IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 24 November 2023, accepted 28 March 2024, date of publication 1 April 2024, date of current version 11 April 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3384095

STANDARDS

Secure Communication for the IoT: EDHOC and (Group) OSCORE Protocols

RIKARD HÖGLUND[1,2], MARCO TILOCA[1], GÖRAN SELANDER[3], JOHN PREUß MATTSSON[3], MALIŠA VUČINIĆ[4], (Senior Member, IEEE), AND THOMAS WATTEYNE[4], (Senior Member, IEEE)

[1]RISE Research Institutes of Sweden, 164 29 Kista, Sweden
[2]Department of Electrical Engineering, Uppsala University, 752 37 Uppsala, Sweden
[3]Ericsson Research, 164 80 Stockholm, Sweden
[4]Inria Paris, 75589 Paris, France

Corresponding author: Mališa Vučinić (malisa.vucinic@inria.fr)

This work was supported in part by the H2020 Project SIFIS-Home under Grant 952652, and in part by the Horizon Europe Project OpenSwarm under Grant 101093046.

ABSTRACT Communication security of an Internet-of-Things (IoT) product depends on the variety of protocols employed throughout its lifetime. The underlying low-power radio communication technologies impose constraints on maximum transmission units and data rates. Surpassing maximum transmission unit thresholds has an important effect on the efficiency of the solution: transmitting multiple fragments over low-power IoT radio technologies is often prohibitively expensive. Furthermore, IoT communication paradigms such as one-to-many require novel solutions to support the applications executing on constrained devices. Over the last decade, the Internet Engineering Task Force (IETF) has been working through its various Working Groups on defining lightweight protocols for Internet-of-Things use cases. ''Lightweight'' refers to the minimal processing overhead, memory footprint and number of bytes in the air, compared to the protocol counterparts used for non-constrained devices in the Internet. This article overviews the standardization efforts in the IETF on lightweight communication security protocols. It introduces EDHOC, a key exchange protocol, as well as OSCORE and Group OSCORE, application data protection protocols adapted for securing IoT applications. The article additionally highlights the design considerations taken into account during the design of these protocols, an aspect not present in the standards documents. Finally, we present an evaluation of these protocols in terms of the message sizes, and we compare them with the non-constrained counterpart, the (D)TLS protocol. We demonstrate that the novel key exchange protocol EDHOC achieves ×5 reduction over DTLS 1.3 authenticated with pre-shared keys in terms of total number of bytes transmitted over the air, while keeping the benefits of authentication with asymmetric credentials.

INDEX TERMS IETF, standardization, security, Internet of Things, CoAP, EDHOC, OSCORE.

I. INTRODUCTION

Internet of Things (IoT) wireless networking technologies enable battery-operated products with a lifetime spanning multiple years. IoT radio standards such as LoRaWAN [1] and 6TiSCH [2] attain low-power operations as follows. First, they limit their maximum transmission unit to be 1-2 orders of magnitude smaller than that in non-constrained networks. Second, they prioritize ''upstream'' traffic from the IoT device to the gateway, over delay-prone downstream traffic. Third, they allow for long sleep periods. There are also cellular low-power technologies, such as NB-IoT [3], which are designed for reducing power consumption by limiting peak data rates, bandwidth and output power, and through extended discontinuous reception to allow for longer sleep periods.

These low-power networks are formed of constrained devices. At the heart of a constrained device is a low-power microcontroller running at 10's of MHz, with 10's of kB of random-access memory and 100's of kB of code memory.

The associate editor coordinating the review of this manuscript and approving it for publication was Derek Abbott.

The microcontrollers often have some sort of hardware acceleration available for cryptographic algorithms, including asymmetric operations. Implementing cryptographic algorithms in software is possible, although avoided as they are slow and reduce the available code space for the actual application. Pulling in generic software processing libraries may similarly be prohibitively expensive from a memory point of view.

To implement the Internet Protocol (IP) and seamlessly integrate the constrained devices into the Internet, the network packets necessarily include some overhead from the IP protocol suite. Standardized protocols developed for constrained radio technologies need to be efficient by carefully encoding data into bytes to minimize the number of bytes in the air. Data compression techniques represent a different method of reducing the number of bytes in the air. However, without any context about the data to compress, they are less efficient than the native encoding.

Over the last decade, the Internet Engineering Task Force (IETF) has been working through its various Working Groups on defining lightweight protocols for constrained networks and devices. "Lightweight" refers to the minimal processing overhead, memory footprint and number of bytes in the air, compared to the protocol counterparts used for non-constrained devices in the Internet. Protocols standardized within the IETF are used as building blocks by industry alliances to build their protocol stacks for specific application domains. One example is the Open Mobile Alliance (OMA) SpecWorks Device Management Working Group, which has specified the Lightweight Machine to Machine (LwM2M) [4] device management stack.

The overall communication security of an IoT networking stack relies on different protocols used throughout the product lifetime. This includes key exchange accompanied by proper key storage and performed with different application entities possibly during device onboarding, as well as communication security and message protection. Each of these protocols relies on authentication credentials such as public certificates or shared symmetric keys, to establish trust between entities and to achieve the desired security properties.

This article presents standardization activities at the IETF on the definition of such a lightweight security protocol stack. The contribution of this article is twofold:

- It describes in a pedagogic and comprehensive manner the new protocol and data format standards from the IETF targeting a lightweight communication security stack.
- It presents a comparative analysis of the message overhead for key establishment and application data protection protocols.

The rest of this article is organized as follows. In Section II, we first discuss the building blocks that enable a secure stack for constrained devices: the Constrained Application Protocol (CoAP), a web transfer protocol similar to HTTP; the Concise Binary Object Representation (CBOR) data format and the related object security format CBOR Object Signing and Encryption (COSE); and C509 certificates, an IoT-equivalent of the X.509 standard. Section III discusses the protocols for secure one-to-one communication: how the Ephemeral Diffie-Hellman over COSE (EDHOC) protocol enables authenticated key exchange for the most constrained IoT use cases; and the end-to-end message protection through the Object Security for Constrained RESTful Environments (OSCORE) protocol. Section IV introduces the protocols for secure group communication and discusses the Group OSCORE protocol. In Section V, we present an evaluation of the message sizes of the protocols and compare with the (Datagram) Transport Layer Security ((D)TLS) protocol typically used in non-constrained environments. In Section VI, we present the related works. In Section VII, we provide a high-level, qualitative overview of the formats and protocols presented in the paper, with respect to corresponding protocols and formats used in non-constrained environments and applications. Finally, in Section VIII, we draw our conclusions.

To illustrate the protocols in action, we outline an industrial scenario and give examples of how the presented security components can be used, as they are introduced throughout the article.

II. BUILDING BLOCKS

The main building blocks for constrained low-power wireless systems are: CoAP, CBOR, COSE and C509. Both secure one-to-one communication and secure one-to-many group communication, detailed below, build on those.

A. CoAP: CONSTRAINED APPLICATION PROTOCOL

The Constrained Application Protocol (CoAP) standard [5] is a web transfer protocol adapted for constrained networking technologies and similar to HTTP.

CoAP especially supports IoT communication paradigms such as group communication and asynchronicity. The protocol is compact: a message starts with a fixed 4-byte header, possibly extended by options before the application payload. CoAP operates on a request-response model with optional response asyncronicity through the `Observe` extension. Group communication is supported through an extension to the core protocol, and relies on one-to-many requests (e.g., over IP multicast) and unicast responses. One core feature of CoAP is response caching at intermediaries. This allows a client to retrieve an IoT device's data even when that device has its radio off.

B. CBOR: CONCISE BINARY OBJECT REPRESENTATION

The Concise Binary Object Representation (CBOR) standard [6] is a data format designed for encoding compactness, small code size and extensibility. It is based on the JavaScript Object Notation (JSON) data model [7], but uses binary encoding. CBOR defines several "major types" such as unsigned and negative integers, byte and text strings, arrays, and dictionaries. CBOR is a basic building block of many

IoT protocols standardized in the IETF, including the security protocols discussed in this article.

