications between two endpoints in IoT constrained scenarios [9]. Similarly to TLS [3], DTLS is a two-layer protocol, where the lower layer protocol encapsulates messages of an upper layer protocol to provide certain functionality (e.g., the security parameter establishment). However, the common presence of intermediate entities in these scenarios, such as proxies or publish/subscribe brokers, implies that DTLS can only provide hop-by-hop security [20]. Unlike DTLS, EDHOC is able to ensure end-to-end security by selectively protecting specific fields of the message. This way, proxies can still provide their functionality. According to it, and taking into account this work focuses on the establishment of security associations in IoT infrastructures, we have included EDHOC as part of our proposal in order to enable secure communications. However, the EDHOC specification [4] does not define any mechanism to establish the authentication credentials that are required to perform the 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) [8] is a bootstrapping service that is implemented by CoAP as EAP lower layer protocol. Indeed, LO-COAP-EAP is built on three pillars: CoAP [9], EAP [43] and AAA [11]. 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 a seamless 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, 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. Towards this end, such key material is derived according to the EAP Key Management Framework [10]. 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 [44]. Furthermore, it can be also used to generate the key material needed to run different Security Association Protocols (SAP) [39, 8]. In the context of a heterogeneous IoT, with different radio technologies and severe resource constraints, in this work we use LO-CoAP-EAP to derive the required cryptographic material to establish an EDHOC security association. 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 protocol to derive the authentication credentials required by EDHOC. Next sections provides a detailed description of the resulting approach.

4. Enabling EDHOC through LO-CoAP-EAP

As already described, the EDHOC protocol allows to establish end-to-end security associations 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, raw public keys or certificates. According to it, we propose the usage of LO-CoAP EAP as bootstrapping protocol to establish such credentials.

4.1. Proposed architecture

Based on the integration of LO-CoAP-EAP and EDHOC, Figure 1 shows an overview of the proposed architecture, in which we consider three entities:

- *Smart Object*. It represents an IoT device that aims to join a certain security domain. The Smart Objects acts as a *CoAP-EAP Client* and *EDHOC Client* to carry out the functionality of LO-CoAP-EAP and EDHOC, respectively.
- *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 *CoAP-EAP Server* and *RADIUS Client* for handling the bootstrapping process through LO-CoAP-EDHOC.

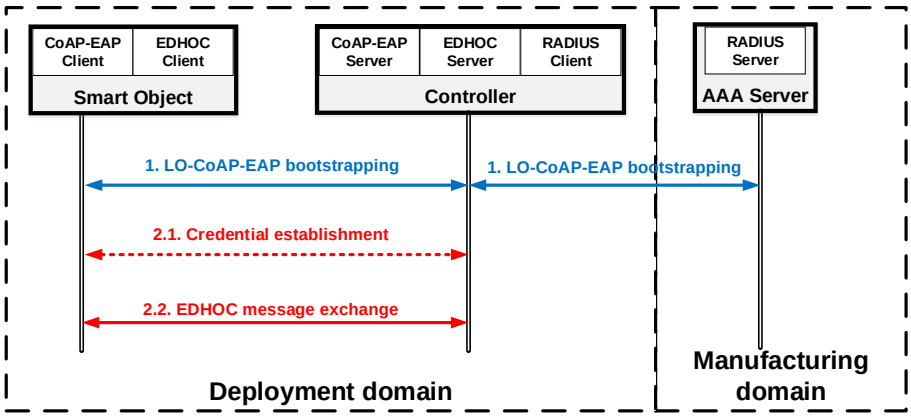

Figure 1: Proposed architecture and interactions overview

- *AAA Server*. It represents the AAA server of the Smart Object Identity Provider (e.g., the manufacturer organization). To realize the corresponding functionality, the AAA Server acts a *RADIUS Server*.

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. As shown in Section 6, 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. On the one hand, the first phase (*LO-CoAP-EAP Bootstrapping*) is focused on the bootstrapping process, which aims to provide certain cryptographic material to the Smart Object and Controller. On the other hand, the second phase consists of the *Authentication Credential Establishment* by using the previously obtained cryptographic material, and the *Security Association Establishment* by the EDHOC protocol.

During the bootstrapping with LO-CoAP-EAP, the Smart Object is authenticated and joins the deployment domain as a trustworthy entity. As a consequence of the bootstrapping process, key material is derived that is used, among other things, to establish a security association (SA) with the Controller. The Controller handles the authentication by interacting with the AAA Server. This entity authenticates the Smart Object and notifies the Controller when the authentication is completed correctly. The AAA Server sends authorization information about the Smart Object to 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 (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* phase is successfully finished, both entities initiate the *Authentication Credential Establishment* process. Towards this end, the Smart Object and the Controller make use of cryptographic material obtained from such previous phase. As described in Section 5, depending on the authentication mode (i.e., PSK, RPK, or certificates), the required operations and interactions will differ. Once the corresponding credentials are established, the Smart Object (acting as *EDHOC Client*) and the Controller (acting as *the EDHOC Server*) start the *Security Association Establishment*. Then, these entities are able to compute a shared symmetric key, which could be used to enable other security solutions intended to protect future communications between the Smart Object and the Controller (e.g., OSCORE [22]).

