Workgroup: ACE Working Group
Internet-Draft:
draft-ietf-ace-pubsub-profile-04
Published: 29 December 2021
Intended Status: Standards Track
Expires: 2 July 2022
Authors: F. Palombini C. Sengul
Ericsson Brunel University

Pub-Sub Profile for Authentication and Authorization for Constrained Environments (ACE)

Abstract

This specification defines an application profile for authentication and authorization for Publishers and Subscribers in a constrained pub-sub scenario, using the ACE framework. This profile relies on transport layer or application layer security to authorize the pub-sub clients to the broker. Moreover, it describes the use of application layer security to protect the content of the pub-sub client message exchange through the broker. The profile covers pub-sub scenarios using either the Constrained Application Protocol (CoAP) [I-D.ietf-core-coap-pubsub] or the Message Queue Telemetry Transport (MQTT) [MQTT-OASIS-Standard-v5] protocol.

Note to Readers

Source for this draft and an issue tracker can be found at https://github.com/ace-wg/pubsub-profile.

Status of This Memo

Table of Contents

1. Introduction

In the publish-subscribe (pub-sub) scenario, devices with limited reachability communicate via a broker, which enables store-and-forward messaging between the devices. This document defines a way

to authorize pub-sub clients using the ACE framework [I-D.ietf-ace-oauth-authz] to obtain the keys for protecting the content of their pub-sub messages when communicating through the broker. The pub-sub communication using the Constrained Application Protocol (CoAP) [RFC7252] is specified in [I-D.ietf-core-coap-pubsub], while the one using MQTT is specified in [MQTT-OASIS-Standard-v5]. This document gives detailed specifications for MQTT and CoAP pub-sub, but can easily be adapted for other transport protocols as well.

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 RFC 2119 [RFC2119].

Readers are expected to be familiar with the terms and concepts described in [I-D.ietf-ace-oauth-authz], [I-D.ietf-ace-key-groupcomm]. In particular, analogously to [I-D.ietf-ace-oauth-authz], terminology for entities in the architecture such as Client (C), Resource Server (RS), and Authorization Server (AS) is defined in OAuth 2.0 [RFC6749] and [I-D.ietf-ace-actors], and terminology for entities such as the Key Distribution Center (KDC) and Dispatcher in [I-D.ietf-ace-key-groupcomm].

Readers are expected to be familiar with terms and concepts of pub-sub group communication, as described in [I-D.ietf-core-coap-pubsub], or MQTT [MQTT-OASIS-Standard-v5].

2. Application Profile Overview

The objective of this document is to specify how to authorize nodes, provide keys, and protect a pub-sub communication, using [I-D.ietf-ace-key-groupcomm], which expands from the ACE framework ([I-D.ietf-ace-oauth-authz]), and transport profiles ([I-D.ietf-ace-dtls-authorize], [I-D.ietf-ace-oscore-profile], [I-D.ietf-ace-mqtt-tls-profile]). The pub-sub communication protocol can be based on CoAP, as described in [I-D.ietf-core-coap-pubsub], MQTT [MQTT-OASIS-Standard-v5] , or other transport. Note that both Publishers and Subscribers use the same pub-sub communication protocol and the same transport profile of ACE in their interaction with the broker. However, all clients need to use CoAP when communicating to the KDC.

The architecture of the scenario is shown in Figure 1.

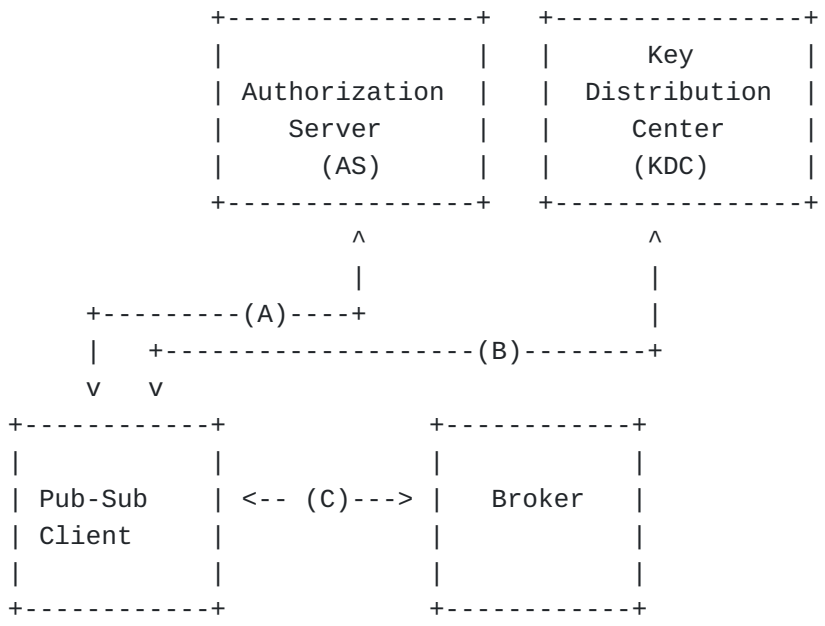

Figure 1: Architecture for Pub-Sub with Authorization Server and Key Distribution Center

Publisher or Subscriber Clients is referred to as Client in short. A Client can act both as a publisher and a subscriber, publishing to some topics, and subscribing to others. However, for the simplicity of presentation, this profile describes Publisher and Subscriber clients separately. The Broker acts as the ACE RS, and also corresponds to the Dispatcher in [I-D.ietf-ace-key-groupcomm]).

