ACE Working Group
Internet-Draft
Intended status: Informational
Expires: October 31, 2015

S. Gerdes
Universitaet Bremen TZI
L. Seitz
SICS Swedish ICT AB
G. Selander
Ericsson
C. Bormann, Ed.
Universitaet Bremen TZI

April 29, 2015

An architecture for authorization in constrained environments
draft-gerdes-ace-actors-05

Abstract

Constrained-node networks are networks where some nodes have severe constraints on code size, state memory, processing capabilities, user interface, power and communication bandwidth (RFC 7228).

This document provides terminology, and elements of an architecture / a problem statement, for authentication and authorization in these networks.

Table of Contents

1. Introduction

Constrained nodes are small devices with limited abilities which in many cases are made to fulfill a specific simple task. They have limited hardware resources such as processing power, memory, non-volatile storage and transmission capacity and additionally in most cases do not have user interfaces and displays. Due to these constraints, commonly used security protocols are not always easily applicable.

Constrained nodes are expected to be integrated in all aspects of everyday life and thus will be entrusted with vast amounts of data. Without appropriate security mechanisms attackers might gain control over things relevant to our lives. Authentication and authorization mechanisms are therefore prerequisites for a secure Internet of Things.

In some cases authentication and authorization can be addressed by static configuration provisioned during manufacturing or deployment by means of fixed trust anchors and access control lists. This is particularly applicable to siloed, fixed-purpose deployments. However, as the need for flexible access to assets already deployed increases, the legitimate set of authorized entities as well as their privileges cannot be conclusively defined during deployment, without any need for change during the lifetime of the device. Moreover, several use cases illustrate the need for fine-grained access control policies, for which the access control lists concept may not be sufficiently generic.

The limitations of the constrained nodes ask for security mechanisms which take the special characteristics of constrained environments into account; not all constituents may be able to perform all necessary tasks by themselves. In order to meet the security requirements in constrained scenarios, the necessary tasks need to be assigned to logical functional entities.

This document provides some terminology, as well as elements of an architecture to represent the relationships between the logical functional entities involved; on this basis, a problem description for authentication and authorization in constrained-node networks is provided.

1.1. Terminology

Readers are required to be familiar with the terms and concepts defined in [RFC4949], including "authentication", "authorization", "confidentiality", "(data) integrity", "message authentication code", and "verify".

REST terms including "resource", "representation", etc. are to be understood as used in HTTP [RFC7231] and CoAP [RFC7252].

Terminology for constrained environments including "constrained device", "constrained-node network", "class 1", etc. are defined in [RFC7228].

In addition, this document uses the following terminology:

Resource (R): an item of interest which is represented through an interface. It might contain sensor or actuator values or other information.

Constrained node: a constrained device in the sense of [RFC7228].

Actor: A logical functional entity that performs one or more tasks. Multiple actors may be present within a single device or a single piece of software.

Resource Server (RS): An entity which hosts and represents a Resource.

Client (C): An entity which attempts to access a resource on an RS.

Principal: (Used in its English sense here, and specifically as:) An individual that is either RqP or RO or both.

Resource Owner (RO): The principal that is in charge of the resource and controls its access permissions.

Requesting Party (RqP): The principal that is in charge of the Client and controls the requests a Client makes and its acceptance of responses.

Authorization Server (AS): An entity that prepares and endorses authentication and authorization data for a Resource Server.

Client Authorization Server (CAS): An entity that prepares and endorses authentication and authorization data for a Client.

Authenticated Authorization: A synthesis of mechanisms for authentication and authorization.

Note that other authorization architectures such as OAuth [RFC6749] or UMA [I-D.hardjono-oauth-umacore] focus on the authorization problems on the RS side, in particular what accesses to resources the RS is to allow. In this document the term authorization includes this aspect, but is also used for client-side authorization, i.e., more generally to describe allowed interactions with other endpoints.

2. Architecture and High-level Problem Statement

2.1. Elements of an Architecture

This document deals with how to control and protect resource-based interaction between potentially constrained endpoints. The following setting is assumed:

- An endpoint may host functionality of C, RS or both C and RS.
- C in one endpoint requests to access R on a RS in another endpoint.
- A priori, the endpoints do not necessarily have a pre-existing security relationship to each other.
- Either of the endpoints, or both, may be constrained.

Without loss of generality, we focus on the C functionality in one endpoint, which we therefore also call C, accessing the RS functionality in another endpoint, which we therefore also call RS.

More on the security objectives of the constrained level in Section 5.1.

Figure 1: Constrained Level

The authorization decisions at the endpoints are made on behalf of the principals that control the endpoints. To reuse OAuth and UMA terminology, the present document calls C’s controlling principal the Requesting Party (RqP), and calls RS’s controlling principal the

Resource Owner (RO). Each principal makes authorization decisions (possibly encapsulating them into security policies) which the endpoint it controls then enforces.

The specific security objectives will vary, but for any specific version of this scenario will include one or more of:

- Objectives of type 1: No entity not authorized by the RO has access to (or otherwise gains knowledge of) R.
- Objectives of type 2: C is exchanging information with (sending a request to, accepting a response from) a resource only where it can ascertain that RqP has authorized the exchange with R.

Objectives of type 1 require performing authorization on the Resource Server side while objectives of type 2 require performing authorization on the Client side.

More on the security objectives of the principal level in Section 5.2.

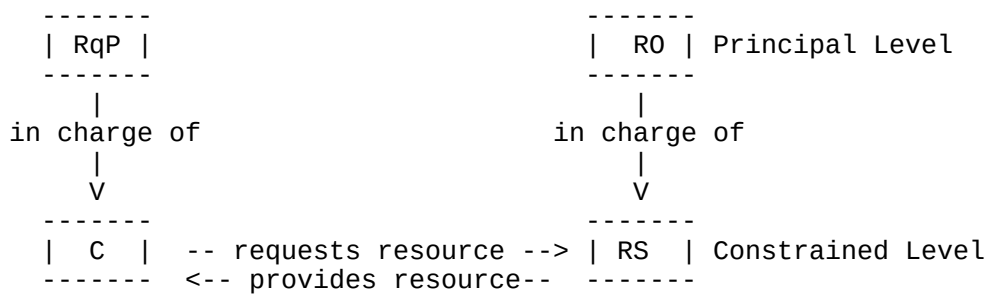


Figure 2: Constrained Level and Principal Level

The use cases defined in [I-D.ietf-ace-usecases] demonstrate that constrained devices are often used for scenarios where their principals are not present at the time of the communication, are not able to communicate directly with the device because of a lack of user interfaces or displays, or may prefer the device to communicate autonomously.

Moreover, constrained endpoints may need support with tasks requiring heavy processing, large memory or storage, or interfacing to humans, such as management of security policies defined by a principal. The principal, in turn, requires some agent maintaining the policies governing how its endpoints will interact.

For these reasons, another level of nodes is introduced in the architecture, the less-constrained level. Using OAuth terminology, AS acts on behalf of the RO to control and support the RS in handling access requests, employing a pre-existing security relationship with RS. We complement this with CAS acting on behalf of RqP to control and support the C in making resource requests and acting on the responses received, employing a pre-existing security relationship with C. To further relieve the constrained level, authorization (and related authentication) mechanisms may be employed between CAS and AS (Section 6.2). (Again, both CAS and AS are conceptual entities controlled by their respective principals. Many of these entities, often acting for different principals, can be combined into a single server implementation; this of course requires proper segregration of the control information provided by each principal.)

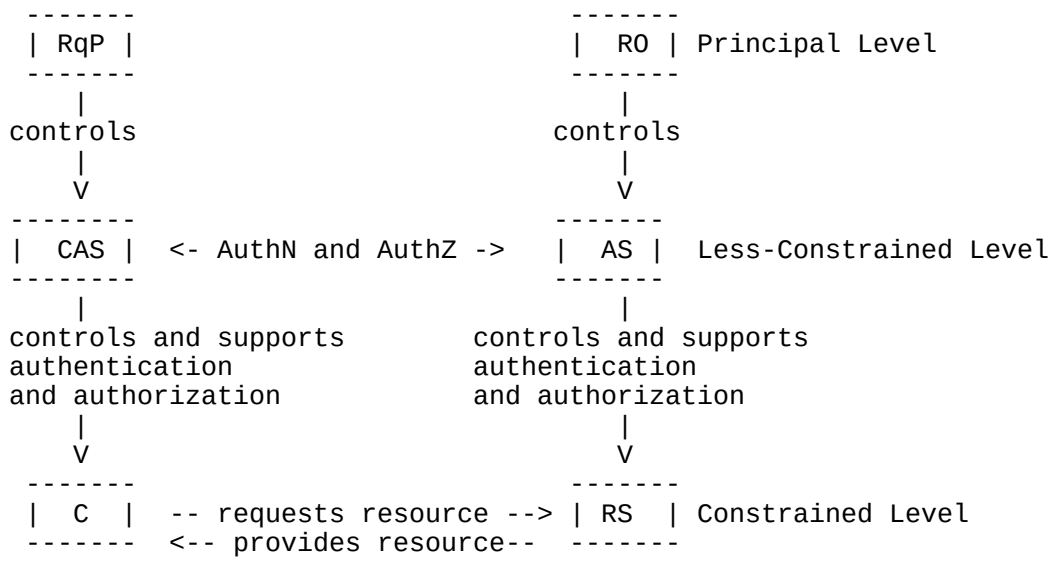


Figure 3: Overall architecture

2.2. Architecture Variants

The elements of the architecture described above are architectural. In a specific scenario, several elements can share a single device or even be combined in a single piece of software. If C is located on a more powerful device, it can be combined with CAS:

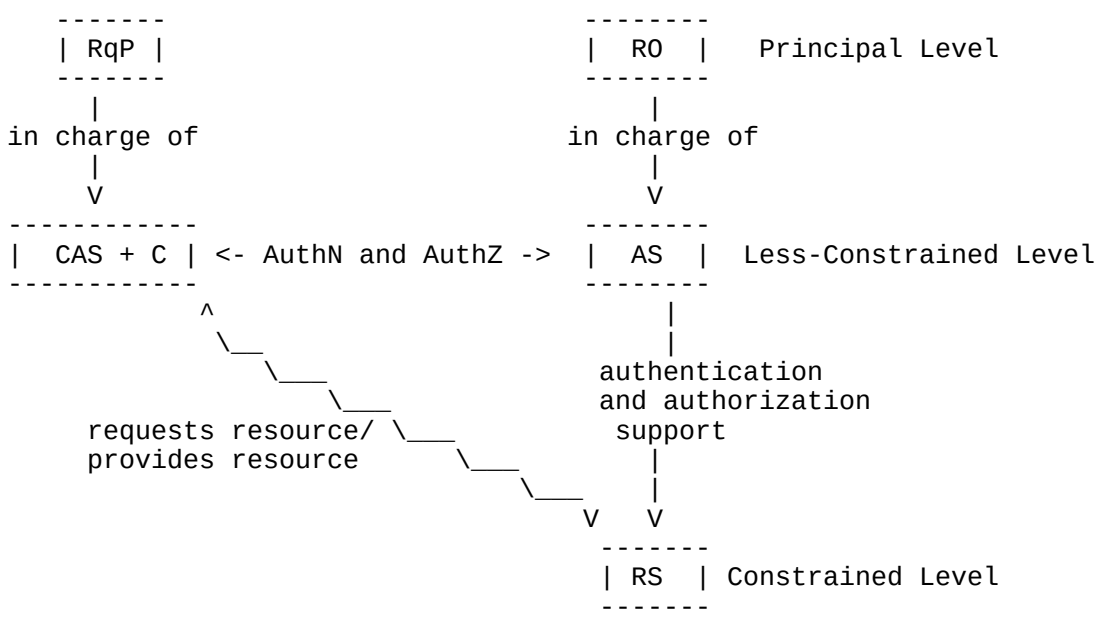


Figure 4: Combined C and CAS

If RS is located on a more powerful device, it can be combined with AS:

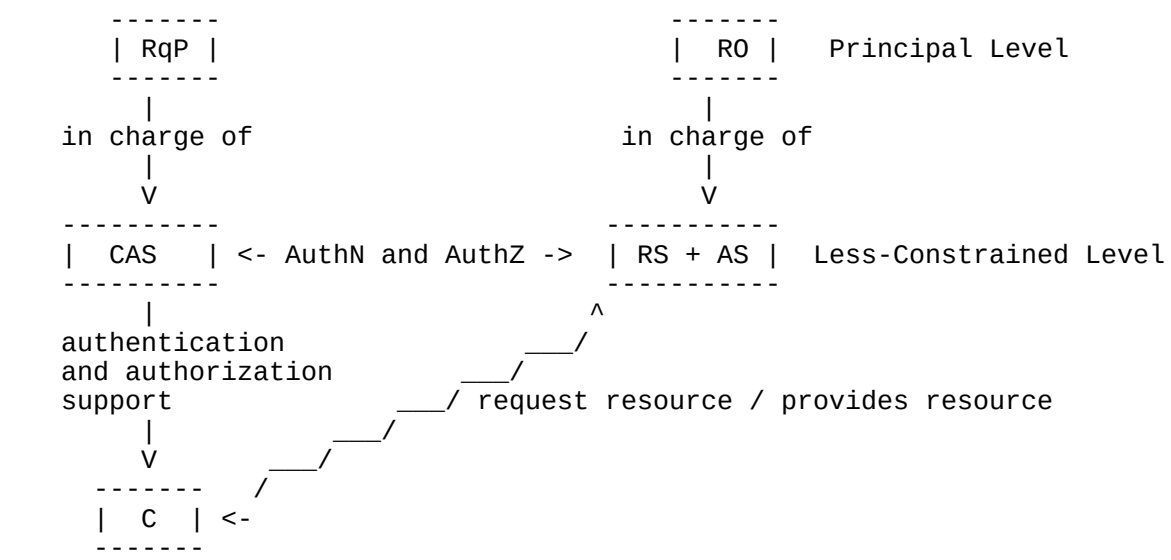


Figure 5: Combined AS and RS

If C and RS have the same principal, CAS and AS can be combined.

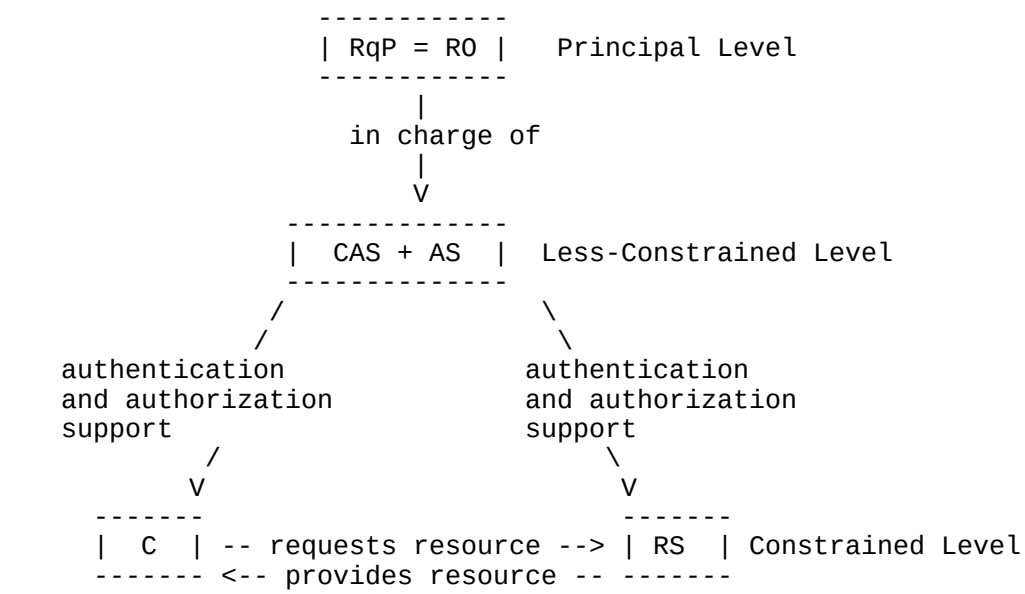


Figure 6: CAS combined with AS

2.3. Information flows

In this subsection, we complement the abstracted architecture described above with a discussion of the information flows in scope, mentioning that each endpoint may assume both a client and a server role and that communication may be via intermediaries.

The less-constrained nodes, CAS and AS, control the interactions between the endpoints by supporting the potentially constrained nodes with control information, for example permissions of clients, conditions on resources, attributes of client and resource servers, keys and credentials. The control information may be rather different for C and RS, reflecting the intrinsic asymmetry with C initiating the request for access to a resource, and RS acting on a received request, and C finally acting on the received response.

The information flows are shown in Figure 7. The arrows with control information only indicate origin and destination of information, actual message flow may pass intermediary nodes (both nodes that are identified in the architecture and other nodes).

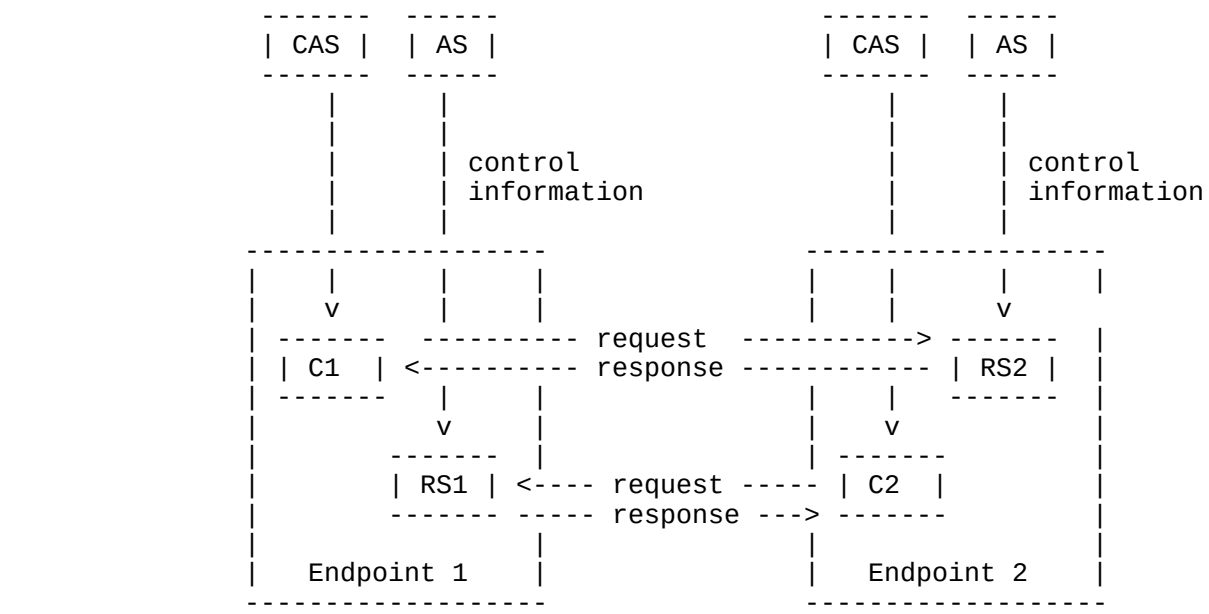


Figure 7: Information flows that need to be protected

- We assume that the necessary keys/credentials for protecting the control information between the potentially constrained nodes and

their associated less-constrained nodes are pre-established, for example as part of the commissioning procedure.

- The messages between the endpoints also need to be protected, potentially end-to-end through intermediary nodes (Section 3.1). Any necessary keys/credentials for protecting the interaction between the endpoints will need to be established and maintained as part of a solution.

2.4. Problem statement

The problem statement for authorization in constrained environments can be summarized as follows:

- The interaction between potentially constrained endpoints is controlled by control information provided by less-constrained nodes on behalf of the principals of the endpoints.

- The interaction between the endpoints needs to be secured, as well as the establishment of the necessary keys for securing the interaction, potentially end-to-end through intermediary nodes.

- The mechanism for transferring control information needs to be secured, potentially end-to-end through intermediary nodes. Pre-established keying material may need to be employed for establishing the keys used to protect these information flows.

3. Security Objectives

The security objectives that are addressed by an authorization solution include confidentiality and integrity. Additionally, allowing only selected entities limits the burden on system resources, thus helping to achieve availability. Misconfigured or wrongly designed authorization solutions can result in availability breaches: Users might no longer be able to use data and services as they are supposed to.

Authentication mechanisms can achieve additional security objectives such as non-repudiation and accountability. These additional objectives are not related to authorization and thus are not in scope of this draft, but may nevertheless be relevant. Non-repudiation and accountability may require authentication on a device level, if it is necessary to determine which device performed an action. In other cases it may be more important to find out who is responsible for the device’s actions.

The security objectives and their relative importance differ for the various constrained environment applications and use cases [I-D.ietf-ace-usecases].

In many cases, one participating party has different security objectives than another. To achieve a security objective of one party, another party may be required to provide a service. E.g., if RqP requires the integrity of representations of a resource R that RS is hosting, both C and RS need to partake in integrity-protecting the transmitted data. Moreover, RS needs to protect any write access to this resource as well as to relevant other resources (such as configuration information, firmware update resources) to prevent unauthorized users from manipulating R.