4.2. Key management and security associations update

According to the proposed architecture (Figure 1), we consider the LO-CoAP-EAP bootstrapping protocol as enabler of the EDHOC protocol. 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. In case of authentication based on asymmetric keys (i.e., raw public keys or certificates), both entities previously require exchange their corresponding public parts. Towards this end, they make use of cryptographic material gained from the LO-CoAP-EAP bootstrapping to authenticate such exchange. It should be pointed out that, unlike authentication with symmetric keys, in this case the Controller could release public keys associated to different Smart Objects that are within the same domain, so that they are able to directly establish security associations each other, without Controller's participation. Such operation mode is out of scope of this work, although it represents part of our future work.

Moreover, the bootstrapping and the authorization credential establishment processes are only performed once, specifically when such Smart Object need to join the *Deployment domain*. At this point, every time the Smart Object and the Controller require to establish a new security association or update a previous one, both entities only need to launch the EDHOC protocol, rather than starting all bootstrapping process again. 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 [45].

4.3. Transporting EDHOC messages

In order to transport EDHOC messages between the Smart Object and the Controller, an application protocol is required. In this sense, we have selected the CoAP [9] 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) [4] 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 as application protocol. Therefore, by also using this protocol for transporting EDHOC messages, Smart Objects' burden is alleviated.

According to it, Figure 2 shows an overview of the EDHOC three-message exchange over CoAP between the Smart Object (EDHOC client) and the Controller (EDHOC server). Nevertheless, it should be pointed out that other application protocols can be adopted to carry out such exchange. Particularly, the *Message-1* is included in a CoAP POST request, which is sent by the Smart Object to start EDHOC. Note that this EDHOC message includes the Smart Object's ephemeral public key. Regarding the *Message 2*, it is contained in a CoAP ACK (2.04 Changed), which is sent by the Controller. Similarly to the first message, the *Message-2* contains the Controller's ephemeral public key. Finally, the *Message-3* is again included in a new CoAP POST request, which is sent by the Smart Object to conclude the EDHOC message exchange. At this point, both entities are able to compute a shared symmetric key by employing the exchanged ephemeral keys, as detailed in Section 5.

5. Interactions description

According to the proposed architecture, in this section we delve and 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 bootstrapping interactions

Prior to running EHDOC, the bootstrapping with LO-CoAP-EAP has to be completed successfully. Once the Smart Object becomes a trustworthy party in the domain, it can start its normal operation. This normal operation can involve running a security association mechanism to secure the communications associated with a specific service. To provide credentials for launching EDHOC, we use the LO-CoAP-EAP protocol. The process of establishing authorization credentials depends on the EDHOC authentication mode, that is, with symmetric keys or asymmetric keys.

In this section we summarize the LO-CoAP-EAP message flow of operation. In the first step, the Smart Object sends an initial message to the Controller to signal that it is present and ready to start the bootstrapping process. This message contains the identity of the Smart Object.

After this first "trigger" message, the Controller initiate the EAP exchange with the AAA Server, sending the identity of the Smart Object received in the