This profile specifies:

1. The establishment of a secure connection between a Client and Broker, using an ACE transport profile such as DTLS [I-D.ietf-ace-dtls-authorize], OSCORE [I-D.ietf-ace-oscore-profile], or MQTT-TLS [I-D.ietf-ace-mqtt-tls-profile] (A and C).

2. The Clients retrieval of keying material for the Publisher Client to publish protected publications to the Broker, and for the Subscriber Client to read protected publications (B).

These exchanges aim at setting up two different security associations. On the one hand, the Publisher and the Subscriber clients have a security association with the Broker, so that, as the ACE RS, it can verify that the Clients are authorized (Security Association 1). On the other hand, the Publisher has a security association with the Subscriber, to protect the publication content (Security Association 2) while sending it through the broker. The Security Association 1 is set up using AS and a transport profile of [I-D.ietf-ace-oauth-authz], the Security Association 2 is set up using AS, KDC and [I-D.ietf-ace-key-groupcomm]. Note that, given

that the publication content is protected, the Broker MAY accept unauthorised Subscribers. In this case, the Subscriber client can skip setting up Security Association 1 with the Broker and connect to it as an anonymous client to subscribe to topics of interest at the Broker.

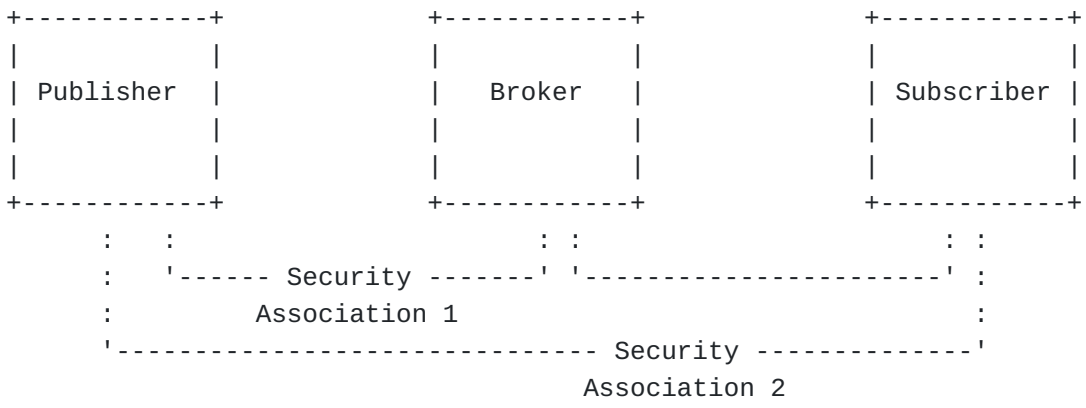

Figure 2: Security Associations between Publisher, Broker, Subscriber pairs.

3. PubSub Authorisation

Since [I-D.ietf-ace-oauth-authz] recommends the use of CoAP and CBOR, this document describes the exchanges assuming CoAP and CBOR are used. However, using HTTP instead of CoAP is possible, using the corresponding parameters and methods. Analogously, JSON [RFC8259] can be used instead of CBOR, using the conversion method specified in Sections 6.1 and 6.2 of [RFC8949]. In case JSON is used, the Content Format or Media Type of the message has to be changed accordingly. Exact definition of these exchanges are considered out of scope for this document.

Figure 3 shows the message flow for authorisation purposes.