3.1. End-to-End Security Objectives

In many cases, the information flows described in Section 2.3 need to be protected end-to-end. For example, AS may not be connected to RS (or may not want to exercise such a connection), relying on C for transferring authorization information. As the authorization information is related to the permissions granted to C, C must not be in a position to manipulate this information, which therefore requires integrity protection on the way between AS and RS.

As another example, resource representations sent between endpoints may be stored in intermediary nodes, such as caching proxies or pub-sub brokers. Where these intermediaries cannot be relied on to fulfill the security objectives of the endpoints, these will need to protect the exchanges end-to-end.

Note that there may also be cases of intermediary nodes that very much partake in the security objectives to be achieved. What is the endpoint to which communication needs end-to-end protection is defined by the use case.

In order to support the required communication and application security, keying material needs to be established between the relevant nodes in the architecture.

4. Authentication and Authorization

Server-side authorization solutions aim at protecting the access to items of interest, e.g. hardware or software resources or data: They enable the resource owner to control who can access it and how.

To determine if an entity is authorized to access a resource, an authentication mechanism is needed. According to the Internet Security Glossary [RFC4949], authentication is "the process of

verifying a claim that a system entity or system resource has a certain attribute value." Examples for attribute values are the ID of a device, the type of the device or the name of its owner.

The security objectives the authorization mechanism aims at can only be achieved if the authentication and the authorization mechanism work together correctly. We speak of authenticated authorization to refer to the required synthesis of mechanism for authentication and authorization.

Where used for authorization, the set of authenticated attributes must be meaningful for this purpose, i.e., authorization decisions must be possible based on these attributes. If the authorization policy assigns permissions to an individual entity, the set of authenticated attributes must be suitable to uniquely identify this entity.

In scenarios where devices are communicating autonomously there is often less need to uniquely identify an individual device: For a principal, the fact that a device belongs to a certain company or that it has a specific type (e.g. light bulb) or location may be more important than that it has a unique identifier.

(As a special case for the authorization of read access to a resource, RS may simply make an encrypted representation available to anyone [OSCAR]. In this case, controlling read access to that resource can be reduced to controlling read access to the key; partially removing access also requires a timely update of the key for RS and all participants still authorized.)

Principals (RqP and RO) need to decide about the required level of granularity for the authorization. For example, we distinguish device authorization from owner authorization, and flat authorization from unrestricted authorization. In the first case different access permissions are granted to individual devices while in the second case individual owners are authorized. If flat authorization is used, all authenticated entities are implicitly authorized and have the same access permissions. Unrestricted authorization for an item of interest means that no authorization mechanism is used for accessing this resource (not even by authentication) and all entities are able to access the item as they see fit (note that an authorization mechanism may still be used to arrive at the decision to employ unrestricted authorization).

More fine-grained authorization does not necessarily provide more security but can be more flexible. Principals need to consider that an entity should only be granted the permissions it really needs

(principle of least privilege), to ensure the confidentiality and integrity of resources.

For all cases where an authorization solution is needed (all but Unrestricted Authorization), the enforcing party needs to be able to authenticate the party that is to be authorized. Authentication is therefore required for messages that contain (or otherwise update) representations of an accessed item. More precisely: The enforcing party needs to make sure that the receiver of a message containing a representation is authorized to receive it, both in the case of a client sending a representation to a server and vice versa. In addition, it needs to ensure that the actual sender of a message containing a representation is indeed the one authorized to send this message, again for both the client-to-server and server-to-client case. To achieve this, integrity protection of these messages is required: Authenticity cannot be assured if it is possible for an attacker to modify the message during transmission.

In some cases, only one side (client or server side) requires the integrity and / or confidentiality of a resource value. Principals may decide to omit authentication (unrestricted authorization), or use flat authorization (just employing an authentication mechanism). However, as indicated in Section 3, the security objectives of both sides must be considered, which can often only be achieved when the the other side can be relied on to perform some security service.

5. Actors and their Tasks

This and the following section look at the resulting architecture from two different perspectives: This section provides a more detailed description of the various "actors" in the architecture, the logical functional entities performing the tasks required. The following section then will focus on the protocols run between these functional entities.

For the purposes of this document, an actor consists of a set of tasks and additionally has a security domain (client domain or server domain) and a level (constrained, principal, less-constrained). Tasks are assigned to actors according to their security domain and required level.

Note that actors are a concept to understand the security requirements for constrained devices. The architecture of an actual solution might differ as long as the security requirements that derive from the relationship between the identified actors are considered. Several actors might share a single device or even be combined in a single piece of software. Interfaces between actors

may be realized as protocols or be internal to such a piece of software.

5.1. Constrained Level Actors

As described in the problem statement (see Section 2), either C or RS or both of them may be located on a constrained node. We therefore define that C and RS must be able to perform their tasks even if they are located on a constrained node. Thus, C and RS are considered to be Constrained Level Actors.

C performs the following tasks:

- Communicate in a secure way (provide for confidentiality and integrity of messages), including access requests.
- Validate that an entity is an authorized server for R.

RS performs the following tasks:

- Communicate in a secure way (provide for confidentiality and integrity of messages), including responses to access requests.
- Validate the authorization of the requester to access the requested resource as requested.

R is an item of interest such as a sensor or actuator value. R is considered to be part of RS and not a separate actor. The device on which RS is located might contain several resources of different ROs. For simplicity of exposition, these resources are described as if they had separate RS.

As C and RS do not necessarily know each other they might belong to different security domains.

(See Figure 8.)

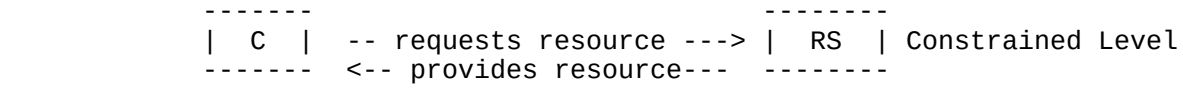


Figure 8: Constrained Level Actors

5.2. Principal Level Actors

Our objective is that C and RS are under control of principals in the physical world, the Requesting Party (RqP) and the Resource Owner (RO) respectively. The principals decide about the security policies of their respective endpoints and belong to the same security domain.

RqP is in charge of C, i.e. RqP specifies security policies for C, e.g. with whom C is allowed to communicate. By definition, C and RqP belong to the same security domain.

RqP must fulfill the following task:

- Configure for C authorization information for sources for R.

RO is in charge of R and RS. RO specifies authorization policies for R and decides with whom RS is allowed to communicate. By definition, R, RS and RO belong to the same security domain.

RO must fulfill the following task:

- Configure for RS authorization information for accessing R.

(See Figure 2.)

5.3. Less-Constrained Level Actors

Constrained level actors can only fulfill a limited number of tasks and may not have network connectivity all the time. To relieve them from having to manage keys for numerous endpoints and conducting computationally intensive tasks, another complexity level for actors is introduced. An actor on the less-constrained level belongs to the same security domain as its respective constrained level actor. They also have the same principal.

The Client Authorization Server (CAS) belongs to the same security domain as C and RqP. CAS acts on behalf of RqP. It assists C in authenticating RS and determining if RS is an authorized server for R. CAS can do that because for C, CAS is the authority for claims about RS.

CAS performs the following tasks:

- Validate on the client side that an entity has certain attributes.
- Obtain authorization information about an entity from C’s principal (RqP) and provide it to C.

- Negotiate means for secure communication to communicate with C.

The Authorization Server (AS) belongs to the same security domain as R, RS and RO. AS acts on behalf of RO. It supports RS by authenticating C and determining C's permissions on R. AS can do that because for RS, AS is the authority for claims about C.

AS performs the following tasks:

- Validate on the server side that an entity has certain attributes.
- Obtain authorization information about an entity from RS' principal (RO) and provide it to RS.
- Negotiate means for secure communication to communicate with RS.

6. Kinds of Protocols

Devices on the less-constrained level potentially are more powerful than constrained level devices in terms of processing power, memory, non-volatile storage. This results in different characteristics for the protocols used on these levels.

6.1. Constrained Level Protocols

A protocol is considered to be on the constrained level if it is used between the actors C and RS which are considered to be constrained (see Section 5.1). C and RS might not belong to the same security domain. Therefore, constrained level protocols need to work between different security domains.

Commonly used Internet protocols can not in every case be applied to constrained environments. In some cases, tweaking and profiling is required. In other cases it is beneficial to define new protocols which were designed with the special characteristics of constrained environments in mind.

On the constrained level, protocols need to address the specific requirements of constrained environments. Examples for protocols that consider these requirements is the transfer protocol CoAP (Constrained Application Protocol) [RFC7252] and the Datagram Transport Layer Security Protocol (DTLS) [RFC6347] which can be used for channel security.

Constrained devices have only limited storage space and thus cannot store large numbers of keys. This is especially important because constrained networks are expected to consist of thousands of nodes.

Protocols on the constrained level should keep this limitation in mind.

6.1.1. Cross Level Support Protocols

Protocols which operate between a constrained device on one side and the corresponding less-constrained device on the other are considered to be (cross level) support protocols. Protocols used between C and CAS or RS and AS are therefore support protocols.

Support protocols must consider the limitations of their constrained endpoint and therefore belong to the constrained level protocols.

6.2. Less-Constrained Level Protocols

A protocol is considered to be on the less-constrained level if it is used between the actors CAS and AS. CAS and AS might belong to different security domains.

On the less-constrained level, HTTP [RFC7230] and Transport Layer Security (TLS) [RFC5246] can be used alongside or instead of CoAP and DTLS. Moreover, existing security solutions for authentication and authorization such as the OAuth web authorization framework [RFC6749] and Kerberos [RFC4120] can likely be used without modifications and there are no limitations for the use of a Public Key Infrastructure (PKI).

7. Elements of a Solution

Without anticipating specific solutions, the following considerations may be helpful in discussing them.

7.1. Authorization

The core problem we are trying to solve is authorization. The following problems related to authorization need to be addressed:

- AS needs to transfer authorization information to RS and CAS needs to transfer authorization information to C.
- The transferred authorization information needs to follow a defined format and encoding, which must be efficient for constrained devices, considering size of authorization information and parser complexity.
- C and RS need to be able to verify the authenticity of the authorization information they receive. Here as well, there is a trade-off between processing complexity and deployment complexity.

- The RS needs to enforce the authorization decisions of the AS, while C needs to abide with the authorization decisions of the CAS. The authorization information might require additional policy evaluation (e.g. matching against local access control lists, evaluating local conditions). The required "policy evaluation" at the constrained actors needs to be adapted to the capabilities of the devices implementing them.

- Finally, as is indicated in the previous bullet, for a particular authorization decision there may be different kinds of authorization information needed, and these pieces of information may be transferred to C and RS at different times and in different ways prior to or during the client request.

7.2. Authentication

The following problems need to be addressed, when considering authentication:

- RS needs to authenticate AS, and C needs to authenticate CAS, to ensure that the authorization information and related data comes from the correct source.

- CAS and AS may need to to authenticate each other, both to perform the required business logic and to ensure that CAS gets security information related to the resources from the right source.

- In some use cases RS needs to authenticate some property of C, in order to map it to the relevant authorization information. In other use cases, authentication and authorization of C may be implicit, e.g. by encrypting the resource representation the RS only providing access to those who possess the key to decrypt.

- C may need to authenticate RS, in order to ensure that it is interacting with the right resources. Alternatively C may just verify the integrity of a received resource representation.

- CAS and AS need to authenticate their communication partner (C or RS), in order to ensure it serves the correct device.

7.3. Communication Security

There are different alternatives to provide communication security, and the problem here is to choose the optimal one for each scenario. We list the available alternatives:

- Session-based security at transport layer such as DTLS [RFC6347] offers security, including integrity and confidentiality

protection, for the whole application layer exchange. However, DTLS may not provide end-to-end security over multiple hops. Another problem with DTLS is the cost of the handshake protocol, which may be too expensive for constrained devices especially in terms of memory and power consumption for message transmissions.

- An alternative is object security at application layer, e.g. using [I-D.selander-ace-object-security] Secure objects can be stored or cached in network nodes and provide security for a more flexible communication model such as publish/subscribe (compare e.g. CoRE Mirror Server [I-D.koster-core-coap-pubsub]). A problem with object security is that it can not provide confidentiality for the message headers.

- Hybrid solutions using both session-based and object security are also possible. An example of a hybrid is where authorization information and cryptographic keys are provided by AS in the format of secure data objects, but where the resource access is protected by session-based security.

7.4. Cryptographic Keys

With respect to cryptographic keys, we see the following problems that need to be addressed:

Symmetric vs Asymmetric Keys
: We need keys both for protection of resource access and for protection of transport of authentication and authorization information. Do we want to support solutions based on asymmetric keys or symmetric keys in both cases? There are classes of devices that can easily perform symmetric cryptography, but consume considerably more time/battery for asymmetric operations. On the other hand asymmetric cryptography has benefits e.g. in terms of deployment.

Key Establishment
: How are the corresponding cryptographic keys established? Considering Section 7.1 there must be a mapping between these keys and the authorization information, at least in the sense that AS must be able to specify a unique client identifier which RS can verify (using an associated key). One of the use cases of [I-D.ietf-ace-usecases] describes spontaneous change of access policies - e.g. giving a hitherto unknown client the right to temporarily unlock your house door. In this case C is not previously known to RS and a key must be provisioned by AS.

Revocation and Expiration

How are keys replaced and how is a key that has been compromised revoked in a manner that reaches all affected parties, also keeping in mind scenarios with intermittent connectivity?

8. Assumptions and Requirements

In this section we list a set of candidate assumptions and requirements to make the problem description in the previous sections more concise and precise.

8.1. Architecture

The architecture consists of at least the following types of nodes:

- RS hosting resources, and responding to access requests
- C requesting access to resources
- AS supporting the access request/response procedure by providing authorization information to RS
  - AS may support this by aiding RS in authenticating C, or providing cryptographic keys or credentials to C and/or RS to secure the request/response procedure.
- CAS supporting the access request/response procedure by providing authorization information to C
  - CAS may support this by aiding C in authenticating RS, forwarding information between AS and C (possibly ultimately for RS), or providing cryptographic keys or credentials to C and/or RS to secure the request/response procedure.
- The architecture allows for intermediary nodes between any pair of C, RS, AS, and CAS, such as forward or reverse proxies in the CoRE architecture. (Solutions may or may not support all combinations.)
  - The architecture does not make a choice between session based security and data object security.

8.2. Constrained Devices

- C and/or RS may be constrained in terms of power, processing, communication bandwidth, memory and storage space, and moreover:
  - unable to manage complex authorization policies

    * unable to manage a large number of secure connections

    * without user interface

    * without constant network connectivity

    * unable to precisely measure time

    * required to save on wireless communication due to high power consumption

  o CAS and AS are not assumed to be constrained devices.

  o All devices under consideration can process symmetric cryptography without incurring an excessive performance penalty.

    * We assume the use of a standardized symmetric key algorithm, such as AES.

    * Except for the most constrained devices we assume the use of a standardized cryptographic hash function such as SHA-256.

  o Public key cryptography requires additional resources (e.g. RAM, ROM, power, specialized hardware).

  o A DTLS handshake involves significant computation, communication, and memory overheads in the context of constrained devices.

    * The RAM requirements of DTLS handshakes with public key cryptography are prohibitive for certain constrained devices.

    * Certificate-based DTLS handshakes require significant volumes of communication, RAM (message buffers) and computation.

  o A solution will need to consider support for a simple scheme for expiring authentication and authorization information on devices which are unable to measure time (cf. section Section 9.2).

8.3. Authentication

  o RS needs to authenticate AS to ensure that the authorization information and related data comes from the correct source.

  o Similary, C needs to authenticate CAS to ensure that the authorization information and related data comes from the correct source.

- Depending on use case and authorization requirements, C, RS, CAS, or AS may need to authenticate messages from each other.

8.4. Server-side Authorization

- RS enforces authorization for access to a resource based on credentials presented by C, the requested resource, the REST method, and local context in RS at the time of the request, or on any subset of this information.
- The credentials presented by C may have been provided by CAS.
- The underlying authorization decision is taken either by AS or RS.
- The authorization decision is enforced by RS.
  - RS needs to have authorization information in order to verify that C is allowed to access the resource as requested.
  - RS needs to make sure that it provides resource access only to authorized clients.
- Apart from authorization for access to a resource, authorization may also be required for access to information about a resource (e.g. resource descriptions).
- The solution may need to be able to support the delegation of access rights.

8.5. Client-side Authorization Information

- C enforces client-side authorization by protecting its requests to RS and by authenticating results from RS, making use of decisions and policies as well as keying material provided by CAS.

8.6. Server-side Authorization Information

- Authorization information is transferred from AS to RS using Agent, Push or Pull mechanisms [RFC2904].
- RS needs to authenticate that the authorization information is coming from AS (integrity).
- The authorization information may also be encrypted end-to-end between AS and RS (confidentiality).
- The architecture supports the case where RS may not be able to communicate with AS at the time of the request from C.

- RS may store or cache authorization information.
- Authorization information may be pre-configured in RS.
- Authorization information stored or cached in RS needs to be possible to change. The change of such information needs to be subject to authorization.
- Authorization policies stored on RS may be handled as a resource, i.e. information located at a particular URI, accessed with RESTful methods, and the access being subject to the same authorization mechanics. AS may have special privileges when requesting access to the authorization policy resources on RS.
- There may be mechanisms for C to look up the AS which provides authorization information about a particular resource.

8.7. Resource Access

- Resources are accessed in a RESTful manner using GET, PUT, POST, DELETE.
- By default, the resource request needs to be integrity protected and may be encrypted end-to-end from C to RS. It needs to be possible for RS to detect a replayed request.
- By default, the response to a request needs to be integrity protected and encrypted end-to-end from RS to C. It needs to be possible for C to detect a replayed response.
- RS needs to be able to verify that the request comes from an authorized client
- C needs to be able to verify that the response to a request comes from the intended RS.
- There may be resources whose access need not be protected (e.g. for discovery of the responsible AS).

8.8. Keys and Cipher Suites

- A constrained node and its authorization manager (i.e., RS and AS, and C and CAS) have established cryptographic keys. For example, they share a secret key or each have the other’s public key.
- The transfer of authorization information is protected with symmetric and/or asymmetric keys.

- The access request/response can be protected with symmetric and/or asymmetric keys.
- There must be a mechanism for RS to establish the necessary key(s) to verify and decrypt the request and to protect the response.
- There must be a mechanism for C to establish the necessary key(s) to protect the request and to verify and decrypt the response.
- There must be a mechanism for C to obtain the supported cipher suites of a RS.

8.9. Network Considerations

- A solution will need to consider network overload due to avoidable communication of a constrained node with its authorization manager (C with CAS, RS with AS).
- A solution will need to consider network overload by compact authorization information representation.
- A solution may want to optimize the case where authorization information does not change often.
- A solution may consider support for an efficient mechanism for providing authorization information to multiple RSs, for example when multiple entities need to be configured or change state.

8.10. Legacy Considerations

- A solution may consider interworking with existing infrastructure.
- A solution may consider supporting authorization of access to legacy devices.

9. Security Considerations

This document discusses authorization-related tasks for constrained environments and describes how these tasks can be mapped to actors in the architecture.

The entire document is about security. Security considerations applicable to authentication and authorization in RESTful environments are provided in e.g. OAuth 2.0 [RFC6749].

In this section we focus on specific security aspects related to authorization in constrained-node networks. Section 11.6 of [RFC7252], "Constrained node considerations", discusses implications

of specific constraints on the security mechanisms employed. A wider view of security in constrained-node networks is provided in [I-D.garcia-core-security].

9.1. Physical Attacks on Sensor and Actuator Networks

The focus of this work is on constrained-node networks consisting of connected sensors and actuators. The main function of such devices is to interact with the physical world by gathering information or performing an action. We now discuss attacks performed with physical access to such devices.

The main threats to sensors and actuator networks are:

- Unauthorized access to data to and from sensors and actuators, including eavesdropping and manipulation of data.

- Denial-of-service making the sensor/actuator unable to perform its intended task correctly.

  A number of attacks can be made with physical access to a device including probing attacks, timing attacks, power attacks, etc. However, with physical access to a sensor or actuator device it is possible to directly perform attacks equivalent of eavesdropping, manipulating data or denial of service. For example:

- Instead of eavesdropping the sensor data or attacking the authorization system to gain access to the data, the attacker could make its own measurements on the physical object.

- Instead of manipulating the sensor data the attacker could change the physical object which the sensor is measuring, thereby changing the payload data which is being sent.

- Instead of manipulating data for an actuator or attacking the authorization system, the attacker could perform an unauthorized action directly on the physical object.

- A denial-of-service attack could be performed physically on the object or device.

  All these attacks are possible by having physical access to the device, since the assets are related to the physical world. Moreover, this kind of attacks are in many cases straightforward (requires no special competence or tools, low cost given physical access, etc.)

As a conclusion, if an attacker has full physical access to a sensor or actuator device, then much of the security functionality elaborated in this draft is not effective to protect the asset during the physical attack.

Since it does not make sense to design a solution for a situation that cannot be protected against we assume there is no need to protect assets which are exposed during a physical attack. In other words, either an attacker does not have physical access to the sensor or actuator device, or if it has, the attack shall only have effect during the period of physical attack, and shall be limited in extent to the physical control the attacker exerts (e.g., must not affect the security of other devices.)