C. COSE: CBOR OBJECT SIGNING AND ENCRYPTION

The CBOR Object Signing and Encryption (COSE) standard [8], [9], [10] describes how to create and process representations for cryptographic keys, ciphertexts, signatures, message authentication codes and key exchange, using CBOR for serialization. COSE builds on JSON Object Signing and Encryption, but, since it uses CBOR, COSE objects are smaller. COSE is designed for allowing constrained devices to verify protected messages efficiently.

D. C509 CERTIFICATES

CBOR-encoded X.509 (C509) Certificates is an ongoing IETF work [11] to standardize a CBOR encoding of X.509 certificates suitable especially for the IoT. The CBOR encoding already supports a large subset of X.509 certificates. The C509 encoding can, in many cases, reduce the size of certificates by over 50%, and can be used independently of certificate compression, which, when combined, may provide additional reduction. Two different types of C509 certificates are specified, and differ only in the content of the signature field: 1) for CBOR re-encoded X.509 certificates, the signature is made over the original DER- and ASN.1-encoded X.509 certificate; 2) for natively signed C509 certificates, the CBOR encoding itself is signed. The latter avoids the ASN.1 processing and the decoding process, both of which are complex for constrained devices. While natively signed C509 significantly simplifies the processing at constrained devices, it loses backwards compatibility with existing X.509 certificates until CBOR encoding is deployed in CAs, for which the CBOR re-encoded X.509 provides a migration path.

III. SECURE ONE-TO-ONE COMMUNICATION

We distinguish ''one-to-one communication'', where a device exchanges information with another device, from ''group communication'', where one device communicates with a group of other devices. The security solutions described in this article are designed to apply to both cases, by providing confidentiality, integrity, freshness, and other security properties detailed further. We start by describing one-to-one communication using EDHOC for key establishment and OSCORE for message protection.

A. EXAMPLE SCENARIO

Without loss of generality, we refer to an industrial scenario to illustrate what the needs are and what the standards offer.

We consider the scenario of a factory floor at an industrial plastics manufacturer, where 20 machines on the floor produce plastic bottles. Each machine is equipped with a hopper in which plastic granulate gets poured every now and then. The machine melts these granulates, injects the molten plastic into a mold, and produces a plastic bottle which falls in a container every 30 s.

Previously, workers had to manually monitor the hopper state, pour granulates when its level was low, and monitor the container to swap it when full of bottles. Failing to keep the hopper full caused the machine to ''run empty'', and restarting it took a skilled technician 1-2 hours of work, during which the machine was down. Across three shifts, this happened 1-2 times per month.

Today, this is automated through each machine equipped with three sensors and two actuators. A level sensor measures the fill level of the hopper, and granulates can be poured into the hopper from a pipe using a solenoid valve. Then, a weight sensor monitors the fill level of the container of bottles, and an industrial blinking light can be switched on so that a worker can swap the container. On top of this, a wireless transceiver is installed on the extension port of the machine and publishes internal values (number of bottles produced, internal temperature, state of the motors). Each of these five devices is battery powered and communicates wirelessly to avoid complex cabling. The wireless medium exposes the operations to a variety of threats including denial-of-service, decreased production quality or quantity, safety of employees, ransomware and industrial espionage, from adversary agents within coverage.

With 20 machines on the factory floor, there are 100 devices in total. A wireless network interconnects all these devices with the gateway, itself connected to control software running on a server in the company's cloud. The control software receives and logs all sensor values, and triggers the opening of the valves and the switching of the lights, when needed. Given its criticality in the production system, communication between the wireless devices and the control software must be secure. Unfortunately, the Internet connectivity between the gateway on the factory floor and the control software may go down for short periods of time. Therefore, the gateway implements a simple proxy functionality by caching packets whenever the Internet connection is down.

The following sections describe the different protocols used for securely integrating devices in the factory network, and to be able to establish and maintain secure interactions among those. All protocols build on the building blocks discussed above.

B. EDHOC: ESTABLISHMENT OF KEYING MATERIAL

The Ephemeral Diffie-Hellman Over COSE (EDHOC) [12] is the protocol that establishes a shared symmetric key between two devices. EDHOC is based on an authenticated Diffie-Hellman key exchange. It is designed to be lightweight with low overhead, and suitable to resource-constrained environments.

To achieve mutual authentication, EDHOC uses peer authentication credentials, e.g., public-key certificates. To reduce the size of EDHOC messages, the authentication credentials can also be transported by reference, instead of by value. If an authentication credential is transported by reference, its value is typically distributed out-of-band and

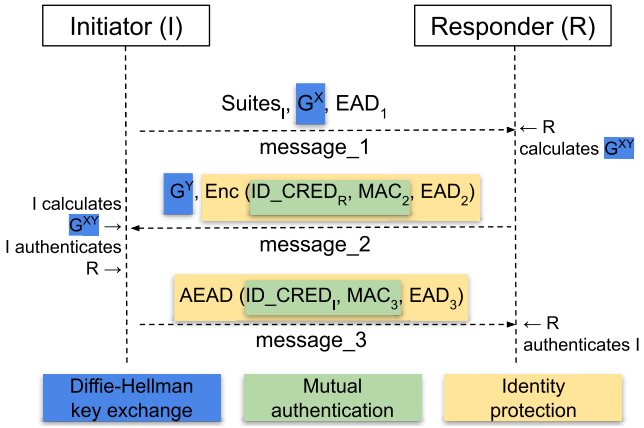


FIGURE 1. Selected fields of an EDHOC exchange in its static Diffie-Hellman authentication mode. The optional message_4 is omitted. Enc denotes binary additive stream cipher encryption. AEAD stands for authenticated encryption with additional data.

then retrieved from the local storage using the reference. Through this and other design decisions, EDHOC achieves a small message size which compares favorably to alternatives such as (D)TLS [12].

The overall security design of the protocol is based on the SIGMA-I MAC-then-SIGN variant [13] that provides: forward secrecy, the guarantee that a compromise of long-term authentication credentials or of a session key does not compromise past session keys; protection of identities; and peer mutual authentication. EDHOC uses COSE for cryptographic operations and CBOR for data encoding, and is typically transported over CoAP. The EDHOC design also provides negotiation and support for a range of cipher suites and authentication methods, as based on either signatures or Message Authentication Codes (MACs) calculated from a shared Diffie-Hellman secret. EDHOC has undergone formal verification of its security properties, which was taken as input in the protocol design [14].

The execution of EDHOC takes place between two peers: one has the role of Initiator, the other the role of Responder (see Fig. 1). Based on the exchanged ephemeral Diffie-Hellman keys G^X and G^Y, the two peers derive the shared secret G^{XY}. By additionally exchanging authentication credentials (ID_CRED$_I$ and ID_CRED$_R$) and MACs (MAC$_2$ and MAC$_3$), EDHOC adds mutual authentication to the basic Diffie-Hellman exchange. By encrypting the exchanged identities with temporary keys, EDHOC protects the Responder identity from passive attacks, and the Initiator identity from active attacks.

The authorization data fields (EAD$_1$, EAD$_2$ and EAD$_3$) allow external security applications to be integrated in EDHOC without increasing the number of round trips, and enabling additional use of parameters transported in EDHOC. One example is to involve a trusted third party performing online authorization of the interaction between message_1 and message_2, through a voucher requested in EAD_1 and retrieved in EAD_2. This enables the Initiator to simultaneously authenticate and authorize the Responder after the second message. As another example, by including a certificate enrolment request in EAD_3, the Responder is able to authenticate and authorize the Initiator after the third message, and to involve a Certification Authority to request the issue of a C509 operational certificate for the Initiator to use in this particular deployment. The C509 certificate may be returned to the Initiator in message_4 (in the message field EAD_4), or alternatively only a reference is sent back, and the certificate is cached in a repository for non-constrained devices to access without loading the constrained links.

Both examples above are applicable in the industrial scenario to optimize the secure on-boarding of new units in the factory, by performing mutual authentication and authorization, and issuing dedicated device certificates from a factory local Certification Authority in two round trips with minimal overhead.