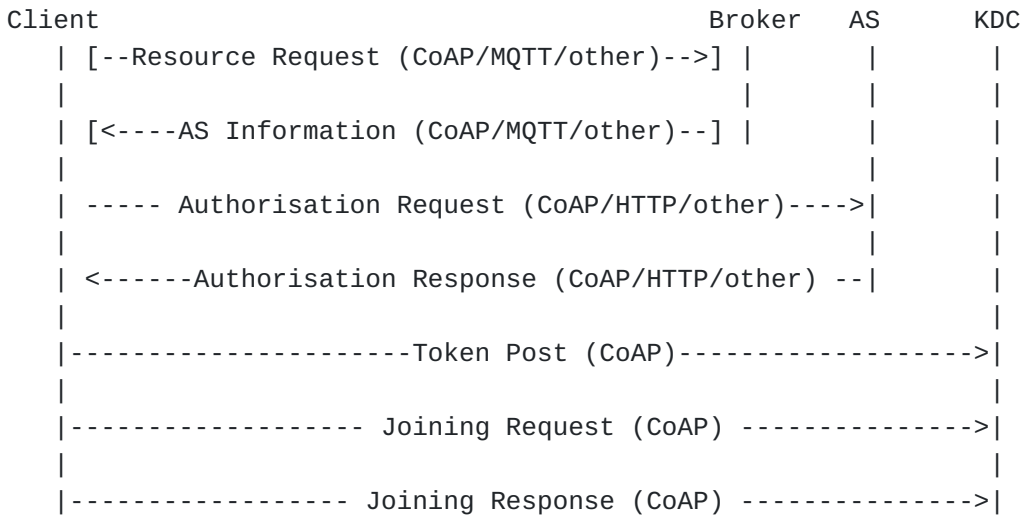

Figure 3: Authorisation Flow

3.1. AS Discovery (Optional)

Complementary to what is defined in [I-D.ietf-ace-oauth-authz] (Section 5.1) for AS discovery, the Broker MAY send the address of the AS to the Client in the 'AS' parameter in the AS Information as a response to an Unauthorized Resource Request (Section 5.2). An example using CBOR diagnostic notation and CoAP is given below:

```
4.01 Unauthorized
Content-Format: application/ace-groupcomm+cbor
{"AS": "coaps://as.example.com/token"}
```

Figure 4: AS Information example

Authorisation Server (AS) Discovery is also defined in Section 2.2.6.1 of [I-D.ietf-ace-mqtt-tls-profile] for MQTT v5 clients (and not supported for MQTT v3 clients).

3.2. Authorising to the KDC and the Broker

After retrieving the AS address, the Client sends two Authorisation Requests to the AS for the KDC and the Broker, respectively.

Note that the AS authorises what topics a Client is allowed to Publish or Subscribe to the Broker, which means authorising which application and security groups a Client can join. This is because being able to publish or subscribe to a topic at the Broker is considered as being part of an application group. As this profile secures the message contents, an application group may be a part of a security group, or can be associated to multiple security groups. Therefore, a Client MUST send Authorization Requests for both.

Both requests include the following fields from the Authorization Request (Section 3.1 of [I-D.ietf-ace-key-groupcomm]):

*'scope', containing the group identifiers, that the Client wishes to access

*'audience', an identifier, corresponding to either the KDC or the Broker. Other additional parameters can be included if necessary, as defined in [I-D.ietf-ace-oauth-authz].

It must be noted that for pub-sub brokers, the scope represents pub-sub topics i.e., the application group. On the other hand, for the KDC, the scope represents the security group. If there is a one-to-one mapping between the application group and the security group, the client uses the same scope for both requests. If there is not a one-to-one mapping, the correct policies regarding both sets of scopes MUST be available to the AS. To be able to join the right security group associated with requested application groups (i.e., pub-sub topics), the client The client MUST ask for the correct scopes in its Authorization Requests. How the client discovers the (application group, security group) association is out of scope of this document. ToDo: Check OSCORE Groups with the CoRE Resource Directory to see if it applies.

The 'scope' parameter is encoded as follows, where 'gname' is treated as topic identifier or filter.

```
gname = tstr

role = tstr

scope_entry = [ gname , ? ( role / [ 2*role ] ) ]

scope = << [ + scope_entry ] >>
```

Figure 5: CDLL definition of scope, using as example group name encoded as tstr and role as tstr.

Other scope representations are also possible and are described in (Section 3.1 of [I-D.ietf-ace-key-groupcomm]). Note that in the AIF-MQTT data model described in Section 3 of the [I-D.ietf-ace-mqtt-tls-profile], the role values have been further constrained to "pub" and "sub".

The AS responds with an Authorization Response to each request as defined in Section 5.8.2 of [I-D.ietf-ace-oauth-authz] and Section 3.2 of [I-D.ietf-ace-key-groupcomm]. The client needs to keep track of which response corresponds to which entity to use the right token for the right audience, i.e., the KDC or the Broker. In case CoAP

PubSub is used as communication protocol, 'profile' claim is set to "coap_pubsub_app" as defined in Section 8.1.1. In case MQTT PubSub is used as communication protocol, 'profile' claim is set to "mqtt_pubsub_app" as defined in Section 8.1.2.

4. Key Distribution for PubSub Content Protection

4.1. Token POST

After receiving a token from the AS, the Client posts the token to the KDC (Section 3.3 [I-D.ietf-ace-key-groupcomm]). In addition to the token post, a Subscriber Client MAY ask for the format of the public keys in the group, used for source authentication, as well as any other group parameters. In this case, the message MUST have Content-Format set to "application/ace+cbor" defined in Section 8.16 of [I-D.ietf-ace-oauth-authz]. The message payload MUST be formatted as a CBOR map, which MUST include the access token and the 'sign_info' parameter. The details for the 'sign_info' parameter can be found in Section 3.3 of [I-D.ietf-ace-key-groupcomm]. Alternatively, the joining node may retrieve this information by other means as described in [I-D.ietf-ace-key-groupcomm].

The KDC verifies the token to check of the Client is authorized to access the topic with the requested role. After successful verification, the Client is authorized to receive the group keying material from the KDC and join the group. The KDC replies to the Client with a 2.01 (Created) response, using Content-Format "application/ace+cbor". The payload of the 2.01 response is a CBOR map.

A Publisher Client MUST send its own public key to the KDC when joining the group. Since the access token from a Publisher Client will have "pub" role, the KDC MUST include 'kdcchallenge' in the CBOR map, specifying a dedicated challenge N_S generated by the KDC. The Client uses this challenge to prove possession of its own private key (see [I-D.ietf-ace-key-groupcomm] for details).