9.2. Time Measurements

Measuring time with certain accuracy is important to achieve certain security properties, for example to determine whether a public key certificate, access token or some other assertion is valid.

Dynamic authorization in itself requires the ability to handle expiry or revocation of authorization decisions or to distinguish new authorization decisions from old.

For certain categories of devices we can assume that there is an internal clock which is sufficiently accurate to handle the time measurement requirements. If RS can connect directly to AS it could get updated in terms of time as well as revocation information.

If RS continuously measures time but can’t connect to AS or other trusted source, time drift may have to be accepted and it may not be able to manage revocation. However, it may still be able to handle short lived access rights within some margins, by measuring the time since arrival of authorization information or request.

Some categories of devices in scope may be unable measure time with any accuracy (e.g. because of sleep cycles). This category of devices is not suitable for the use cases which require measuring validity of assertions and authorizations in terms of absolute time.

10. IANA Considerations

This document has no actions for IANA.

11. Acknowledgements

The authors would like to thank Olaf Bergmann, Robert Cragie, Klaus Hartke, Sandeep Kumar, John Mattson, Corinna Schmitt, Mohit Sethi, Hannes Tschofenig, Vlasios Tsiatsis and Erik Wahlstroem for contributing to the discussion, giving helpful input and commenting on previous forms of this draft. The authors would also like to specifically acknowledge input provided by Hummen and others [HUM14delegation].

12. Informative References

[HUM14delegation]
Hummen, R., Shafagh, H., Raza, S., Voigt, T., and K. Wehrle, "Delegation-based Authentication and Authorization for the IP-based Internet of Things", 11th IEEE International Conference on Sensing, Communication, and Networking (SECON’14), June 30 - July 3, 2014.

[I-D.garcia-core-security]
Garcia-Morchon, O., Kumar, S., Keoh, S., Hummen, R., and R. Struik, "Security Considerations in the IP-based Internet of Things", draft-garcia-core-security-06 (work in progress), September 2013.

[I-D.hardjono-oauth-umacore]
Hardjono, T., Maler, E., Machulak, M., and D. Catalano, "User-Managed Access (UMA) Profile of OAuth 2.0", draft-hardjono-oauth-umacore-13 (work in progress), April 2015.

[I-D.ietf-ace-usecases]
Seitz, L., Gerdes, S., Selander, G., Mani, M., and S. Kumar, "ACE use cases", draft-ietf-ace-usecases-03 (work in progress), March 2015.

[I-D.koster-core-coap-pubsub]
Koster, M., Keranen, A., and J. Jimenez, "Publish-Subscribe Broker for the Constrained Application Protocol (CoAP)", draft-koster-core-coap-pubsub-01 (work in progress), March 2015.

[I-D.selander-ace-object-security]
Selander, G., Mattsson, J., and L. Seitz, "March 9, 2015", draft-selander-ace-object-security-01 (work in progress), March 2015.

[OSCAR] Vucinic, M., Tourancheau, B., Rousseau, F., Duda, A., Damon, L., and R. Guizzetti, "OSCAR: Object Security Architecture for the Internet of Things", CoRR vol. abs/1404.7799, 2014.

[RFC2904] Vollbrecht, J., Calhoun, P., Farrell, S., Gommans, L., Gross, G., de Bruijn, B., de Laat, C., Holdrege, M., and D. Spence, "AAA Authorization Framework", RFC 2904, August 2000.

[RFC4120] Neuman, C., Yu, T., Hartman, S., and K. Raeburn, "The Kerberos Network Authentication Service (V5)", RFC 4120, July 2005.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", RFC 4949, August 2007.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

[RFC6749] Hardt, D., "The OAuth 2.0 Authorization Framework", RFC 6749, October 2012.

[RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, May 2014.

[RFC7230] Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, June 2014.

[RFC7231] Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, June 2014.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, June 2014.

Authors' Addresses

Stefanie Gerdes
Universitaet Bremen TZI
Postfach 330440
Bremen D-28359
Germany

Phone: +49-421-218-63906
Email: gerdes@tzi.org

Ludwig Seitz
SICS Swedish ICT AB
Scheelevaegen 17
Lund 223 70
Sweden

Email: ludwig@sics.se

Goeran Selander
Ericsson
Faroegatan 6
Kista 164 80
Sweden

Email: goran.selander@ericsson.com

Carsten Bormann (editor)
Universitaet Bremen TZI
Postfach 330440
Bremen D-28359
Germany

Phone: +49-421-218-63921
Email: cabo@tzi.org

ACE
Internet-Draft
Intended status: Informational
Expires: June 20, 2015

B. Greevenbosch
D. He
R. Sun
Huawei Technologies
December 17, 2014

Comparison of different proposals for ACE
draft-greevenbosch-ace-comparison-01

Abstract

This document investigates the different solutions in the ACE working group. It highlights both similarities and differences. In addition, it provides a security analysis for the different solutions.

Note that the views and comments in this document are solely based on its authors' interpretation, and do not necessarily represent the opinions of the authors of the discussed solutions.

Table of Contents

1. Requirements notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Introduction

In the ACE working group, there are various proposals to resolve the authentication and authorization issue. This document investigates the different proposals, and compares their similarities and differences.

The following proposals are investigated:

- DCAF
- EAP
- OAUTH
- Two-way authentication for IoT (TWAI)
- Pull Model (PULL)

In addition, this document compares different possible architectures, and evaluates their advantages and disadvantages.

The views and comments in this document are solely based on its authors' interpretation, and do not necessarily represent the opinions of the authors of drafts for the discussed solutions.

3. Proposals Comparison

3.1. DCAF

3.1.1. Overview

The DCAF proposal is defined in [I-D.gerdes-ace-dcaf-authorize].

In the proposal, the client (C) and resource server (RS) want to communicate securely. To setup a cryptographic channel, they need to communicate with an Authorization Server (AS), which generates a token with a secret to be used during the DTLS handshake.

C does not directly communicate with the AS, but uses an

Authorization Manager (AM) as intermediary. The AM and AS communicate through a secure channel, the nature of which is out of scope. This AM extracts the key for the client and provides it to the client through a secure channel. In addition, an access token contains another copy of the key, but encrypted using a key shared between AS and RS.

The access token format is specified. It signals permissions through [I-D.bormann-core-ace-aif].

3.1.2. Scope

The following items are left out of scope of DCAF:

- Secure channel setup between AM and AS
- Shared secret exchange between AS and RS

3.2. EAP

3.2.1. Overview

The document [I-D.marin-ace-wg-coap-eap] defines an EAP [RFC3748] authentication solution for CoAP. The EAP protocol is a framework for authentication, allowing for customized solutions. Examples of protocols using EAP include RADIUS [RFC2865] and Diameter [RFC6733].

The EAP solution includes a handshake between the EAP peer and the EAP authenticator. In most cases, the EAP peer can be considered a CoAP server, whereas the EAP authenticator is a CoAP client.

After the handshake, the EAP peer and EAP authenticator have a common Master Session Key, which they can use to setup a DTLS connection or for authentication for (as yet) unencrypted CoAP exchanges.

3.2.2. CoAP server and CoAP client functionality

Authentication could be started when the EAP peer and EAP authenticator discover each other.

When starting the authentication, the EAP Peer sends a CoAP GET to the EAP Authenticator, whereas in subsequent messages the EAP Authenticator sends CoAP POST and PUT messages. This means that both EAP Peer and EAP Authenticator need to implement both CoAP server and CoAP client functionality.

It is, however, to note that the EAP Authenticator can initiate the authentication by itself, in which case the client functionality is

not required for the EAP peer.

An Authentication, Authorization and Accounting (AAA) server may be deployed. Such AAA server is mentioned in [RFC3748], and both RADIUS [RFC2865] and Diameter [RFC6733] define AAA servers. However, the authorization in RADIUS and Diameter is not fine grained, it is a simple all or nothing question. Thus, RADIUS and/or Diameter would need adaptation to cater for CoAP specific requirements, or a new AAA protocol may need to be defined.

3.2.3. "AUTH" CoAP option

The authentication allows the exchange of cryptographic material. After authentication, this material is referred to using the new "AUTH" CoAP option. This AUTH option is only used for integrity protection of the related message, encryption is yet to be defined.

The EAP proposal mentions defining the cryptographic material needed to setup an encrypted DTLS channel in a future version of the proposal.

3.3. OAUTH

3.3.1. Overview

There are currently three documents that aim at adapting OAUTH v2.0 [RFC6749] for CoAP:

- [I-D.tschofenig-ace-oauth-iot] gives a general overview of how OAuth 2.0 can be adopted to cater for the IOT.
- [I-D.tschofenig-ace-oauth-bt] defines how a bearer token can be used in IoT.
- [I-D.wahlstroem-ace-oauth-introspection] defines a method for a client or resource server to query an OAuth authorization server to determine meta-information about an OAuth token using the CoAP protocol.

3.3.2. OAuth IoT

The document [I-D.tschofenig-ace-oauth-iot] describes the communication between the client and the authorization server. It is to be noted that OAuth was originally designed for HTTP and hence needs adaptations to cater for CoAP.

To obtain an access permission to a resource, the client first needs to acquire an access token from the AS. This access token contains

signalling of the Access Token Scope, which indicates which kind of operations the Access Token enables. The Access Token Scope is proprietary in OAuth, and hence would need to be defined for this particular application.

The Access Token Request and Response are sent over DTLS. This means that the client and AS must have a DTLS connection before exchange of the Access Token can be achieved. Authentication of the client and AS is considered part of the DTLS channel setup, and not further specified.

The OAuth 2.0 specification leaves discovery of AS and registration of the client with the AS out of scope. It does, however, require verification of client credentials by the AS. In HTTP, such credentials could consist of a password ([RFC6749], section 2.3.1).

It is currently unclear which key material is used to setup an encrypted channel between client and RS.

3.3.3. OAuth bearer token

The document [I-D.tschofenig-ace-oauth-bt] defines how OAuth v2.0 Bearer Tokens from [RFC6750] can be used for CoAP. The Bearer Token is used by the client to prove to the server that it has access rights to certain resources. The server has to inspect the existence and value of the Bearer Token, and return an error if the check fails.

The Bearer Token is used as proof of the identity of the client to the server, and hence needs to be kept secret to avoid spoofing.

3.3.4. OAuth introspection

The Access Token is associated with permissions for the client to access a resource. However, the Access Token is an opaque string, which is hard to interpret from the client or server side. Hence there is a need for [I-D.wahlstroem-ace-oauth-introspection], which allows the client to send the token to an Introspection Endpoint (usually the AS), which acquires the metadata (such as access permissions) for the token and sends it back to the client in JSON.

The document is based on [I-D.ietf-oauth-introspection], which defines the same technology for standard, HTTP based, OAuth.

3.4. Two-way authentication for IoT (TWAI)

The document [I-D.schmitt-ace-twowayauth-for-iot] establishes a framework for authentication and authorization, with the explicit

goal of using existing and deployed standards where possible.

The document mentions both class 2 and class 1 devices (defined in [I-D.bormann-lwig-terms]). The document wants to provide two solutions for the two classes, but currently only defines the solution for class 2, whereas the solution for class 1 is still to be defined. There are, however, already some indications of adaptations needed to make the class 2 solution work for class 1 devices.

3.4.1. Basic authentication

The basic solution for class 2 devices is standard authentication during the DTLS handshake, based on X.509 certificates. For class 2, the draft proposes to use RSA key pairs. Since an RSA public key exceeds a kilobyte, the size is not considered viable for class 1 devices. Remember also that in CoAP, the recommended maximum message size is around 1024 bytes to avoid fragmentation. The certificates are exchanged in DTLS, not CoAP, but the fragmentation remains a concern. Hence for class 1 devices, the draft proposes to use Eliptic Curve Cryptography (ECC), although the details have not yet been described.

The X.509 certificates are authenticated by a certificate authority, either net-wide or domain-wide.

Parsing of X.509 certificates is difficult for constrained devices. Not only would they need to parse ASN.1 data and verify a signatures and a possible certificate chain, they also would need to contact an OSCP responder or consult a CRL to verify revocation status of the X.509 certificates. In addition, they need a secure time source to verify that the X.509 certificate has not yet expired.

3.4.2. Authorization

The draft also describes an authorization architecture. The architecture is based on four entities:

- Subscriber
- Access control server
- Gateway
- Publisher

The idea is that the publisher aggregates data of multiple sensors, and provides it to possibly multiple subscribers.

The access control server provides access control tickets, which are separately delivered to the publisher and the subscribers, after which they can perform standard two-way authentication and setup the DTLS channel.

The definition of the ticket format and the messages between subscriber, access control server and publisher is still to be done.

The publisher may delegate its security functionality to a gateway.

3.5. Pull Model

The document [I-D.greevenbosch-ace-pull-model] defines a solution using a pull model. When the client wants to access a resource on the RS, the RS asks the AS for authorization. The AS then grants or refuses the authorization.

Since the RS talks directly to the AS, there is no need for the client to communicate with the AS. Hence the AS can (but does not have to) reside in a domain separated from the Client.

The authorization from the AS is signalled to the RS using an authorization ticket. This ticket has the same format as as the authorization ticket in DCAF. It can signal permissions on a per resource and per action basis. The client is not presented with the access ticket, the RS parses and interprets it by itself.

The communication between RS and AS, as well as between Client and RS, is DTLS + CoAP.

3.6. Comparison of the proposals

3.6.1. Comparison table

The following provides a list comparing the different solutions. For TWAI, we consider the access control server the equivalent to the AS, and abbreviate it as such.

Registration between C and AS:
- DCAF: out of scope
- EAP: not applicable
- OAuth: out of scope
- TWAI: standard DTLS
- PULL: out of scope

Client credentials:

- DCAF: through relationship with AM
- EAP: out of scope
- OAuth: out of scope
- TWAI: X.509 certificates
- PULL: out of scope

Revocation of client or server:
- DCAF: out of scope, but could be done by AM or AS (no new ticket)
- EAP: AAA server could refuse new authorization
- OAuth: could be done by AS (no new ticket)
- TWAI: not defined, but OSCP and CRL should be feasible
- PULL: can be done by AS (no access grant)

Client needs to verify certificates?
- DCAF: no - delegated to AM
- EAP: ?
- OAuth: yes, from AS
- TWAI: yes, from AS
- PULL: no - delegated to AM

Pre-shared key assumptions:
- DCAF: C and AM have a pre-shared key. RS and AS have a pre-shared key.
- EAP: ?
- OAuth: ?
- TWAI: maybe between AS and publisher/gateway?
- PULL: RS and AS have a pre-shared key.

Discovery of AS:
- DCAF: RS informs C about AS when refusing an unauthorized response from C.
- EAP: ?
- OAuth: Preconfigured
- TWAI: Out of scope
- PULL: Preconfigured, RS contacts AS directly

Protocol for exchange of authorization information:
- DCAF: between C and AM, and between C and RS: DTLS; between AM and AS: proprietory.
- EAP: CoAP
- OAuth: between C and AS: CoAP+DTLS. Between C and RS: DTLS?
- TWAI: to be defined
- PULL: CoAP + DTLS

Definition of new CoAP option(s):
- DCAF: no

- EAP: maybe, "AUTH"
- OAuth: yes, "Bearer" and "Error"
- TWAI: no
- PULL: re-uses new "Node-Id" option

Protocol for actual data exchange between C and RS:
- DCAF: DTLS + CoAP
- EAP: CoAP + EAP or DTLS + CoAP
- OAuth: DTLS + CoAP
- TWAI: DTLS + CoAP
- PULL: DTLS + CoAP

Object security or transport security:
- DCAF: transport security
- EAP: object security (with AUTH option) or transport security (with DTLS)
- OAuth: transport security
- TWAI: transport security
- PULL: transport security

Level of authorization:
- DCAF: resource and method level
- EAP: All or nothing
- OAuth: Resource and method level possible, through definition of Access Token Scope
- TWAI: Resource and method level at publisher
- PULL: resource and method level

Ticket format:
- DCAF: defined
- EAP: N.A.
- OAuth: needs definition
- TWAI: needs definition
- PULL: defined (same as DCAF)

Ticket lifetime:
- DCAF: limited
- EAP: diameter provides an AVP for authorization lifetime
- OAuth: limited
- TWAI: ?
- PULL: limited

Number of parties:
- DCAF: 3/4 (AM can be merged with client)
- EAP: 2/3 (depending on deployment of AAA server)
- OAuth: 3

- o TWAI: 4/5 (gateway and access control server could be combined)
- o PULL: 3

4. Architectural Models Comparison

According to the uploaded drafts and previous discussions in IETF 89 Toronto F2F meeting, there are several possible architectural models to achieve authorization in a constrained environment:

- o Push model
- o Pull model
- o Agent model
- o Push/Confirm model
- o Indirect push model

This section provides an analysis of these models, discusses their advantages and disadvantages, and gives some recommendations.

4.1. Push Model

4.1.1. Brief Introduction

The push model is described in [RFC2904] and proposed in the DCAF draft ([I-D.gerdes-ace-dcaf-authorize]).

From high level point of view, it can be depicted as follows:

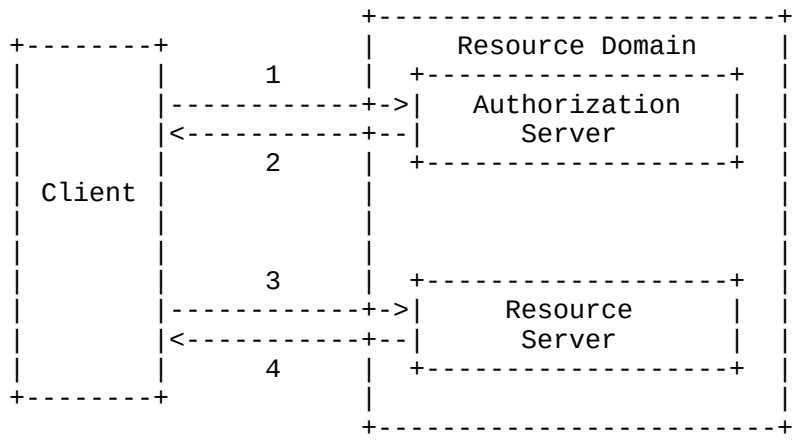


Figure 1: Push Model

To simplify the architectural model, the Authentication Manager is combined with the client in this architecture diagram.

In this model, the high level flows are described as follows:

Step 1: the client sends a request to the Authorization Server to get an access ticket for a specific resource access request.

Step 2: the Authorization Server sends the access ticket to the client.

Step 3: the client sends the resource access request to the Resource Server, containing the access ticket.

Step 4: the Resource Server verifies the access ticket and returns a resource access response.

4.1.2. Analysis

Advantage: in the push model, the message transmission and processing workload for the Resource Server is relatively low, and message transmission and processing workload for the client is relatively high. Since the Resource Server is usually the most constrained device, this model is suitable for the constrained environment.

Disadvantage: in some use cases, the client may not be able to have a connection with Authorization Server. Then this model is not applicable. Also, a ticket revocation mechanism may be needed, for example in case of a security breach or policy modification. If there is no such mechanism, during the validity period of the ticket,

the client is still able to access the resource in accordance to the old policy.

4.2. Pull Model

4.2.1. Brief Introduction

The pull model is described in [RFC2904] and proposed in the Pull Model draft [I-D.greevenbosch-ace-pull-model].

From a high level point of view, it can be depicted as below:

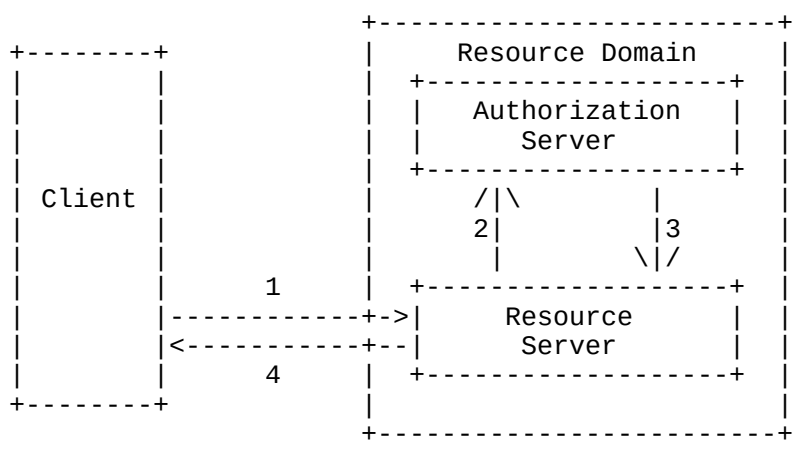


Figure 2: Pull Model

In this model, the high level flows are described as follows:

Step 1: the client sends a resource access request to the Resource Server.

Step 2: the Resource Server sends an authorization request to the Authorization Server for the resource access request from the client.

Step 3: the Authorization Server evaluates the authorization request and returns an access ticket to the Resource Server.

Step 4: the Resource Server verifies the access ticket and returns a resource access response.

4.2.2. Analysis

Advantage: in the pull model, there is no need to cache or forward the access ticket. This can save message transmission and processing workload for the client. This will especially be helpful if the client is constained. Also, this model can work for use cases where the client cannot have a direct connection with the Authorization Server.

Disadvantage: the message transmission and processing workload for the Resource Server is relatively high compared with the push model.