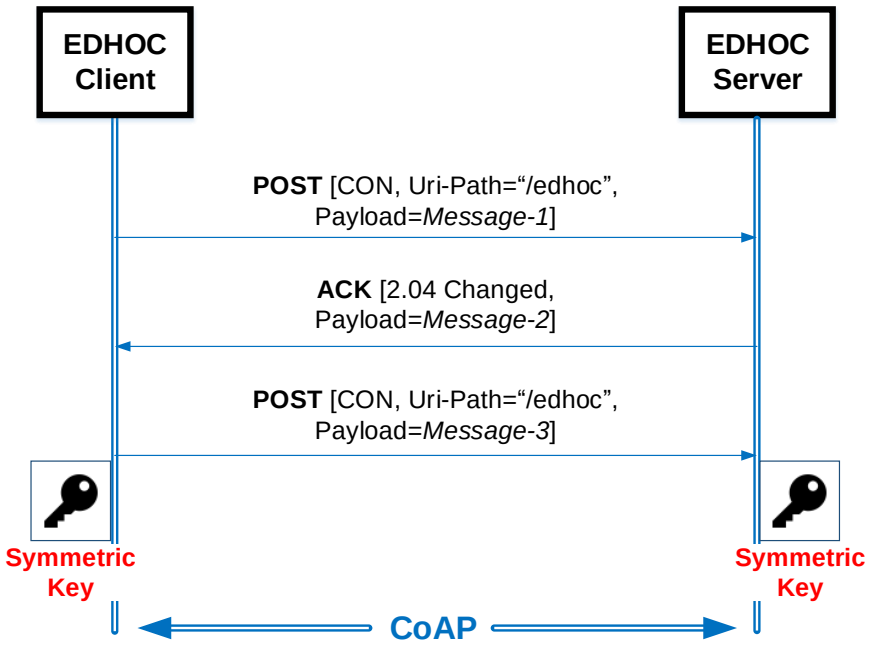

Figure 2: EDHOC three-message exchange over CoAP

previous message. Then, the EAP method, chosen by the AAA, starts between the Smart Object and the AAA, acting the Controller as a forwarder of these messages. When the EAP method has finished correctly and the Smart Object is successfully authenticated, the AAA Server sends the EAP success message to the Controller along with the MSK. The Controller then, is able to derive Transient Session Keys (TSKs) to establish a security association at the EAP lower layer level to secure the LO-COAP-EAP traffic between the Smart Object and the Controller. In this last exchange, there is a key named COAP_PSK, that is derived from the MSK and nonces exchanged in LO-CoAP-EAP, which are used to establish a security association (SA) between the Controller and the Smart Object. The SA is established when the EAP success and its acknowledgement are exchanged and correctly verified. These two message contain a CoAP option called AUTH option, that contains the HMAC of the entire message, providing integrity and authentication to the messages. This AUTH option is generated by the COAP_PSK, a 16-byte key that is computed with AES-CMAC-PRF-128 [46] as Key Derivation Function (KDF), which, in turn, uses AES-CMAC-128 [47]. Both primitives use AES-128 [48]. COAP_PSK is generated as:

$$COAP\_PSK = KDF(MSK, \text{“IETF\_COAP\_PSK”} \,||\, nonce\text{-}c \,||\, nonce\text{-}s, 64, length)$$

where MSK is the key derived from the EAP method, "IETF_COAP_PSK" is the non-NULL ASCII string without quotation marks; nonce-s and nonce-c are the nonces exchanged in the LO-COAP-EAP protocol; 64 is the length of the

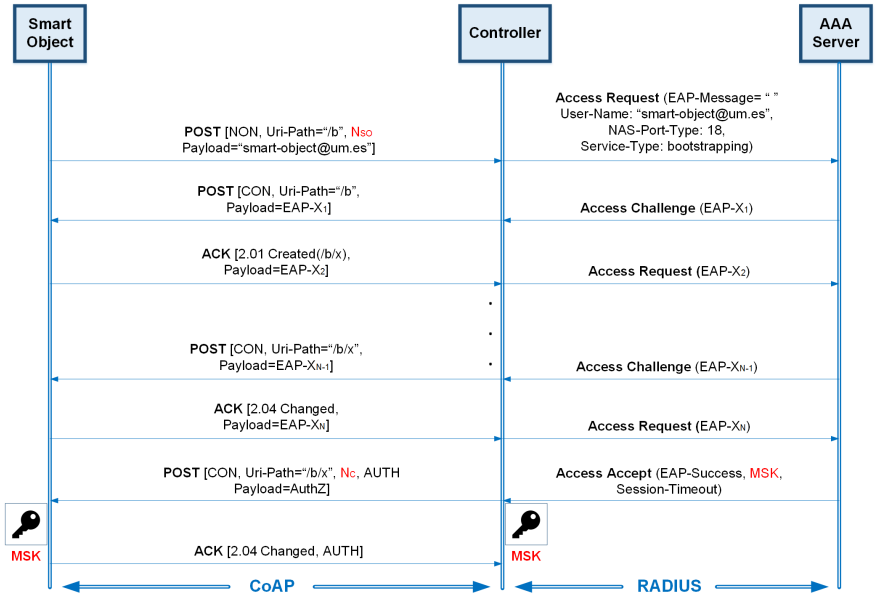

Figure 3: LO-COAP-EAP bootstrapping interactions to establish a MSK between the Smart Object and the Controller

MSK; length is the length of the nonces plus the ASCII string.

With this COAP_PSK we generate the AUTH Option as shown:

$$AUTH\ Option\ value = AES\text{-}CMAC\text{-}128(COAP\_PSK,\ message,\ length)$$

Where message is the entire CoAP message to protect, including the AUTH Option, and length is the length of the message.

5.2. EDHOC key exchange interactions

Once the LO-CoAP-EAP bootstrapping has successfully finished and the Smart Object and Controller are in possession of the MSK, both entities initiate the authentication credential establishment process in order to enable the EDHOC protocol. According to it, the EDHOC three-message exchange depends on the selected authentication mode, that is, with symmetric keys (PSKs) or asymmetric keys (RPKs or certificates), as already mentioned. It should be pointed that we have not considered certificates-based authentication in this work since its corresponding EDHOC message exchange is similar to that employed with authentication based on RPKs.

5.2.1. Message exchange with PSK-based authentication

In case of EDHOC with PSK-based authentication, the Smart Object and the Controller make use of a key derivation function for generating the corresponding pre-shared key (PSK) from the MSK, as shown in Figure 4 (*Authentication Credential Establishment*). Particularly, we select the $prf+$ function

that is defined in [49] and recommended in [50]. It should be noted that such function employs the AES-CMAC-PRF-128 algorithm.

$$PSK = prf+(MSK,\ \textit{“IETF-EDHOC-PSK”} \mid NULL \mid N_{SO} \mid N_C,\ 64,\ 128)$$

According to this equation, the next parameters are required:

- MSK is the master session key from the LO-CoAP-EAP bootstrapping.
- 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.
  - $Nonce_{SO}$ is a random nonce for the Smart Object.
  - $Nonce_C$ is a random nonce for the Controller.
- *64* indicates the MSK's length (in bytes).
- *16* indicates the PSK's length (in bytes).