4.2. Join Request and Join Response

In the next step, a joining node MUST have a secure communication association established with the KDC, before starting to join a group under that KDC. Possible ways to provide a secure communication association are described in the DTLS transport profile [I-D.ietf-ace-dtls-authorize] and OSCORE transport profile [I-D.ietf-ace-oscore-profile] of ACE.

After establishing a secure communication, the Client sends a Joining Request to the KDC as described in Section 4.3 of [I-D.ietf-ace-key-groupcomm]. More specifically, the Client sends a POST request to the /ace-group/GROUPNAME endpoint on KDC, with Content-

Format "application/ace-groupcomm+cbor" that MUST contain in the payload (formatted as a CBOR map, Section 4.1.2.1 of [I-D.ietf-ace-key-groupcomm]):

* 'scope' parameter set to the specific group that the Client is attempting to join, i.e., the group name, and the roles it wishes to have in the group. This value corresponds to one scope entry, as defined in Section 3.2.

* 'get_pub_keys' parameter set to the empty array if the Client needs to retrieve the public keys of the other pubsub members,

* 'client_cred' parameter containing the Client's public key formatted according to the encoding of the public keys used in the group, if the Client is a Publisher,

* 'cnonce', encoded as a CBOR byte string, and including a dedicated nonce N_C generated by the Client, if 'client_cred' is present,

* 'client_cred_verify', set to a signature computed over the 'rsnonce' concatenated with cnonce, if 'client_cred' is present,

* OPTIONALLY, if needed, the 'pub_keys_repos' parameter

TODO: Check 'cnonce'

Note that for a Subscriber-only Client, the Joining Request MUST NOT contain the 'client_cred parameter', the role element in the 'scope' parameter MUST be set to "sub". The Subscriber MUST have access to the public keys of all the Publishers; this MAY be achieved in the Joining Request by using the parameter 'get_pub_keys' encoding the CBOR simple value 'null' (0xf6) (as described in Section 4.3.1 of [I-D.ietf-ace-key-groupcomm]) to retrieve the public keys of all the Publishers.

If the 'client_cred' parameter is present, KDC stores the public key of the Client. Note that the alg parameter in the 'client_cred' COSE_Key MUST be a signing algorithm, as defined in [I-D.ietf-cose-rfc8152bis-algs] [I-D.ietf-cose-rfc8152bis-struct], and that it is the same algorithm used to compute the signature sent in 'client_cred_verify'.

The KDC responds with a Joining Response, which has the Content-Format "application/ace-groupcomm+cbor". The payload (formatted as a CBOR map) MUST contain the following fields from the Joining Response (Section 4.3.1 of [I-D.ietf-ace-key-groupcomm]):

* 'gkty' identifies a key type for the 'key' parameter.

* 'key', which contains a "COSE_Key" object (defined in [I-D.ietf-cose-rfc8152bis-algs][I-D.ietf-cose-rfc8152bis-struct], containing:

  - 'kty' with value 4 (symmetric)

  - 'alg' with value defined by the AS (Content Encryption Algorithm)

  - 'Base IV' with value defined by the AS

  - 'k' with value the symmetric key value

  - OPTIONALLY, 'kid' with an identifier for the key value

* OPTIONALLY, 'exp' with the expiration time of the key

* 'pub_keys', containing the public keys of all Publisher Clients, formatted according to the public key encoding for the group, if the 'get_pub_keys' parameter was present and set to the empty array in the Key Distribution Request. For Subscriber Clients, the Joining Response MUST contain the 'pub_keys' parameter. The encoding accepted for this document is UCCS (Unprotectec CWT Claims Set) [I-D.draft-ietf-rats-uccs-01]. ToDo: Consider allowing other public key formats with the following text. If CBOR Web Tokens (CWTs) or CWT Claims Sets (CCSs) [RFC8392] are used as public key format, the public key algorithm is fully described by a COSE key type and its "kty" and "crv" parameters. If X.509 certificates [RFC7925] or C509 certificates [I-D.ietf-cose-cbor-encoded-cert] are used as public key format, the public key algorithm is fully described by the "algorithm" field of the "SubjectPublicKeyInfo" structure, and by the "subjectPublicKeyAlgorithm" element, respectively.

An example of the Joining Request and corresponding Response for a CoAP Publisher using CoAP and CBOR is specified in Figure 6 and Figure 7, where SIG is a signature computed using the private key associated to the public key and the algorithm in 'client_cred'.

```
{
  "scope" : ["Broker1/Temp", "pub"],
  "client_cred" :
    { / COSE_Key /
      / type / 1 : 2, / EC2 /
      / kid / 2 : h'11',
      / alg / 3 : -7, / ECDSA with SHA-256 /
      / crv / -1 : 1 , / P-256 /
      / x / -2 : h'65eda5a12577c2bae829437fe338701a10aaa375e1bb5b5de1
      08de439c08551d',
      / y /-3 : h'1e52ed75701163f7f9e40ddf9f341b3dc9ba860af7e0ca7ca7e
      9eecd0084d19c',
  "cnonce" : h'd36b581d1eef9c7c,
  "client_cred_verify" : SIG
    }
}
```

