Application Layer Security Protocols for the Internet of Things

Göran Selander, Francesca Palombini, John Mattsson, and Ludwig Seitz

***Abstract*—The Internet of Things (IoT) has a tremendous impact on our society, and provides challenges in terms of safety and security. Although the IoT presents a large variety of deployment settings, many security objectives are identical and possible to address with a common set of security protocols. However, current standards are not optimized for connected embedded devices, and a new set of security protocols is being developed in the Internet community. This paper describes the IoT application layer security suite which are lightweight protocols providing end-to-end security for HTTP-like operations over any transport and also supporting secure group communications.**

I. Introduction

THE Internet of Things (IoT) consists of a large variety of connected devices in diverse deployment settings, with devices ranging from relatively powerful to very constrained in terms of processing data rates, memory, power, bandwidth, etc. [1]. The safety and security of the IoT has a tremendous impact on our society, and remains a challenge:

- The things being connected in the IoT are the physical objects surrounding us and with which we interact in our daily life. As a consequence they may have a significant impact on us and our society when disabled or compromised, and furthermore can in many cases not be protected from external tampering, e.g., by being confined to locked rooms;
- The sheer number of things being connected is estimated to exceed 20 billions by 2020 [2], which is higher than any previous set of connected devices. This creates both a larger attack surface and, when compromised, a larger base to launch attacks from, as demonstrated by the recent denial of service attacks perpetrated by IoT botnets [3];
- The IoT inherits the combined security challenges of embedded devices and of the Internet. While existing security protocols and mechanisms are suitable for powerful Internet-connected devices such as mobile phones, more constrained devices and networks require solutions adapted to their capabilities;
- The large variety of IoT deployments leads to different security solutions being developed in different industry segments, which is both costly and more prone to security errors than it would be to apply a suitable generic security standard instead.

Although deployments may be very different many security objectives are common, including: secure onboarding of new devices in a network, mutual authentication of network nodes, encrypted and integrity protected communication, secure group communication, securing actuator commands and protection against unauthorized access. All these objectives have an impact on the security protocols used between the nodes in the network, and since current security protocols do not sufficiently address the issues of connected embedded devices, it is desirable to standardize new IoT security protocols which are suitable for a wider range of deployments.

This paper presents work in progress on standardization of application layer security protocols for constrained devices within the Internet Engineering Task Force (IETF). It should be noted that since this paper provides a snapshot of the state of standards proposals as of June 2017, the reader who wants to know more details about referenced draft standards should consult the latest version of the corresponding specification.

Specifically addressing the challenges above: the focus in this paper is on security protocols, and there are obvious safety aspects implied, for example, in securing actuator operations. If default-password authentication had been replaced with a security protocol suitable for IoT devices implementing stronger authentication, then the mass-scale compromise in [3] would not have been possible. However, the scope of these security protocols is primarily to protect against attackers that fall under the Dolev Yao model [4]; intrusion or malware in endpoints are not in scope of this paper.

The rest of this paper is organized as follows: section II introduces the concept of constrained environments; section III discusses the security design objectives, in particular different protocol layers; section IV describes the existing technologies designed for constrained devices and networks; section V outlines the new IoT security protocols developed in the IETF; finally, section VI illustrates some applications of these protocols, while section VII gives a summary and concludes this document.

II. Security for constrained environments

During the last decade, new Internet protocols have been standardized addressing the requirements of constrained devices and networks [1]. One notable example is the Constrained Application Protocol (CoAP [5]) which is similar to HTTP but adapted to constrained environments (Section IV-B).

This paper targets security protocols suitable for constrained environments, and in this section we provide some examples of how protocols are affected by these limiting conditions. Solutions derived from constrained device requirements can be used seamlessly on non-constrained devices thus enabling a wide range of deployments. Even if a device is not constrained

G. Selander, F. Palombini, and J. Mattsson are at Ericsson Research
L. Seitz is at RISE SICS

in all aspects considered here, a low security overhead is always desirable to get the highest performance out of the available resources. Therefore the constrained environment requirements are relevant for many deployments, even outside the scope of IoT. We emphasize that the security protocols suitable for constrained environments discussed here do not compromise on security in comparison with less constrained security protocols. The optimizations made are in terms of specific performance characteristics and making other non-security trade-offs – all for the purpose to adapt to constrained device conditions.

The constrainedness of IoT devices can take many different forms, such as, for example, slow processing units, small amount of RAM or persistent memory, battery powered or energy harvesting devices which need to run for decades and therefore must save on power consumption. Interestingly, the processing power in even the most constrained devices in general does not prevent the use of state-of-the-art cryptographic algorithms and key lengths [6]. Indeed, hardware implementation of standard algorithms are common on many chipsets and also modern crypto algorithms are often more efficient than older ones, e.g., the older signature algorithm ECDSA [7] is outperformed by the newer and more secure EdDSA [8]. Power consumption requirements, however, may have a direct implication on network protocols and architecture, which in turn impacts the security solution:

- In case of wireless communication, transmitting or listening for messages to receive may consume relatively large amounts of energy [9], as do high data rates. As a consequence, protocols must comply with a very restricted budget in terms of message size and number of messages exchanged, and this applies also to security protocols.
- To save power when idle, devices can turn off the radio interface and go to sleep or enter a low-power mode. As a consequence, IoT devices are not always reachable, and intermediary nodes such as gateways or proxies perform load-balancing or store-and-forward tasks, which are used to support settings of intermittent connectivity. Performing intermediary node functionality efficiently requires the proxy to read or change the message metadata, which has consequences on end-to-end security including privacy.
- Maintaining state in a device has a cost, both in terms of storage and in terms of writing the state information to persistent memory. In settings where power is intermittent, only persistent memory can be relied upon. Security protocols need to consider the effect of losing state and restarting to prevent attack vectors based on power loss.
- A constrained device may not be capable of maintaining a synchronized time, which impacts the ability to verify validity of assertions, e.g., certificates or access control policies. However, even devices with a strict power budget and that almost always sleep are not prevented from having a local sense of time, since clocks are commonly available and require less power than the natural discharge of batteries, even when the device is sleeping[1]. Since clocks may be drifting and a dedicated synchronization process may be costly, security protocols need to provide suitable freshness information.

III. Design objectives

A. Security at what layer?

Applying security at different layers of a protocol stack provides different security properties. Applying security at one layer protects the data of higher layers from being accessed at lower layers of the stack. Hence the lower in the stack protection is applied, the more is protected. On the other hand, by protecting a high layer in the protocol stack, independence of underlying layers may be achieved, which makes the protocol applicable to a wider set of interconnecting technologies.

In many cases security protocols are applied on more than one layer to comply with the security objectives of a particular deployment, but for efficiency reasons it is often necessary to restrict the number of security solutions on different layers.

- Security at a low layer, say at the data link layer, enables early rejection of invalid messages, reducing unnecessary processing and mitigating Denial-of-Service attacks. However, link layer security only protects one out of potentially many hops in the communication between the endpoints.
- Communication security at transport layer (TLS/DTLS [10], [11]) is the default security solution on the Internet. The protocol messages are protected across different link layers, but since TLS encrypts higher layer information, application layer protocol metadata (e.g., HTTP/CoAP method, URL, etc.) cannot be read by, e.g., a proxy. For constrained deployments that rely on proxy operations, e.g., for store and forward during sleep cycles, end-to-end transport layer security is thus broken at the proxy. Other issues with transport layer security in constrained environments are previously studied. Lithe [13] describes a header compression mechanism for DTLS in order to reduce the node's energy consumption. OSCAR [12] specifies an object-based security architecture removing the notion of state between communicating entities, but uses DTLS for key exchange.
- If security is applied above transport layer then the protected message can be carried also over different transport protocols (e.g., UDP, TCP, SMS) with preserved integrity. There are two flavors of application layer security:
  1) Only application layer payload is protected. In this case, the intermediate nodes can arbitrarily read or change the application layer metadata. If the payload in itself contains all the information that needs to be protected, then this provides the right level of protection. One example of solution is encapsulating payload with JOSE/COSE (see section V-A).
  2) Application layer payload and metadata are protected. In this case, the security protocol can be designed to strike the balance between making the

[1] See, e.g., https://tinyurl.com/y86j653g for power consumption calculations

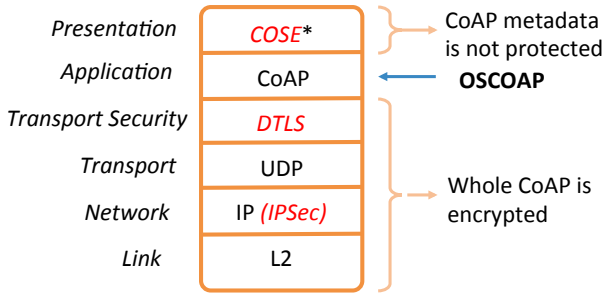

Fig. 1. Example of security on different layers

application layer payload useable by proxies, and at the same time protect sensitive metadata between the endpoints. One example of this is the use of OSCOAP [14] together with CoAP (see section V-B).

Fig. 1 shows a comparison of what is protected by security protocols at different layers of a constrained device protocol stack.

B. Other design principles

- Use optimized primitives, whenever possible. The use of CBOR for encoding, COSE for secure message formatting and CoAP for messaging provides compact messages, lightweight processing and yet a rich platform for designing a large variety of protocols.
- Reuse primitives to save on code space.
- Reduce the number of options in the protocols, to enable lower complexity implementations, which reduces code size and the risk for errors.

IV. Existing Technologies

In this section we outline two IETF standards designed for constrained devices and networks. The new proposed security standards presented in section V are based on the existing technologies described in this section.

A. CBOR

Concise Binary Object Representation (CBOR) [15] is a binary data format that allows compact serialization paired with an extremely small code size. Its data model is similar to the JavaScript Object Notation (JSON) data model, with some differences that come from providing a data format adapted for constrained environments. CBOR has other main advantages compared to JSON: CBOR representation allows to encode and decode the most common data formats used in Internet standards, such as the basic data types (strings, numbers, etc.), with the addition of byte strings, and supports JSON common structures arrays and trees. Analogously to JSON, CBOR data is self-describing, and does not need a schema description. Moreover, CBOR was designed with a constrained environment in mind, which makes the decoding and encoding of data reasonably frugal in CPU usage. The main advantages of CBOR are the very small code size of encoder and decoder,

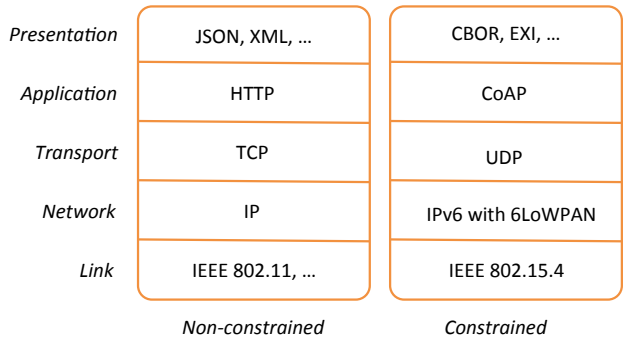

Fig. 2. Example protocol stacks for unconstrained and constrained devices.

and compactness of the data serialization. In fact, other binary formats such as ASN.1 DER, BER, or PER are not particularly compact, and the code overhead is significant, or does not allow for the extensibility that CBOR provides.

