ACE Working Group   C. Sengul
Internet-Draft   Nominet
Intended status: Standards Track   A. Kirby
Expires: June 22, 2020   Oxbotica
                                              P. Fremantle
                                     University of Portsmouth
                                             December 20, 2019

MQTT-TLS profile of ACE
draft-ietf-ace-mqtt-tls-profile-03

Abstract

   This document specifies a profile for the ACE (Authentication and
   Authorization for Constrained Environments) framework to enable
   authorization in an MQTT-based publish-subscribe messaging system.
   Proof-of-possession keys, bound to OAuth2.0 access tokens, are used
   to authenticate and authorize MQTT Clients. The protocol relies on
   TLS for confidentiality and server authentication.

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 https://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 June 22, 2020.

Table of Contents

1. Introduction

This document specifies a profile for the ACE framework
[I-D.ietf-ace-oauth-authz]. In this profile, Clients and a Broker
use MQTT to exchange Application Messages. The protocol relies on
TLS for communication security between entities. The MQTT protocol
interactions are described based on the MQTT v5.0 - the OASIS
Standard [MQTT-OASIS-Standard-v5]. It is expected that MQTT
deployments will retain backward compatibility for MQTT v3.1.1
clients, and therefore, this document also describes a reduced set of
protocol interactions for MQTT v3.1.1 - the OASIS Standard
[MQTT-OASIS-Standard]. However, MQTT v5.0 is the RECOMMENDED version
as it works more naturally with ACE-style authentication and
authorization.

MQTT is a publish-subscribe protocol and after connecting to the MQTT
Broker, a Client can publish and subscribe to multiple topics. The
MQTT Broker is responsible for distributing messages published by the
publishers to their subscribers. Messages are published under a
Topic Name, and subscribers must subscribe to the Topic Names to
receive the corresponding messages. The Broker uses the Topic Name
in a published message to determine which subscribers to relay the
messages.

In this document, topics, more specifically, Topic Names, are treated
as resources. The Clients are assumed to have identified the
publish/subscribe topics of interest out-of-band (topic discovery is
not a feature of the MQTT protocol). A resource owner can pre-
configure policies at the AS that give Clients publish or subscribe
permissions to different topics.

Clients use an access token, bound to a proof-of-possession (PoP) key
to authorize with the MQTT Broker their connection and publish/
subscribe permissions to topics. In the context of this ACE profile,
the Broker acts as the Resource Server (RS). In the rest of the
document the terms "RS" and "Broker" are used interchangeably. This
document describes how to authorise the following exchanges between
the Clients and the Broker.

o Connection requests from the Clients to the Broker

o Publish requests from the Clients to the Broker, and from the
   Broker to the Clients

o Subscribe requests from Clients to the Broker

This document does not protect the contents of the PUBLISH message
from the Broker, and hence, the content of the the PUBLISH message is

not signed or encrypted separately for the subscribers. This
functionality may be implemented using the proposal outlined in the
CoAP Pub-Sub Profile [I-D.palombini-ace-coap-pubsub-profile].

To provide communication confidentiality and RS authentication, TLS
is used and TLS 1.3 is RECOMMENDED. This document makes the same
assumptions as the Section 4 of the ACE framework
[I-D.ietf-ace-oauth-authz] regarding Client and RS registration with
the AS and setting up keying material. While the Client-Broker
exchanges are only over MQTT, the required Client-AS and RS-AS
interactions are described for HTTPS-based communication, using
'application/ace+json' content type, and unless otherwise specified,
using JSON encoding. The token may be a reference, or JSON Web Token
(JWT). For JWT tokens, this document follows RFC 7800 [RFC7800] for
PoP semantics for JWTs. The Client-AS and RS-AS may also be other
than HTTPS e.g., CoAP or MQTT. Implementations MAY also use
'application/ace+cbor' content type, and CBOR encoding, and CBOR Web
Token (CWT) and associated PoP semantics to reduce the protocol
memory and bandwidth requirements. For more information on Proof of
Possession semantics for CWTs, see Proof-of-Possession Key Semantics
for CBOR Web Tokens (CWTs) [I-D.ietf-ace-cwt-proof-of-possession].

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
"SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and
"OPTIONAL" in this document are to be interpreted as described in BCP
14 [RFC2119] [RFC8174], when, and only when, they appear in all
capitals, as shown here.