Figure 6: Joining Request payload for a Publisher

```
{
  "gkty" : "COSE_Key",
  "key" : {1: 4, 2: h'1234', 3: 12, 5: h'1f389d14d17dc7',
          -1: h'02e2cc3a9b92855220f255fff1c615bc'}
}
```

Figure 7: Joining Response payload for a Publisher

An example of the payload of a Joining Request and corresponding Response for a Subscriber using CoAP and CBOR is specified in Figure 8 and Figure 9.

```
{
  "scope" : ["Broker1/Temp", "sub"],
  "get_pub_keys" : null
}
```

Figure 8: Joining Request payload for a Subscriber

```
{
  "scope" : ["Broker1/Temp", "sub"],
  "gkty" : "COSE_Key"
  "key" : {1: 4, 2: h'1234', 3: 12, 5: h'1f389d14d17dc7',
          -1: h'02e2cc3a9b92855220f255fff1c615bc'},
  "pub_keys" : [
   {/UCCS/
      2:  "42-50-31-FF-EF-37-32-39", /sub/
      8: {/cnf/
      1: {/COSE_Key/
      1 : 1, /alg/
      3 : -8 /kty/
      -1 : 6 , /crv/
      -2 : h'C6EC665E817BD064340E7C24BB93A11E /x/
8EC0735CE48790F9C458F7FA340B8CA3', / x /
    }
    }
   }
  ]
}
```

Figure 9: Joining Response payload for a Subscriber

ToDO: Fix Example for COSE_Key for public key

5. PubSub Protected Communication

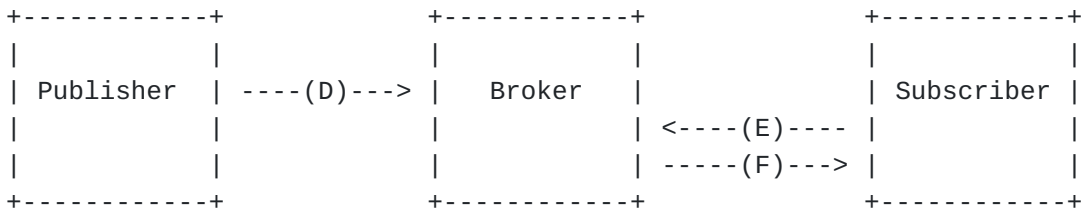

Figure 10: Secure communication between Publisher and Subscriber

(D) corresponds to the publication of a topic on the Broker. The publication (the resource representation) is protected with COSE ([I-D.ietf-cose-rfc8152bis-algs] [I-D.ietf-cose-rfc8152bis-struct]) by the Publisher. The (E) message is the subscription of the Subscriber. The subscription MAY be unprotected. The (F) message is the response from the Broker, where the publication is protected with COSE by the Publisher.

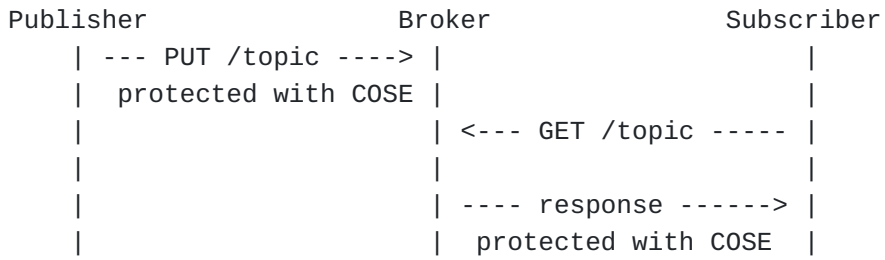

Figure 11: (E), (F), (G): Example of protected communication for CoAP

The flow graph is presented below for CoAP. The message flow is similar for MQTT, where PUT corresponds to a PUBLISH message, and GET corresponds to a SUBSCRIBE message. Whenever a Client publishes a new message, the Broker sends this message to all valid subscribers.

5.1. Using COSE Objects To Protect The Resource Representation

The Publisher uses the symmetric COSE Key received from the KDC (Section 4) to protect the payload of the PUBLISH operation (Section 4.3 of [I-D.ietf-core-coap-pubsub] and [MQTT-OASIS-Standard-v5]). Specifically, the COSE Key is used to create a COSE_Encrypt0 object with algorithm specified by KDC. The Publisher uses the private key corresponding to the public key sent to the KDC in exchange B (Section 4) to countersign the COSE Object as specified in [I-D.ietf-cose-rfc8152bis-algs] [I-D.ietf-cose-rfc8152bis-struct]. The payload is replaced by the COSE object before the publication is sent to the Broker.

The Subscriber uses the 'kid' in the 'countersignature' field in the COSE object to retrieve the right public key to verify the countersignature. It then uses the symmetric key received from KDC to verify and decrypt the publication received in the payload from the Broker (in the case of CoAP the publication is received by the CoAP Notification and for MQTT, it is received as a PUBLISH message from the Broker to the subscribing client).

The COSE object is constructed in the following way:

* The protected Headers (as described in [I-D.ietf-cose-rfc8152bis-algs] [I-D.ietf-cose-rfc8152bis-struct]) MUST contain the kid parameter if it was provided in the Joining Response, with value the kid of the symmetric COSE Key received in Section 4 and MUST contain the content encryption algorithm.