4.3. Agent Model

4.3.1. Brief Introduction

Agent model is described in [RFC2904] and this model was discussed in IETF 89 ACE Toronto F2F meeting.

From high level point of view, it can be depicted below:

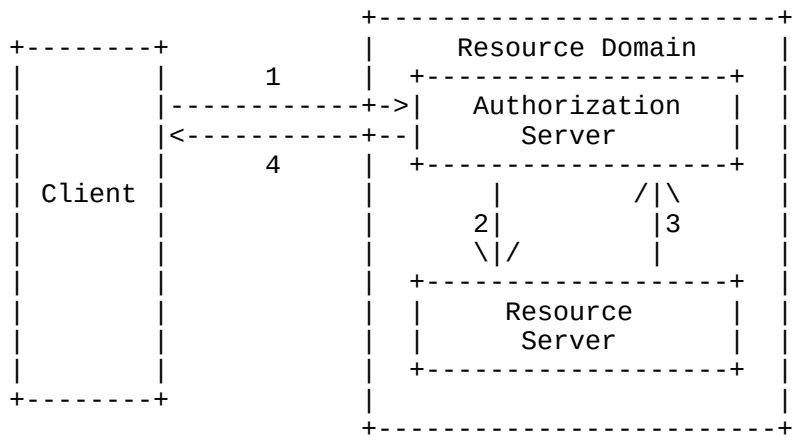


Figure 3: Agent Model

In this model, the high level flows are described as follows:

Step 1: the client sends a resource access request to the Authorization Server.

Step 2: the Authorization Server checks the client’s access rights, and sends a resource access request to the Resource Server, containing the access ticket.

Step 3: the Resource Server verifies the access ticket and returns a resource access response to the Authorization Server.

Step 4: the Resource Server verifies the access ticket and returns a resource access response to the client.

4.3.2. Analysis

Advantage: this model can work when the client cannot have a direct connection with the Resource Server.

Disadvantage: the Authorization Server works as an agent, and it is involved in the resource access process. Logically, the Authorization Server should just take care of authentication and authorization issues and should not mix other functionalities.

4.4. Push/Confirm Model

4.4.1. Brief Introduction

This model was discussed in IETF 89 ACE Toronto F2F meeting. The OAuth introspection solution [I-D.wahlstroem-ace-oauth-introspection] uses this model.

From high level point of view, it can be depicted as follows:

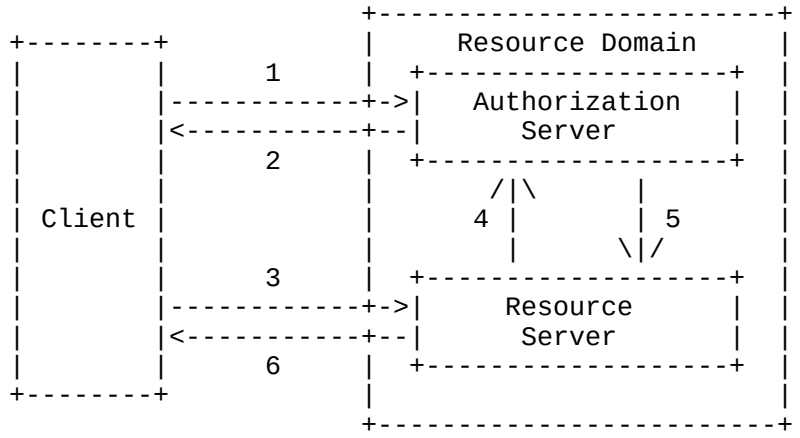


Figure 4: Push/Confirm Model

In this model, the high level flows are described as follows:

Step 1: the client sends a request to the Authorization Server to get

an access ticket for a specific resource access request.

Step 2: the Authorization Server sends the access ticket identifier to the client.

Step 3: the client sends a resource access request to the Resource Server, containing the access ticket identifier.

Step 4: the Resource Server sends an authorization request to the Authorization Server, containing the access ticket identifier.

Step 5: the Authorization Server gets the access ticket according to the received access ticket identifier, and sends the access ticket to the Resource Server.

Step 6: the Resource Server verifies the access ticket and returns the resource access response to the client.

4.4.2. Analysis

Advantage: this model does not require direct transfer of the access ticket to the client, such that transmission traffic for the client is reduced. This model is efficient if the access ticket is relatively large and the client is resource constrained.

Disadvantage: this model requires more processes in the whole authorization flow. It adds a burden for the Resource Server to send an authorization request to and get a response from the Authorization Server. It also adds a burden for the Authorization Server to handle requests from both the client and the Resource Server.

4.5. Indirect Push Model

4.5.1. Brief Introduction

This model was proposed in [I-D.schmitt-ace-twowayauth-for-iot].

From a high level point of view, it can be depicted as follows:

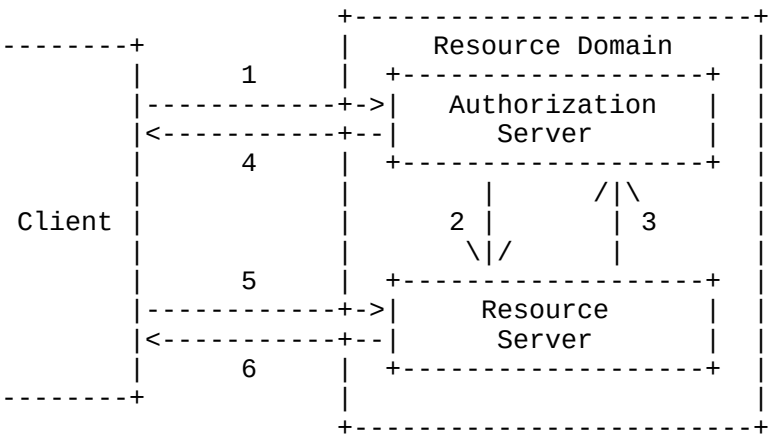


Figure 5: Indirect Push Model

In this model, the high level flows are described as follows:

Step 1: the client sends an authorization request to the Authorization Server for a specific resource access request.

Step 2: the Authorization Server verifies the authorization request and sends an access ticket to the Resource Server.

Step 3: the Resource Server caches the received access ticket and returns confirmation about the receipt of the access ticket.

Step 4: the Authorization Server sends an authorization response to the client.

Step 5: the client sends a resource access request to the Resource Server.

Step 6: the Resource Server verifies the resource access request based on the cached access ticket and returns a resource access response to the client.

4.5.2. Analysis

Advantage: this model does not require the client to receive the access ticket from the Authorization Server and send the access ticket to the Resource Server. This can reduce transmission traffic for the client. This model is efficient if the access ticket is relatively large and the client is resource constrained.

Disadvantage: this model requires more processes in the whole

authorization flow. It adds a burden to the Resource Server to cache the access ticket.

4.6. Recommendations

To cover different use cases, one architectural model may be difficult to cover all the use cases. Also, too many alternative architectural models may introduce interoperability problems. Based on the analysis above, this memo recommends two models: the push model and the pull model. These two models can work complementary to each other, and can satisfy different use cases.

5. Security considerations

The complete document concerns security considerations.

6. IANA considerations

This document does not require any IANA registrations.

7. Acknowledgements

Thanks to the authors of the various drafts for their efforts to provide a solution for authentication and authorization in constrained environments.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

8.2. Informative References

[RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.

[RFC2904] Vollbrecht, J., Calhoun, P., Farrell, S., Gommans, L., Gross, G., de Bruijn, B., de Laat, C., Holdrege, M., and D. Spence, "AAA Authorization Framework", RFC 2904, August 2000.

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004.

[RFC6733] Fajardo, V., Arkko, J., Loughney, J., and G. Zorn, "Diameter Base Protocol", RFC 6733, October 2012.

[RFC6749] Hardt, D., "The OAuth 2.0 Authorization Framework", RFC 6749, October 2012.

[RFC6750] Jones, M. and D. Hardt, "The OAuth 2.0 Authorization Framework: Bearer Token Usage", RFC 6750, October 2012.

[I-D.ietf-oauth-introspection] Richer, J., "OAuth Token Introspection", draft-ietf-oauth-introspection-00 (work in progress), August 2014.

[I-D.bormann-core-ace-aif] Bormann, C., "An Authorization Information Format (AIF) for ACE", draft-bormann-core-ace-aif-01 (work in progress), July 2014.

[I-D.bormann-lwig-terms] Bormann, C. and M. Ersue, "Terminology for Constrained Node Networks", draft-bormann-lwig-terms-00 (work in progress), November 2012.

[I-D.gerdes-ace-dcaf-authorize] Gerdes, S., Bergmann, O., and C. Bormann, "Delegated CoAP Authentication and Authorization Framework (DCAF)", draft-gerdes-ace-dcaf-authorize-00 (work in progress), July 2014.

[I-D.greevenbosch-ace-pull-model] Greevenbosch, B., ana.hedanping@huawei.com, a., and D. Zhang, "ACE Pull Model", draft-greevenbosch-ace-pull-model-00 (work in progress), October 2014.

[I-D.marin-ace-wg-coap-eap] Garcia, D., "EAP-based Authentication Service for CoAP", draft-marin-ace-wg-coap-eap-01 (work in progress), October 2014.

[I-D.schmitt-ace-twowayauth-for-iot] Schmitt, C. and B. Stiller, "Two-way Authentication for IoT", draft-schmitt-ace-twowayauth-for-iot-00 (work in

progress), June 2014.

[I-D.tschofenig-ace-oauth-bt]
Tschofenig, H., "The OAuth 2.0 Bearer Token Usage over the Constrained Application Protocol (CoAP)", draft-tschofenig-ace-oauth-bt-00 (work in progress), July 2014.

[I-D.tschofenig-ace-oauth-iot]
Tschofenig, H., "The OAuth 2.0 Internet of Things (IoT) Client Credentials Grant", draft-tschofenig-ace-oauth-iot-00 (work in progress), July 2014.

[I-D.wahlstroem-ace-oauth-introspection]
Wahlstroem, E., "OAuth 2.0 Introspection over the Constrained Application Protocol (CoAP)", draft-wahlstroem-ace-oauth-introspection-00 (work in progress), October 2014.

Authors' Addresses

Bert Greevenbosch
Huawei Technologies Co., Ltd.
Huawei Industrial Base
Bantian, Longgang District
Shenzhen 518129
P.R. China

Email: bert.greevenbosch@huawei.com

Danping He
Huawei Technologies
Q14, Huawei, Huanbao Yuan, 156 Beiqing Road, Haidian District
Beijing 100095
China

Email: ana.hedanping@huawei.com

Ruinan Sun
Huawei Technologies Co., Ltd.
Huawei Industrial Base
Bantian, Longgang District
Shenzhen 518129
P.R. China

Email: sunruinan@huawei.com

ACE Working Group
Internet-Draft
Intended status: Informational
Expires: April 30, 2015

L. Seitz, Ed.
SICS Swedish ICT AB
S. Gerdes, Ed.
Universitaet Bremen TZI
G. Selander
Ericsson
M. Mani
Itron
S. Kumar
Philips Research
October 27, 2014

ACE use cases
draft-seitz-ace-usecases-02

Abstract

Constrained devices are nodes with limited processing power, storage space and transmission capacities. These devices in many cases do not provide user interfaces and are often intended to interact without human intervention.

This document comprises a collection of representative use cases for the application of authentication and authorization in constrained environments. These use cases aim at identifying authorization problems that arise during the lifecylce of a constrained device and are intended to provide a guideline for developing a comprehensive authentication and access control solution for this class of scenarios.

Where specific details are relevant, it is assumed that the devices use the Constrained Application Protocol (CoAP) as communication protocol, however most conclusions apply generally.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Table of Contents

1. Introduction

Constrained devices [RFC7228] are nodes with limited processing power, storage space and transmission capacities. These devices are often battery-powered and in many cases do not provide user interfaces.

Constrained devices benefit from being interconnected using Internet protocols. However, due to the devices' limitations, commonly used security protocols are not always easily applicable. As the devices are expected to be integrated in all aspects of everyday life, the application of adequate security mechanisms is required to prevent attackers from gaining control over data or functions important to our lives.

This document comprises a collection of representative use cases for the application of authentication and authorization in constrained environments. These use cases aim at identifying authorization problems that arise during the lifecycle of a constrained device.

We assume that the communication between the devices is based on the Representational State Transfer (REST) architectural style, i.e. a device acts as a server that offers resources such as sensor data and actuators. The resources can be accessed by clients, sometimes without human intervention (M2M). In some situations the communication will happen through intermediaries (e.g. gateways, proxies).

Where specific detail is necessary it is assumed that the devices communicate using CoAP [RFC7252], although most conclusions are generic.

1.1. Terminology

Resource Server (RS): The constrained device which hosts resources the Client wants to access.

Client (C): A device which wants to access a resource on the Resource Server.
This could also be a constrained device.

Resource Owner (RO): The subject who owns the resource and controls its access permissions.

2. Use Cases

This section lists use cases involving constrained devices with certain authorization problems to be solved. Each use case first presents a general description of the application area, then one or more specific use cases, and finally a summary of the authorization-related problems device owners need to be solved.

There are various reasons for assigning a function (client or resource server) to a device, e.g. which device initiates the conversation, how do devices find each other, etc. The definition of the function of a device in a certain use case is not in scope of this document. Readers should be aware that there might be reasons for each setting and that devices might even have different functions at different times.

2.1. Container monitoring

The ability of sensors to communicate environmental data wirelessly opens up new application areas. The use of such sensor systems makes it possible to continuously track and transmit specific characteristics such as temperature, humidity and gas content during the transportation and storage of goods.

The proper handling of the sensors in this scenario is not easy to accomplish. They have to be associated to the appropriate pallet of the respective container. Moreover, the goods and the corresponding sensors belong to specific customers.

During the shipment to their destination the goods often pass stops where they are transloaded to other means of transportation, e.g. from ship transport to road transport.

The transportation and storage of perishable goods is especially challenging since they have to be stored at a constant temperature and with proper ventilation. Additionally, it is very important for

the vendors to be informed about irregularities in the temperature and ventilation of fruits to avoid the delivery of decomposed fruits to their customers. The need for a constant monitoring of perishable goods has led to projects such as The Intelligent Container (http://www.intelligentcontainer.com).

2.1.1. Bananas for Munich

A fruit vendor grows bananas in Costa Rica for the German market. It instructs a transport company to deliver the goods via ship to Rotterdam where they are picked up by trucks and transported to a ripening facility. A Munich supermarket chain buys ripened bananas from the fruit vendor and transports them with their own company trucks.

The fruit vendor’s quality management wants to assure the quality of their products and thus equips the banana boxes with sensors. The state of the goods is monitored consistently during shipment and ripening and abnormal sensor values are recorded. Additionally, the sensor values are used to control the climate within the cargo containers. Since a wrong sensor value leads to a wrong temperature and thus to spoiled goods, the integrity of the sensor data must be assured.

Due to the high water content of the fruits, the propagation of radio waves is hindered, thus often inhibiting direct communication between nodes [Jedermann14]. Instead, messages are forwarded over multiple hops. Those relaying nodes might belong to different owners. The sensors in the banana boxes cannot always reach the internet during the journey.

The personnel that transloads the goods must be able to locate the goods meant for a specific customer. However the fruit vendor does not want to disclose sensor information pertaining to the condition of the goods to other companies and therefore wants to assure the confidentiality of this data.

When the goods arrive at the supermarket in Munich, the supermarket conducts its own quality check. If no anomalies occurred during the transport, the bananas are admitted for sale.

2.1.2. Authorization Problems Summary

- U1.1 The device owner wants to grant different access rights to a resource to different parties.

- U1.2 The device owner wants to control which devices are allowed to present data to the device.

- U1.3 The device owner wants to grant different access rights for different resources on a device.
- U1.4 The device owner requires the integrity of sensor data.
- U1.5 The device owner requires the confidentiality of sensor data.
- U1.6 The device owner is not always present at the time of access and cannot manually intervene in the authorization process.
- U1.7 The device owner wants to grant temporary access permissions to a party.
- U1.8 Messages between client and resource server might need to be forwarded over multiple hops.
- U1.9 The constrained device might not always be able to reach the internet.

2.2. Home Automation

Automation of the home has the potential to become a big future market for the Internet of Things. A home automation system connects devices in a house to the Internet and thus makes them accessible and manageable remotely. Such devices might control for example heating, ventilation, lighting, home entertainment or home security.

Such a system needs to accommodate a number of regular users (inhabitants, close friends, cleaning personnel) as well as a heterogeneous group of dynamically varying users (visitors, repairmen, delivery men).

As the users are not typically trained in security (or even computer use), the configuration must use secure default settings, and the interface must be well adapted to novice users.

2.2.1. Controlling the Smart Home Infrastructure

Jane and her husband George own a flat which is equipped with home automation devices such as HVAC and shutter control, and they have a motion sensor in the corridor which controls the light bulbs there.

Jane and George can control the shutters and the temperature in each room using either wall-mounted touch panels or their smartphones. Since Jane and George both have a full-time job, they want to be able to change settings remotely, e.g. turn up the heating on a cold day if they will be home earlier than expected.

The couple does not want people in radio range of their devices, e.g. their neighbors, to be able to control them without authorization. Moreover, they don't want burglars to be able to deduce behavioral patterns from eavesdropping on the network.

2.2.2. Seamless Authorization

Jane buys a new light bulb for the corridor and integrates it into the home network (how she does that is not in scope). George is not at home, but Jane wants him to be able to control the new device with his smart phone without the need for additional administration effort.

2.2.3. Remotely letting in a visitor

Jane and George have equipped their home with automated connected door-locks and an alarm system at the door and the windows. The couple can control this system remotely.

Jane and George have invited Jane's parents over for dinner, but are stuck in traffic and can not arrive in time, while Jane's parents who use the subway will arrive punctually. Jane calls her parents and offers to let them in remotely, so they can make themselves comfortable while waiting.

Jane's parents download an application that lets them communicate with Jane's door-lock and alarm system. Then Jane sets temporary permissions that allow them to open the door, and shut down the alarm when they arrive.

The security system controlling the door-locks and alarm system needs to be at least as secure as for a comparable unautomated home.

2.2.4. Authorization Problems Summary

- U2.1 A home owner wants to spontaneously provision authorization means to visitors.
- U2.2 A home owner wants to spontaneously change the home's access control policies.
- U2.3 A home owner wants to apply different access rights for different users.
- U2.4 A home owner wants to apply context-based conditions (presence, time) to authorizations, and the devices need to be able to verify these conditions.

- U2.5 The smart home devices need to be able to communicate with different control devices (e.g. wall-mounted touch panels, smartphones, electronic key fobs).

- U2.6 The access control configuration of the automated home needs to be secure by default.

- U2.7 The access control policies need to be easy to edit, even remotely and it needs to be easy to get access with correct authorization.

- U2.8 The owners of the automated home wants to prevent eavesdroppers form being able to deduce behavioral profiles from the home network.

- U2.9 Usability is particularly important in this scenario since administrative tasks such as installation, configuration and decommissioning of devices likely need to be performed by the home owners who in most cases have little knowledge of security.

- U2.10 Home Owners want their devices to seamlessly (and in some cases even unnoticeably) fulfill their purpose. The administration effort needs to be kept at a minimum.

2.3. Personal Health Monitoring

The use of wearable health monitoring technology is expected to grow strongly, as a multitude of novel devices are developed and marketed. The need for open industry standards to ensure interoperability between products has lead to initiatives such as Continua Alliance (continuaalliance.org) and Personal Connected Health Alliance (pchalliance.org). Personal health devices are typically battery driven, and located physically on the user. They monitor some bodily function, such as e.g. temperature, blood pressure, or pulse. They are connected to the Internet through an intermediary base-station, using wireless technologies. Through this connection they report the monitored data to some entity, which may either be the user herself, or some medical personnel in charge of the user.

Medical data has always been considered as very sensitive, and therefore requires good protection against unauthorized disclosure. A frequent, conflicting requirement is the capability for medical personnel to gain emergency access, even if no specific access rights exist. As a result, the importance of secure audit logs increases in such scenarios.

Since the users are not typically trained in security (or even computer use), the configuration must use secure default settings,

and the interface must be well adapted to novice users. Parts of the system must operate with minimal maintenance. Especially frequent changes of battery are unacceptable.

2.3.1. John and the heart rate monitor

John has a heart condition, that can result in sudden cardiac arrests. He therefore uses a device called HeartGuard that monitors his heart rate and his position. In case of a cardiac arrest it automatically sends an alarm to an emergency service, transmitting John's current location. The HeartGuard also broadcasts emergency information in the neighborhood to notify doctors or people with certain skills who have been enrolled in an emergency program, e.g. people who got training in heart and lung rescue. For doctors, medical information or diagnosis can be provided with the notification to improve immediate treatment.

The device includes some smart logic, with which it identifies its owner John and allows him to configure the device's settings, including access control.
This prevents situation where someone else wearing that device can act as the owner and mess up the access control and security settings.

John can configure additional persons that get notified in an emergency, for example his daughter Jill. Furthermore the device stores data on John's heart rate, which can later be accessed by a physician to assess the condition of John's heart.

However John is a rather private person, and is worried that Jill might use HeartGuard to monitor his location while there is no emergency. Furthermore he doesn't want his health insurance to get access to the HeartGuard data, or even to the fact that he is wearing a HeartGuard, since they might refuse to renew his insurance if they decided he was too big a risk for them.