When the Smart Object and the Controller derive and share the PSK, both entities are able to start the EDHOC three-message exchange, as shown in Figure 4 (*EDHOC message exchange*). It should be pointed out that such exchange between the Smart Object and the Controller employs the CoAP protocol to transport the corresponding messages, as already mentioned.

In order to launch the EDHOC protocol, the Smart Object firstly generate 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.
- $HKDFs$ 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 associated to the PSK.

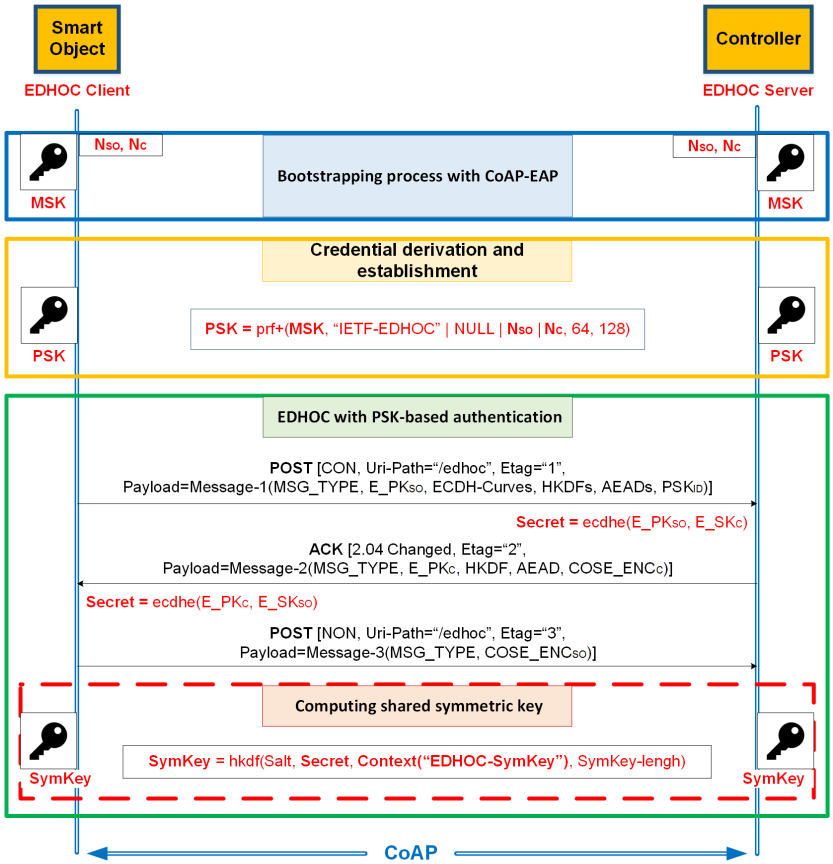

Figure 4: EDHOC message exchange with PSK-based authentication between the Smart Object and the Controller

Then, the Smart Object embeds the *Message-1* in a CoAP 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 generate 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 [6, 4], the $COSE\_ENC_C$ allows to authenticate the Controller, as well as protect the *Message-1* and *Message-2* integrity.

Then, the Controller sends this message in a CoAP 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 CoAP 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 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 is successful, the EDHOC three-message exchange finishes. 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 ([6, 4]) 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 [51], 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 other security protocols (i.e., OSCORE [22] or IPsec [52]), thus allowing them to protect future communications between them.

5.2.2. Message exchange with RPK-based authentication

Before running the EDHOC with authentication based on RPKs, both public keys must be securely exchanged between the Smart Object and the Controller during the *Authentication Credential Establishment*. This exchange is done on LO-CoAP-EAP, when the bootstrapping is completed. To exchange their RPKs, the Smart Object firstly generates its own key pair, and sending the public part to the Controller. This is done by using a new service associated with the Controller, called *RPK Exchange*. Then, the Smart Object sends a CoAP POST message to such service (Uri-Path CoAP header: */rpk*) by including the corresponding RPK. Note that this CoAP message is protected with the AUTH option. When the Controller receives this message, it knows the identity of the Smart Object, so it stores this public key associating it to the Smart Object. After this, the Controller sends it public part to the Smart Object in the corresponding CoAP response, which is also secured with the AUTH option.

After this process, the Smart Object have the Controller's raw public (RPK_C), while the Controller have the Smart Object's raw public key (RPK_{SO}).

Furthermore, it should be pointed out that the corresponding three-message exchange (Figure 5, *EDHOC protocol* sub-phase) is similar to that employed with PSK-based authentication. 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 RPKs 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.