*The unprotected Headers MUST contain the Partial IV, with value a sequence number that is incremented for every message sent, and the counter signature that includes:

  -the algorithm (same value as in the asymmetric COSE Key received in (B)) in the protected header;

  -the kid (same value as the kid of the asymmetric COSE Key received in (B)) in the unprotected header;

  -the signature computed as specified in [I-D.ietf-cose-rfc8152bis-algs] [I-D.ietf-cose-rfc8152bis-struct].

*The ciphertext, computed over the plaintext that MUST contain the message payload.

The 'external_aad' is an empty string.

An example is given in Figure 12:

```
16(
  [
    / protected / h'a2010c04421234' / {
        \ alg \ 1:12, \ AES-CCM-64-64-128 \
        \ kid \ 4: h'1234'
      } / ,
    / unprotected / {
      / iv / 5:h'89f52f65a1c580',
      / countersign / 7:[
        / protected / h'a10126' / {
          \ alg \ 1:-7
        } / ,
        / unprotected / {
          / kid / 4:h'11'
        },
        / signature / SIG / 64 bytes signature /
      ]
    },
    / ciphertext / h'8df0a3b62fccff37aa313c8020e971f8aC8d'
  ]
)
```

Figure 12: Example of COSE Object sent in the payload of a PUBLISH operation

The encryption and decryption operations are described in [I-D.ietf-cose-rfc8152bis-algs] [I-D.ietf-cose-rfc8152bis-struct].

6. Profile-specific Considerations

This section summarises the CoAP and MQTT specific pub-sub communications, and considerations respectively.

6.1. CoAP PubSub Application Profile

A CoAP Pub-Sub Client and Broker use an ACE transport profile such as DTLS [I-D.ietf-ace-dtls-authorize], OSCORE [I-D.ietf-ace-oscore-profile].

As shown in Figure 1, (A) is an Access Token Request and Response exchange between Publisher and Authorization Server to retrieve the Access Token and RS (Broker) Information. As specified, the Client has the role of a CoAP client, the Broker has the role of the CoAP server.

(B) corresponds to the retrieval of the keying material to protect the publication end-to-end (see Section 5.1), and uses [I-D.ietf-ace-key-groupcomm]. The details are defined in Section 4.

(C) corresponds to the exchange between the Client and the Broker, where the Client sends its access token to the Broker and establishes a secure connection with the Broker. Depending on the Information received in (A), this can be for example DTLS handshake, or other protocols. Depending on the application, there may not be the need for this set up phase: for example, if OSCORE is used directly. Note that, in line with what defined in the ACE transport profile used, the access token includes the scope (i.e. pubsub topics on the Broker) the Publisher is allowed to publish to. For implementation simplicity, it is RECOMMENDED that the ACE transport profile used.

After the previous phases have taken place, the pub-sub communication can commence. The operations of publishing and subscribing are defined in [I-D.ietf-core-coap-pubsub].

6.2. MQTT PubSub Application Profile

The steps MQTT clients go through are similar to the CoAP clients as described in Section 6.1. The payload that is carried in MQTT messages will be protected using COSE.

In MQTT, topics are organised as a tree, and in the [I-D.ietf-ace-mqtt-tls-profile] 'scope' captures permissions for not a single topic but a topic filter. Therefore, topic names (i.e., group names) may include wildcards spanning several levels of the topic tree. Hence, it is important to distinguish application groups and security groups defined in [I-D.ietf-core-groupcomm-bis]. An application group has relevance at the application level - for

example, in MQTT an application group could denote all topics stored under ""home/lights/". On the other hand, a security group is a group of endpoints that each store group security material to exchange secure communication within the group. The group communication in [I-D.ietf-ace-key-groupcomm] refers to security groups. ToDo: Give a more complete example

For an MQTT client we envision the following steps to take place:

1. Client sends a token request to AS for the requested topics (application groups) using the broker as the audience.

2. Client sends a token request to AS for the corresponding security groups for its application groups using the KDC as the audience.

3. Client sends join requests to KDC to gets the keys for these security groups.

4. Client authorises to the Broker with the token (described in [I-D.ietf-ace-mqtt-tls-profile]).

5. A Publisher Client sends PUBLISH messages for a given topic and protects the payload with the corresponding key for the associated security group. RS validates the PUBLISH message by checking the topic stored token.

6. A Subscriber Client may send SUBSCRIBE messages with one or multiple topic filters. A topic filter may correspond to multiple topics. RS validates the SUBSCRIBE message by checking the stored token for the Client.

7. Security Considerations

In the profile described above, the Publisher and Subscriber use asymmetric crypto, which would make the message exchange quite heavy for small constrained devices. Moreover, all Subscribers must be able to access the public keys of all the Publishers to a specific topic to be able to verify the publications. Such a database could be set up and managed by the same entity having control of the key material for that topic, i.e. KDC.

An application where it is not critical that only authorized Publishers can publish on a topic may decide not to make use of the asymmetric crypto and only use symmetric encryption/MAC to confidentiality and integrity protection of the publication. However, this is not recommended since, as a result, any authorized Subscribers with access to the Broker may forge unauthorized publications without being detected. In this symmetric case the Subscribers would only need one symmetric key per topic, and would

not need to know any information about the Publishers, that can be anonymous to it and the Broker.