1.2. ACE-Related Terminology

The terminology for entities in the architecture is defined in OAuth
2.0 RFC 6749 [RFC6749] such as "Client" (C), "Resource Server" (RS)
and "Authorization Server" (AS).

The term "Resource" is used to refer to an MQTT Topic Name, which is
defined in Section 1.3. Hence, the "Resource Owner" is any entity
that can authoritatively speak for the topic.

Certain security-related terms such as "authentication",
"authorization", "confidentiality", "(data) integrity", "message
authentication code", and "verify" are taken from RFC 4949 [RFC4949].

1.3. MQTT-Related Terminology

   The document describes message exchanges as MQTT protocol
   interactions. The Clients are MQTT Clients, which connect to the
   Broker to publish and subscribe to Application Messages, labeled with
   their topics. For additional information, please refer to the MQTT
   v5.0 - the OASIS Standard [MQTT-OASIS-Standard-v5] or the MQTT v3.1.1
   - the OASIS Standard [MQTT-OASIS-Standard].

   MQTTS
           Secured transport profile of MQTT. MQTTS runs over TLS.

   Broker
           The Server in MQTT. It acts as an intermediary between the
           Clients that publishes Application Messages, and the Clients
           that made Subscriptions. The Broker acts as the Resource
           Server for the Clients.

   Application Message
           The data carried by the MQTT protocol. The data has an
           associated QoS level and a Topic Name.

   QoS level
           The level of assurance for the delivery of an Application
           Message. The QoS level can be 0-2, where "0" indicates "At
           most once delivery", "1" "At least once delivery", and "2"
           "Exactly once delivery".

   Topic Name
           The label attached to an Application Message, which is
           matched to a Subscription.

   Subscription
           A subscription comprises a Topic Filter and a maximum Quality
           of Service (QoS).

   Topic Filter
           An expression that indicates interest in one or more Topic
           Names. Topic Filters may include wildcards.

   MQTT sends various control messages across a network connection. The
   following is not an exhaustive list and the control packets that are
   not relevant for authorization are not explained. These include, for
   instance, the PUBREL and PUBCOMP packets used in the 4-step handshake
   required for the QoS level 2.

   CONNECT

Client request to connect to the Broker. After a network
connection is established, this is the first packet sent by a
Client.

CONNACK
        The Broker connection acknowledgment. The first packet sent
        from the Broker to a Client is a CONNACK packet. CONNACK
        packets contain return codes indicating either a success or
        an error state to a Client.

AUTH

        Authentication Exchange. An AUTH packet is sent from the
        Client to the Broker or to the Broker to the Client as part
        of an extended authentication exchange. AUTH Properties
        include Authentication Method and Authentication Data. The
        Authentication Method is set in the CONNECT packet, and
        consequent AUTH packets follow the same Authentication
        Method. The contents of the Authentication Data are defined
        by the Authentication Method.

PUBLISH
        Publish request sent from a publishing Client to the Broker,
        or from the Broker to a subscribing Client.

PUBACK
        Response to PUBLISH request with QoS level 1. PUBACK can be
        sent from the Broker to a Client or a Client to the Broker.

PUBREC
        Response to PUBLISH request with QoS level 2. PUBREC can be
        sent from the Broker to a Client or a Client to the Broker.

SUBSCRIBE
        The Client subscribe request.

SUBACK
        Subscribe acknowledgment.

PINGREQ
        A ping request sent from a Client to the Broker. It signals
        to the Broker that the Client is alive, and is used to
        confirm that the Broker is also alive. The "Keep Alive"
        period is set in the CONNECT message.

PINGRESP
        Response sent by the Broker to the Client in response to
        PINGREQ. It indicates the Broker is alive.

Will

>   If the network connection is not closed normally, the Server
>   sends a last Will message for the Client, if the Client
>   provided one in its CONNECT message. If the Will Flag is
>   set, then the payload of the CONNECT message includes
>   information about the Will. The information consists of the
>   Will Properties, Will Topic, and Will Payload fields.