NOTE: Monitoring of some state parameter (e.g. an alarm button) and the position of a person also fits well into an elderly care service. This is particularly useful for people suffering from dementia, where the relatives or caregivers need to be notified of the whereabouts of the person under certain conditions. In this case it is not the patient that decides about access.

2.3.2. Authorization Problems Summary

- U3.1 A device owner wants to pre-configure access rights to specific data for persons or groups, in the context of an emergency.
- U3.2 A device owner wants to selectively allow different persons or groups to access medical data.
- U3.3 A device owner wants to block access to specific persons in an otherwise allowed group (e.g. doctors in an emergency), if he mistrusts them.
- U3.4 The security measures could affect battery lifetime of the devices and should changes of battery are highly inconvenient.
- U3.5 Devices are often used with default access control settings.
- U3.6 Device users are often not trained in computer use and especially computer security.
- U3.7 Security mechanisms themselves could provide opportunities for denial of service attacks on the device.
- U3.8 The device provides a service that can be fatal for the device owner if it fails. Accordingly, the device owner wants a security mechanism to provide a high level of security.

2.4. Building Automation

Buildings for commercial use such as shopping malls or office buildings nowadays are equipped increasingly with semi-automatic components to enhance the overall living quality and to save energy where possible. This includes for example heating, ventilation and air condition (HVAC) as well as illumination and security systems such as fire alarms.

Different areas of these buildings are often exclusively leased to different companies. However they also share some of the common areas of the building.
Accordingly, a company must be able to control the light and HVAC system of its own part of the building and must not have access to control rooms that belong to other companies.

Some parts of the building automation system such as entrance illumination and fire alarm systems are controlled either by all parties together or by a service company.

2.4.1. Device Lifecycle

2.4.1.1. Installation and Commissioning

A building is hired out to different companies for office space. This building features various automated systems, such as a fire alarm system, which is triggered by several smoke detectors which are spread out across the building. It also has automated HVAC, lighting and physical access control systems.

A vacant area of the building has been recently leased to company A. Before moving into its new office, Company A wishes to replace the lighting with a more energy efficient and a better light quality luminaries. They hire an installation and commissioning company C to redo the illumination. Company C is instructed to integrate the new lighting devices, which may be from multiple manufacturers, into the existing lighting infrastructure of the building which includes presence sensors, switches, controllers etc.

Company C gets the necessary authorization from the service company to interact with the existing Building and Lighting Management System (BLMS). To prevent disturbance to other occupants of the building, Company C is provided authorization to perform the commissioning only during non-office hours and only to modify configuration on devices belonging to the domain of Company A’s space. After installation (wiring) of the new lighting devices, the commissioner adds the devices into the company A’s lighting domain.

Once the devices are in the correct domain, the commissioner authorizes the interaction rules between the new lighting devices and existing devices like presence sensors. For this, the commissioner creates the authorization rules on the BLMS which define which lights form a group and which sensors /switches/controllers are allowed to control which groups. These authorization rules may be context based like time of the day (office or non-office hours) or location of the handheld lighting controller etc.

2.4.1.2. Operational

Company A’s staff move into the newly furnished office space. Most lighting is controlled by presence sensors which control the lighting of specific group of lights based on the authorization rules in the BLMS. Additionally employees are allowed to manually override the lighting brightness and color in their office by using the switches or handheld controllers. Such changes are allowed only if the authorization rules exist in the BLMS. For example lighting in the corridors may not be manually adjustable.

At the end of the day, lighting is dimmed down or switched off if no occupancy is detected even if manually overridden during the day.

On a later date company B also moves into the same building, and shares some of the common spaces with company A. On a really hot day James who works for company A turns on the air condition in his office. Lucy who works for company B wants to make tea using an electric kettle. After she turned it on she goes outside to talk to a colleague until the water is boiling. Unfortunately, her kettle has a malfunction which causes overheating and results in a smoldering fire of the kettle's plastic case.

Due to the smoke coming from the kettle the fire alarm is triggered. Alarm sirens throughout the building are switched on simultaneously (using a broadcastor multicast) to alert the staff of both companies. Additionally, the ventilation system of the whole building is closed off to prevent the smoke from spreading and to withdraw oxygen from the fire. The smoke cannot get into James' office although he turned on his air condition because the fire alarm overrides the manual setting by sending commands (broadcast or multicast) to switch off all the air conditioning.

The fire department is notified of the fire automatically and arrives within a short time. After inspecting the damage and extinguishing the smoldering fire a fire fighter resets the fire alarm because only the fire department is authorized to do that.

2.4.1.3. Maintenance

Company A's staff are annoyed that the lights switch off too often in their rooms if they work silently in front of their computer. Company A notifies the commissioning Company C about the issue and asks them to increase the delay before lights switch off.

Company C again gets the necessary authorization from the service company to interact with the BLMS. The commissioner's tool gets the necessary authorization from BMLS to send a configuration change to all lighting devices in Company A's offices to increase their delay before they switch off.

2.4.1.4. Decommissioning

Company A has noticed that the handheld controllers are often misplaced and hard to find when needed. So most of the time staff use the existing wall switches for manual control. Company A decides it would be better to completely remove handheld controllers and asks Company C to decommission them from the lighting system.

Company C again gets the necessary authorization from the service company to interact with the BLMS. The commissioner now deletes any rules that allowed handheld controllers authorization to control the lighting. Additionally the commissioner instructs the BLMS to push these new rules to prevent cached rules at the end devices from being used.

2.4.2. Authorization Problems Summary

- U4.1 Device owners want to be able to add a new device to their administrative domain (commissioning).
- U4.2 Device owners want to be able to integrate a device that formerly belonged to a different administrative domain to their own administrative domain (handover).
- U4.3 Device owner want to be able to remove a device from their administrative domain (decomissioning).
- U4.4 Device owners want to be able to delegate selected administration tasks for their devices to others.
- U4.5 The device owner wants to be able to define context-based Authorization rules.
- U4.6 The device owner wants to be able to revoke granted permissions and delegations.
- U4.7 The device owner wants to allow only authorized access to device resources (default deny).
- U4.8 The device owner wants to be able to authorize a device to control several devices at the same time using a multicast protocol.
- U4.9 Device owners want to be able to interconnect their own subsystems with those from a different operational domain while keeping the control over the authorizations (e.g. granting and revoking permissions) for their devices.

2.5. Smart Metering

Automated measuring of customer consumption is an established technology for electricity, water, and gas providers. Increasingly these systems also feature networking capability to allow for remote management. Such systems are in use for commercial, industrial and residential customers and require a certain level of security, in order to avoid economic loss to the providers, vulnerability of the

distribution system, as well as disruption of services for the customers.

The smart metering equipment for gas and water solutions is battery driven and communication should be used sparingly due to battery consumption. Therefore the types of meters sleep most of the time, and only wake up every minute/hour to check for incoming instructions. Furthermore they wake up a few times a day (based on their configuration) to upload their measured metering data.

Different networking topologies exist for smart metering solutions. Based on environment, regulatory rules and expected cost, one or a mixture of these topologies may be deployed to collect the metering information. Drive-By metering is one of the most current solutions deployed for collection of gas and water meters.

2.5.1. Drive-by metering

A service operator offers smart metering infrastructures and related services to various utility companies. Among these is a water provider, who in turn supplies several residential complexes in a city. The smart meters are installed in the end customer’s homes to measure water consumption and thus generate billing data for the utility company. The meters do so by sending data to a base station. Several base stations are installed around the city to collect the metering data. However in the denser urban areas, the base stations would have to be installed very close to the meters. This would require a high number of base stations and expose this more expensive equipment to manipulation or sabotage. The service operator has therefore chosen another approach, which is to drive around with a mobile base-station and let the meters connect to that in regular intervals in order to gather metering data.

2.5.2. Meshed Topology

In another deployment, the water meters are installed in a building that already has power meters installed, the latter are mains powered, and are therefore not subject to the same power saving restrictions. The water meters can therefore use the power meters as proxies, in order to achieve better connectivity. This requires the security measures on the water meters to work through intermediaries.

2.5.3. Advanced Metering Infrastructure

A utility company is updating its old utility distribution network with advanced meters and new communication systems, known as an Advanced Metering Infrastructure (AMI). AMI refers to a system that measures, collects and analyzes usage, and interacts with metering

devices such as electricity meters, gas meters, heat meters, and water meters, through various communication media either on request (on-demand) or on pre-defined schedules. Based on this technology, new services make it possible for consumers to control their utility consumption and reduce costs by supporting new tariff models from utility companies, and more accurate and timely billing.

The technical solution is based on levels of data aggregation between smart meters located at the consumer premises and the Meter Data Management (MDM) system located at the utility company. Two possible intermediate levels are:

- Head-End System (HES) which is hardware and software that receives the stream of meter data and exposes an interface to the MDM.

- Data Collection (DC) units located in a local network communicating with a number of smart meters and with a backhaul interface communicating with the HES, e.g. using cellular communication.

For reasons of efficiency and cost end-to-end connectivity is not always feasible, so metering data is stored in batches in DC for some time before being forwarded to the HES, and in turn accessed by the MDM. The HES and the DC units may be operated by a third party service operator on behalf of the utility company. One responsibility of the service operator is to make sure that meter readings are performed and delivered to the HES. An example of a Service Level Agreement between the service operator and the utility company is e.g. "at least 95 % of the meters have readings recorded during the last 72 hours".

2.5.4. Authorization Problems Summary

- U5.1 Devices are installed in hostile environments where they are physically accessible by attackers. Device owners want to make sure that an attacker cannot use a captured device to attack other parts of their infrastructure.

- U5.2 Device owners want to restrict which entities are allowed to write data to the devices and thus ensure the integrity of the data on their devices.

- U5.3 The device owner wants to control which entities are allowed to read data on the devices and protect such data in transfer.

- U5.4 The devices may have intermittent Internet connectivity.

- U5.5 The device owner is not always present at the time of access and cannot manually intervene in the authorization process.

- U5.6 When authorization policies are updated it is impossible, or at least very inefficient to contact all affected devices directly.

- U5.7 Messages between a client and the device may need to be stored and forwarded over multiple nodes.

2.6. Sports and Entertainment

In the area of leisure time activities, applications can benefit from the small size and weight of constrained devices. Sensors and actuators with various functionalities can be integrated into fitness equipment, games and even clothes. Owners can carry their devices around with them at all times.

Usability is especially important in this area since owners will often want to spontaneously interconnect their devices with others. Therefore the configuration of access permissions must be simple and fast and not require much effort at the time of access (preferably none at all).

The required level of security will in most cases be low since security breaches will likely have less severe consequences. The continuous monitoring of data might however enable an attacker to create behavioral or movement profiles. Moreover, the aggregation of data can seriously increase the impact on the privacy of device owners.

2.6.1. Dynamically Connecting Smart Sports Equipment

Jody is a an enthusiastic runner. To keep track of her training progress, she has smart running shoes that measure the pressure at various points beneath her feet to count her steps, detect irregularities in her stride and help her to improve her posture and running style. On a sunny afternoon, she goes to the Finnbahn track near her home to work out. She meets her friend Lynn who shows her the smart fitness watch she bought a few days ago. The watch can measure the wearer’s pulse, show speed and distance, and keep track of the configured training program. The girls detect that the watch can be connected with Jody’s shoes and then can additionally display the information the shoes provide.

Jody asks Lynn to let her try the watch and lend it to her for the afternoon. Lynn agrees but doesn’t want Jody to access her training plan. She configures the access policies for the watch so that

Jody's shoes are allowed to access the display and measuring features but cannot read or add training data. Jody's shoes connect to Lynn's watch after only a press of a button because Jody already configured access rights for devices that belong to Lynn a while ago.

After an hour, Jody gives the watch back and both girls terminate the connection between their devices.

2.6.2. Authorization Problems Summary

- U6.1 The owner of a device wants to be able to grant access rights dynamically when needed.
- U6.2 The owner wants the configuration of access rights to work with very little effort.
- U6.3 The device owner wants to be able to preconfigure access policies that grant certain access permissions to devices with certain attributes (e.g. devices of a certain user) without additional configuration effort at the time of access.
- U6.4 Device owners wants to protect the confidentiality of their data for privacy reasons.
- U6.5 Devices might not have an Internet connection at the time of access.

2.7. Industrial Control Systems

Industrial control systems (ICS) and especially supervisory control and data acquisition systems (SCADA) use a multitude of sensors and actuators in order to monitor and control industrial processes in the physical world. Example processes include manufacturing, power generation, and refining of raw materials.

Since the advent of the Stuxnet worm it has become obvious to the general public how vulnerable this kind of systems are, especially when connected to the Internet. The severity of these vulnerabilities are exacerbated by the fact that many ICS are used to control critical public infrastructure, such as power, water treatment of traffic control. Nevertheless the economical advantages of connecting such systems to the Internet can be significant if appropriate security measures are put in place.

2.7.1. Oil Platform Control

An oil platform uses an industrical control system to monitor data and control equipment. The purpose of this system is to gather and

process data from a large number of sensors, and control actuators such as valves and switches to steer the oil extraction process on the platform. Raw data, alarms, reports and other information are also available to the operators, who can intervene with manual commands. Many of the sensors are connected to the controlling units by direct wire, but the operator is slowly replacing these units by wireless ones, since this makes maintenance easier.

The controlling units are connected to the Internet, to allow for remote administration, since it is expensive and inconvenient to fly in a technician to the platform.

The main interest of the operator is to ensure the integrity of control messages and sensor readings. The access to some resources needs to be restricted to certain clients, e.g. the operator wants wireless actuators only to accept commands by authorized control units.

The owner of the platform also wants to collect auditing information for liability reasons.

2.7.2. Authorization Problems Summary

- U7.1 The device owner wants to ensure that only authorized clients can read data from sensors and sent commands to actuators.
- U7.2 The device owner wants to ensure that data coming from sensors and commands sent to actuators are authentic.
- U7.3 Some devices do not have direct Internet connection.
- U7.4 Some devices have wired connection while other use wireless.
- U7.5 The execution of unauthorized commands in an ICS can lead to significant financial damage, and threaten the availability of critical infrastructure services. Accordingly, the device owner wants a security solution that provides a very high level of security.

3. Security Considerations

As the use cases listed in this document demonstrate, constrained devices are used in various application areas. The appeal of these devices is that they are small and inexpensive. That makes it easy to integrate them into many aspects of everyday life. Therefore, the devices will be entrusted with vast amounts of valuable data or even control functions, that need to be protected from unauthorized access.

Moreover, the aggregation of data must be considered: attackers might not only collect data from a single device but from many devices, thus increasing the potential damage.

Not only the data on the constrained devices themselves is threatened, the devices might also be abused as an intrusion point to infiltrate a network. Once an attacker gained control over the device, it can be used to attack other devices as well. Due to their limited capabilities, constrained devices appear as the weakest link in the network and hence pose an attractive target for attackers.

This section summarizes the security problems highlighted by the use cases above and provides guidelines for the design of protocols for authentication and authorization in constrained RESTful environments.

3.1. Attacks

This document lists security problems that owners of constrained devices want to solve. Further analysis of attack scenarios is not in scope of the document. However, there are attacks that must be considered by solution developers.

Because of the expected large number of devices and their ubiquity, constrained devices increase the danger from Pervasive Monitoring [RFC7258] attacks.

As some of the use cases indicate, constrained devices may be installed in hostile environments where they are physically accessible (see Section 2.5). Protection from physical attacks is not in the scope of ACE, but should be kept in mind by developers of authorization solutions.

Denial of service (DoS) attacks threaten the availability of services a device provides. E.g., an attacker can induce a device to perform steps of a heavy weight security protocol (e.g. Datagram Transport Layer Security (DTLS) [RFC6347]) before authentication and authorization can be verified, thus exhausting the device’s system resources. This leads to a temporary or - e.g. if the batteries are drained - permanent failure of the service. For some services of constrained devices, availability is especially important (see Section 2.3). Because of their limitations, constrained devices are especially vulnerable to denial of service attacks. Solution designers must be particularly careful to consider these limitations in every part of the protocol. This includes:

- Battery usage
- Number of message exchanges required by security measures

- Size of data that is transmitted (e.g. authentication and access control data)
- Size of code required to run the protocol
- Size of RAM memory and stack required to run the protocol

Another category of attacks that needs to be considered by solution developers is session interception and hijacking.

3.2. Configuration of Access Permissions

- The access control policies of the Resource Owner need to be enforced (all use cases): The access control policies set by the Resource Owner need to be provisioned to the device that enforces the authorization and applied to every incoming request.
- A single resource might have different access rights for different requesting entities (all use cases).

  Rationale: In some cases different types of users need different access rights, as opposed to a binary approach where the same access permissions are granted to all authenticated users.
- A device might host several resources where each resource has its own access control policy (all use cases).
- The device that makes the policy decisions should be able to evaluate context-based permissions such as location or time of access (see e.g. Section 2.2, Section 2.3, Section 2.4). Access may depend on local conditions, e.g. access to health data in an emergency. The device that makes the policy decisions should be able to take such conditions into account.

3.3. Design Considerations for Authorization Solutions

- Devices need to be enabled to enforce the owner’s authorization policies without the owner’s intervention at the time of the access request (see e.g. Section 2.1, Section 2.2, Section 2.4, Section 2.5).
- Authorization solutions need to consider that constrained devices might not have internet access at the time of the access request (see e.g. Section 2.1, Section 2.3, Section 2.5, Section 2.6).
- It should be possible to update access control policies without manually re-provisioning individual devices (see e.g. Section 2.2, Section 2.3, Section 2.5, Section 2.6).

Rationale: Peers can change rapidly which makes manual re-provisioning unreasonably expensive.

- Owners might define authorization policies for a large number of devices that might only have intermittent connectivity. Distributing policy updates to every device for every update might not be a feasible solution.

- It must be possible to dynamically revoke authorizations (see e.g. Section 2.4).

- The authentication and access control protocol can put undue burden on the constrained resources of a device participating in the protocol. An authorization solutions must take the limitations of the constrained devices into account (see also Section 3.1).

- Secure default settings are needed for the initial state of the authentication and authorization protocols (all use cases).

  Rationale: Many attacks exploit insecure default settings, and experience shows that default settings are frequently left unchanged by the end users.

- Access to resources on other devices should only be permitted if a rule exists that explicitly allows this access (default deny).

- Usability is important for all use cases. The configuration of authorization policies as well as the gaining access to devices must be simple for the users of the devices. Special care needs to be taken for home scenarios where access control policies have to be configured by users that are typically not trained in security (see Section 2.2, Section 2.6).

3.4. Proxies

In some cases, the traffic between Client and Resource Server might go through intermediary nodes (e.g. proxies, gateways). This might affect the function or the security model of authentication and access control protocols e.g. end-to-end security between Client and Resource Server with DTLS might not be possible (see Section 2.5).

4. Privacy Considerations

Many of the devices that are in focus of this document register data from the physical world (sensors) or affect processes in the physical world (actuators), which may involve data or processes belonging to individuals. To make matters worse the sensor data may be recorded

continuously thus allowing to gather significant information about an individual subject to the sensor readings. Therefore privacy protection is especially important, and Authentication and Access control are important tools for this, since they make it possible to control who gets access to private data.

Privacy protection can also be weighted in when evaluating the need for end-to-end confidentiality, since otherwise intermediary nodes will learn the content of potentially sensitive messages sent between a client and a resource server and thereby endanger the privacy of the individual that may be subject of this data.

In some cases, even the possession of a certain type of device can be confidential, e.g. owners might not want to others to know that they are wearing a certain medical device (see Section 2.3).

The personal health monitoring use case (see Section 2.3) indicates the need for secure audit logs which impose specific requirements on a solution. Auditing is not in the scope of ACE. However, if an authorization solution provides means for audit logs, it must consider the impact of logged data for the privacy of the owner and other parties involved.
Suitable measures for protecting and purging the logs must be taken during operation, maintenance and decommissioning of the device.

5. Acknowledgments

The authors would like to thank Olaf Bergmann, Sumit Singhal, John Mattson, Mohit Sethi, Carsten Bormann, Martin Murillo, Corinna Schmitt, Hannes Tschofenig, Erik Wahlstroem, and Andreas Backman for reviewing and/or contributing to the document. Also, thanks to Markus Becker, Thomas Poetsch and Koojana Kuladinithi for their input on the container monitoring use case.

6. IANA Considerations

This document has no IANA actions.

7. Informative References

[Jedermann14]
Jedermann, R., Poetsch, T., and C. LLoyd, "Communication techniques and challenges for wireless food quality monitoring", Philosophical Transactions of the Royal Society A Mathematical, Physical and Engineering Sciences, May 2014.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

[RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, May 2014.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, June 2014.

[RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, May 2014.