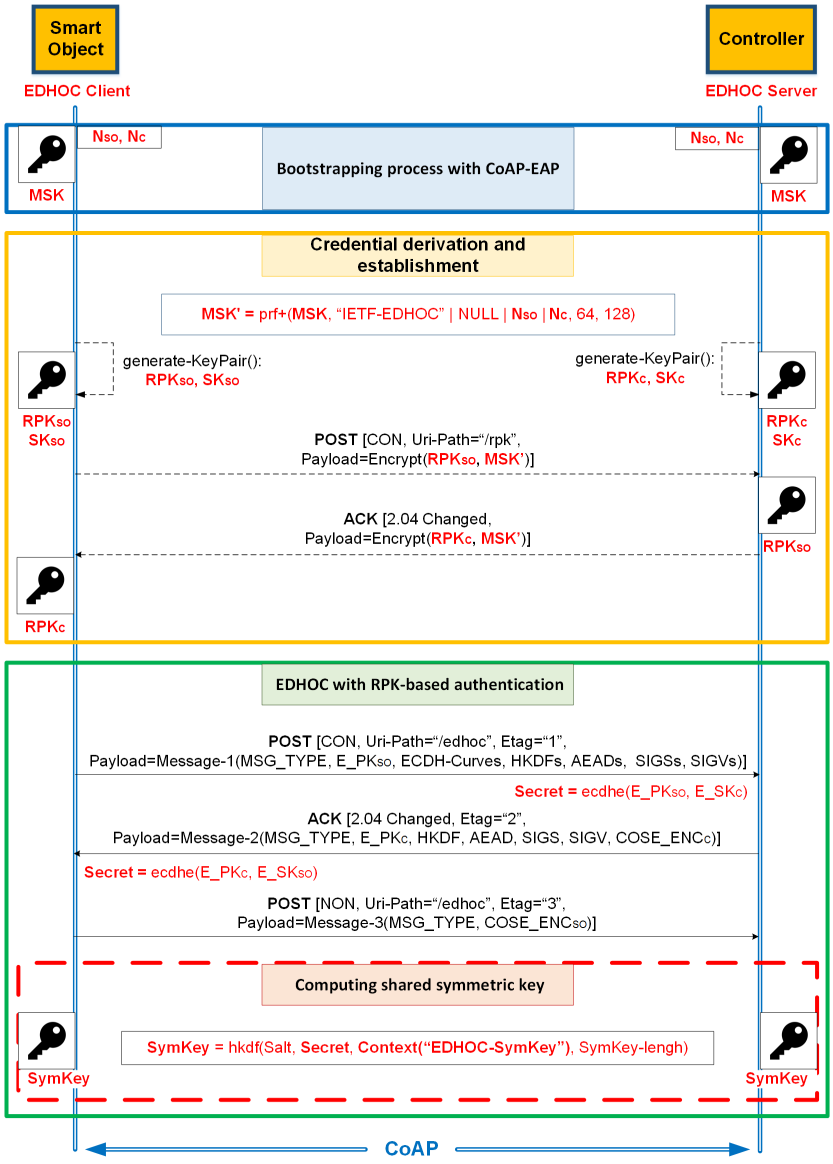

Figure 5: EDHOC message exchange with RPK-based authentication between the Smart Object and the Controller

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 with the PSK-based authentication case, 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 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 PSKs or RPKs, where the corresponding credentials are derived from certain cryptographic material obtained by the LO-CoAP-EAP bootstrapping protocol According to it, next section describes a real use case, in which our proposal has been deployed.

6. Use case of 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, control access and so on.

In particular, the scenario considered is a real building called Technology Transfer Center located in University of Murcia (Spain). The ground floor of this building is shown in figure 6 where several laboratories are presented on the lower part of the map. This screen-shot 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 temperature, lighting, presence, power consumption, etc. The high variety of equipment deployed is shown in figure 7. The smart objects act as data sources providing critical information related to the building. Moreover, the smart objects enable the data transmission to the BA system using standardized Internet protocol (i.e. COAP). All collected information from smart objects is finally provided to users/administrators through a SCADA-web platform deployed in the cloud.

However, such critical data can be compromised by cyber-attacks if security technologies are not implemented. For this reason, the building automation 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 architecture implements smart technologies such as secure bootstrapping and key-establishment process.

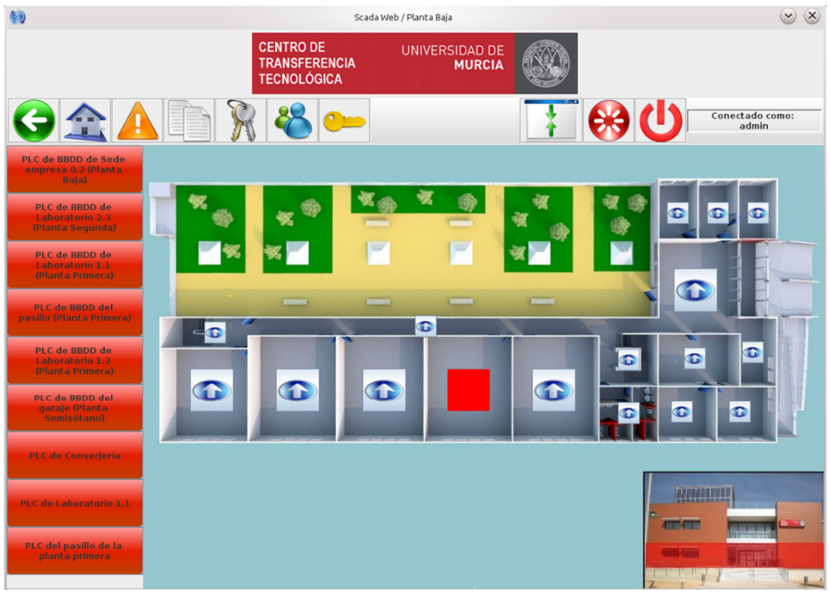

Figure 6: View of the ground floor of the testbed building.