2. Authorizing Connection Requests

This section specifies how Client connections are authorized by the
MQTT Broker.Figure 1 shows the basic protocol flow during connection
set-up.The token request and response use the /token endpoint at the
AS, specified in the Section 5.6 of the ACE framework
[I-D.ietf-ace-oauth-authz]. Steps (D) and (E) are optional, and use
the introspection endpoint, specified in the Section 5.7 of the ACE
framework. The Client and Broker use HTTPS to communicate to AS via
these endpoints. The Client and Broker use only MQTT to communicate
between them.

If the Client is resource-constrained, a Client Authorisation Server
may carry out the token request on behalf of the Client, and later,
onboard the Client with the token. Also, the C-AS and Broker-AS
interfaces may be implemented using protocols other than HTTPS, e.g.,
CoAP or MQTT. The interactions between a Client and its Client
Authorization Server for token onboarding, and the MQTTS support for
token requests are out of scope of this document.

```
                        +--------------------+
                        | Client             |
                        |                    |
      +---(A) Token request--| Client -       |
      |                   | Authorization    |
      |    +-(B) Access token-> Server Interface |
      |    |              |        (HTTPS)    |
      |    |              |_____|
      |    |              |                    |
   +--v------------+      |  Pub/Sub Interface |
   | Authorization |      |      (MQTTS)       |
   | Server        |      +-----------^--------+
   |_____|           |       |
      |     ^         (C)Connection  (F)Connection
      |     |            request +     response
      |     |            access token  |
      |     |                 |         |
      |     |              +---v--------------+
      |     |              |  Broker (MQTTS) |
      |     |              |_____|
      |    +(D)Introspection-|                 |
      |    request (optional) | RS-AS interface |
      |                   |       (HTTPS)     |
      +-(E)Introspection---->|_____|
        response (optional)
```
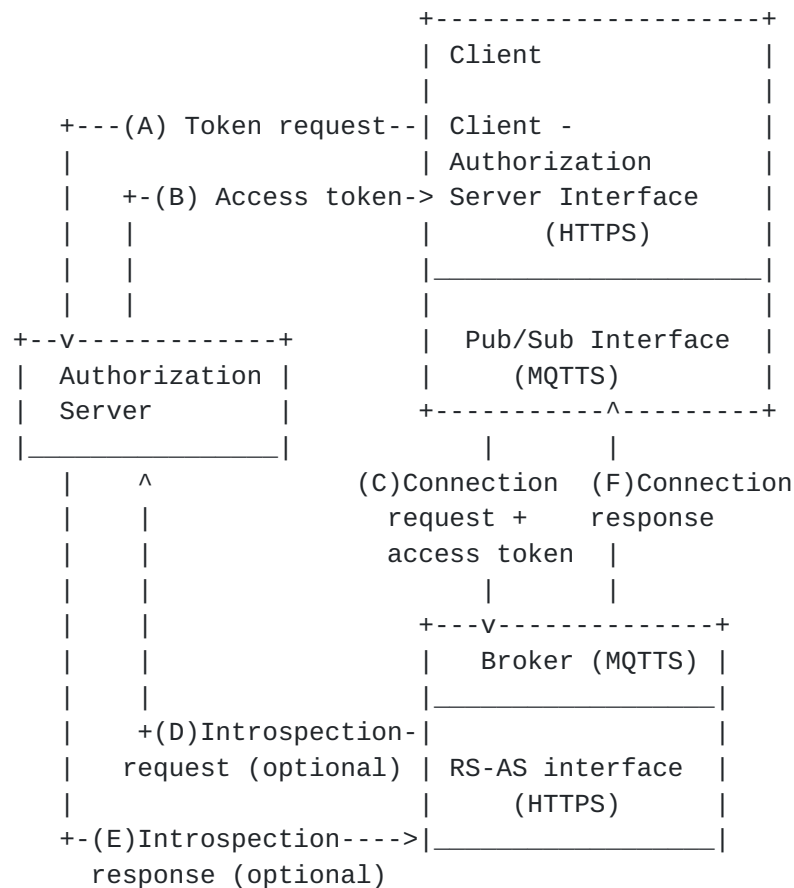
                     Figure 1: Connection set-up

2.1. Client Token Request to the Authorization Server (AS)

   The first step in the protocol flow (Figure 1 (A)) is the token
   acquisition by the Client from the AS. When requesting an access
   token from the AS, the Client follows the token request as described
   in Section 5.6.1 of the ACE framework [I-D.ietf-ace-oauth-authz],
   howevever, it MUST set the profile parameter to 'mqtt_tls'. The
   media format is 'application/ace+json'. The AS uses JSON in the
   payload of its responses to both to the Client and the RS.