Authors' Addresses

Ludwig Seitz (editor)
SICS Swedish ICT AB
Scheelevaegen 17
Lund 223 70
Sweden

Email: ludwig@sics.se

Stefanie Gerdes (editor)
Universitaet Bremen TZI
Postfach 330440
Bremen 28359
Germany

Phone: +49-421-218-63906
Email: gerdes@tzi.org

Goeran Selander
Ericsson
Faroegatan 6
Kista 164 80
Sweden

Email: goran.selander@ericsson.com

Mehdi Mani
Itron
52, rue Camille Desmoulins
Issy-les-Moulineaux 92130
France

Email: Mehdi.Mani@itron.com

Sandeep S. Kumar
Philips Research
High Tech Campus
Eindhoven 5656 AA
The Netherlands

Email: sandeep.kumar@philips.com

ACE Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 14, 2017

G. Selander
J. Mattsson
F. Palombini
Ericsson AB
L. Seitz
SICS Swedish ICT

October 11, 2016

Object Security of CoAP (OSCOAP)
draft-selander-ace-object-security-06

Abstract

This memo defines Object Security of CoAP (OSCOAP), a method for application layer protection of message exchanges with the Constrained Application Protocol (CoAP), using the CBOR Object Signing and Encryption (COSE) format. OSCOAP provides end-to-end encryption, integrity and replay protection to CoAP payload, options, and header fields, as well as a secure binding between CoAP request and response messages. The use of OSCOAP is signaled with the CoAP option Object-Security, also defined in this memo.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 14, 2017.

Copyright Notice

Table of Contents

1. Introduction

The Constrained Application Protocol (CoAP) [RFC7252] is a web application protocol, designed for constrained nodes and networks [RFC7228]. CoAP specifies the use of proxies for scalability and efficiency. At the same time CoAP references DTLS [RFC6347] for security. Proxy operations on CoAP messages require DTLS to be terminated at the proxy. The proxy therefore not only has access to the data required for performing the intended proxy functionality, but is also able to eavesdrop on, or manipulate any part of the CoAP payload and metadata, in transit between client and server. The proxy can also inject, delete, or reorder packages without being protected or detected by DTLS.

This memo defines Object Security of CoAP (OSCOAP), a data object based security protocol, protecting CoAP message exchanges end-to-end, across intermediary nodes. An analysis of end-to-end security for CoAP messages through intermediary nodes is performed in [I-D.hartke-core-e2e-security-reqs], this specification addresses the forwarding case.

The solution provides an in-layer security protocol for CoAP which does not depend on underlying layers and is therefore favorable for providing security for "CoAP over foo", e.g. CoAP messages passing over both unreliable and reliable transport [I-D.ietf-core-coap-tcp-tls], CoAP over IEEE 802.15.4 IE [I-D.bormann-6lo-coap-802-15-ie].

OSCOAP builds on CBOR Object Signing and Encryption (COSE) [I-D.ietf-cose-msg], providing end-to-end encryption, integrity, and replay protection. The use of OSCOAP is signaled with the CoAP option Object-Security, also defined in this memo. The solution transforms an unprotected CoAP message into a protected CoAP message in the following way: the unprotected CoAP message is protected by including payload (if present), certain options, and header fields in a COSE object. The message fields that have been encrypted are removed from the message whereas the Object-Security option and the COSE object are added. We call the result the "protected" CoAP message. Thus OSCOAP is a security protocol based on the exchange of protected CoAP messages (see Figure 1).

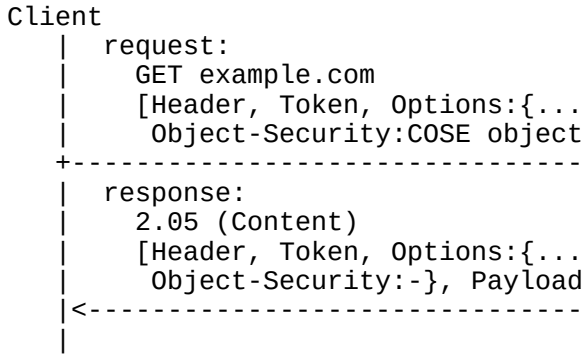


Figure 1: Sketch of OSCOAP

OSCOAP provides protection of CoAP payload, certain options, and header fields, as well as a secure binding between CoAP request and response messages, and freshness of requests and responses. It may be used in extremely constrained settings, where DTLS cannot be supported. Alternatively, OSCOAP can be combined with DTLS, thereby enabling end-to-end security of CoAP payload, in combination with hop-by-hop protection of the entire CoAP message, during transport between end-point and intermediary node. Examples of the use of OSCOAP are given in Appendix B.

The message protection provided by OSCOAP can alternatively be applied only to the payload of individual messages. We call this object security of content (OSCON) and it is defined in Appendix C.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119]. These words may also appear in this document in lowercase, absent their normative meanings.

Readers are expected to be familiar with the terms and concepts described in [RFC7252] and [RFC7641].

Terminology for constrained environments, such as "constrained device", "constrained-node network", is defined in [RFC7228].

Two different scopes of object security are defined:

- OSCOAP = object security of CoAP, signaled with the Object-Security option.

- OSCON = object security of content, signaled with Content Format/ Media Type set to application/oscon (defined in Appendix C).

2. The Object-Security Option

The Object-Security option indicates that OSCOAP is used to protect the CoAP message exchange. The protection is achieved by means of a COSE object included in the protected CoAP message, as detailed in Section 5.

The Object-Security option is critical, safe to forward, part of the cache key, and not repeatable. Figure 2 illustrates the structure of the Object-Security option.

A CoAP proxy SHOULD NOT cache a response to a request with an Object-Security option, since the response is only applicable to the original client's request. The Object-Security option is included in the cache key for backward compatibility with proxies not recognizing the Object-Security option. The effect of this is that messages with the Object-Security option will never generate cache hits. To further prevent caching, a Max-Age option with value zero SHOULD be added to the protected CoAP responses.

No.	C	U	N	R	Name	Format	Length
TBD	x				Object-Security	opaque	0-

C=Critical, U=Unsafe, N=NoCacheKey, R=Repeatable

Figure 2: The Object-Security Option

The length of the Object-Security option depends on whether the unprotected message has payload, on the set of options that are included in the unprotected message, the length of the integrity tag, and the length of the information identifying the security context.

- If the unprotected message has payload, then the COSE object is the payload of the protected message (see Section 6.2 and Section 6.4), and the Object-Security option has length zero. An endpoint receiving a CoAP message with payload, that also contains a non-empty Object-Security option SHALL treat it as malformed and reject it.

- If the unprotected message does not have payload, then the COSE object is the value of the Object-Security option and the length of the Object-Security option is equal to the size of the COSE object. An endpoint receiving a CoAP message without payload,

that also contains an empty Object-Security option SHALL treat it as malformed and reject it.

More details about the message overhead caused by the Object-Security option is given in Appendix A.

3. The Security Context

OSCOAP uses COSE with an Authenticated Encryption with Additional Data (AEAD) algorithm. The specification requires that client and server establish a security context to apply to the COSE objects protecting the CoAP messages. In this section we define the security context, and also specify how to establish a security context in client and server based on common shared secret material and a key derivation function (KDF).

The EDHOC protocol [I-D.selander-ace-cose-ecdhe] enables the establishment of secret material with the property of forward secrecy, and negotiation of KDF and AEAD, it thus provides all necessary pre-requisite steps for using OSCOAP as defined here.

3.1. Security Context Definition

The security context is the set of information elements necessary to carry out the cryptographic operations in OSCOAP. Each security context is identified by a Context Identifier. A Context Identifier that is no longer in use can be reassigned to a new security context.

For each endpoint, the security context is composed by a "Common Context", a "Sender Context" and a "Recipient Context". The Common Context includes common security material. The endpoint protects the messages sent using the Sender Context. The endpoint verifies the messages received using the Recipient Context. In communication between two endpoints, the Sender Context of one endpoint matches the Recipient Context of the other endpoint, and vice versa. Note that, because of that, the two security contexts identified by the same Context Identifiers in the two endpoints are not the same, but they are partly mirrored.

An example is shown in Figure 3.

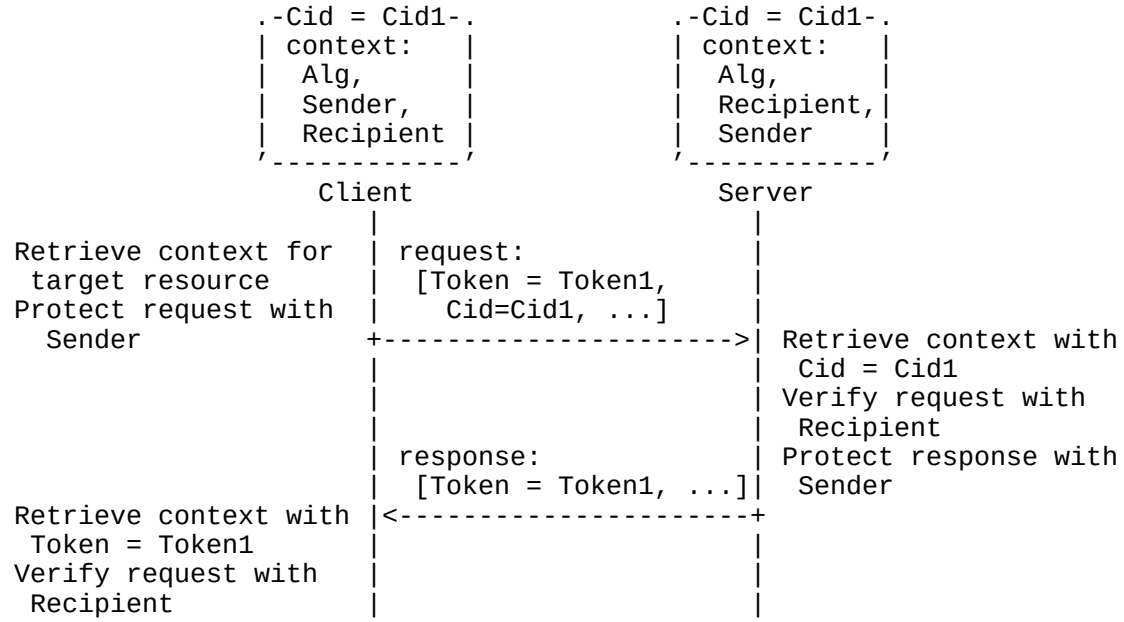


Figure 3: Retrieval and use of the Security Context

The Common Context structure contains the following parameters:

- Context Identifier (Cid). Variable length byte string that identifies the security context. Its value is immutable once the security context is established.

- Algorithm (Alg). Value that identifies the COSE AEAD algorithm to use for encryption. Its value is immutable once the security context is established.

- Base Key (base_key). Byte string containing the key used to derive the security context Section 3.2.

The Sender Context structure contains the following parameters:

- Sender ID. Variable length byte string identifying oneself. Its value is immutable once the security context is established.

- Sender Key. Byte string containing the symmetric key to protect messages to send. Length is determined by Algorithm. Its value is immutable once the security context is established.

- Sender IV. Byte string containing the fixed portion of IV (context IV in [I-D.ietf-cose-msg]) to protect messages to send.

Length is determined by Algorithm. Its value is immutable once the security context is established.

- Sender Sequence Number. Non-negative integer enumerating the COSE objects that the endpoint sends, associated to the Context Identifier. It is used for replay protection, and to generate unique IVs for the AEAD. Maximum value is determined by Algorithm.

The Recipient Context structure contains the following parameters:

- Recipient ID. Variable length byte string identifying the endpoint messages are received from or sent to. Its value is immutable once the security context is established.

- Recipient Key. Byte string containing the symmetric key to verify messages received. Length is determined by the Algorithm. Its value is immutable once the security context is established.

- Recipient IV. Byte string containing the context IV to verify messages received. Length is determined by Algorithm. Its value is immutable once the security context is established.

- Recipient Sequence Number. Non-negative integer enumerating the COSE objects received, associated to the Context Identifier. It is used for replay protection, and to generate unique IVs for the AEAD. Maximum value is determined by Algorithm.

- Replay Window. The replay protection window for messages received, equivalent to the functionality described in Section 4.1.2.6 of [RFC6347].

The 3-tuple (Cid, Sender ID, Sender Sequence Number) is called Transaction Identifier (Tid), and SHALL be unique for each COSE object and server. The Tid is used as a unique challenge in the COSE object of the protected CoAP request. The Tid is part of the Additional Authenticated Data (AAD, see Section 5) of the protected CoAP response message, which is how the challenge becomes signed by the server.

The client and server may change roles while maintaining the same security context. The former server will then make the request using the Sender Context, the former client will verify the request using its Recipient Context etc.

3.2. Security Context Establishment

This section aims at describing how to establish the security context, given some input parameters. The input parameters, which are established in a previous phase, are:

- Context Identifier (Cid)
- Algorithm (Alg)
- Base Key (base_key)
- Sender ID
- Recipient ID
- Replay Window (optionally)

These are included unchanged in the security context. We give below some indications on how applications should select these parameters. Moreover, the following parameters are established as described below:

- Sender Key
- Sender IV
- Sender Sequence Number
- Recipient Key
- Recipient IV
- Recipient Sequence Number
- Replay Window

3.2.1. Derivation of Sender Key/IV, Recipient Key/IV

Given a common shared secret material and a common key derivation function, the client and server can derive the security context necessary to run OSCOAP. The derivation procedure described here MUST NOT be executed more than once on a set of common secret material. Also, the same base_key SHOULD NOT be used in different security contexts (identified by different Cids).

The procedure assumes that the common shared secret material is uniformly random and that the key derivation function is HKDF

[RFC5869]. This is for example the case after having used EDHOC [I-D.selander-ace-cose-ecdhe].

Assumptions:

- The hash function, denoted HKDF, is the HMAC based key derivation function defined in [RFC5869] with specified hash function
- The common shared secret material, denoted base_key, is uniformly pseudo-random of length at least equal to the output of the specified hash function

The security context parameters Sender Key/IV, Recipient Key/IV SHALL be derived using the HKDF-Expand primitive [RFC5869]:

output parameter = HKDF-Expand(base_key, info, key_length),

where:

- base_key is defined above
- info = Cid || Sender ID/Recipient ID || "IV"/"Key" || Algorithm || key_length
- key_length is the key size of the AEAD algorithm

The Sender/Recipient Key shall be derived using the Cid concatenated with the Sender/Recipient ID, the label "Key", the Algorithm and the key_length. The Sender/Recipient IV shall be derived using the Cid concatenated with the Sender/Recipient ID, the label "IV", the Algorithm and the key_length.

For example, for the algorithm AES-CCM-64-64-128 (see Section 10.2 in [I-D.ietf-cose-msg]), key_length for the keys is 128 bits and key_length for the context IVs is 56 bits.

3.2.2. Sequence Numbers and Replay Window

The values of the Sequence Numbers are initialized to 0 during establishment of the security context. The default Replay Window size of 64 is used if no input parameter is provided in the set up phase.

3.2.3. Context Identifier and Sender/Recipient ID

As mentioned, Cid, Sender ID and Recipient ID are established in a previous phase. How this is done is application specific, but some guidelines are given in this section.

It is RECOMMENDED that the application uses 64-bits long pseudo-random Cids, in order to have globally unique Context Identifiers. Cid SHOULD be unique in the sets of all security contexts used by all the endpoints. If it is not the case, it is the role of the application to specify how to handle collisions.

In the same phase during which the Cid is established in the endpoint, the application informs the endpoint what resource can be accessed using the corresponding security context. The granularity of that is decided by the application (resource, host, etc). The endpoint SHALL save the association resource-Cid, in order to be able to retrieve the correct security context to access a resource.

The Sender ID and Recipient ID are also established in the endpoint during the previous set up phase. The application SHOULD make sure that these identifiers are locally unique in the set of all endpoints using the same security context. If it is not the case, it is again the role of the application to specify how to handle collisions.

In case of EDHOC [I-D.selander-ace-cose-ecdhe]) the Cid is the hash of the messages exchanged.

4. Protected CoAP Message Fields

This section defines how the CoAP message fields are protected. OSCOAP protects as much of the unprotected CoAP message as possible, while still allowing forward proxy operations [I-D.hartke-core-e2e-security-reqs].

The CoAP Payload SHALL be encrypted and integrity protected.

The CoAP Header fields Version and Code SHALL be integrity protected but not encrypted. The CoAP Message Layer parameters, Type and Message ID, as well as Token and Token Length SHALL neither be integrity protected nor encrypted.

Protection of CoAP Options can be summarized as follows:

- To prevent information leakage, Uri-Path and Uri-Query SHALL be encrypted. As a consequence, if Proxy-Uri is used, those parts of the URI SHALL be removed from the Proxy-Uri. The CoAP Options Uri-Host, Uri-Port, Proxy-Uri, and Proxy-Scheme SHALL neither be encrypted, nor integrity protected (cf. protection of the effective request URI in Section 5.2).

- The other CoAP options SHALL be encrypted and integrity protected.

A summary of which options are encrypted or integrity protected is shown in Figure 4.

No.	C	U	N	R	Name	Format	Length	E	D
1	x			x	If-Match	opaque	0-8	x	
3	x	x	-		Uri-Host	string	1-255		
4				x	ETag	opaque	1-8	x	
5	x				If-None-Match	empty	0	x	
6		x	-		Observe	uint	0-3	x	x
7	x	x	-		Uri-Port	uint	0-2		
8				x	Location-Path	string	0-255	x	
11	x	x	-	x	Uri-Path	string	0-255	x	
12					Content-Format	uint	0-2	x	
14		x	-		Max-Age	uint	0-4	x	x
15	x	x	-	x	Uri-Query	string	0-255	x	
17	x				Accept	uint	0-2	x	
20				x	Location-Query	string	0-255	x	
23	x	x	-	-	Block2	uint	0-3	x	x
27	x	x	-	-	Block1	uint	0-3	x	x
28			x		Size2	unit	0-4	x	x
35	x	x	-		Proxy-Uri	string	1-1034		
39	x	x	-		Proxy-Scheme	string	1-255		
60			x		Size1	uint	0-4	x	x

C=Critical, U=Unsafe, N=NoCacheKey, R=Repeatable,
E=Encrypt and Integrity Protect, D=Duplicate.

Figure 4: Protection of CoAP Options

Unless specified otherwise, CoAP options not listed in Figure 4 SHALL be encrypted and integrity protected.

The encrypted options are in general omitted from the protected CoAP message and not visible to intermediary nodes (see Section 6.2 and Section 6.4). Hence the actions resulting from the use of corresponding options is analogous to the case of communicating directly with the endpoint. For example, a client using an ETag option will not be served by a proxy.

However, some options which are encrypted need to be readable in the protected CoAP message to support certain proxy functions. A CoAP option which may be both encrypted in the COSE object of the protected CoAP message, and also unencrypted as CoAP option in the protected CoAP message, is called "duplicate". The "encrypted" value of a duplicate option is intended for the destination endpoint and

the "unencrypted" value is intended for a proxy. The unencrypted value is not integrity protected.

- The Max-Age option is duplicate. The unencrypted Max-Age SHOULD have value zero to prevent caching of responses. The encrypted Max-Age is used as defined in [RFC7252] taking into account that it is not accessible to proxies.

- The Observe option is duplicate. If Observe is used, then the encrypted Observe and the unencrypted Observe SHALL have the same value. The Observe option as used here targets the requirements on forwarding of [I-D.hartke-core-e2e-security-reqs] (Section 2.2.1.2).

- The block options Block1 and Block2 are duplicate. The encrypted block options is used for end-to-end secure fragmentation of payload into blocks and protected information about the fragmentation (block number, last block, etc.). The MAC from each block is included in the calculation of the MAC for the next block's (see Section 5.2). In this way, each block in ordered sequence from the first block can be verified as it arrives. The unencrypted block option allows for arbitrary proxy fragmentation operations which cannot be verified by the endpoints. An intermediary node can generate an arbitrarily long sequence of blocks. However, since it is possible to protect fragmentation of large messages, there SHALL be a security policy defining a maximum unfragmented message size such that messages exceeding this size SHALL be fragmented by the sending endpoint. Hence an endpoint receiving fragments of a message that exceeds maximum message size SHALL discard this message.

- The size options Size1 and Size2 are duplicate, analogously to the block options.

Specifications of new CoAP options SHOULD specify how they are processed with OSCOAP. New COAP options SHALL be encrypted and integrity protected. New COAP options SHOULD NOT be duplicate unless a forwarding proxy needs to read the option. If an option is registered as duplicate, the duplicate value SHOULD NOT be the same as the end-to-end value, unless the proxy is required by specification to be able to read the end-to-end value.

5. The COSE Object

This section defines how to use the COSE format [I-D.ietf-cose-msg] to wrap and protect data in the unprotected CoAP message. OSCOAP uses the COSE_Encrypt0 structure with an Authenticated Encryption with Additional Data (AEAD) algorithm.

The mandatory to support AEAD algorithm is AES-CCM-64-64-128 defined in Section 10.2 of [I-D.ietf-cose-msg]. For AES-CCM-64-64-128 the length of Sender Key and Recipient Key SHALL be 128 bits, the length of IV, Sender IV, and Recipient IV SHALL be 7 bytes, and the maximum Sender Sequence Number and Recipient Sequence Number SHALL be 2^56-1. The IV is constructed using a Partial IV exactly like in Section 3.1 of [I-D.ietf-cose-msg], i.e. by padding the Sender Sequence Number or the Recipient Sequence Number with zeroes and XORing it with the Sender IV or Recipient IV, respectively.

Since OSCOAP only makes use of a single COSE structure, there is no need to explicitly specify the structure, and OSCOAP uses the untagged version of the COSE_Encrypt0 structure (Section 2. of [I-D.ietf-cose-msg]). If the COSE object has a different structure, the recipient MUST reject the message, treating it as malformed.