B. CoAP

The Constrained Application Protocol [5] is a RESTful web protocol designed to work within constrained environments, especially considering machine-to-machine applications. It provides a subset of HTTP REST services, and additional features, such as built-in discovery, multicast support and asynchronous message exchange. Another fundamental feature supported by CoAP is the proxy and caching capabilities, which allow to improve network performance and provide access to devices that are not constantly online. These features, together with the low message overhead and parsing complexity, give CoAP a major advantage compared with HTTP in the case of constrained environments. CoAP is designed to run over UDP, but can run over any underlying protocol, such as SMS, TCP or SCTP, or even directly on link layer [16]. Fig. 2 shows a comparison of protocol stacks for the web and for constrained environments.

V. Security protocols

In this section we describe the new IoT application layer security protocols developed in the IETF.

A. COSE

The CBOR data format is used in the CBOR Object Signing and Encryption (COSE) specification [17]. COSE provides a way to have basic security services for the CBOR data format (secure wrapping of data). Analogously to the JOSE family of specifications, including JSON Web Signature [18] and JSON Web Encryption [19], COSE describes how to create and process encryption, signatures and message authentication codes (MACs), using CBOR data serialization, so to create a secure CBOR object, called a COSE object. In practice, the COSE object is composed at least by a header part including the necessary parameters to perform the security processing

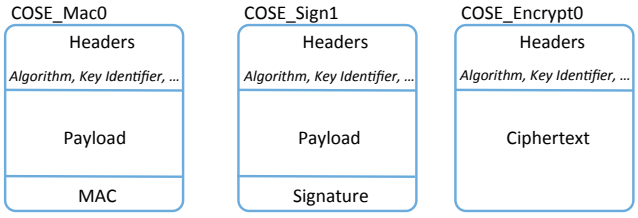

Fig. 3. Structure of different COSE objects.

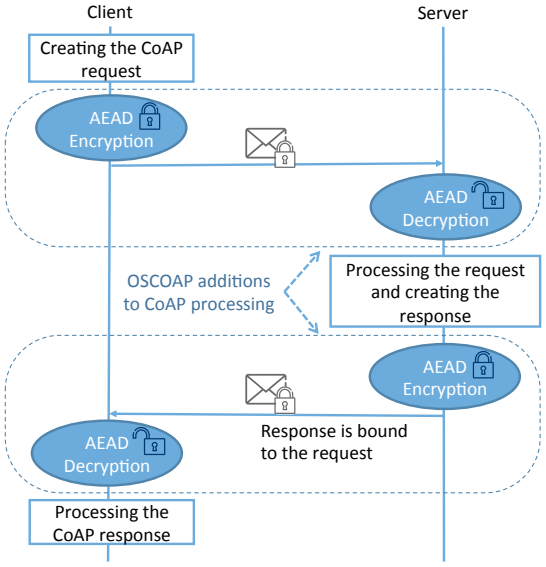

Fig. 4. Overview of the CoAP with OSCOAP processing flow.

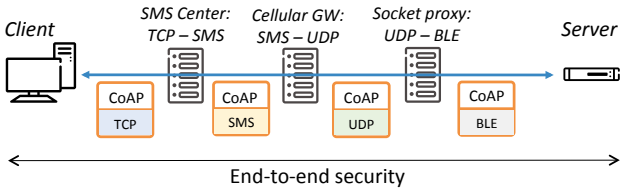

Fig. 5. Proxy functionality with OSCOAP.

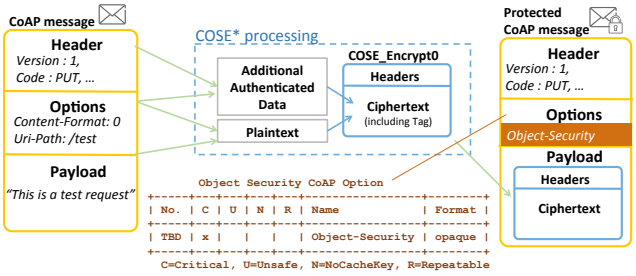

Fig. 6. Protecting an OSCOAP message.

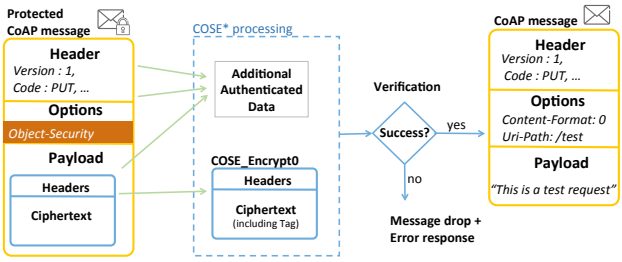

Fig. 7. Verifying an OSCOAP message.