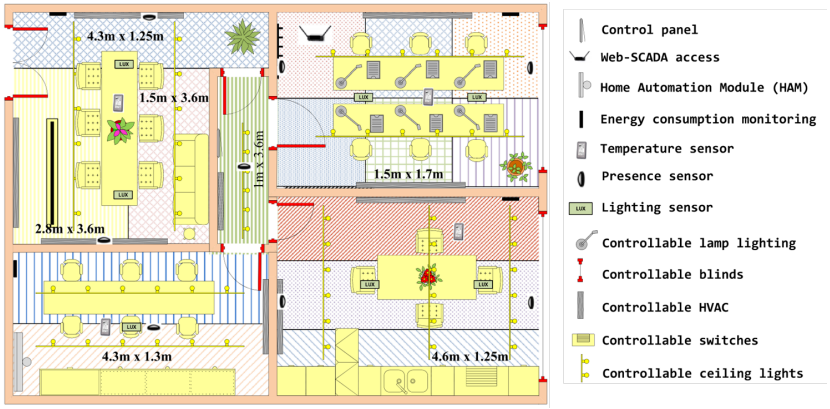

Figure 7: Equipment in the laboratories of the building automation testbed.

Now we describe the secure process by which a smart object is deployed, bootstrapped and joined to the security domain. This process finishes with the creation of a security association using EDHOC to enable the secure data communication. This secure process for BA system is shown in figure 8 where different smart objects with a wireless communication through a *6LoWPAN Border Router* must be deployed. First, each smart object has to be authenticated in the network before sending any data traffic. To do that, each smart object starts the LO-CoAP-EAP bootstrapping with the IoT Controller (step

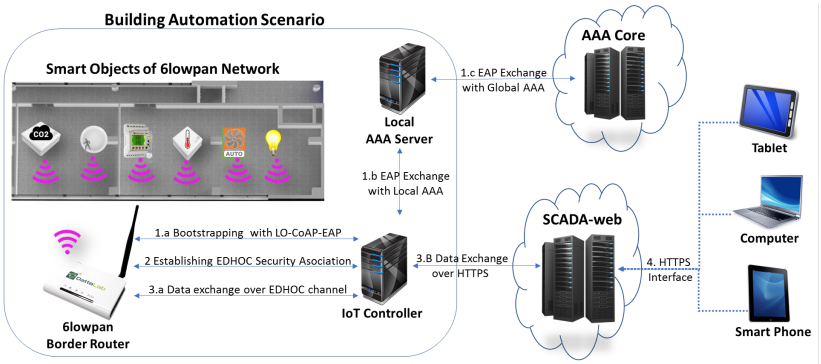

Figure 8: General overview of the use case

1.a). If the smart object is from the local organization, it will only contact the local AAA server (step 1.b) (for which we used a RADIUS implementation in this case), but if it is from an external organization it will have to go to the global AAA server of smart object's organization (step 1.c), as explained in our previous works [39, 8]. 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 further security associations (SAs). Finished the bootstrapping, the smart object starts the EDHOC protocol (step 3.a) to establish a security association with the IoT controller. Then, the smart object can perform secure data exchange with the IoT Controller also enabling the EDHOC key refreshment. The data exchange between the IoT Controller to the SCADA-web is also secured by HTTP over SSL (HTTPS) because these components have high computing resources to manage HTTPS data communications.

Once this information is available in the SCADA-web, the building administrator is able to analyze and monitor the information, such as energy use, and establish the proper automatic operations in the SCADA-web. For instance, the SCADA-web 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 received data from SCADA-web application through secure HTTPS interface for mobile devices such as laptop, tablet or smartphone.

In this use case, the IoT controller represents the main enabler of secure data communications within the building and the SCADA-web is the brain of the data processing and automatic decision generation according to the collected data and end-user configurations.

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

7. Evaluation and Performance Analysis

This section aims to provide a performance evaluation of our proposal by comparing different configuration for each phase, that is, the *Bootstrapping*, *Authentication Credential Establishment* and *Security Association Establishment* phases. Towards this end, we consider relevant practical aspects, such as message size and runtime.

7.1. 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 managing both LO-CoAP-EAP and PANA messages, while the later 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 the PIC32 communicate each other through a USART serial port in order to support all functionality of the Smart Object, that is, LO-CoAP-EAP/PANA 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.

Once we have specified the main features of real devices employed to perform the performance tests, we provide the evaluation of our proposal taking into account message size and runtime with different configurations, as already stated. Particularly, the LO-CoAP-EAP and PANA[2] protocols are tested for the *Bootstrapping* phase. It should be pointed out that we have considered PANA in such phase due to it is widely proposed as bootstrapping mechanisms in IoT deployments ([44, 34]). Additionally, these protocols are also employed and compared for the *Authentication Credential Establishment* phase. Finally, EDHOC with authentication based on both pre-shared keys and raw public keys is considered for *Security Association Establishment* phase. In this case, note that the 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].