We denote by Plaintext the data that is encrypted and integrity protected, and by Additional Authenticated Data (AAD) the data that is integrity protected only, in the COSE object.

The fields of COSE_Encrypt0 structure are defined as follows (see example in Appendix C.4).

- The "Headers" field is formed by:
  - The "protected" field, which SHALL include:
    - The "Partial IV" parameter. The value is set to the Sender Sequence Number. The Partial IV is a byte string (type: bstr), where the length is the minimum length needed to encode the sequence number. An Endpoint that receives a COSE object with a sequence number encoded with leading zeroes (i.e. longer than the minimum needed length) SHALL reject the corresponding message as malformed.
    - If the message is a CoAP request, the "kid" parameter. The value is set to the Context Identifier (see Section 3).
    - Optionally, the parameter called "sid", defined below. The value is set to the Sender ID (see Section 3). Note that since this parameter is sent in clear, privacy issues SHOULD be considered by the application defining the Sender ID.
  - The "unprotected" field, which SHALL be empty.
- The "cipher text" field is computed from the Plaintext (see Section 5.1) and the Additional Authenticated Data (AAD) (see

Section 5.2) and encoded as a byte string (type: bstr), following Section 5.2 of [I-D.ietf-cose-msg].

sid: This parameter is used to identify the sender of the message. Applications MUST NOT assume that 'sid' values are unique. This is not a security critical field. For this reason, it can be placed in the unprotected headers bucket.

name	label	value type	value registry	description
sid	TBD	bstr		Sender identifier

Table 1: Additional COSE Header Parameter

5.1. Plaintext

The Plaintext is formatted as a CoAP message without Header (see Figure 5) consisting of:

- all CoAP Options present in the unprotected message which are encrypted (see Section 4), in the order as given by the Option number (each Option with Option Header including delta to previous included encrypted option); and
- the CoAP Payload, if present, and in that case prefixed by the one-byte Payload Marker (0xFF).

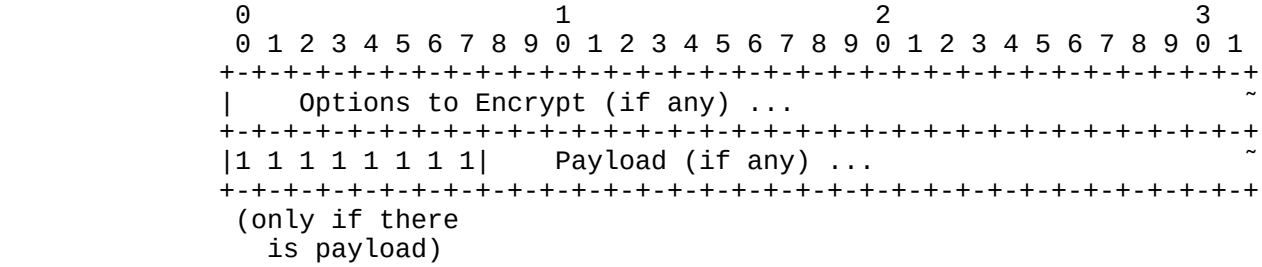


Figure 5: Plaintext

5.2. Additional Authenticated Data

The Additional Authenticated Data ("Enc_structure") as described is Section 5.3 of [I-D.ietf-cose-msg] includes:

- the "context" parameter, which has value "Encrypted"

- the "protected" parameter, which includes the "protected" part of the "Headers" field;

- the "external_aad" is a serialized CBOR array (see Figure 8) that contains, in the given order:

  * ver: uint, contains the CoAP version number of the unprotected CoAP message, as defined in Section 3 of [RFC7252]

  * code: bstr, contains is the CoAP Code of the unprotected CoAP message, as defined in Section 3 of [RFC7252].

  * alg: bstr, contains the serialized Algorithm from the security context used for the exchange (see Section 3.1);

  * request-uri: tstr, contains the plaintext "effective" request URI composed from the request scheme and Uri-* options according to the method described in Section 6.5 of [RFC7252], if the message is a CoAP request;

  * transaction-id: bstr, only included if the message to protect or verify is a CoAP response, contains the Transaction Identifier (Tid) of the associated CoAP request (see Section 3). Note that the Tid is the 3-tuple (Cid, Sender ID, Sender Sequence Number) for the endpoint sending the request and verifying the response; which means that for the endpoint sending the response, the Tid has value (Cid, Recipient ID, seq), where seq is the value of the "Partial IV" in the COSE object of the request (see Section 5); and

  * mac-previous-block: bstr, contains the MAC of the message containing the previous block in the sequence, as enumerated by Block1 in the case of a request and Block2 in the case of a response, if the message is fragmented using a block option [RFC7959].

```
external_aad_req = [
    ver : uint,
    code : bstr,
    alg : bstr,
    request-uri : tstr,
    ? mac-previous-block : bstr
]
```

Figure 6: external_aad for a request

```
external_aad_resp = [
    ver : uint,
    code : bstr,
    alg : bstr,
    transaction-id : bstr,
    ? mac-previous-block : bstr
]
```

Figure 7: external_aad for a response

```
external_aad = external_aad_req / external_aad_resp
```

Figure 8: external_aad

The encryption process is described in Section 5.3 of [I-D.ietf-cose-msg].

6. Protecting CoAP Messages

6.1. Replay and Freshness Protection

In order to protect from replay of messages and verify freshness, a CoAP endpoint SHALL maintain a Sender Sequence Number, and a Recipient Sequence Number associated to a security context, which is identified with a Context Identifier (Cid). The two sequence numbers are the highest sequence number the endpoint has sent and the highest sequence number the endpoint has received. An endpoint uses the Sender Sequence Number to protect messages to send and the Recipient Sequence Number to verify received messages, as described in Section 3.

Depending on use case and ordering of messages provided by underlying layers, an endpoint MAY maintain a sliding replay window for Sequence Numbers of received messages associated to each Cid. In case of reliable transport, the receiving endpoint MAY require that the Sequence Number of a received message equals last Sequence Number + 1.

A receiving endpoint SHALL verify that the Sequence Number received in the COSE object has not been received before in the security context identified by the Cid. The receiving endpoint SHALL also reject messages with a sequence number greater than 2^56-1.

OSCOAP is a challenge-response protocol, where the response is verified to match a prior request, by including the unique transaction identifier (Tid as defined in Section 3) of the request in the Additional Authenticated Data of the response message.

If a CoAP server receives a request with the Object-Security option, then the server SHALL include the Tid of the request in the AAD of the response, as described in Section 6.4.

If the CoAP client receives a response with the Object-Security option, then the client SHALL verify the integrity of the response, using the Tid of its own associated request in the AAD, as described in Section 6.5.

6.2. Protecting the Request

Given an unprotected CoAP request, including header, options and payload, the client SHALL perform the following steps to create a protected CoAP request using a security context associated with the target resource (see Section 3.2.3).

1. Increment the Sender Sequence Number by one (note that this means that sequence number 0 is never used). If the Sender Sequence Number exceeds the maximum number for the AEAD algorithm, the client MUST NOT process any requests with the given security context. The client SHOULD acquire a new security context (and consequently inform the server about it) before this happens. The latter is out of scope of this memo.

2. Compute the COSE object as specified in Section 5

   * the IV in the AEAD is created by XORing the Sender IV (context IV) with the Sender Sequence Number (partial IV).

   * If the block option is used, the AAD includes the MAC from the previous fragment sent (from the second fragment and following) Section 5.2. This means that the endpoint MUST store the MAC of each last-sent fragment to compute the following.

   * Note that the 'sid' field containing the Sender ID is included in the COSE object (Section 5) if the application needs it.

3. Format the protected CoAP message as an ordinary CoAP message, with the following Header, Options, and Payload, based on the unprotected CoAP message:

   * The CoAP header is the same as the unprotected CoAP message.

   * The CoAP options which are encrypted and not duplicate (Section 4) are removed. Any duplicate option which is present has its unencrypted value. The Object-Security option is added.

   * If the message type of the unprotected CoAP message does not allow Payload, then the value of the Object-Security option is the COSE object. If the message type of the unprotected CoAP message allows Payload, then the Object-Security option is empty and the Payload of the protected CoAP message is the COSE object.

4. Store in memory the association Token - Cid. The Client SHALL be able to find the correct security context used to protect the request and verify the response with use of the Token of the message exchange.

6.3. Verifying the Request

A CoAP server receiving a message containing the Object-Security option SHALL perform the following steps, using the security context identified by the Context Identifier in the "kid" parameter in the received COSE object:

1. Verify the Sequence Number in the Partial IV parameter, as described in Section 6.1. If it cannot be verified that the Sequence Number has not been received before, the server MUST stop processing the request.

2. Recreate the Additional Authenticated Data, as described in Section 5.

   * If the block option is used, the AAD includes the MAC from the previous fragment received (from the second fragment and following) Section 5.2. This means that the endpoint MUST store the MAC of each last-received fragment to compute the following.

3. Compose the IV by XORing the Recipient IV (context IV) with the Partial IV parameter, received in the COSE Object.

4. Retrieve the Recipient Key.

5. Verify and decrypt the message. If the verification fails, the server MUST stop processing the request.

6. If the message verifies, update the Recipient Sequence Number or Replay Window, as described in Section 6.1.

7. Restore the unprotected request by adding any decrypted options or payload from the plaintext. Any duplicate options (Section 4) are overwritten. The Object-Security option is removed.

6.4. Protecting the Response

A server receiving a valid request with a protected CoAP message (i.e. containing an Object-Security option) SHALL respond with a protected CoAP message.

Given an unprotected CoAP response, including header, options, and payload, the server SHALL perform the following steps to create a protected CoAP response, using the security context identified by the Context Identifier of the received request:

1. Increment the Sender Sequence Number by one (note that this means that sequence number 0 is never used). If the Sender Sequence Number exceeds the maximum number for the AEAD algorithm, the server MUST NOT process any more responses with the given security context. The server SHOULD acquire a new security context (and consequently inform the client about it) before this happens. The latter is out of scope of this memo.

2. Compute the COSE object as specified in Section Section 5

   * The IV in the AEAD is created by XORing the Sender IV (context IV) and the Sender Sequence Number.

   * If the block option is used, the AAD includes the MAC from the previous fragment sent (from the second fragment and following) Section 5.2. This means that the endpoint MUST store the MAC of each last-sent fragment to compute the following.

3. Format the protected CoAP message as an ordinary CoAP message, with the following Header, Options, and Payload based on the unprotected CoAP message:

   * The CoAP header is the same as the unprotected CoAP message.

   * The CoAP options which are encrypted and not duplicate (Section 4) are removed. Any duplicate option which is present has its unencrypted value. The Object-Security option is added.

   * If the message type of the unprotected CoAP message does not allow Payload, then the value of the Object-Security option is the COSE object. If the message type of the unprotected CoAP message allows Payload, then the Object-Security option is empty and the Payload of the protected CoAP message is the COSE object.

Note the differences between generating a protected request, and a protected response, for example whether "kid" is present in the header, or whether Destination URI or Tid is present in the AAD, of the COSE object.

6.5. Verifying the Response

A CoAP client receiving a message containing the Object-Security option SHALL perform the following steps, using the security context identified by the Token of the received response:

1. Verify the Sequence Number in the Partial IV parameter as described in Section 6.1. If it cannot be verified that the Sequence Number has not been received before, the client MUST stop processing the response.

2. Recreate the Additional Authenticated Data as described in Section 5.

   * If the block option is used, the AAD includes the MAC from the previous fragment received (from the second fragment and following) Section 5.2. This means that the endpoint MUST store the MAC of each last-received fragment to compute the following.

3. Compose the IV by XORing the Recipient IV (context IV) with the Partial IV parameter, received in the COSE Object.

4. Retrieve the Recipient Key.

5. Verify and decrypt the message. If the verification fails, the client MUST stop processing the response.

6. If the message verifies, update the Recipient Sequence Number or Replay Window, as described in Section 6.1.

7. Restore the unprotected response by adding any decrypted options or payload from the plaintext. Any duplicate options (Section 4) are overwritten. The Object-Security option is removed.

7. Security Considerations

In scenarios with intermediary nodes such as proxies or brokers, transport layer security such as DTLS only protects data hop-by-hop. As a consequence the intermediary nodes can read and modify information. The trust model where all intermediate nodes are considered trustworthy is problematic, not only from a privacy perspective, but also from a security perspective, as the

intermediaries are free to delete resources on sensors and falsify commands to actuators (such as "unlock door", "start fire alarm", "raise bridge"). Even in the rare cases, where all the owners of the intermediary nodes are fully trusted, attacks and data breaches make such an architecture brittle.

DTLS protects hop-by-hop the entire CoAP message, including header, options, and payload. OSCOAP protects end-to-end the payload, and all information in the options and header, that is not required for forwarding (see Section 4). DTLS and OSCOAP can be combined, thereby enabling end-to-end security of CoAP payload, in combination with hop-by-hop protection of the entire CoAP message, during transport between end-point and intermediary node.

The CoAP message layer, however, cannot be protected end-to-end through intermediary devices since the parameters Type and Message ID, as well as Token and Token Length may be changed by a proxy. Moreover, messages that are not possible to verify should for security reasons not always be acknowledged but in some cases be silently dropped. This would not comply with CoAP message layer, but does not have an impact on the application layer security solution, since message layer is excluded from that.

The use of COSE to protect CoAP messages as specified in this document requires an established security context. The method to establish the security context described in Section 3.2 is based on a common shared secret material and key derivation function in client and server. EDHOC [I-D.selander-ace-cose-ecdhe] describes an augmented Diffie-Hellman key exchange to produce forward secret keying material and agree on crypto algorithms necessary for OSCOAP, authenticated with pre-established credentials. These pre-established credentials may, in turn, be provisioned using a trusted third party such as described in the OAuth-based ACE framework [I-D.ietf-ace-oauth-authz]. An OSCOAP profile of ACE is described in [I-D.seitz-ace-oscoap-profile].

For symmetric encryption it is required to have a unique IV for each message, for which the sequence numbers in the COSE message field "Partial IV" is used. The context IVs (Sender IV and Recipient IV) SHOULD be established between sender and recipient before the message is sent, for example using the method in [I-D.selander-ace-cose-ecdhe], to avoid the overhead of sending it in each message.

The MTI AEAD algorithm AES-CCM-64-64-128 is selected for broad applicability in terms of message size (2^64 blocks) and maximum no. messages (2^56-1). For 128 bit CCM*, use instead AES-CCM-16-64-128 [I-D.ietf-cose-msg].

If the recipient accepts any sequence number larger than the one previously received (less than the maximum sequence number), then the problem of sequence number synchronization is avoided. With reliable transport it may be defined that only messages with sequence number which are equal to previous sequence number + 1 are accepted. The alternatives to sequence numbers have their issues: very constrained devices may not be able to support accurate time, or to generate and store large numbers of random IVs. The requirement to change key at counter wrap is a complication, but it also forces the user of this specification to think about implementing key renewal.

The encrypted block options enable the sender to split large messages into protected fragments such that the receiving node can verify blocks before having received the complete message. In order to protect from attacks replacing fragments from a different message with the same block number between same endpoints and same resource at roughly the same time, the MAC from the message containing one block is included in the external_aad of the message containing the next block.

The unencrypted block options allow for arbitrary proxy fragmentation operations which cannot be verified by the endpoints, but can by policy be restricted in size since the encrypted options allow for secure fragmentation of very large messages. A maximum message size (above which the sending endpoint fragments the message and the receiving endpoint discards the message, if complying to the policy) may be obtained as part of normal resource discovery.

8. Privacy Considerations

Privacy threats executed through intermediate nodes are considerably reduced by means of OSCOAP. End-to-end integrity protection and encryption of CoAP payload and all options that are not used for forwarding, provide mitigation against attacks on sensor and actuator communication, which may have a direct impact on the personal sphere.

CoAP headers sent in plaintext allow for example matching of CON and ACK (CoAP Message Identifier), matching of request and responses (Token) and traffic analysis.

9. IANA Considerations

Note to RFC Editor: Please replace all occurrences of "[[this document]]" with the RFC number of this specification.

9.1. Sid Registration

IANA is requested to enter a new parameter entitled "sid" to the registry "COSE Header Parameters". The parameter is defined in Table 1.

9.2. CoAP Option Number Registration

The Object-Security option is added to the CoAP Option Numbers registry:

Number	Name	Reference
TBD	Object-Security	[[this document]]

9.3. Media Type Registrations

The "application/oscon" media type is added to the Media Types registry:

Type name: application

Subtype name: cose

Required parameters: N/A

Optional parameters: N/A

Encoding considerations: binary

Security considerations: See the Security Considerations section of [[this document]].

Interoperability considerations: N/A

Published specification: [[this document]]

Applications that use this media type: To be identified

Fragment identifier considerations: N/A

Additional information:

* Magic number(s): N/A

* File extension(s): N/A

* Macintosh file type code(s): N/A

Person & email address to contact for further information: iesg@ietf.org

Intended usage: COMMON

Restrictions on usage: N/A

Author: Goeran Selander, goran.selander@ericsson.com

Change Controller: IESG

Provisional registration? No

9.4. CoAP Content Format Registration

The "application/oscon" content format is added to the CoAP Content Format registry:

Media type	Encoding	ID	Reference
application/oscon	-	70	[[this document]]

10. Acknowledgments

Klaus Hartke has independently been working on the same problem and a similar solution: establishing end-to-end security across proxies by adding a CoAP option. We are grateful to Malisa Vucinic and Marco Tiloca for providing helpful and timely reviews of previous versions of the draft. We are also grateful to Carsten Bormann and Jim Schaad for providing input and interesting discussions.

11. References

11.1. Normative References

[I-D.ietf-cose-msg] Schaad, J., "CBOR Object Signing and Encryption (COSE)", draft-ietf-cose-msg-20 (work in progress), October 2016.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <http://www.rfc-editor.org/info/rfc6347>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <http://www.rfc-editor.org/info/rfc7252>.

[RFC7641] Hartke, K., "Observing Resources in the Constrained Application Protocol (CoAP)", RFC 7641, DOI 10.17487/RFC7641, September 2015, <http://www.rfc-editor.org/info/rfc7641>.

[RFC7959] Bormann, C. and Z. Shelby, Ed., "Block-Wise Transfers in the Constrained Application Protocol (CoAP)", RFC 7959, DOI 10.17487/RFC7959, August 2016, <http://www.rfc-editor.org/info/rfc7959>.

11.2. Informative References

[I-D.bormann-6lo-coap-802-15-ie]
Bormann, C., "Constrained Application Protocol (CoAP) over IEEE 802.15.4 Information Element for IETF", draft-bormann-6lo-coap-802-15-ie-00 (work in progress), April 2016.

[I-D.hartke-core-e2e-security-reqs]
Selander, G., Palombini, F., and K. Hartke, "Requirements for CoAP End-To-End Security", draft-hartke-core-e2e-security-reqs-01 (work in progress), July 2016.

[I-D.ietf-ace-oauth-authz]
Seitz, L., Selander, G., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "Authentication and Authorization for Constrained Environments (ACE)", draft-ietf-ace-oauth-authz-02 (work in progress), June 2016.

[I-D.ietf-core-coap-tcp-tls]
Bormann, C., Lemay, S., Tschofenig, H., Hartke, K., Silverajan, B., and B. Raymor, "CoAP (Constrained Application Protocol) over TCP, TLS, and WebSockets", draft-ietf-core-coap-tcp-tls-04 (work in progress), August 2016.

[I-D.seitz-ace-oscoap-profile]
Seitz, L., "OSCOAP profile of ACE", draft-seitz-ace-oscoap-profile-00 (work in progress), July 2016.

[I-D.selander-ace-cose-ecdhe]
Selander, G., Mattsson, J., and F. Palombini, "Ephemeral Diffie-Hellman Over COSE (EDHOC)", draft-selander-ace-cose-ecdhe-02 (work in progress), July 2016.

[RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010, <http://www.rfc-editor.org/info/rfc5869>.

[RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, DOI 10.17487/RFC7228, May 2014, <http://www.rfc-editor.org/info/rfc7228>.

Appendix A. Overhead

OSCOAP transforms an unprotected CoAP message to a protected CoAP message, and the protected CoAP message is larger than the unprotected CoAP message. This appendix illustrates the message expansion.

A.1. Length of the Object-Security Option

The protected CoAP message contains the COSE object. The COSE object is included in the payload if the message type of the unprotected CoAP message allows payload or else in the Object-Security option. In the former case the Object-Security option is empty. So the length of the Object-Security option is either zero or the size of the COSE object, depending on whether the CoAP message allows payload or not.

Length of Object-Security option = { 0, size of COSE Object }

A.2. Size of the COSE Object

The size of the COSE object is the sum of the sizes of

- the Header parameters,
- the Cipher Text (excluding the Tag),
- the Tag, and
- data incurred by the COSE format itself (including CBOR encoding).

Let’s analyse the contributions one at a time:

- The header parameters of the COSE object are the Context Identifier (Cid) and the Sequence Number (Seq) (also known as the Transaction Identifier (Tid)) if the message is a request, and Seq only if the message is a response (see Section 5).
  - The size of Cid depends on the number of simultaneous clients, as discussed in Section 3.2
  - The size of Seq is variable, and increases with the number of messages exchanged.
  - As the IV is generated from the padded Sequence Number and a previously agreed upon context IV it is not required to send the whole IV in the message.