(such as the key identifier, algorithm identifier, etc.) and by the content of the message (plain text or ciphertext). Fig. 3 shows the structure of the different COSE objects.

B. Object Security of CoAP (OSCOAP)

The OSCOAP (Object Security of CoAP) protocol [14] defines a way to secure the CoAP communication between two endpoints, in the presence of untrusted intermediaries. On a high level, the sensitive parts of the CoAP message are protected using COSE, creating a secure object. This secure object is added to a modified CoAP message and sent over CoAP. OSCOAP defines exactly what parts of CoAP are to be protected, and what processing the endpoints must execute to create protected messages and to verify such messages they receive. Fig. 4 shows the OSCOAP processing flow.

The CoAP specification [5] defines the binding to the Datagram Transport Layer Security (DTLS) over UDP, and a separate specification [20] defines the binding to TLS over TCP. As discussed in section III-A if CoAP is protected on transport layer, all the parties in the communication are required to trust every intermediary, or alternatively remove important CoAP capabilities such as proxying.

In contrast, since OSCOAP is defined in terms of CoAP it can run over any underlying protocol and at the same time allow certain proxy functionality, see Fig. 5. The security requirements for OSCOAP are specified in [21].

Inline with best current security practice, OSCOAP uses authenticated encryption with additional data (AEAD) to protect the messages. Each endpoint using OSCOAP to communicate securely has to store a set of information elements necessary to carry out the cryptographic operations, called the security context, including keys, initialization vectors etc. In particular, the two directions of a CoAP transaction (request and response) are protected using two different sets of keying material. Fig. 6 shows the creation of OSCOAP messages, and Fig. 7 shows the verification of such messages.

OSCOAP, like DTLS, makes use of sequence numbers to provide freshness and ordering of messages. In contrast to DTLS, the sequence numbers are encoded with CBOR for compact representation. Additionally, OSCOAP also provides a binding between the request and the response, securing against replay of previously sent messages, increases security and reduces the message overhead in the response. Specifically, the use of different keys and IV only differing on one bit between request and response enables the omission of sequence number in the response.

OSCOAP is also designed to handle sudden loss of power or RAM without requiring establishment of a new security context, by a combined strategy of intermittently using persistant storage and synchronizing security state. The message overhead resulting from the use of OSCOAP is very low. In Fig. 8, the overhead of OSCOAP is compared to TLS and

Protocol	*Overhead (Bytes) Seq Num = '05'*	*Overhead (Bytes) Seq Num= '1005'*	*Overhead (Bytes) Seq Num = '100005'*
DTLS 1.2	29	29	29
DTLS 1.3	21	21	21
TLS 1.2	21	21	21
TLS 1.3	21	21	21
DTLS 1.2 (GHC)	16	16	17
DTLS 1.2 (Raza)	13	13	14
DTLS 1.3 (GHC)	14	14	15
DTLS 1.3 (Raza)	13	13	14
TLS 1.2 (GHC)	17	18	19
TLS 1.3 (GHC)	17	18	19
OSCOAP Request	13	14	15
OSCOAP Response	9	9	9

Fig. 8. Overhead comparison of between DTLS, TLS, and OSCOAP.

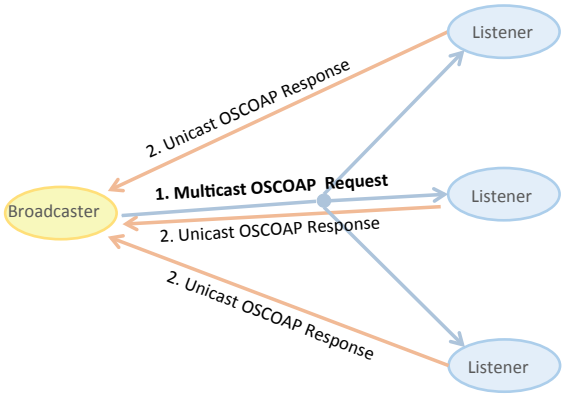

Fig. 9. Overview of OSCOAP multicast.

DTLS record layer, with and without compression algorithms (GHC = Generic Header Compression [22], Raza = [23])

OSCOAP also secures CoAP group communication, e.g., multicast [24]. In fact, using OSCOAP to protect end-to-end group communication is possible with little to no modification to the COSE wrapping or the processing of messages. In this case, each broadcaster would have a security association (or shared keying material) with any listener in the group, and each response to a multicast request would be securely associated to the corresponding request. Fig. 9 shows an example of using OSCOAP for securing multicast with unicast responses.

OSCOAP is not bundle with a key exchange protocol (such as EDHOC), which reduces code size for implementations that use other key distribution mechanisms, e.g., ACE.

C. Ephemeral Diffie-Hellman Over COSE (EDHOC)

In order for a communication session to provide forward secrecy, the communicating parties can run an Elliptic Curve Diffie-Hellman (ECDH) key exchange protocol with ephemeral keys, from which shared key material can be derived. Ephemeral Diffie-Hellman Over COSE (EDHOC) [25] is a compact and lightweight authenticated Diffie-Hellman key exchange protocol with ephemeral keys.

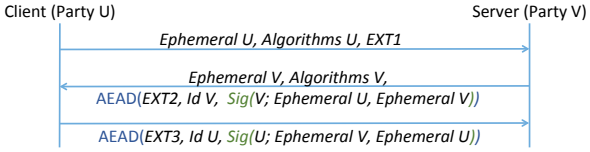

Fig. 10. The EDHOC protocol for asymmetric keys (simplification).