[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
		POST/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**
Authentication Credential Establishment	LO-CoAP-EAP	POST	91
		ACK	89
		Total	**180**
	PANA	PNR	112
		PNA	112
		Total	**224**
Security Association Establishment	CoAP (EDHOC-PSK)	Message-1	84
		Message-2	99
		Message-3	42
		Total	**225**
	CoAP (EDHOC-RPK)	Message-1	86
		Message-2	211
		Message-3	152
		Total	**449**

7.2. 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 is 51% lower than by PANA when these protocols are employed in the *Bootstrapping* phase. In addition, PANA require 4 more messages to be exchanged in comparison with LO-CoAP-EAP. Note that the authentication mode employed in both cases is EAP-PSK, as already mentioned. Similarly, when LO-CoAP-EAP and PANA are considered in the *Authentication Credential Establishment* phase, the total size of all messages is 20% lower with the former. Regarding the last phase (*Security Association Establishment*), the total messages size is 50% lower when EDHOC authentication is based on PSK. However, it should be pointed out that the usage of RPKs in EDHOC would facilitate the establishments of security associations between Smart Objects, as previously stated.

Furthermore, Figure 9 shows a message size comparison among all potential configurations, that is, our proposal (LO-CoAP-EAP and EDHOC with

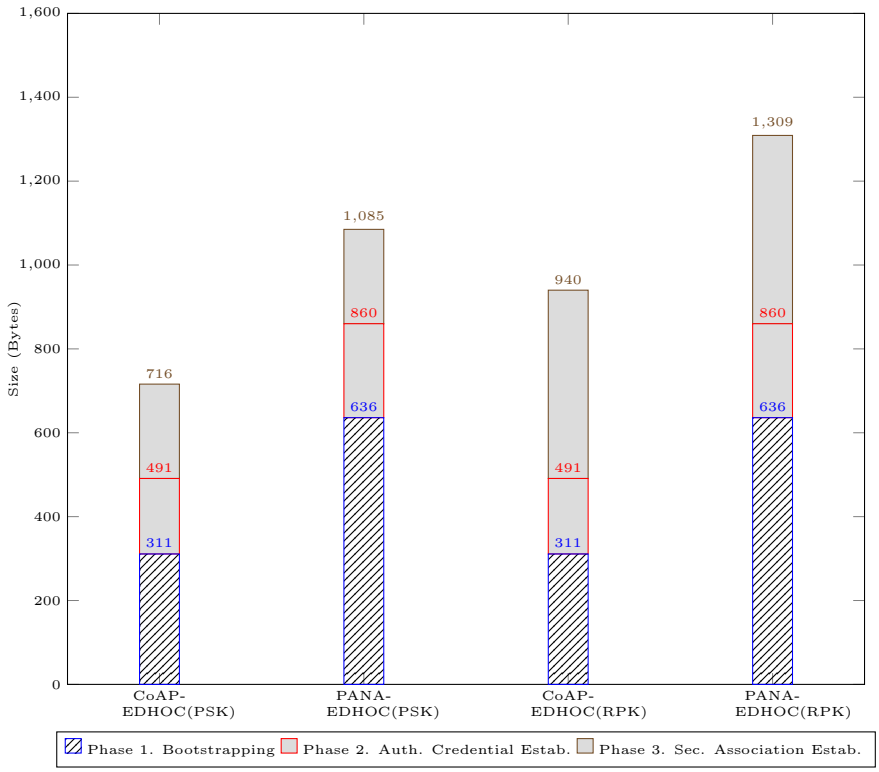

Figure 9: Total length of all messages exchanged with different configurations

authentication based on PSK or RPKs) against PANA and EDHOC with PSK-based 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, while it is 28% less in case of employing RPKs. According to it, it should be pointed out that this reduction of the total size of all messages achieved by our solution is specially relevant in IoT scenarios where network presents a limited bandwidth or is made up by resource-constrained devices.

7.3. Runtime

Another aspect to be considered for evaluation purposes is the runtime spent by the different configurations, which is shown in Figure 10. It should be pointed out that we have performed 10 executions of each configuration, so that results indicate the average value of the runtime of such executions. Additionally, we also include 95% confidence intervals in order to represent the runtime variation in each case. By considering the obtained results, we find the average runtime taken by our proposal to establish a security association with PSK-based authentication (Phase 1: 1509 ms + Phase 2: 6 ms + Phase 3: 1282 ms) is 11%