Design Considerations: One of the main protocol design requirements was low implementation complexity and consequently low code footprint. Meeting this requirement while ensuring that the protocol is usable in a wide range of use cases and deployment scenarios is not trivial. For example, one of the initial goals of the protocol was to support authentication based on symmetric keys, apart from asymmetric authentication credentials. To meet the requirement of low complexity, a design decision was made to rule out authentication based on symmetric keys and focus exclusively on asymmetric credentials.

Another example was the selection of the mandatory to implement cipher suites, which is important from an interoperability point of view, as the common denominator among different applications. An EDHOC cipher suite consists of several algorithms, for authenticated encryption, hashing, key exchange curve and signing. There was no controversy on mandating cipher suites based on Advanced Encryption Standard (AES), as it is a symmetric primitive that is widely implemented in hardware even for constrained devices. Similarly, the SHA-256 algorithm was considered the go-to choice for hashing, due to the many available hardware implementations for constrained devices. The community was split in its preferred choice to mandate an elliptic curve and the corresponding signature algorithm: the question was whether to support the NIST P-256 curve and Elliptic Curve Digital Signature Algorithm (ECDSA), or instead Curve25519 and the Edwards-curve Digital Signature Algorithm (EdDSA). The fact that the EdDSA algorithm is specified with a different hash algorithm than SHA-256 meant that a device implementing EDHOC would have to support two different hash algorithms. This was deemed unacceptable from the code footprint point of view. A decision was made to mandate the NIST P-256 curve and ECDSA, also due to the good support available in constrained hardware.

C. OSCORE

The protocol Object Security for Constrained RESTful Environments (OSCORE) [15] allows for protecting CoAP

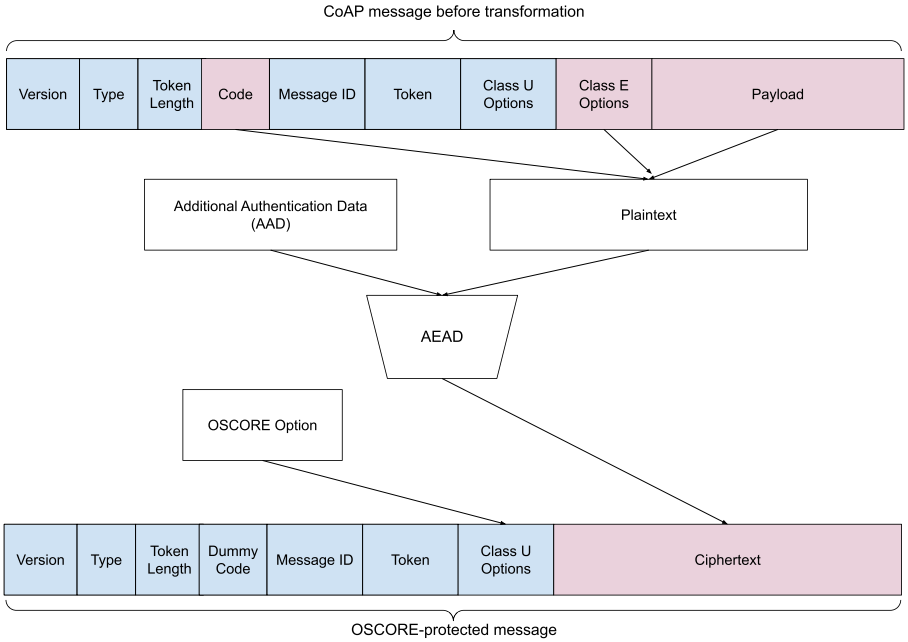


FIGURE 2. The OSCORE transformation and protection of CoAP messages. AEAD stands for "Authenticated Encryption with Additional Data".

messages at the application layer, while enabling those to be forwarded by a proxy (the gateway in our illustrative scenario). OSCORE ensures confidentiality, replay protection, authentication, integrity and ordering of protected messages. In particular, it provides end-to-end security from message producer to message consumer, also in the presence of intermediaries such as proxies.

OSCORE is especially suited for resource-constrained environments, and its design prioritizes small message size and small code footprint, in order to ensure operation in constrained networks. To this end, it relies on CBOR for data encoding and on COSE for security-related services such as encryption. In addition to CoAP, the OSCORE protocol is applicable also to HTTP messages and supports proxy operations for translating between CoAP and HTTP. The security of OSCORE is based on authenticated symmetric encryption.

To communicate using OSCORE, two peers need to first establish an OSCORE Security Context with one another. This can be pre-provisioned to the two peers, or derived by using EDHOC, in which case the two peers start from asymmetric key material to establish a shared secret for keying OSCORE.

The OSCORE Security Context comprises a number of parameters needed for communication with OSCORE. These include: OSCORE identifiers for the two peers, as conceptually distinct from their EDHOC identifiers; settings on the encryption algorithm to use; a counter keeping track of message sequence numbers; a replay window; and cryptographic key information. OSCORE relies on a Master Secret and a Master Salt to generate individual symmetric keys for message protection and unprotection through a key derivation function. Messages in different directions are processed with different keys: a Sender Key for outgoing messages, and a Recipient Key for incoming messages.

OSCORE takes a CoAP message as input and produces a protected CoAP message as output (see Fig. 2). In an OSCORE-protected CoAP message, the CoAP method, the payload and most of the CoAP options are encrypted.

When using EDHOC and OSCORE together, their operation can be optimized (see Fig. 3). In case the EDHOC Initiator has data to send immediately after EDHOC has completed, it is possible to use the optimized flow and piggyback EDHOC message_3 with the first OSCORE request. This optimization, defined in an ongoing work [16], reduces the communication overhead by one round trip.

Design Considerations. The design of OSCORE was led by the motivation to enable end-to-end security in the presence of proxies. This means that the OSCORE security association between two peers does not terminate at a proxy, like in the case of DTLS, but is instead truly end-to-end. A consequence of this design decision is that the proxy has to be able to read the elements of the CoAP message necessary to be able to forward it to the intended recipient. As we show in Fig. 2, an OSCORE-protected message is a valid CoAP message including the OSCORE option, which a proxy can seamlessly forward. The OSCORE-protected message, however, hides as much information as possible from the proxy: it encrypts the CoAP code and leaves in the clear only the options necessary for the proxy to see, e.g., those indicating the host part of the resource identifier at the server.

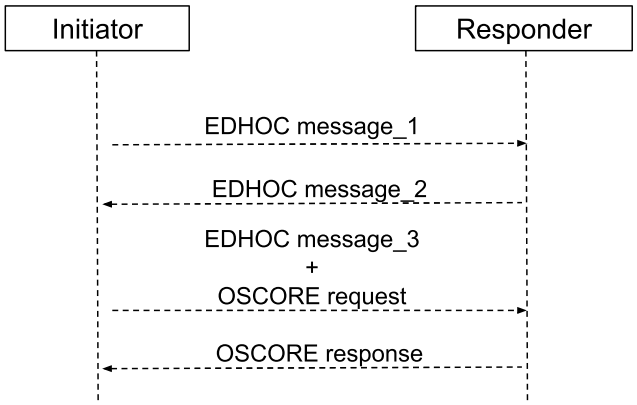


FIGURE 3. The piggybacking of an OSCORE request with EDHOC message_3 to reduce communication overhead is specified in an ongoing work [16] of the IETF CoRE working group.

In its initial design, OSCORE was not based on CBOR and COSE, but defined its own encoding and cryptographic processing. While this initial encoding was very efficient, it required custom processing logic for an OSCORE implementation. Later on in the standardization process, as the CBOR and COSE specifications became stable, it was deemed beneficial to rely on them for reducing the overall code footprint of the security stack for constrained devices.

D. RENEWAL OF KEYING MATERIAL

In our example scenario, EDHOC establishes keying material from the certificates that are pre-installed on the devices. To ensure secure communication over years, it is important to be able to renew that keying material. We want the protocols to allow for this renewal without requiring a worker to walk to each device and connect specialized equipment using a cable.

There are several reasons why renewing keying material is important, such as the exhaustion of sequence numbers used for the construction of nonces, or the attainment of usage limits for the keying material. For AEAD algorithms, specific limits to be respected exist on the number of times that a key has been used to encrypt data or for an unsuccessful decryption (as potentially due to message forgery) [17]. Not following these limits enables an attacker to break the security properties of the used algorithm. The limits depend on the specific algorithm and its properties. The typical limit values for the algorithms used in OSCORE are 2^{20} message encryptions and 2^{20} failed decryptions, using a specific key [17]. If we consider each of our sensor devices generating one packet every minute (a typical value), it takes approximately 2 years to reach this limit. That is, during the typical lifetime of an IoT device of around 10 years, the device needs to be rekeyed several times.