EDHOC is built on an AEAD based variant of the well-known SIGMA (SIGn-and-MAc) [26] protocol and designed to encrypt and integrity protect as much information as possible, in particular the identities. Authentication is based on credentials established out of band, e.g., from a trusted third party, such as an Authorization Server as specified by ACE. EDHOC supports authentication using pre-shared keys, ”raw” public keys [34], and certificates.

EDHOC messages are encoded with CBOR and COSE and designed to be as compact and lightweight as possible, in terms of message sizes, processing, and the ability to reuse already existing CBOR and COSE libraries. EDHOC does not put any requirement on the lower layers and can therefore also be used, e.g., in environments without IP. The three EDHOC messages maps directly to the three messages in SIGMA-I and protected application data can be sent in parallel with the third message. The application data may, e.g., be protected using the AEAD algorithm negotiated in EDHOC. EDHOC allows application-defined extensions to be sent in the respective messages.

Some of the differences compared to the TLS handshake include a different procedure to mitigate bidding-down attacks, allowing the verification that the best common curve for ECDH. EDHOC also allows the peers to select an explicit identifier of the resulting security context, enabling locally unique compact identifier. EDHOC is independent on transport protocol, it may, e.g., be used with CoAP but without IP.

A main use case for EDHOC is to establish a shared security context for OSCOAP. The EDHOC protocol enables the establishment of OSCOAP input parameters with the property of forward secrecy and negotiation of key derivation function and AEAD, it thus provides all necessary pre-requisite steps for using OSCOAP.

Fig. 10 shows a simplified version of the EDHOC protocol in the case of asymmetric keys. The client and server exchange signed ephemeral public keys and MAC:d identities (certificates or raw public keys), and derive the shared secret keys used in the AEAD calculations as well as the output of the protocol. The protocol also includes negotiation of algorithms and placeholders for opaque application specific data (”EXT”).

D. The ACE Framework

IoT deployments require a fine-grained and flexible access control solution which restricts access to sensitive resources on a need-to-know/need-to-change basis, and must be suitable for constrained endpoints. The ACE (Authentication and Authorization in Constrained Environments) framework [27] provides such a solution, and is based on the widely used web authorization protocol OAuth 2.0 [28]. The ACE framework is designed to adapt OAuth mechanisms to the requirements

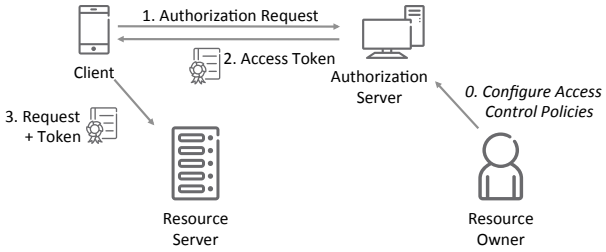

Fig. 11. ACE framework communication flows.

of constrained devices and to the predominance of machine-to-machine (M2M) communication in IoT use cases.

The two main requirements these adaptations addresses are the reduction of network communication (both number and size of messages), and the fact that either client or resource server (RS) may have intermittent connectivity with back-end services like the Authorization Server (AS).

To this end the following modifications to the plain OAuth protocol are defined:

- The AS provides proof-of-possession keys linked to the access token to both client and RS (in case of intermittent connectivity for one party, the other is used as intermediary).
- The AS specifies which communication security profile client and RS must use.
- The RS provides a resource that the client can send its access token to.
- All messages are encoded in CBOR instead of JSON, in order to reduce message sizes.

Since the other OAuth grant flows involve manual intervention by the resource owner (RO), the ACE framework focuses on the client credentials grant, where the client can request an access token directly at the AS using just its own credentials. In this flow it is assumed that the RO has defined access control policies at the AS that define which kinds of access tokens can be granted to requesting clients. Fig. 11 shows the basic flow of the ACE framework with the client credentials grant, with the following steps:

0) The RO configures access control policies at the AS, thus authorizing it to issue access tokens for the RO's resources, based on certain client credentials.
1) The client requests an access token from the AS based on its own credentials.
2) After verifying that the request is authorized, the AS issues the requested access token.
3) The client sends its access token and its request to the RS. This can be two separate communication steps or combined together.

1) Profiles of the ACE Framework: Since IoT consists of many different devices using a multitude of communication protocols, the ACE framework cannot be restricted to one communication stack. Therefore the ACE specifications are divided into the communication protocol agnostic framework, and several specific profiles that define adaptations for specific communication security suites.

Currently the following profiles are in the process of being specified: CoAP over DTLS [29], CoAP with OSCOAP [30], CoAP Publish-Subscribe [31], and MQTT over TLS [32].

The profiles' main tasks are to specify the communication protocol and the communication security protocol that is used between client and RS. Moreover, they also define how the parties authenticate and how the proof-of-possession of the access token is performed.

For example in the CoAP over DTLS profile, CoAP is specified as communication protocol and DTLS as communication security protocol, furthermore the authentication is taken care of as part of the DTLS handshake, and proof-of-possession is performed by using the proof-of-possession key in the DTLS handshake, e.g., as pre-shared key [33] or as raw public key [34].