- The Cipher Text, excluding the Tag, is the encryption of the payload and the encrypted options Section 4, which are present in the unprotected CoAP message.

- The size of the Tag depends on the Algorithm. For example, for the algorithm AES-CCM-64-64-128, the Tag is 8 bytes.

- The overhead from the COSE format itself depends on the sizes of the previous fields, and is of the order of 10 bytes.

A.3. Message Expansion

The message expansion is not the size of the COSE object. The cipher text in the COSE object is encrypted payload and options of the unprotected CoAP message - the plaintext of which is removed from the protected CoAP message. Since the size of the cipher text is the same as the corresponding plaintext, there is no message expansion due to encryption; payload and options are just represented in a different way in the protected CoAP message:

- The encrypted payload is in the payload of the protected CoAP message

- The encrypted options are in the Object-Security option or within the payload.

Therefore the OSCOAP message expansion is due to Cid (if present), Seq, Tag, and COSE overhead:

```
Message Overhead = Cid + Seq + Tag + COSE Overhead
```

Figure 9: OSCOAP message expansion

A.4. Example

This section gives an example of message expansion in a request with OSCOAP.

In this example we assume an extreme 4-byte Cid, based on the assumption of an ACE deployment with billions of clients requesting access to this particular server. (A typical Cid, will be 1-2 byte as is discussed in Appendix A.2.)

- Cid: 0xa1534e3c

In the example the sequence number is 225, requiring 1 byte to encode. (The size of Seq could be larger depending on how many messages that has been sent as is discussed in Appendix A.2.)

o Seq: 225

The example is based on AES-CCM-64-64-128.

o Tag is 8 bytes

The COSE object is represented in Figure 10 using CBOR's diagnostic notation.

```
[
  h'a20444a1534e3c0641e2', # protected:
                             {04:h'a1534e3c',
                              06:h'e2'}
  {},                      # unprotected: -
  Tag                      # cipher text + 8 byte authentication tag
]
```

Figure 10: Example of message expansion

Note that the encrypted CoAP options and payload are omitted since we target the message expansion (see Appendix A.3). Therefore the size of the COSE Cipher Text equals the size of the Tag, which is 8 bytes.

The COSE object encodes to a total size of 22 bytes, which is the message expansion in this example. The COSE overhead in this example is 22 - (4 + 1 + 8) = 9 bytes, according to the formula in Figure 9. Note that in this example two bytes in the COSE overhead are used to encode the length of Cid and the length of Seq.

Figure 11 summarizes these results.

Tid	Tag	COSE OH	Message OH
5 bytes	8 bytes	9 bytes	22 bytes

Figure 11: Message overhead for a 5-byte Tid and 8-byte Tag.

Appendix B. Examples

This section gives examples of OSCOAP. The message exchanges are made, based on the assumption that there is a security context established between client and server. For simplicity, these examples only indicate the content of the messages without going into detail of the COSE message format.

B.1. Secure Access to Sensor

Here is an example targeting the scenario in the Section 2.2.1. - Forwarding of [I-D.hartke-core-e2e-security-reqs]. The example illustrates a client requesting the alarm status from a server. In the request, CoAP option Uri-Path is encrypted and integrity protected, and the CoAP header fields Code and Version are integrity protected (see Section 4). In the response, the CoAP Payload is encrypted and integrity protected, and the CoAP header fields Code and Version are integrity protected.

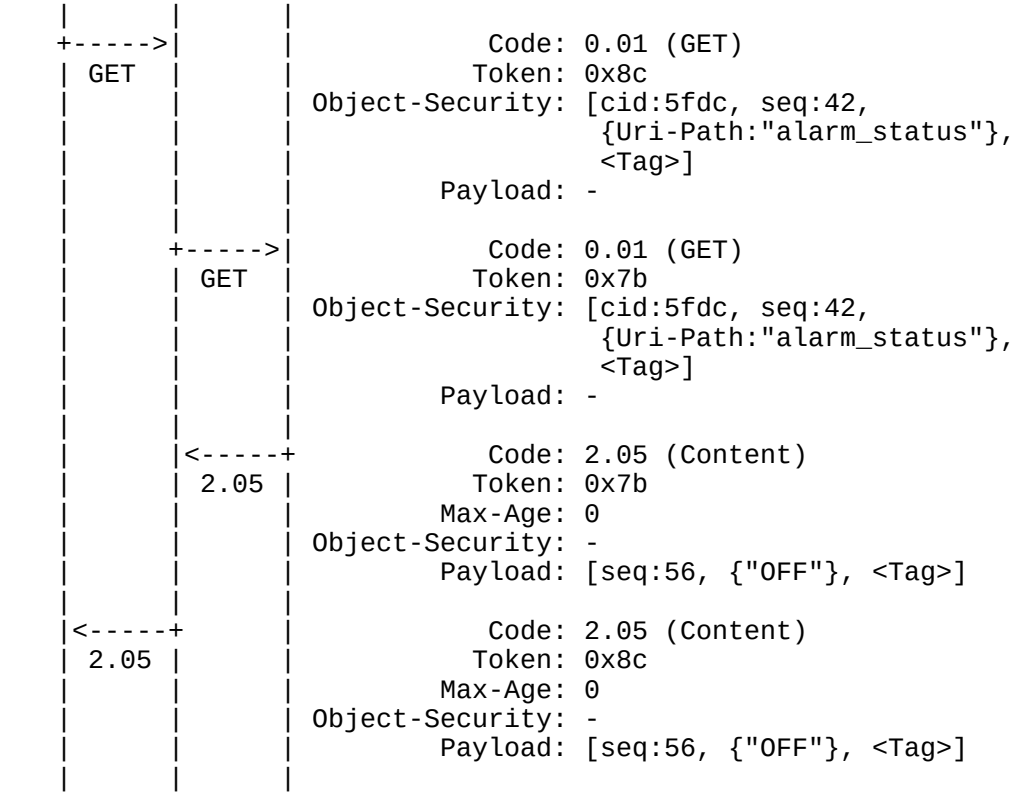


Figure 12: Indication of CoAP GET protected with OSCOAP. The brackets [...] indicate a COSE object. The brackets { ... } indicate encrypted data.

Since the unprotected request message (GET) has no payload, the Object-Security option carries the COSE object as its value. Since the unprotected response message (Content) has payload ("OFF"), the COSE object (indicated with [...]) is carried as the CoAP payload.

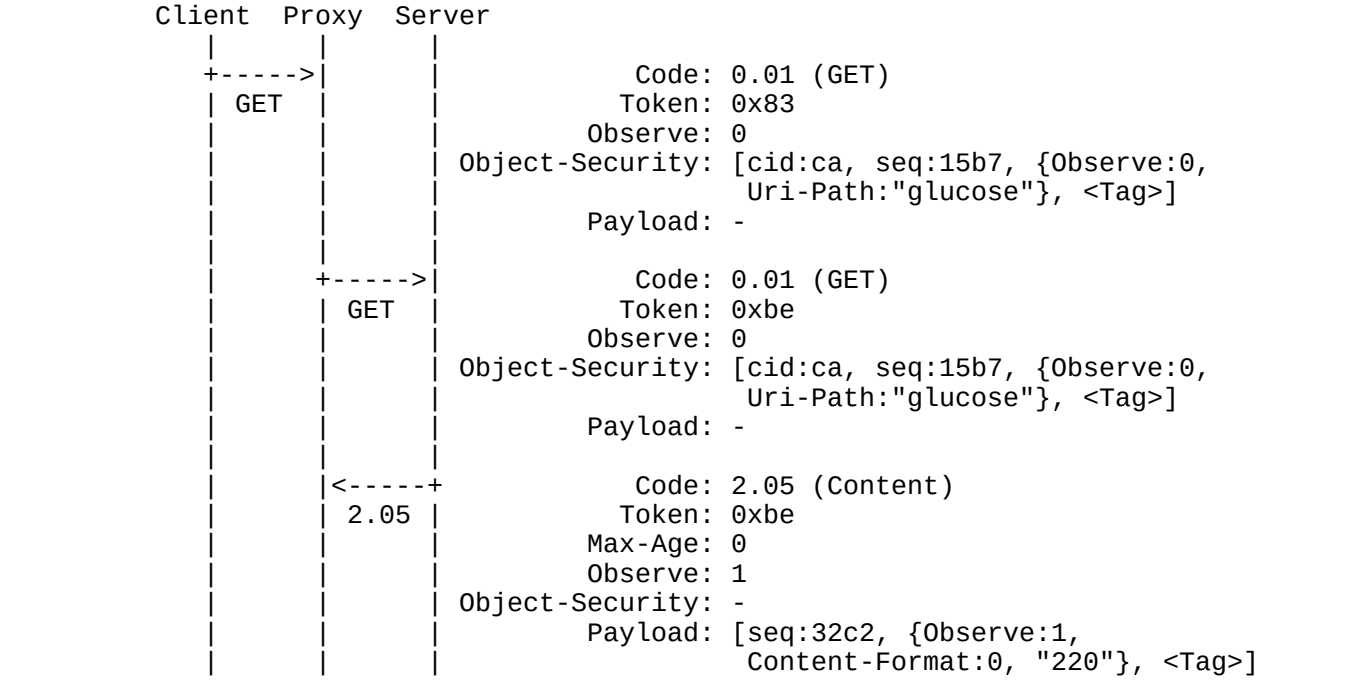
The COSE header of the request contains a Context Identifier (cid:5fdc), indicating which security context was used to protect the message and a Sequence Number (seq:42).

The option Uri-Path (alarm_status) and payload ("OFF") are formatted as indicated in Section 5, and encrypted in the COSE Cipher Text (indicated with { ... }).

The server verifies that the Sequence Number has not been received before (see Section 6.1). The client verifies that the Sequence Number has not been received before and that the response message is generated as a response to the sent request message (see Section 6.1).

B.2. Secure Subscribe to Sensor

Here is an example targeting the scenario in the Forwarding with observe case of [I-D.hartke-core-e2e-security-reqs]. The example illustrates a client requesting subscription to a blood sugar measurement resource (GET /glucose), and first receiving the value 220 mg/dl, and then a second reading with value 180 mg/dl. The CoAP options Observe, Uri-Path, Content-Format, and Payload are encrypted and integrity protected, and the CoAP header field Code is integrity protected (see Section 4).



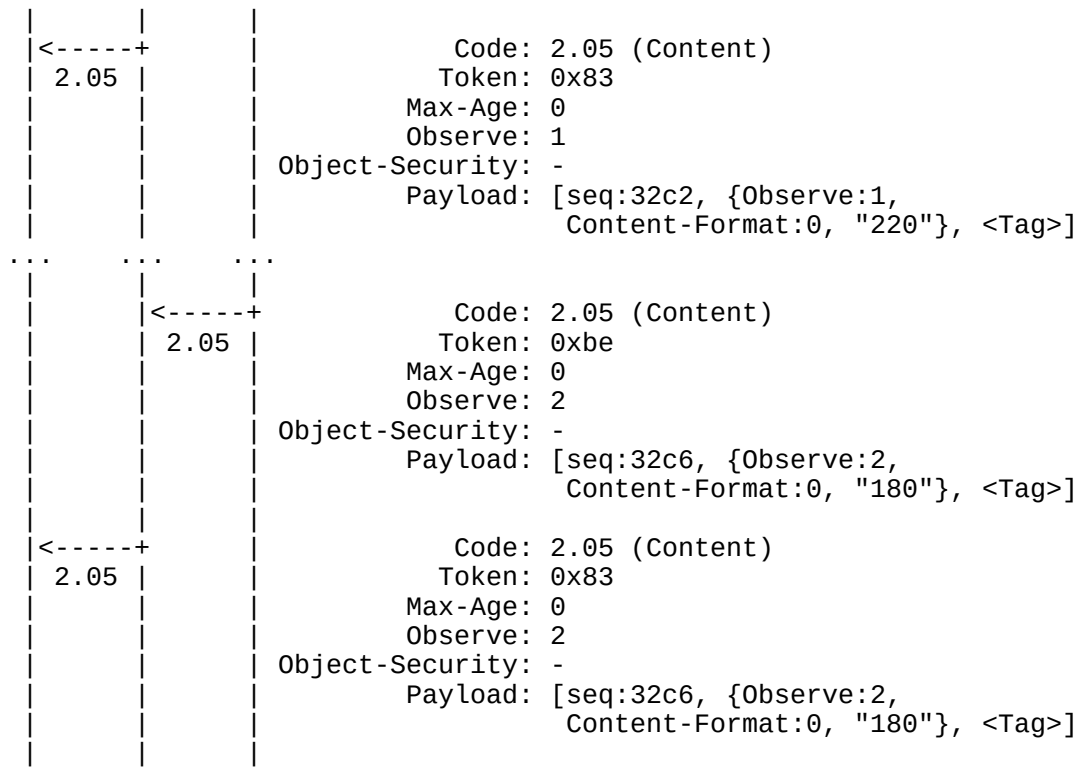


Figure 13: Indication of CoAP GET protected with OSCOAP. The brackets [...] indicates COSE object. The bracket { ... } indicates encrypted data.

Since the unprotected request message (GET) allows no payload, the COSE object (indicated with [...]) is carried in the Object-Security option value. Since the unprotected response message (Content) has payload, the Object-Security option is empty, and the COSE object is carried as the payload.

The COSE header of the request contains a Context Identifier (cid:ca), indicating which security context was used to protect the message and a Sequence Number (seq:15b7).

The options Observe, Content-Format and the payload are formatted as indicated in Section 5, and encrypted in the COSE cipher text (indicated with { ... }).

The server verifies that the Sequence Number has not been received before (see Section 6.1). The client verifies that the Sequence

Number has not been received before and that the response message is generated as a response to the subscribe request.

Appendix C. Object Security of Content (OSCON)

OSCOAP protects message exchanges end-to-end between a certain client and a certain server, targeting the security requirements for forward proxy of [I-D.hartke-core-e2e-security-reqs]. In contrast, many use cases require one and the same message to be protected for, and verified by, multiple endpoints, see caching proxy section of [I-D.hartke-core-e2e-security-reqs]. Those security requirements can be addressed by protecting essentially the payload/content of individual messages using the COSE format ([I-D.ietf-cose-msg]), rather than the entire request/response message exchange. This is referred to as Object Security of Content (OSCON).

OSCON transforms an unprotected CoAP message into a protected CoAP message in the following way: the payload of the unprotected CoAP message is wrapped by a COSE object, which replaces the payload of the unprotected CoAP message. We call the result the "protected" CoAP message.

The unprotected payload shall be the plaintext/payload of the COSE object. The 'protected' field of the COSE object 'Headers' shall include the context identifier, both for requests and responses. If the unprotected CoAP message includes a Content-Format option, then the COSE object shall include a protected 'content type' field, whose value is set to the unprotected message Content-Format value. The Content-Format option of the protected CoAP message shall be replaced with "application/oscon" (Section 9)

The COSE object shall be protected (encrypted) and verified (decrypted) as described in ([I-D.ietf-cose-msg]).

In the case of symmetric encryption, the same key and IV shall not be used twice. Sequence numbers for partial IV as specified for OSCOAP may be used for replay protection as described in Section 6.1. The use of time stamps in the COSE header parameter 'operation time' [I-D.ietf-cose-msg] for freshness may be used.

OSCON shall not be used in cases where CoAP header fields (such as Code or Version) or CoAP options need to be integrity protected or encrypted. OSCON shall not be used in cases which require a secure binding between request and response.

The scenarios in Sections 3.3 - 3.5 of [I-D.hartke-core-e2e-security-reqs] assume multiple recipients for a particular content. In this case the use of symmetric keys does not

provide data origin authentication. Therefore the COSE object should in general be protected with a digital signature.

C.1. Overhead OSCON

In general there are four different kinds of ciphersuites that need to be supported: message authentication code, digital signature, authenticated encryption, and symmetric encryption + digital signature. The use of digital signature is necessary for applications with many legitimate recipients of a given message, and where data origin authentication is required.

To distinguish between these different cases, the tagged structures of COSE are used (see Section 2 of [I-D.ietf-cose-msg]).

The size of the COSE message for selected algorithms are detailed in this section.

The size of the header is shown separately from the size of the MAC/ signature. A 4-byte Context Identifier and a 1-byte Sequence Number are used throughout all examples, with these values:

- Cid: 0xa1534e3c
- Seq: 0xa3

For each scheme, we indicate the fixed length of these two parameters ("Cid+Seq" column) and of the Tag ("MAC"/"SIG"/"TAG"). The "Message OH" column shows the total expansions of the CoAP message size, while the "COSE OH" column is calculated from the previous columns following the formula in Figure 9.

Overhead incurring from CBOR encoding is also included in the COSE overhead count.

To make it easier to read, COSE objects are represented using CBOR’s diagnostic notation rather than a binary dump.

C.2. MAC Only

This example is based on HMAC-SHA256, with truncation to 8 bytes (HMAC 256/64).

Since the key is implicitly known by the recipient, the COSE_Mac0_Tagged structure is used (Section 6.2 of [I-D.ietf-cose-msg]).

The object in COSE encoding gives:

```
996(                          # COSE_Mac0_Tagged
  [
    h'a20444a1534e3c0641a3', # protected:
                                {04:h'a1534e3c',
                                 06:h'a3'}
    {},                       # unprotected
    h'',                      # payload
    MAC                       # truncated 8-byte MAC
  ]
)
```

This COSE object encodes to a total size of 26 bytes.

Figure 14 summarizes these results.

Structure	Tid	MAC	COSE OH	Message OH
COSE_Mac0_Tagged	5 B	8 B	13 B	26 B

Figure 14: Message overhead for a 5-byte Tid using HMAC 256/64

C.3. Signature Only

This example is based on ECDSA, with a signature of 64 bytes.

Since only one signature is used, the COSE_Sign1_Tagged structure is used (Section 4.2 of [I-D.ietf-cose-msg]).

The object in COSE encoding gives:

```
997(                          # COSE_Sign1_Tagged
  [
    h'a20444a1534e3c0641a3', # protected:
                                {04:h'a1534e3c',
                                 06:h'a3'}
    {},                       # unprotected
    h'',                      # payload
    SIG                       # 64-byte signature
  ]
)
```

This COSE object encodes to a total size of 83 bytes.

Figure 15 summarizes these results.

Structure	Tid	SIG	COSE OH	Message OH
COSE_Sign1_Tagged	5 B	64 B	14 B	83 bytes

Figure 15: Message overhead for a 5-byte Tid using 64 byte ECDSA signature.

C.4. Authenticated Encryption with Additional Data (AEAD)

This example is based on AES-CCM with the MAC truncated to 8 bytes.

It is assumed that the IV is generated from the Sequence Number and some previously agreed upon context IV. This means it is not required to explicitly send the whole IV in the message.

Since the key is implicitly known by the recipient, the COSE_Encrypt0_Tagged structure is used (Section 5.2 of [I-D.ietf-cose-msg]).

The object in COSE encoding gives:

```
993(                          # COSE_Encrypt0_Tagged
  [
    h'a20444a1534e3c0641a3', # protected:
                               {04:h'a1534e3c',
                                06:h'a3'}
    {},                       # unprotected
    TAG                       # cipher text + truncated 8-byte TAG
  ]
)
```

This COSE object encodes to a total size of 25 bytes.

Figure 16 summarizes these results.

Structure	Tid	TAG	COSE OH	Message OH
COSE_Encrypt0_Tagged	5 B	8 B	12 B	25 bytes

Figure 16: Message overhead for a 5-byte Tid using AES_128_CCM_8.

C.5. Symmetric Encryption with Asymmetric Signature (SEAS)

This example is based on AES-CCM and ECDSA with 64 bytes signature. The same assumption on the security context as in Appendix C.4. COSE defines the field 'counter signature w/o headers' that is used here to sign a COSE_Encrypt0_Tagged message (see Section 3 of [I-D.ietf-cose-msg]).

The object in COSE encoding gives:

```
  993(                          # COSE_Encrypt0_Tagged
    [
      h'a20444a1534e3c0641a3', # protected:
                                 {04:h'a1534e3c',
                                  06:h'a3'}
      {9:SIG},                 # unprotected:
                                  09: 64 bytes signature
      TAG                      # cipher text + truncated 8-byte TAG
    ]
  )
```

This COSE object encodes to a total size of 92 bytes.

Figure 17 summarizes these results.

Structure	Tid	TAG	SIG	COSE OH	Message OH
COSE_Encrypt0_Tagged	5 B	8 B	64 B	15 B	92 B

Figure 17: Message overhead for a 5-byte Tid using AES-CCM countersigned with ECDSA.

Authors' Addresses

Goeran Selander
Ericsson AB
Farogatan 6
Kista SE-16480 Stockholm
Sweden

Email: goran.selander@ericsson.com

John Mattsson
Ericsson AB
Farogatan 6
Kista SE-16480 Stockholm
Sweden

Email: john.mattsson@ericsson.com

Francesca Palombini
Ericsson AB
Farogatan 6
Kista SE-16480 Stockholm
Sweden

Email: francesca.palombini@ericsson.com

Ludwig Seitz
SICS Swedish ICT
Scheelevagen 17
Lund 22370
Sweden

Email: ludwig@sics.se