There are several ways of rekeying, such as by using the EDHOC `KeyUpdate` function, rerunning the full EDHOC handshake or rekeying at the application layer, e.g., using the KUDOS protocol defined in an ongoing work [18].

IV. SECURE GROUP COMMUNICATION

The protocols described above allow for secure one-to-one communication, including establishing the keying material, securely communicating through proxies, and rekeying. Some use cases require efficient secure group communication, where one device communicates with many devices at the same time, i.e., the same information is intended for multiple recipients. While it could be achieved through one-to-one communication with each device of the group, this adds significant overhead in terms of network bandwidth and discovery. We hence focus on "true" group communication, as relying on the "one-to-many" paradigm (e.g., over IP multicast), where the network natively supports this type of communication.

A. EXAMPLE SCENARIO

Let us imagine that our factory has been operating for a couple of months. We realize that the valve is sometimes not filling correctly, thus causing the downtime we wanted to avoid. After we trace the malfunction to a bug in the actuators firmware, a fix is quickly implemented, and we now need to update the firmware on the devices in the factory floor. This is a typical use case for group communication: the central controller needs to send a new 100-200 kB firmware image to the devices in the network. Because each packet typically contains at most 100 B of payload, it takes 100 packets to carry the image from the controller to a device. It is hence much more efficient to enable group communication rather than repeating the same 100 packets, once for each device.

Also, an inspection of the factory safety reveals that, due to the ambient noise, workers may not hear the general evacuation alarm when standing next to a machine. Because they are equipped with powerful lights, the recommendation is to switch all the lights at the same time to signal an evacuation. This means sending a command from the control software as addressed to all devices. As the latency is in the order of 1 s in the low-power wireless network, it is important to use group communication to turn on the lights all at the same time, rather than relying on one-to-one communication and turn them on one after the other.

However, how can such group communication occur securely?

B. GROUP OSCORE

Applications relying on one-to-many and many-to-many exchanges of CoAP messages can benefit from secure communication through the security protocol Group OSCORE [19]. Group OSCORE extends the OSCORE protocol and protects end-to-end CoAP requests addressed to one or multiple recipients in a group, as well as the corresponding unicast CoAP responses. Like in OSCORE, the responses are cryptographically bound to the same associated request. Group OSCORE fulfills similar security requirements as OSCORE and provides confidentiality, replay protection, source authentication, integrity and

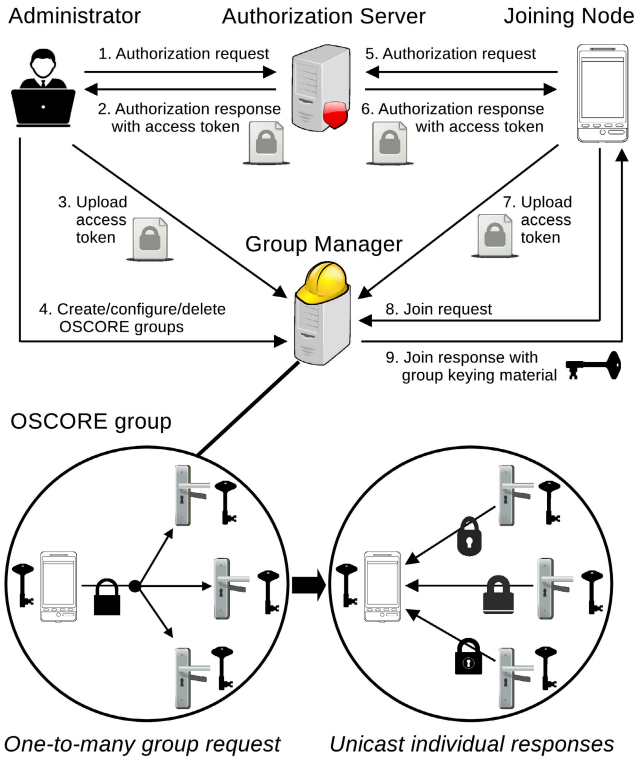


FIGURE 4. Architecture of group OSCORE.

ordering of protected messages. All members of an OSCORE group share the same symmetric keying material used for encryption, hence they are able to decrypt messages exchanged in the group. Source message authentication is achieved through signatures when using the group mode.

Group OSCORE relies on a trusted third party, the Group Manager, as responsible for managing OSCORE groups. Each OSCORE group is identified by a Group ID, unique under the same Group Manager. As a new group member joins, the Group Manager provides it with the necessary group keying material, the Group ID and an available OSCORE Sender ID. When protecting a message, Group OSCORE also relies on the authentication credential of the Group Manager, which prevents attacks based on group cloning and message injection.

When protecting a message intended for multiple recipients, its source authentication is ensured by using a digital signature computed with the sender's private key. The sender first encrypts the message with symmetric keying material shared with the whole group, and then signs the result with its private key.

Fig. 4 overviews how group communication with Group OSCORE works. Administrative operations for creating and configuring the OSCORE groups, as well as the authorized joining and key provisioning operations are defined in detail in the ongoing works by the IETF ACE Working Group [20], [21].

Design Considerations: With the intent of building on top of the EDHOC- and OSCORE-based security stack, Group OSCORE reuses building blocks and concepts from OSCORE. This reduces the code footprint and adds minimal overhead for an implementation of Group OSCORE complementing an existing OSCORE implementation.

Since the Group OSCORE signature is of non-negligible length (e.g., 64 bytes), it is transported at the end of the CoAP message payload, instead of in the message header. This enables possible fragmentation of the CoAP message at the application layer, rather than less conveniently at lower transport layers. Furthermore, the signature is in turn encrypted with separate, per-group keying material in the interest of privacy, as preventing from tracking a group member across different OSCORE groups. That is, it is not possible to infer that a node is a member of two different OSCORE groups, without also being a member of both of those groups.

V. EVALUATION

We evaluate the protocols discussed in this paper in terms of message size. We analyze the protocol fields in detail as they are standardized in the IETF, and deduce the corresponding message sizes as a function of protocol parameters. The following analysis is based on the results of the work-in-progress document in the IETF at [22].

We compare the message sizes of the protocols discussed in this paper with those for the (D)TLS protocol. To have a fair comparison, we make the following assumptions. We use the shortest available authentication tag length, i.e., 8 bytes. During cipher suite negotiation, a minimum number of cipher suites is offered by the protocols. The length of the key identifiers and of the connection identifiers is 1 byte. We include only the mandatory extensions of DTLS. We do not consider the Denial-of-Service mitigation feature of the protocols.

A. KEY ESTABLISHMENT OVERHEAD

We compare the overhead of the EDHOC protocol and the DTLS 1.3 Handshake protocol. We consider different credential types: certificates, raw public keys (RPKs), or pre-shared symmetric keys (PSKs). Table 1 summarizes the message sizes of the key establishment protocols for different configurations.

With reference to DTLS, compression refers to the encoding of an elliptic curve point in the compressed format, such that only the X coordinate is carried over the wire. In the uncompressed format, both X and Y coordinates are carried over the wire.

DTLS provides a feature to cache the certificate carried over the wire in one of the previous handhshake runs. In subsequent runs, the certificate does not need to be carried again, which reduces the message size. We denote this configuration with "Cached RPK" or "Cached X.509", depending on the type of credential used.

EDHOC offers different authentication methods, based either on digital signatures (denoted as "Signature") or static Diffie-Hellman keys (denoted as "Static DH").

TABLE 1. Message sizes (in bytes) for different key establishment protocols and configurations. ECDHE stands for elliptic-curve Diffie-Hellman exchange.