Subscribers can be excluded from future publications through re-keying for a certain topic. This could be set up to happen on a regular basis, for certain applications. How this could be done is out of scope for this work.

The Broker is only trusted with verifying that the Publisher is authorized to publish, but is not trusted with the publications itself, which it cannot read nor modify. In this setting, caching of publications on the Broker is still allowed.

TODO: expand on security and privacy considerations

8. IANA Considerations

8.1. ACE Groupcomm Profile Registry

The following registrations are done for the "ACE Groupcomm Profile" Registry following the procedure specified in [I-D.ietf-ace-key-groupcomm].

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

8.1.1. CoAP Profile Registration

Name: coap_pubsub_app

Description: Profile for delegating client authentication and authorization for publishers and subscribers in a CoAP pub-sub setting scenario in a constrained environment.

CBOR Key: TBD

Reference: [[This document]]

8.1.2. MQTT Profile Registration

Name: mqtt_pubsub_app

Description: Profile for delegating client authentication and authorization for publishers and subscribers in a MQTT pub-sub setting scenario in a constrained environment.

CBOR Key: TBD

Reference: [[This document]]

8.2. ACE Groupcomm Key Registry

The following registrations are done for the ACE Groupcomm Key Registry following the procedure specified in [I-D.ietf-ace-key-groupcomm].

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

Name: COSE_Key

Key Type Value: TBD

Profile: coap_pubsub_app, mqtt_pubsub_app

Description: COSE_Key object

References: [I-D.ietf-cose-rfc8152bis-algs] [I-D.ietf-cose-rfc8152bis-struct], [[This document]]

9. References

9.1. Normative References

[I-D.draft-ietf-rats-uccs-01] Birkholz, H., O'Donoghue, J., Cam-Winget, N., and C. Bormann, "A CBOR Tag for Unprotected CWT Claims Sets", Work in Progress, Internet-Draft, draft-ietf-rats-uccs-01, 12 July 2021, <https://www.ietf.org/archive/id/draft-ietf-rats-uccs-01.txt>.

[I-D.ietf-ace-key-groupcomm] Palombini, F. and M. Tiloca, "Key Provisioning for Group Communication using ACE", Work in Progress, Internet-Draft, draft-ietf-ace-key-groupcomm-15, 23 December 2021, <https://www.ietf.org/archive/id/draft-ietf-ace-key-groupcomm-15.txt>.

[I-D.ietf-ace-oauth-authz] Seitz, L., Selander, G., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "Authentication and Authorization for Constrained Environments (ACE) using the OAuth 2.0 Framework (ACE-OAuth)", Work in Progress, Internet-Draft, draft-ietf-ace-oauth-authz-46, 8 November 2021, <https://www.ietf.org/archive/id/draft-ietf-ace-oauth-authz-46.txt>.

[I-D.ietf-core-coap-pubsub] Koster, M., Keranen, A., and J. Jimenez, "Publish-Subscribe Broker for the Constrained Application Protocol (CoAP)", Work in Progress, Internet-Draft, draft-ietf-core-coap-pubsub-09, 30 September 2019,

<https://www.ietf.org/archive/id/draft-ietf-core-coap-pubsub-09.txt>.

[I-D.ietf-core-groupcomm-bis] Dijk, E., Wang, C., and M. Tiloca, "Group Communication for the Constrained Application Protocol (CoAP)", Work in Progress, Internet-Draft, draft-ietf-core-groupcomm-bis-05, 25 October 2021, <https://www.ietf.org/archive/id/draft-ietf-core-groupcomm-bis-05.txt>.

[I-D.ietf-cose-cbor-encoded-cert]
Mattsson, J. P., Selander, G., Raza, S., Höglund, J., and M. Furuhed, "CBOR Encoded X.509 Certificates (C509 Certificates)", Work in Progress, Internet-Draft, draft-ietf-cose-cbor-encoded-cert-02, 12 July 2021, <https://www.ietf.org/archive/id/draft-ietf-cose-cbor-encoded-cert-02.txt>.

[I-D.ietf-cose-rfc8152bis-algs]
Schaad, J., "CBOR Object Signing and Encryption (COSE): Initial Algorithms", Work in Progress, Internet-Draft, draft-ietf-cose-rfc8152bis-algs-12, 24 September 2020, <https://www.ietf.org/archive/id/draft-ietf-cose-rfc8152bis-algs-12.txt>.

[I-D.ietf-cose-rfc8152bis-struct]
Schaad, J., "CBOR Object Signing and Encryption (COSE): Structures and Process", Work in Progress, Internet-Draft, draft-ietf-cose-rfc8152bis-struct-15, 1 February 2021, <https://www.ietf.org/archive/id/draft-ietf-cose-rfc8152bis-struct-15.txt>.

[MQTT-OASIS-Standard-v5] Banks, A., Briggs, E., Borgendale, K., and R. Gupta, "OASIS Standard MQTT Version 5.0", 2017, <http://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html>.

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

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <https://www.rfc-editor.org/info/rfc6749>.