E. Security for Actuators

In autonomous cyber-physical systems, integrity and availability become more important than confidentiality. Losing control of locks, vehicles, or medical equipment is far worse than having someone eavesdrop on them, and there is an explicit or implicit safety component in many actuator use cases. Therefore, properties like message freshness, proximity, and channel binding also become essential, sometimes in unexpected ways. As a current example, consider the proximity-based security systems used in smart car keys, access cards, and contactless payment systems. While those systems all verify freshness, they do not verify proximity, so two attackers can relatively easily relay the signal from a device in a victim's pocket, gaining access to office buildings, opening and starting cars, or transferring money.

While an attacker delaying a request to a sensor is often not a security problem, an attacker delaying a request to an actuator performing an action is often a serious problem. A request to an actuator (for example a request to unlock a lock) is often only meant to be valid for a short time frame, and if the request does not reach the actuator during this short timeframe, the request should not be fulfilled. A solution must enable the server to verify that the request was received within a certain time frame after it was sent. This can be accomplished with either a challenge-response pattern, by exchanging timestamps, or by only allowing requests a short period after client authentication. More details are provided in [35].

VI. Applications

In this section we give some examples of how the new security protocols may be applied.

A. Device Joining a Network

When a new device seeks admission to a network, it needs to authenticate and get authorized for joining the network. This may require communicating with an AAA server over the network to which it does not yet have full access. This problem statement is well known and common solutions involve, e.g., port-based access control (802.1X) and the Extensible Authentication Protocol (EAP) [36]. An alternative lightweight

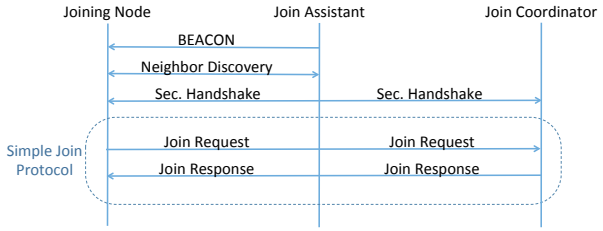

Fig. 12. 6TiSCH join protocol proposal using OSCOAP.

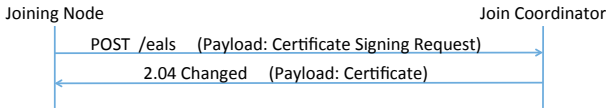

Fig. 13. An enrollment protocol proposal using OSCOAP.

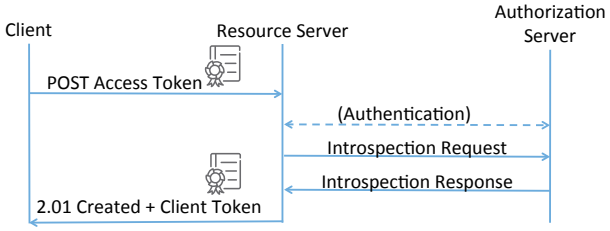

Fig. 14. The ACE Introspection message flow with Client Token

version is provided by the IETF 6TiSCH Minimal Security protocol [37], in which the authentication and authorization procedure is run over CoAP.

In the latter protocol, the Joining Node (JN) listens to beacons from a Join Assistant (JA) nodes in the network, and first run neighbor discovery. To communicate with the Join Coordinator (JC), the JN can use CoAP with the JA as stateless CoAP proxy. Using pre-established credentials in JN, and thanks to the end-to-end security of the application layer protocols over multiple hops, the JN and JC can communicate securely. Different join procedures are possible, two examples are provided:

1) Simple join [37]. The join protocol is an optional security handshake, provided by the EDHOC protocol (see section V-C), followed by an OSCOAP message exchange carrying the protected join request and join response messages. The join response contains a network wide key and configuration data required to connect to devices in the network. See Fig. 12.
2) Simple enrollment [38]. The simple enrollment protocol consists of an optional key establishment protocol followed by an OSCOAP message exchange: The client sends a Certificate Signing Request (e.g., PKCS#10 [39]) and receives a Public Key Certificate (e.g., PKCS#7 [40]) that is used for proving membership in this network. See Fig. 13. The key establishment protocol may be EDHOC (see V-C) or ACE (see V-D). In the latter case the Introspection flow using a Client Token may be used, see Fig. 14.

B. Lighting Control

Some IoT deployments require secure group communications between a broadcaster and multiple listeners. In the lighting use case, a set of lighting nodes (e.g., luminaires, wall-switches, sensors) are grouped together such that all authorized listeners in a group can verify the messages. For reasons of latency, message communication must happen in a synchronous manner.

Authorization of members to perform certain actions in a group is handled by a group manager, which mediates keys protecting the communication between group members. The group manager operations can be modeled as an ACE resource server, to which authorized members can request to acquire keys. For some settings a symmetric shared key can be used to protect the communication, e.g., using OSCOAP. In settings where source authentication is required, messages broadcasted in the group include a signature made with the broadcaster's private key. To verify the message the listeners need to acquire the broadcaster's public key.

In settings where the content of the broadcast message is sensitive, the combination of symmetric keys used for confidentiality and asymmetric keys used for authentication can be deployed using these security protocols in the same manner.

VII. Conclusions and Future Directions

Application layer security protocols suitable for the IoT are in development in the IETF. The application layer security protocol suite provides an attractive solution for protecting IoT deployments thanks to, for example, its end-to-end secure protection of RESTful interactions, its independence of transport mechanisms, its applicability to group communication and its lightweight operations. The IoT application layer protocol suite includes the following building blocks:

- COSE - a secure CBOR-based message format optimized for constrained device processing.
- OSCOAP - a lightweight security protocol built into CoAP enabling many deployment and communication scenarios, including multicast.
- EDHOC - a key exchange protocol with forward secrecy designed for constrained devices.
- ACE - an authorization framework suitable for IoT built on the industry standard OAuth 2.0.