Flight	#1	#2	#3	Total
DTLS 1.3 - RPKs, ECDHE	185	454	255	894
DTLS 1.3 - Compressed RPKs, ECDHE	185	422	223	830
DTLS 1.3 - Cached RPK, PRK, ECDHE	224	402	255	881
DTLS 1.3 - Cached X.509, RPK, ECDHE	218	396	255	869
DTLS 1.3 - PSK, ECDHE	219	226	56	501
DTLS 1.3 - PSK	136	153	56	345
EDHOC - Signature X.509s, x5t, ECDHE	37	115	90	242
EDHOC - Signature RPKs, kid, ECDHE	37	102	77	216
EDHOC - Static DH X.509s, x5t, ECDHE	37	58	33	128
EDHOC - Static DH RPKs, kid, ECDHE	37	45	19	101

TABLE 2. Message overhead (in bytes) for protection of application data. GHC stands for generic header compression, a feature of the 6LoWPAN adaptation layer.

Sequence Number	'05'	'1005'	'100005'
DTLS 1.2	29	29	29
DTLS 1.3	11	11	11
DTLS 1.2 (GHC)	16	16	16
DTLS 1.3 (GHC)	12	12	12
TLS 1.2	21	21	21
TLS 1.3	14	14	14
TLS 1.2 (GHC)	17	18	19
TLS 1.3 (GHC)	15	16	17
OSCORE request	13	14	15
OSCORE response	11	11	11
Group OSCORE pairwise request	14	15	16
Group OSCORE pairwise response	11	11	11

X.509 certificates in EDHOC can be identified by their hash value, without the need to carry the whole certificate over the wire (denoted as "x5t"). Raw Public Keys (RPKs) in EDHOC can be identified using a key identifier ("kid").

In terms of raw public keys, if we compare Cached RPK of DTLS 1.3 with the signature mode of EDHOC using RPKs, we can see that EDHOC achieves $\times 4$ reduction in the total number of bytes transported over the wire. In terms of certificates, DTLS 1.3 needs a total of 869 bytes in case the certificate is cached (Cached X.509, RPK, ECDHE), compared to 242 bytes needed by EDHOC. Finally, the most efficient method of EDHOC using authentication with Static Diffie-Hellman keys requires only 101 bytes to complete the handshake. DTLS 1.3 does not support authentication based on Static Diffie-Hellman keys, hence the comparison is not straightforward. However, we can note that in this configuration, EDHOC achieves $\times 5$ reduction over DTLS 1.3 authenticated with symmetric pre-shared keys, while keeping the benefits of asymmetric credentials.

B. OVERHEAD OF APPLICATION DATA PROTECTION PROTOCOLS

To compare the overheads of different application data protection protocols, we additionally consider a fixed 6-byte long plaintext. Also, we consider an 8-byte authentication tag. The following overheads also comprise the presence of sequence numbers and connection identifiers of the same length. We present the overhead in bytes, as a function of the length of sequence numbers in three representative configurations. Table 2 summarizes the application data protection overhead.

We can note that, separately for each protocol version, the (D)TLS overhead is independent of the length of the sequence number, except in the case of TLS with Generic Header Compression (GHC). The OSCORE overhead depends on the CoAP layer, and more specifically on the number of CoAP options introduced. We can see that OSCORE requests and TLS 1.3 have a comparable overhead, while the overhead is slightly lower in the case of OSCORE responses.

We also present the Group OSCORE overhead in pairwise mode, a special mode where each member of the group can efficiently derive a symmetric pairwise key with any other member of the group. This allows us to have a fair comparison in terms of the considered features, as both OSCORE and (D)TLS use symmetric-key cryptography for the protection of application data. Otherwise, if Group OSCORE is used in its Group mode with asymmetric-key cryptography, a digital signature is appended to each message. This increases the overhead by at least 64 bytes, while on the other hand enabling one-to-many, protected messages with source authentication, which is not possible to achieve with the other considered protocols.

VI. RELATED WORK

The protocols outlined in this article have triggered multiple pieces of academic research, including formal analysis and performance evaluations. In this section, we highlight how these protocols have been analyzed or covered in various research settings.

Multiple papers have been written on the topic of formal analysis, employing model solvers or theoreom provers to test the properties of EDHOC. The output of these works has been taken as input during the design and work on the EDHOC protocol within the IETF. A formal analysis of EDHOC was performed in [23], providing insights about the protocol's resilience under various threat models. The study examined version 12 of the EDHOC specification, and employed the SAPIC+ protocol platform that leverages tools like PROVERIF, TAMARIN and DEEPSEC for a comprehensive assessment. While confirming EDHOC's security against basic threats, this research uncovered vulnerabilities in more advanced threat scenarios, leading to modifications in version 14 of the specification.

Another formal analysis of EDHOC was presented in the study [24] that analyzed all the EDHOC methods, by using an enhanced symbolic Dolev-Yao model. The analysis was done by employing the Tamarin tool and revealed certain limitations in achieving injective agreement authentication. At the same time it confirmed the protocol's capability for implicit authentication and forward secrecy. The study also identified scenarios where a session key might inadvertently be established with a compromised peer. The output from this

study was provided as input to the standardization process to consider for the designing of EDHOC.

Another perspective on the formal analysis of EDHOC was provided in [25], which involved a formal analysis focusing on integrity, secrecy and forward secrecy. This research utilized ProVerif as the chosen tool for formal verification and analysis. The study provided a better understanding of EDHOC's security properties, and it identified violations of some security properties in the protocol's reduced round-trip modes. These findings were reported as input to consider in the protocol design process.

The study in [26] evaluated EDHOC, performing an investigation centered around the protocol's security level. The researchers identified vulnerabilities that could potentially undermine the claimed 128-bit security level. Given that EDHOC uses 8-byte MACs in its most constrained configuration, their analysis highlighted attacks feasible within 2^{64} operations and provided proposals for resolving these issues. Some of these enhancements did not impose additional communication costs, thereby maintaining the lightweight nature of the protocol.

A number of papers have presented additional testing or performance evaluation of EDHOC implementations.

[27] presents EDHOC-Fuzzer, which is a protocol state fuzzer designed for EDHOC implementations. This tool uses model learning to construct a state machine model of an EDHOC implementation. EDHOC-Fuzzer can perform comprehensive model-based testing and a range of useful analysis steps. The authors present three instances of usage of the fuzzing tool on existing EDHOC implementations, including some for constrained devices. This allowed them to discover a number of unexpected behaviours in the considered implementations.

The paper [28] presents the development of CompactEDHOC, a version of EDHOC that externalizes the negotiation of security parameters, using an out-of-band process relying on a third party entity, in order to reduce message overhead. This approach is targeted mainly towards resource-constrained IoT settings. The paper provides practical evaluations using real IoT hardware and simulations, showing reduced message overhead of CompactEDHOC compared to EDHOC and DTLS.

Reference [29] presents the design of an embedded EDHOC implementation for ARM-based devices using the Contiki-NG operating system. Additionally, it provides a performance evaluation exploring runtime, memory overhead and execution time on constrained devices. The evaluation also explores the utilization of hardware acceleration for performing cryptographic operations.

The authors of [30] presented a design of open-source software libraries implementing OSCORE and EDHOC for conventional microcontrollers, and for systems employing a Trusted Execution Environment (TEE) module. The authors perform a performance evaluation on IoT devices, by measuring RAM and flash memory usage in addition to computing time and energy usage. The results show that: a CoAP message can be protected with OSCORE in a matter of milliseconds; the EDHOC authentication based on static-static Diffie-Hellman keys displays better performance than that when signature keys are used; and the use of RPKs as authentication credentials results in a smaller overhead than when using certificates.

Different works also study the evaluation of the OSCORE and Group OSCORE protocols.

In [31], the authors present an open-source implementation of OSCORE for the Contiki-NG operating system. Additionally, they perform a comprehensive experimental evaluation of their implementation running on one resource-constrained hardware platform. The evaluation provides a comparison of DTLS versus OSCORE in terms of payload size, round trip time (RTT), and resource utilization on the device. The results indicate that OSCORE displays better performance compared to DTLS.

Similarly, [32] presents an implementation of the Group OSCORE protocol for the Contiki-NG operating system. The paper provides results from a performance evaluation of Group OSCORE on two separate resource constrained hardware platforms. The evaluation focuses on device resource utilization and RTT, comparing CoAP, OSCORE and Group OSCORE, also considering the use of hardware acceleration for cryptographic operations. The results show that using Group OSCORE on the two tested hardware platforms is feasible, while noting that the signing and verification operations of protected messages are the main responsible for long RTTs.