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

[RFC7925] Tschofenig, H., Ed. and T. Fossati, "Transport Layer Security (TLS) / Datagram Transport Layer Security (DTLS) Profiles for the Internet of Things", RFC 7925, DOI 10.17487/RFC7925, July 2016, <https://www.rfc-editor.org/info/rfc7925>.

[RFC8392] Jones, M., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "CBOR Web Token (CWT)", RFC 8392, DOI 10.17487/RFC8392, May 2018, <https://www.rfc-editor.org/info/rfc8392>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/info/rfc8949>.

9.2. Informative References

[I-D.ietf-ace-actors] Gerdes, S., Seitz, L., Selander, G., and C. Bormann, "An architecture for authorization in constrained environments", Work in Progress, Internet-Draft, draft-ietf-ace-actors-07, 22 October 2018, <https://www.ietf.org/archive/id/draft-ietf-ace-actors-07.txt>.

[I-D.ietf-ace-dtls-authorize] Gerdes, S., Bergmann, O., Bormann, C., Selander, G., and L. Seitz, "Datagram Transport Layer Security (DTLS) Profile for Authentication and Authorization for Constrained Environments (ACE)", Work in Progress, Internet-Draft, draft-ietf-ace-dtls-authorize-18, 4 June 2021, <https://www.ietf.org/archive/id/draft-ietf-ace-dtls-authorize-18.txt>.

[I-D.ietf-ace-mqtt-tls-profile] Sengul, C. and A. Kirby, "Message Queuing Telemetry Transport (MQTT)-TLS profile of Authentication and Authorization for Constrained Environments (ACE) Framework", Work in Progress, Internet-Draft, draft-ietf-ace-mqtt-tls-profile-13, 23 October 2021, <https://www.ietf.org/archive/id/draft-ietf-ace-mqtt-tls-profile-13.txt>.

[I-D.ietf-ace-oscore-profile] Palombini, F., Seitz, L., Selander, G., and M. Gunnarsson, "OSCORE Profile of the Authentication and Authorization for Constrained Environments Framework", Work in Progress, Internet-Draft, draft-ietf-ace-oscore-profile-19, 6 May 2021, <https://www.ietf.org/archive/id/draft-ietf-ace-oscore-profile-19.txt>.

[RFC8259]
Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/info/rfc8259>.

Appendix A. Requirements on Application Profiles

This section lists the specifications on this profile based on the requirements defined in Appendix A of [I-D.ietf-ace-key-groupcomm]

*REQ1: Specify the encoding and value of the identifier of group or topic of 'scope': see Section 4).

*REQ2: Specify the encoding and value of roles of 'scope': see Section 4).

*REQ3: Optionally, specify the acceptable values for 'sign_alg': TODO

*REQ4: Optionally, specify the acceptable values for 'sign_parameters': TODO

*REQ5: Optionally, specify the acceptable values for 'sign_key_parameters': TODO

*REQ6: Optionally, specify the acceptable values for 'pub_key_enc': TODO

*REQ7: Specify the exact format of the 'key' value: COSE_Key, see Section 4.

*REQ8: Specify the acceptable values of 'kty' : "COSE_Key", see Section 4.

*REQ9: Specify the format of the identifiers of group members: TODO

*REQ10: Optionally, specify the format and content of 'group_policies' entries: not defined

*REQ11: Specify the communication protocol the members of the group must use: CoAP pub/sub.

*REQ12: Specify the security protocol the group members must use to protect their communication. This must provide encryption, integrity and replay protection: Object Security of Content using COSE, see Section 5.1.

*REQ13: Specify and register the application profile identifier : "coap_pubsub_app", see Section 8.1.

*REQ14: Optionally, specify the encoding of public keys, of 'client_cred', and of 'pub_keys' if COSE_Keys are not used: NA.

*REQ15: Specify policies at the KDC to handle id that are not included in get_pub_keys: TODO

*REQ16: Specify the format and content of 'group_policies': TODO

*REQ17: Specify the format of newly-generated individual keying material for group members, or of the information to derive it, and corresponding CBOR label : not defined

*REQ18: Specify how the communication is secured between Client and KDC. Optionally, specify transport profile of ACE [I-D.ietf-ace-oauth-authz] to use between Client and KDC: pre-set, as KDC is AS.

*OPT1: Optionally, specify the encoding of public keys, of 'client_cred', and of 'pub_keys' if COSE_Keys are not used: NA

*OPT2: Optionally, specify the negotiation of parameter values for signature algorithm and signature keys, if 'sign_info' and 'pub_key_enc' are not used: NA

*OPT3: Optionally, specify the format and content of 'mgt_key_material': not defined

*OPT4: Optionally, specify policies that instruct clients to retain unsuccessfully decrypted messages and for how long, so that they can be decrypted after getting updated keying material: not defined

Acknowledgments

The author wishes to thank Ari Keraenen, John Mattsson, Ludwig Seitz, Goeran Selander, Jim Schaad and Marco Tiloca for the useful discussion and reviews that helped shape this document.

Authors' Addresses

Francesca Palombini
Ericsson

Email: francesca.palombini@ericsson.com

Cigdem Sengul
Brunel University

Email: csengul@acm.org