Solutions designed for lightweight operations are applicable also for non-constrained deployments, consumes less resources and enables interoperability between a wide range of deployments.

The ongoing work provides a foundation for standardization of secure operations in constrained devices. Additional innovations optimizing other parts of the transactions are yet to be explored. For example, the certificate based authentication is still heavy due to legacy formats, and new certificate formats, as well as enrolment procedures, can significantly improve the performance. In general, decreasing the size of public key signatures would be an important contribution to reduce message overhead when asymmetric cryptography is used, and even more so with most post-quantum signature algorithms.

ACKNOWLEDGMENT

Ludwig Seitz and Göran Selander worked on this document as part of the CelticPlus project CyberWI, with funding from Vinnova.

REFERENCES

[1] C. Bormann, M. Ersue, and A. Keranen, "Terminology for Constrained-Node Networks," RFC 7228 (Informational), Internet Engineering Task Force, May 2014. [Online]. Available: http://www.ietf.org/rfc/rfc7228.txt

[2] P. Middleton, J. Hines, B. Tratz-Ryan, E. Goodness, D. Freeman, M. Yamaji, A. McIntyre, A. Gupta, D. Rueb, and T. Tsai, "Forecast: Internet of Things Endpoints and Associated Services, Worldwide, 2016," https://www.gartner.com/doc/3558917/forecast-internet-things--endpoints, December 2016, Gartner Report, ID G00321441.

[3] B. Krebs, "Krebsonsecurity Hit With Record DDos," https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-with-record-ddos/, accessed 2017-03-13.

[4] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Transactions on information theory*, vol. 29, no. 2, pp. 198–208, 1983.

[5] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," RFC 7252 (Proposed Standard), Internet Engineering Task Force, Jun. 2014. [Online]. Available: http://www.ietf.org/rfc/rfc7252.txt

[6] M. Sethi, J. Arkko, and A. Keraenen, "End-to-end security for sleepy smart object networks," in *Local Computer Networks Workshops (LCN Workshops), 2012 IEEE 37th Conference on*, October 2012, pp. 964–972.

[7] C. Kerry, , P. Gallagher, and C. Romine, "Digital signature standard (dss)," NIST, FIPS 186-4, July 2013.

[8] D. Bernstein, N. Duif, T. Lange, P. Schwabe, and B. Yang, "High-speed high-security signatures," *Journal of Cryptographic Engineering*, vol. 2, no. 2, pp. 77–89, September 2012.

[9] C. Margi, B. de Oliveira, G. de Sousa, M. Simplicio, P. Barreto, T. Carvalho, M. Naslund, and R. Gold, "Impact of Operating Systems on Wireless Sensor Networks (Security) Applications and Testbeds," in *Proceedings of 19th International Conference on Computer Communications and Networks (ICCCN)*, August 2010, pp. 1–6.

[10] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2," RFC 5246 (Proposed Standard), Internet Engineering Task Force, Aug. 2008, updated by RFCs 5746, 5878, 6176, 7465, 7507, 7568, 7627, 7685. [Online]. Available: http://www.ietf.org/rfc/rfc5246.txt

[11] E. Rescorla and N. Modadugu, "Datagram Transport Layer Security Version 1.2," RFC 6347 (Proposed Standard), Internet Engineering Task Force, Jan. 2012, updated by RFC 7507. [Online]. Available: http://www.ietf.org/rfc/rfc6347.txt

[12] M. Vucnic, B. Tourancheau, F. Rousseau, A. Duda, L. Damon, and R. Guizzetti, "OSCAR: Object security architecture for the Internet of Things," *Ad Hoc Networks*, 2015, DOI: 10.1016/j.adhoc.2014.12.005.

[13] S. Raza, H. Shafagh, K. Hewage, and R. Hummen, "Lithe: Lightweight secure coap for the internet of things."

[14] G. Selander, J. Mattsson, F. Palombini, and L. Seitz, "Object security of coap (oscoap)," IETF, Internet-Draft, December 2016, (work in progress). [Online]. Available: http://tools.ietf.org/html/draft-ietf-core-object-security

[15] C. Bormann and P. Hoffman, "Concise Binary Object Representation (CBOR)," RFC 7049 (Proposed Standard), Internet Engineering Task Force, Oct. 2013. [Online]. Available: http://www.ietf.org/rfc/rfc7049.txt

[16] C. Bormann, "Constrained application protocol (coap) over ieee 802.15.4 information element for ietf," IETF, Internet-Draft, April 2016, (work in progress). [Online]. Available: http://tools.ietf.org/html/draft-bormann-6lo-coap-802-15-ie-00

[17] J. Schaad, "Cbor object signing and encryption (cose)," IETF, Internet-Draft, November 2016, (work in progress). [Online]. Available: http://tools.ietf.org/html/draft-ietf-cose-msg

[18] M. Jones, J. Bradley, and N. Sakimura, "JSON Web Signature (JWS)," RFC 7515 (Proposed Standard), Internet Engineering Task Force, May 2015. [Online]. Available: http://www.ietf.org/rfc/rfc7515.txt

[19] M. Jones and J. Hildebrand, "JSON Web Encryption (JWE)," RFC 7516 (Proposed Standard), Internet Engineering Task Force, May 2015. [Online]. Available: http://www.ietf.org/rfc/rfc7516.txt