VII. DISCUSSION

This section discusses formats and protocols that are suitable for constrained environments and applications, and that compose the lightweight security stack presented in this paper. Also, it provides a high-level, qualitative comparison with analogous protocols and formats that are intended for non-constrained environments and applications.

Table 3 overviews the different formats and protocols at-a-glance, positioning them in their respective category.

Constrained environments especially benefit from the use of concise and efficient data formats. A suitable option is CBOR (see Section II-B), which extends the data model of JSON as the corresponding, most common option in non-constrained environments. However, while JSON is text-based, CBOR provides a binary, hence more efficient data representation, which in turn results in a considerably smaller overhead. In particular, CBOR enables extremely small code size, fairly small message size, and extensibility without the need for version negotiation. At the same time CBOR supports all JSON data types for conversion to and from JSON.

The COSE standard (see Section II-C) makes it possible to efficiently create and process signatures, message authentication codes and encryption, as well as to represent cryptographic keys. To this end, it uses CBOR for achieving concise and efficient data serialization. COSE has been

TABLE 3. Overview of formats and protocols. The "constrained" column includes those used in constrained environments and applications, as largely composing the lightweight security stack presented in this paper. The "Non-constrained" column includes the corresponding ones that are intended for non-constrained environments and applications.

Category		Constrained	Non-constrained
Data format		CBOR [6]	JSON [7]
Object security format		COSE [8] [9]	JOSE [33] [34] [35]
Certificate format		C509 [11]	X.509 [36]
Web transfer protocol		CoAP [5]	HTTP [37]
Secure communication protocol	One-to-one communication	OSCORE [15], DTLS Record layer [38]	TLS Record layer [39], IPsec [40]
	Group communication	Group OSCORE [19]	IPsec for IP multicast [41]
Key establishment protocol	One-to-one communication	EDHOC [12], DTLS Handshake [38]	TLS Handshake [39], IKEv2 [42]
	Group communication	Key provisioning for Group OSCORE with ACE [21]	G-IKEv2 [43]

designed taking as starting point JOSE as the corresponding, most common option in non-constrained environments. However, COSE has re-examined some of the decisions originally taken for JOSE, and resulted in a single overall message structure, the use of binary encoding rather than base64 encoding, different message types for signatures and MACs, separate signed message headers for the signed content and the signature, and a partly overlapping set of cryptographic algorithms.

C509 certificates (see Section II-D) enable the use of compact public-key certificates as authentication credentials in constrained environments. With respect to X.509 certificates as the corresponding, most common option in non-constrained environments, C509 certificates rely on CBOR instead of the DER and ASN.1 encoding, thus achieving concise and efficient data serialization. The variant based on CBOR re-encoded X.509 certificates enables a migration path at the price of performing the ASN.1 processing and decoding, while natively signed C509 certificates lack backwards compatibility but do not require re-encoding operations.

CoAP is a lightweight, RESTful web transfer protocol suitable for constrained environments (see Section II-A), which natively supports also group communication, e.g., over IP multicast. The corresponding web transfer protocol for non-constrained environments is HTTP, of which CoAP shares a subset of functionalities. This allows for a straightforward mapping from CoAP to HTTP messages and vice versa with the aid of a cross-protocol proxy, thus enabling the seamless integration of the IoT with traditional web applications. There are several reasons why HTTP was deemed inadequate for low-power IoT. Being connection-oriented, HTTP demands non-negligible resources for connection establishment and maintenance, lacking features that are useful for IoT-based machine-to-machine applications. These include built-in discovery, multicast support and asynchronous message exchanges. Also, HTTP uses messages whose structure and extensibility through headers is not designed to limit communication overhead, an important factor in resource-constrained networks.

As to secure communication, CoAP originally relied on the DTLS Record layer for protecting messages at the transport layer. In the presence of intermediaries such as proxies, this approach does not provide end-to-end security between the origin sender and the origin recipient, since the secure channel terminates at the proxy. In particular, the proxy has to establish two separate secure channels (i.e., one with the sender and one with the recipient), and it can see or modify the plain exchanged data without being noticed. As a more recent alternative, the OSCORE security protocol (see Section III-C) can be used to protect CoAP messages end-to-end between the origin sender and the origin recipient, also in the presence of (untrusted) intermediaries. In particular, OSCORE uses CBOR and COSE, and results in a smaller overhead compared to DTLS (see Table 2). Unlike DTLS running over UDP, OSCORE is not devoted to a particular transport and works wherever CoAP works. Corresponding secure communication protocols for non-constrained environments are: the TLS Record layer, for protecting messages transported over TCP at the transport layer; and the IPsec protocol for protecting IP packets at the IP layer, possibly paired with the enforcement of packet filtering and rule-based packet processing.

Group communication is not attainable at the transport layer, and not even DTLS over UDP supports protection of one-to-many message exchanges. As suitable also to constrained environments and applications, group communication for CoAP can be protected with the Group OSCORE security protocol (see Section IV-B), which extends OSCORE and adapts it for working in group communication setups. Group OSCORE ensures the cryptographic binding between a one-to-many CoAP request and all the multiple, corresponding CoAP responses. A corresponding secure group communication protocols for non-constrained environments is IPsec for IP multicast for protecting IP packets at the IP layer, as an extension of IPsec for group communication setups.

For the establishment of keying material, CoAP originally relied on the DTLS Handshake, in order to establish a DTLS session for protecting communications at the transport layer. More recently, the EDHOC key establishment protocol has also been specified (see Section III-B). Unlike DTLS running over UDP, EDHOC is not devoted to a particular transport, and it always ensures mutual authentication of the two peers through their public authentication credentials. Also, the established cryptographic secret always has forward secrecy, and it can be used to derive arbitrary keying material (e.g., an OSCORE Security Context), while the DTLS Handshake

specifically establishes a DTLS session. EDHOC messages are considerably smaller in size when compared to the DTLS Handshake messages (see Table 1), especially due to its use of CBOR and COSE. The corresponding key establishment protocols for non-constrained environments are: the TLS Handhshake, for establishing TLS sessions to protect communications at the transport layer; and the IKEv2 protocol, for establishing IPSec Secure Associations to protect communications at the IP layer.

When group communication for CoAP is protected with Group OSCORE, key provisioning can be performed through a dedicated application profile of the ACE framework for authentication and authorization in constrained environments (see Section IV-B). In particular, a trusted Group Manager is responsible for such a process as paired with access control enforcement, thus ensuring that only authorized nodes can become members of an OSCORE group. In non-constrained environments, one can rely on the G-IKEv2 protocol for establishing group Security Associations to protect communications over IP multicast at the IP layer.

VIII. CONCLUSION

This article has presented the lightweight security protocols for Internet-of-Things use cases being standardized at the IETF. We have described different security protocols providing authentication, authorization, secret key establishment and secure communication. We have highlighted design considerations taken into account during the design of these protocols. The protocols build on common building blocks suitable for implementation in constrained environments. These protocols and their building blocks together complete the suite of efficiently encoded security items: i) the EDHOC protocol, which enables efficient authentication and Diffie-Hellman key exchange even for the most constrained radio technologies; ii) the OSCORE protocol, which efficiently provides end-to-end protection of CoAP messages in the presence of intermediary proxies; and iii) the Group OSCORE protocol, which provides end-to-end security to group communication for CoAP, e.g., over IP multicast.

The protocols presented in this paper have already been the subject of different academic studies. The EDHOC protocol has been thoroughly analyzed for vulnerabilities using both symbolic and computational model approaches. Performance of EDHOC is also the subject of different academic studies. Authors also examined the performance of the OSCORE and Group OSCORE protocols in different IoT scenarios. This paper additionally presents the evaluation of protocol message sizes in different configurations. Even when authentication based on asymmetric keys in EDHOC is compared with authentication based on symmetric pre-shared keys in DTLS 1.3, we have shown a $\times 5$ reduction in the total number of bytes transmitted over the wire. Protocols protecting application data present a similar overhead than that of the DTLS 1.3 Record protocol, but additionally enable end-to-end protection of application data even in the presence of intermediaries and also in group communication setups. Through the evaluation of message sizes in different configurations, we have shown how these protocols are tailored for the constrained Internet-of-Things environments with limited maximum transmission units.