   If the AS successfully verifies the access token request and
   authorizes the Client for the indicated audience (i.e., RS) and
   scopes (i.e., publish/subscribe permissions over topics), the AS
   issues an access token (Figure 1 (B)). The response includes the
   parameters described in Section 5.6.2 of the ACE framework
   [I-D.ietf-ace-oauth-authz]. The included token is assumed to be
   Proof-of-Possession (PoP) token by default. This document follows
   RFC 7800 [RFC7800] for PoP semantics for JWTs. The PoP token
   includes a 'cnf' parameter with a symmetric or asymmetric PoP key.
   The 'cnf' parameter in the web tokens are to be consumed by the RS

and not the Client. The PoP token may include a 'rs_cnf' parameter
containing the information about the public key used by the RS to
authenticate as described in [I-D.ietf-ace-oauth-params].

The AS returns error responses for JSON-based interactions following
the Section 5.2 of RFC 6749 [RFC6749]. When CBOR is used, the
interactions must implement the Section 5.6.3 of ACE framework
[I-D.ietf-ace-oauth-authz].

2.2. Client Connection Request to the Broker (C)

2.2.1. Client-Server Authentication over TLS and MQTT

The Client and the Broker MUST perform mutual authentication. The
Client MAY authenticate to the Broker over MQTT or TLS. For MQTT,
the options are "None" and "ace". For TLS, the options are "Anon"
for anonynous client, and "Known(RPK/PSK)" for Raw Public Keys (RPK)
and Pre-Shared Keys (PSK), respectively. Combined, the Client
authentication takes the following options:

o "TLS:Anon-MQTT:None": This option is used only for the topics that
   do not require authorization, including the "authz-info" topic.
   Publishing to the "authz-info" topic is described in
   Section 2.2.2.

o "TLS:Anon-MQTT:ace": The token is transported inside the CONNECT
   message, and MUST be validated using one of the methods described
   in Section 2.2.2. This option also supports a tokenless
   connection request for AS discovery.

o "TLS:Known(RPK/PSK)-MQTT:none": For the RPK, the token MUST have
   been published to the "authz-info" topic. For the PSK, the token
   MAY be, alternatively, provided in the "psk_identity". The TLS
   session set-up is as described in DTLS profile for ACE
   [I-D.ietf-ace-dtls-authorize].

o "TLS:Known(RPK/PSK)-MQTT:ace": This option SHOULD NOT be chosen.
   In any case, the token transported in the CONNECT overwrites any
   permissions passed during the TLS authentication.

It is RECOMMENDED that the Client follows TLS:Anon-MQTT:ace.

The Broker MUST be authenticated during the TLS handshake. If the
Client authentication uses TLS:Known(RPK/PSK), then the Broker is
authenticated using the respective method. Otherwise, to
authenticate the Broker, the client MUST validate a public key from a
X.509 certificate or an RPK from the Broker against the 'rs_cnf'
parameter in the token response. The AS MAY include the thumbprint

of the RS's X.509 certificate in the 'rs_cnf' (thumbrint as defined
in [I-D.ietf-cose-x509]), then the client MUST validate the RS
certificate against this thumbprint.

2.2.2. authz-info: The Authorization Information Topic

In the cases when the Client MUST transport the token to the Broker
before the TLS handshake, the Client connects to the Broker and
publishes its token to the "authz-info" topic. The "authz-info"
topic MUST be publish-only (i.e., the Clients are not allowed to
subscribe to it). "authz-info" is not protected, and hence, the
Client authenticates with the RS using the "TLS:Anon-MQTT:None"
option. After publishing the token, the Client disconnects from the
Broker and is expected to try reconnecting over TLS.

The Broker stores and indexes all tokens received to this topic in
its key store similar to DTLS profile for ACE
[I-D.ietf-ace-dtls-authorize]. This profile follows the
recommendation of Section 5.8.1 of ACE framework
[I-D.ietf-ace-oauth-authz], and expects that RS stores only one token
per proof-of-possession key, and any other token linked to the same
key overwrites existing token at the RS.