[20] C. Bormann, S. Lemay, H. Tschofenig, K. Hartke, B. Silverajan, and B. Raymor, "Coap (constrained application protocol) over tcp, tls, and websockets," IETF, Internet-Draft, March 2017, (work in progress). [Online]. Available: http://tools.ietf.org/html/draft-ietf-core-coap-tcp-tls

[21] G. Selander, F. Palombini, and K. Hartke, "Requirements for coap end-to-end security," IETF, Internet-Draft, January 2017, (work in progress). [Online]. Available: http://tools.ietf.org/html/draft-hartke-core-e2e-security-reqs

[22] C. Bormann, "6LoWPAN-GHC: Generic Header Compression for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)," IETF, RFC 7400, Nov 2014. [Online]. Available: https://tools.ietf.org/html/rfc7400

[23] S. Raza, H. Shafagh, and O. Dupont, "Compression of record and handshake headers for constrained environments," http://shahidraza.info/draft-raza-6lo-compressed.txt, March 2017.

[24] M. Tiloca, G. Selander, and F. Palombini, "Secure group communication for coap," IETF, Internet-Draft, March 2017, (work in progress). [Online]. Available: http://tools.ietf.org/html/draft-tiloca-core-multicast-oscoap

[25] G. Selander, J. Mattsson, and F. Palombini, "Ephemeral diffie-hellman over cose (edhoc)," IETF, Internet-Draft, October 2016, (work in progress). [Online]. Available: http://tools.ietf.org/html/draft-selander-ace-cose-ecdhe

[26] H. Krawczyk, *SIGMA: The 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman and Its Use in the IKE Protocols*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 400–425. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-45146-4_24

[27] L. Seitz, G. Selander, E. Wahlstroem, S. Erdtman, and H. Tschofenig, "Authentication and Authorization for Constrained Environments (ACE)," IETF, Internet-Draft, February 2017, (work in progress). [Online]. Available: http://tools.ietf.org/html/draft-ietf-ace-oauth-authz

[28] D. Hardt, "The OAuth 2.0 Authorization Framework," RFC 6749 (Proposed Standard), Internet Engineering Task Force, Oct. 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6749.txt

[29] S. Gerdes, O. Bergmann, C. Bormann, G. Selander, and L. Seitz, "Datagram Transport Layer Security (DTLS) Profile for Authentication and Authorization for Constrained Environments (ACE)," IETF, Internet-Draft, June 2017, (work in progress). [Online]. Available: http://tools.ietf.org/html/draft-ietf-ace-dtls-authorize

[30] L. Seitz and F. Palombini, "OSCOAP profile of ACE," IETF, Internet-Draft, October 2016, (work in progress). [Online]. Available: http://tools.ietf.org/html/draft-seitz-ace-oscoap-profile

[31] F. Palombini, "Coap pub-sub profile for authentication and authorization for constrained environments (ace)," IETF, Internet-Draft, March 2017, (work in progress). [Online]. Available: http://tools.ietf.org/html/draft-palombini-ace-coap-pubsub-profile

[32] C. Sengul and A. Kirby, "MQTT-TLS profile of ACE," IETF, Internet-Draft, January 2017, (work in progress). [Online]. Available: http://tools.ietf.org/html/draft-sengul-ace-mqtt-tls-profile

[33] P. Eronen and H. Tschofenig, "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)," RFC 4279 (Proposed Standard), Internet Engineering Task Force, Dec. 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4279.txt

[34] P. Wouters, H. Tschofenig, J. Gilmore, S. Weiler, and T. Kivinen, "Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)," RFC 7250 (Proposed Standard), Internet Engineering Task Force, Jun. 2014. [Online]. Available: http://www.ietf.org/rfc/rfc7250.txt

[35] J. Mattsson, J. Fornehed, G. Selander, and F. Palombini, "Controlling actuators with coap," IETF, Internet-Draft, November 2016, (work in progress). [Online]. Available: https://tools.ietf.org/html/draft-mattsson-core-coap-actuators

[36] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, and H. Levkowetz, "Extensible Authentication Protocol (EAP)," RFC 3748 (Proposed Standard), Internet Engineering Task Force, Jun. 2004, updated by RFCs 5247, 7057. [Online]. Available: http://www.ietf.org/rfc/rfc3748.txt

[37] M. Vucinic, J. Simon, K. Pister, and M. Richardson, "Minimal security framework for 6tisch," IETF, Internet-Draft, March 2017, (work in progress). [Online]. Available: http://tools.ietf.org/html/draft-ietf-6tisch-minimal-security

[38] G. Selander, S. Raza, M. Vucinic, M. Furuhed, and M. Richardson, "Enrollment with application layer security," IETF, Internet-Draft, March 2017, (work in progress). [Online]. Available: http://tools.ietf.org/html/draft-selander-ace-eals

[39] M. Nystrom and B. Kaliski, "PKCS #10: Certification Request Syntax Specification Version 1.7," RFC 2986 (Informational), Internet Engineering Task Force, Nov. 2000, updated by RFC 5967. [Online]. Available: http://www.ietf.org/rfc/rfc2986.txt

[40] B. Kaliski, "PKCS #7: Cryptographic Message Syntax Version 1.5," RFC 2315 (Informational), Internet Engineering Task Force, Mar. 1998. [Online]. Available: http://www.ietf.org/rfc/rfc2315.txt