We have demonstrated a significant reduction in message overhead for key establishment. This reduction in message overhead has three main implications: 1) it often avoids the need for fragmentation, leading to smaller and simpler code base; 2) it reduces the energy consumption due to less bytes to transmit; 3) it reduces the associated transmission delays. For these reasons, we recommend the usage of the presented security protocols for Internet-of-Things environments such as LoRaWAN, 6TiSCH and NB-IoT, where the maximum transmission units or the available energy budget are limited.

Finally, each use case requires specific security properties. While the protocols discussed in this article provide fundamental security services, future work concerns their extension for more advanced cases. This includes secure enrollment, certificate revocation and remote attestation.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers and the Editor-in-Chief for their insightful comments and suggestions, which have helped to improve the technical and editorial quality of the manuscript and also would like to thank the participants of the IETF Working Groups ACE, CoRE, COSE, and LAKE, for their input during the standardization process.

REFERENCES

[1] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui, and T. Watteyne, "Understanding the limits of LoRaWAN," *IEEE Commun. Mag.*, vol. 55, no. 9, pp. 34–40, Sep. 2017.

[2] X. Vilajosana, T. Watteyne, T. Chang, M. Vučinić, S. Duquennoy, and P. Thubert, "IETF 6TiSCH: A tutorial," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 1, pp. 595–615, 1st Quart., 2020.

[3] O. Liberg, M. Sundberg, E. Wang, J. Bergman, J. Sachs, and G. Wikström, *Cellular Internet of Things: From Massive Deployments To Critical 5G Applications*. New York, NY, USA: Academic, 2019.

[4] *Lightweight Machine to Machine Technical Specification: Core*, document Version 1.2, Open Mobile Alliance, 2020.

[5] Z. Shelby, K. Hartke, and C. Bormann, *The Constrained Application Protocol (CoAP)*, Standard RFC7252, Internet Eng. Task Force (IETF), 2014.

[6] C. Bormann and P. E. Hoffman, *Concise Binary Object Representation (CBOR)*, Standard RFC8949, Internet Eng. Task Force (IETF), 2020.

[7] T. Bray, *The JavaScript Object Notation (JSON) Data Interchange Format*, Standard RFC8259, Internet Eng. Task Force (IETF), 2017.

[8] J. Schaad, *CBOR Object Signing and Encryption (COSE): Structures and Process*, Standard RFC9052, Internet Eng. Task Force (IETF), 2022.

[9] J. Schaad, *CBOR Object Signing and Encryption (COSE): Initial Algorithms*, Standard RFC9053, Internet Eng. Task Force (IETF), 2022.

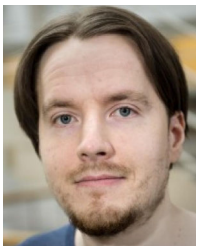
[10] J. Schaad, *CBOR Object Signing and Encryption (COSE): Countersignatures*, Standard RFC9338, Internet Eng. Task Force (IETF), 2022.

[11] J. P. Mattsson, G. Selander, S. Raza, J. Höglund, and M. Furuhed, *CBOR Encoded X.509 Certificates (C509 Certificates)*, document draft-ietf-cose-cbor-encoded-cert-09 (work-in-progress), Internet Eng. Task Force (IETF), 2024.

[12] G. Selander, J. P. Mattsson, and F. Palombini, *Ephemeral Diffie–Hellman Over COSE (EDHOC)*, Standard RFC9528, Internet Eng. Task Force (IETF), 2024.

[13] H. Krawczyk, "SIGMA: The 'SIGn-and-MAc' approach to authenticated Diffie–Hellman and its use in the IKE protocols," in *Proc. Annu. Int. Cryptol. Conf.* Santa Barbara, CA, USA: Springer, 2003, pp. 400–425.

[14] M. Vučinić, G. Selander, J. P. Mattsson, and T. Watteyne, "Lightweight authenticated key exchange with EDHOC," *Computer*, vol. 55, no. 4, pp. 94–100, Apr. 2022.
[15] G. Selander, J. P. Mattsson, F. Palombini, and L. Seitz, *Object Security for Constrained RESTful Environments (OSCORE)*, Standard RFC8613, Internet Eng. Task Force (IETF), 2019.
[16] F. Palombini, M. Tiloca, R. Höglund, S. Hristozov, and G. Selander, *Using Ephemeral Diffie–Hellman Over COSE (EDHOC) With the Constrained Application Protocol (CoAP) and Object Security for Constrained RESTful Environments (OSCORE)*, document draft-ietf-core-oscore-edhoc-10 (work-in-progress), Internet Eng. Task Force (IETF), 2023.
[17] R. Höglund and M. Tiloca, *Key Usage Limits for OSCORE*, document draft-ietf-core-oscore-key-limits-02 (work-in-progress), Internet Eng. Task Force (IETF), 2024.
[18] R. Höglund and M. Tiloca, *Key Update for OSCORE (KUDOS)*, document draft-ietf-core-oscore-key-update-07 (work-in-progress), Internet Eng. Task Force (IETF), 2024.
[19] M. Tiloca, G. Selander, F. Palombini, J. P. Mattsson, and R. Höglund, *Group Object Security for Constrained RESTful Environments (Group OSCORE)*, document draft-ietf-core-oscore-groupcomm-21 (work-in-progress), Internet Eng. Task Force (IETF), 2024.
[20] M. Tiloca, R. Höglund, P. van der Stok, and F. Palombini, *Admin Interface for the OSCORE Group Manager*, document draft-ietf-ace-oscore-gm-admin-11 (work-in-progress), Internet Eng. Task Force (IETF), 2024.
[21] M. Tiloca, J. Park, and F. Palombini, *Key Management for OSCORE Groups in ACE*, document draft-ietf-ace-key-groupcomm-oscore-16 (work-in-progress), Internet Eng. Task Force (IETF), 2023.
[22] J. P. Mattsson, F. Palombini, and M. Vučinić, *Comparison of CoAP Security Protocols*, document draft-ietf-iotops-security-protocol-comparison-06 (work-in-progress), Internet Eng. Task Force (IETF), 2024.
[23] C. Jacomme, E. Klein, S. Kremer, and M. Racouchot, "A comprehensive, formal and automated analysis of the EDHOC protocol," in *Proc. 32nd USENIX Secur. Symp.*, Anaheim, CA, USA, Aug. 2023, pp. 5881–5898.
[24] K. Norrman, V. Sundararajan, and A. Bruni, "Formal analysis of EDHOC key establishment for constrained IoT devices," in *Proc. 18th Int. Conf. Secur. Cryptography*, 2021, pp. 210–221.
[25] A. Bruni, T. S. Jørgensen, T. G. Petersen, and C. Schürmann, "Formal verification of ephemeral Diffie–Hellman over COSE (EDHOC)," in *Security Standardisation Research* (EDHOC). Cham, Switzerland: Springer, 2018, pp. 21–36.
[26] B. Cottier and D. Pointcheval, "Security analysis of improved EDHOC protocol," in *Proc. Int. Symp. Found. Pract. Secur.* Cham, Switzerland: Springer, 2022, pp. 3–18.
[27] K. Sagonas and T. Typaldos, "EDHOC-Fuzzer: An EDHOC protocol state fuzzer," in *Proc. 32nd ACM SIGSOFT Int. Symp. Softw. Test. Anal.* New York, NY, USA: Association for Computing Machinery, Jul. 2023, pp. 1495–1498, doi: 10.1145/3597926.3604922.
[28] S. Pérez, J. L. Hernández-Ramos, S. Raza, and A. Skarmeta, "Application layer key establishment for end-to-end security in IoT," *IEEE Internet Things J.*, vol. 7, no. 3, pp. 2117–2128, Mar. 2020.
[29] L. P. Fraile, A. Fournaris, and C. Koulamas, "Design and performance evaluation of an embedded EDHOC module," in *Proc. 10th Medit. Conf. Embedded Comput. (MECO)*, Jun. 2021, pp. 1–6.
[30] S. Hristozov, M. Huber, L. Xu, J. Fietz, M. Liess, and G. Sigl, "The cost of OSCORE and EDHOC for constrained devices," in *Proc. 11th ACM Conf. Data Appl. Secur. Privacy*. New York, NY, USA: Association for Computing Machinery, Apr. 2021, pp. 245–250.
[31] M. Gunnarsson, J. Brorsson, F. Palombini, L. Seitz, and M. Tiloca, "Evaluating the performance of the OSCORE security protocol in constrained IoT environments," *Internet Things*, vol. 13, Mar. 2021, Art. no. 100333.
[32] M. Gunnarsson, K. M. Malarski, R. Höglund, and M. Tiloca, "Performance evaluation of group OSCORE for secure group communication in the Internet of Things," *ACM Trans. Internet Things*, vol. 3, no. 3, pp. 1–31, Jul. 2022, doi: 10.1145/3523064.
[33] M. B. Jones, J. Bradley, and N. Sakimura, *JSON Web Signature (JWS)*, Standard RFC7515, Internet Eng. Task Force (IETF), 2015.
[34] M. B. Jones and J. Hildebrand, *JSON Web Encryption (JWE)*, Standard RFC7516, Internet Eng. Task Force (IETF), 2015.
[35] M. B. Jones, *JSON Web Key (JWK)*, Standard RFC7517, Internet Eng. Task Force (IETF), 2015.
[36] S. Boeyen, S. Santesson, T. Polk, R. Housley, S. Farrell, and D. Cooper, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, Standard RFC5280, Internet Eng. Task Force (IETF), 2008.
[37] R. T. Fielding, M. Nottingham, and J. Reschke, *HTTP/1.1*, Internet Eng. Task Force (IETF), 2022.
[38] E. Rescorla, H. Tschofenig, and N. Modadugu, *The Datagram Transport Layer Security (DTLS) Protocol Version 1.3*, Standard RFC9147, Internet Eng. Task Force (IETF), 2022.
[39] E. Rescorla, *The Transport Layer Security (TLS) Protocol Version 1.3*, Standard RFC8446, Internet Eng. Task Force (IETF), 2018.
[40] K. Seo and S. Kent, *Security Architecture for the Internet Protocol*, Standard RFC4301, Internet Eng. Task Force (IETF), 2005.
[41] G. Gross, B. Weis, and D. Ignjatic, *Multicast Extensions to the Security Architecture for the Internet Protocol*, Standard RFC5374, Internet Eng. Task Force (IETF), 2008.
[42] C. Kaufman, P. E. Hoffman, Y. Nir, P. Eronen, and T. Kivinen, *Internet Key Exchange Protocol Version 2 (IKEv2)*, Standard RFC7296, Internet Eng. Task Force (IETF), 2014.
[43] V. Smyslov and B. Weis, *Group Key Management Using IKEv2*, document draft-ietf-ipsecme-g-ikev2-11 (work-in-progress), Internet Eng. Task Force (IETF), 2024.