The Broker MUST verify the validity of the token (i.e., through local
validation or introspection) as described in Section 2.2.5. To
validate the token, RS MAY need to introspect the token with the AS
e.g., if the token is a reference. If the token is not valid, the
Broker MUST discard the token. Depending on the QoS level of the
PUBLISH message, the Broker may return the error response as a PUBACK
or a DISCONNECT message.

If the QoS level is equal to 0, and token is invalid or the claims
cannot be obtained in the case of an introspected token, the Broker
MUST send a DISCONNECT message with the reason code '0x87 (Not
authorized)'. If the token does not parse to a token, the RS MUST
send a DISCONNECT with the reason code '0x99 (Payload format
invalid)'.

For the QoS level of the PUBLISH message is greater than or equal to
1, the Broker MAY return 'Not authorized' in PUBACK. If the token
does not parse to a token, the PUBACK reason code is '0x99 (Payload
format invalid)'.

It must be noted that when the AS sends the 'Not authorized'
response, this corresponds to the token being invalid, and not that
the actual PUBLISH message was not authorized. Given that the
"authz-info" is a public topic, this response is not expected to
cause a confusion.

2.2.3. Transporting Access Token Inside the MQTT CONNECT

This section describes how the Client transports the token to the
Broker (RS) inside the CONNECT message. If this method is used, the
Client TLS connection is expected to be anonymous, and the Broker is
authenticated during the TLS connection set-up. The approach
described in this section is similar to an earlier proposal by
Fremantle et al. [fremantle14].

Figure 2 shows the structure of the MQTT CONNECT message used in MQTT
v5.0. A CONNECT message is composed of a fixed header, a variable
header and a payload. The fixed header contains the Control Packet
Type (CPT), Reserved, and Remaining Length fields. The Variable
Header contains the Protocol Name, Protocol Level, Connect Flags,
Keep Alive, and Properties fields. The Connect Flags in the variable
header specify the properties of the MQTT session. It also indicates
the presence or absence of some fields in the Payload. The payload
contains one or more encoded fields, namely a unique Client
identifier for the Client, a Will Topic, Will Payload, User Name and
Password. All but the Client identifier can be omitted depending on
flags in the Variable Header.

```
      0               8               16              24              32
      +------------------------------------------------------------+
      |CPT=1 | Rsvd.|Remaining len.| Protocol  name len. = 4 |
      +------------------------------------------------------------+
      |                    'M' 'Q' 'T' 'T'                         |
      +------------------------------------------------------------+
      | Proto.level=5|Connect flags|        Keep alive          |
      +------------------------------------------------------------+
      |                    Property length                         |
      |        Auth. Method (0x15) | 'ace'                         |
      |        Auth. Data (0x16)   | empty or token or            |
      |                              token + PoP data              |
      +------------------------------------------------------------+
      |                       Payload                              |
      +------------------------------------------------------------+
```
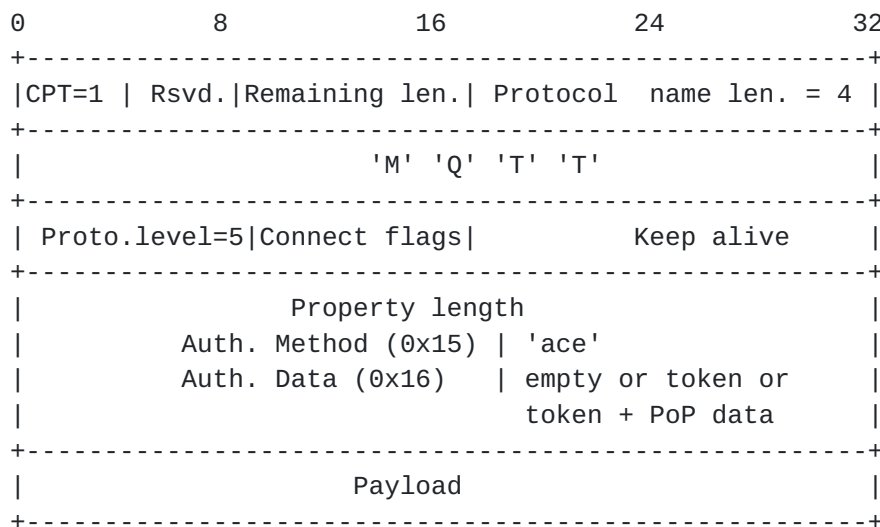
Figure 2: MQTT v5 CONNECT control message with ACE authentication.
(CPT=Control Packet Type)