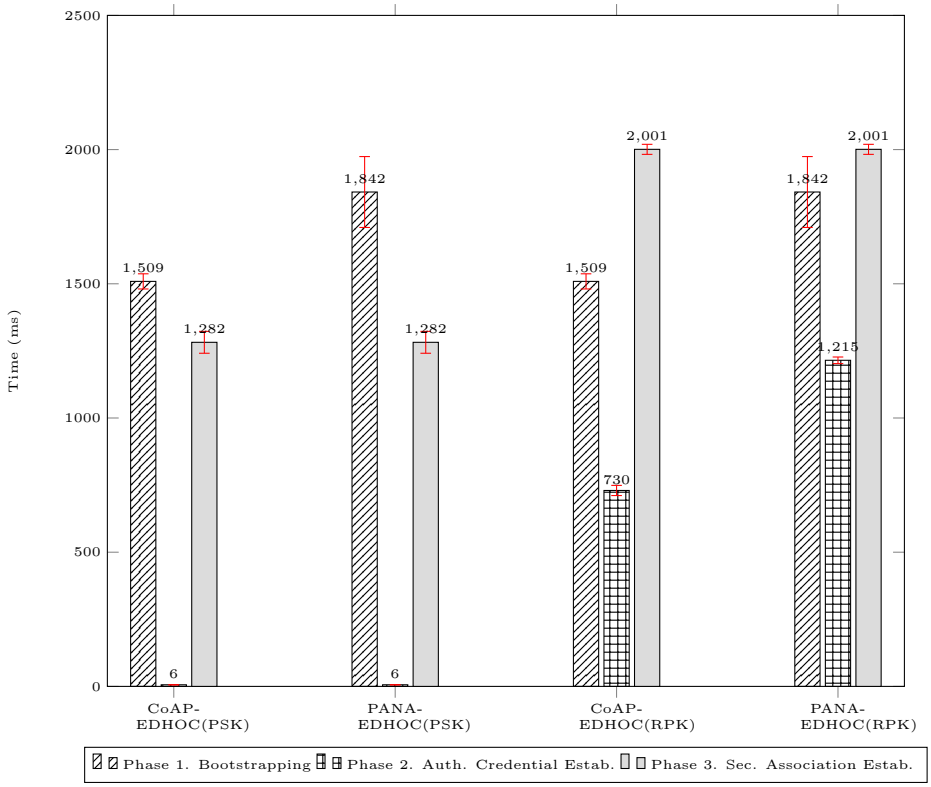

Figure 10: Runtime to establish a security association by using each configurations (Smart Object side)

lower than by PANA and EDHOC with such authentication mode (1842 ms + 6 ms + 1282 ms). Similarly, when RPKs are employed, LO-CoAP-EAP and EDHOC (1509 ms + 730 ms + 2001 ms) takes 16% lower than PANA and EDHOC (1842 ms + 1215 ms + 2001 ms). These results are a direct consequence of the shorter size of LO-CoAP-EAP messages compared to PANA messages (see Table 1).

In this section, we have compared different configurations in order to establish security associations by considering certain relevant aspects, specifically, message size and runtime. To this end, we have employed real resource-constrained devices in order to evaluate their performance in typical IoT scenarios. Additionally, we have also tested another widely employed bootstrapping protocol, PANA, as potential enabler of EDHOC. The results demonstrate that our proposal, which integrates LO-CoAP-EAP and EDHOC, is a feasible solution to be applied in IoT deployments, with the aim to establish different security associations between Smart Objects and Controllers.

8. Conclusions

Key management represents a crucial aspect to build more secure IoT-enabled scenarios. Toward this end, this work proposed an integrative approach to manage the lifecycle of cryptographic material, which is employed to establish security associations between two endpoints. 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 a security association between two endpoints. 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 IETF standardization efforts in the scope of the ACE WG. Furthermore, the solution was deployed and evaluated on real hardware devices as part of the proposed Building Automation use case. Finally, we provide a performance evaluation by employing different configurations, and considering other protocols widely used in IoT scenarios, such as PANA. The results show that our proposal is a feasible solution to be applied in IoT scenarios, in order to establish 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 build an integrative application-layer security approach for the IoT.

Acknowledgements

This work has been partially funded by the H2020 EU ANASTACIA project under Grant Agreement 731558 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, and CHIST-ERA PCIN-2016-010 and PEANA UNMU13-2E-2536 that is partially funded by FEDER funds.

References

[1] J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, Internet of things (iot): A vision, architectural elements, and future directions, Future generation computer systems 29 (7) (2013) 1645–1660.

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

[4] 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

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

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

[7] 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

[8] 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).

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

[10] 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

[11] 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

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

[13] 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-06 (2017).

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

[15] 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.

[16] 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

[17] 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.

[18] 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.

[19] 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.

[20] 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

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

[22] 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

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

[24] 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.

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

[26] 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.

[27] 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

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

[29] 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.

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

[31] 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.

[32] 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

[33] 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.

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

[35] 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

[36] 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

[37] 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

[38] 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

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

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

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

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

[43] 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

[44] 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.

[45] 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).

[46] 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).

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

[48] A. M. Abdullah, Advanced encryption standard (aes) algorithm to encrypt and decrypt data, Cryptography and Network Security.

[49] 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

[50] 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).

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

[52] G. Selander, M. Tiloca, S. Raza, Ipsec profile of ace, Tech. rep., IETF Secretariat (October 2017).
URL https://tools.ietf.org/html/draft-aragon-ace-ipsec-profile-01