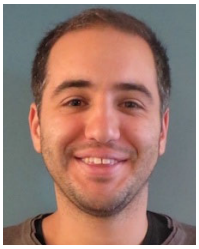


RIKARD HÖGLUND received the bachelor's degree in computer science from Dalarna University, Borlänge, Sweden, and the master's degree in information and communication technology from the KTH Royal Institute of Technology, Stockholm, Sweden. He is currently pursuing the Industrial Ph.D. degree with Uppsala University, Sweden. He has been a Researcher with the Cybersecurity Unit, RISE Research Institute of Sweden, Stockholm, since 2014. He performed his master's thesis project at SICS Swedish ICT, now part of RISE, and subsequently joined as a Researcher with the Cybersecurity Unit, where he has been working since then. He works especially in research areas concerning cybersecurity, with a focus on IoT security, computer networking, and communication protocols. He has been actively involved in the implementation and experimental evaluation of IoT security and communication protocols and in prototype development and their integration into larger software releases. He has contributed as a Designer and a Developer of security services in a number of industrial direct assignments and research projects.

MARCO TILOCA received the Ph.D. degree in computer engineering from the University of Pisa, in 2013. He is currently a Senior Researcher with the Cybersecurity Unit, RISE Research Institutes of Sweden, Stockholm, Sweden. His research interests include network and communication security, security in the Internet of Things, secure group communication, key management, and access control. He has long-time experience in European and national research and development projects, where he was a National Coordinator, a Technical Coordinator, and the Work Package Leader. He is actively involved in standardization activities under the premier international body internet engineering task force (IETF), where he has also been working as the Chair of the Working Group Constrained RESTful Environments (CoRE).

GÖRAN SELANDER received the M.Sc. degree in physics from Stockholm University and the Ph.D. degree in mathematics from the Royal Institute of Technology, Sweden. He is currently a Principal Researcher with Ericsson Research. He has over 20 years' experience in security research and standardization and over ten years, with a focus on cyber-physical systems and development of lightweight security solutions targeting resource efficient processing and communication. He has worked with security protocols and enablers in different technology areas contributing to standardization in the IETF, 3GPP, OMA, Bluetooth SIG, and GlobalPlatform.

JOHN PREUß MATTSSON received the M.Sc. degree in engineering physics from the KTH Royal Institute of Technology, Sweden, and the M.Sc. degree in business administration and economics from Stockholm University. He is currently an Expert in cryptographic algorithms and security protocols with Ericsson Research, Stockholm, Sweden. His work focuses on applied cryptography, security protocols, privacy, the IoT security, post-quantum cryptography, and trade compliance. During his 15 years with Ericsson, he has worked with a lot of different technology areas and been active in many security standardization organizations, including IETF, IRTF, 3GPP, GSMA, and NIST, where he has significantly influenced internet and cellular security standards. In addition to designing new protocols, he has also found significant attacks on many algorithms and protocols.



MALIŠA VUČINIĆ (Senior Member, IEEE) received the Engineering degree from the University of Montenegro, in 2010, the joint master's degree (Hons.) from Politecnico di Torino and the Grenoble Institute of Technology, in 2012, and the Ph.D. degree from Grenoble Alps University, in 2015. He is currently a Research Scientist with the AIO Team, Inria Research Institute, Paris, France. In parallel with the Ph.D. studies, he was a Research Engineer with STMicroelectronics and a Visiting Scholar with the University of California at Berkeley, in 2015. He co-chairs the IETF LAKE working group which standardizes a lightweight authenticated key exchange protocol for Internet of Things use cases. He led the security standardization work in IETF 6TiSCH, specifying the constrained join protocol (CoJP) for 6TiSCH networks. He is also a Core Developer of the OpenWSN Project, the reference 6TiSCH implementation, and the Co-Lead of the 6TiSCH Simulator. He is fluent in English, French, Italian, and Montenegrin.

THOMAS WATTEYNE (Senior Member, IEEE) received the Ph.D. degree in computer science, the M.Sc. degree in networking, and the M.Eng. degree in telecommunications from INSA Lyon, France, in 2008 and 2005, respectively. He is currently an Insatiable Enthusiast in low-power wireless technologies. He holds the research director position with the AIO Research Team, Inria Paris, where he leads a team that designs, models, and builds networking solutions based on a variety of Internet of Things (IoT) standards. He is also a Wireless System Architect with Analog Devices, the Undisputed Leader of Supplying Low Power Wireless Mesh Networking Solutions for Critical Applications for Industrial and Beyond. Since 2013, he has bene co-chairs the IETF 6TiSCH working group, which standardizes how to use IEEE802.15.4e TSCH in IPv6-enabled mesh networks and is a member of the IETF Internet of Things Directorate. In 2019, he co-founded Wattson Elements, the company that develops the award-winning Falco marina management solution. He was the Postdoctoral Research Lead of the Prof. Kristofer Pister's Team, University of California at Berkeley. He founded and co-leads Berkeley's OpenWSN project, an open-source initiative to promote the use of fully standards-based protocol stacks for the IoT. Between 2005 and 2008, he was a Research Engineer with France Telecom, Orange Laboratories. He is fluent in four languages.