The CONNECT message flags are Username, Password, Will retain, Will
QoS, Will Flag, Clean Start, and Reserved. Figure 6 shows how the
MQTT connect flags MUST be set to use AUTH packets for authentication
and authorisation. To use AUTH, the username and password flags MUST
be set to 0. The RS MAY support token transport using username and
password and the CONNECT message for that option is described in

Section 6 for MQTT v3.1.1, which is the only option available to MQTT
v3.1.1.

```
+-------------------------------------------------------------+
|User name|Pass.|Will retain|Will QoS|Will Flag|Clean| Rsvd.|
|   Flag  |Flag |           |        |         |Start|      |
+-------------------------------------------------------------+
| 0       | 0   |    X      | X X    |    X    | X   | 0    |
+-------------------------------------------------------------+
```

Figure 3: CONNECT flags for AUTH

The Will Flag indicates that a Will message needs to be sent if
network connection is not closed normally. The situations in which
the Will message is published include disconnections due to I/O or
network failures, and the server closing the network connection due
to a protocol error. The Client may set the Will Flag as desired
(marked as 'X' in Figure 3). If the Will Flag is set to 1 and the
Broker accepts the connection request, the Broker must store the Will
message, and publish it when the network connection is closed
according to Will QoS and Will retain parameters, and MQTT Will
management rules. To avoid publishing Will Messages in the case of
temporary network disconnections, the Client may specify a Will Delay
Interval in the Will Properties. Section 5 explains how the Broker
deals with the retained messages in further detail.

In MQTT v5, to achieve a clean session (i.e., the session does not
continue an existing session), the Client sets the Clean Start Flag
to 1 and, the Session Expiry Interval to 0 in the CONNECT message.
However, in this profile, the Broker MUST always start with a clean
session regardless of how these parameters are set. The Broker MUST
set the Session Present flag to 0 in the CONNACK packet to signal the
Client that the Broker started a clean session.

2.2.4. Authentication Using AUTH Property

To use AUTH, the Client MUST set the Authentication Method as a
property of a CONNECT packet by using the property identifier 21
(0x15). This is followed by a UTF-8 Encoded String containing the
name of the Authentication Method, which MUST be set to 'ace'. If
the RS does not support this profile, it sends a CONNACK with a
Reason Code of '0x8C (Bad authentication method)'.

The Authentication Method is followed by the Authentication Data,
which has a property identifier 22 (0x16) and is binary data. Based
on the Authentication Data, this profile allows:

o Proof-of-Possession using a challenge from the TLS session

   o Proof-of-Possession via Broker generated challenge/response

   o Unauthorised request and Authorisation Server discovery

2.2.4.1. Proof-of-Possession Using a Challenge from the TLS session

   For this option, the Authentication Data MUST contain the token and
   the keyed message digest (MAC) or the Client signature. The secret
   that is used for the proof-of-possession calculation, i.e., to
   calculate the keyed message digest (MAC) or the Client signature, is
   obtained using a TLS exporter ([RFC5705] for TLS 1.2 and for TLS 1.3,
   Section 7.5 of [RFC8446]). The secret is exported from TLS using the
   exporter label 'EXPORTER-ACE-Sign-Challenge', an empty context, and
   length of 32 bytes. The token is also validated as described in
   Section 2.2.5 and the server responds with a CONNACK with the
   appropriate response code.

2.2.4.2. Proof-of-Possession via Broker-generated Challenge/Response

   For this option, the RS follows a Broker-generated challenge/response
   protocol. The success case is illustrated in Figure 4. If the
   Authentication Data only includes the token, the RS MUST respond with
   an AUTH packet, with the Authenticate Reason Code set to '0x18
   (Continue Authentication)'. This packet includes the Authentication
   Method, which MUST be set to 'ace' and Authentication Data. The
   Authentication Data MUST NOT be empty and contains an 8-byte nonce as
   a challenge for the Client. The Client responds to this with an AUTH
   packet with a reason code '0x18 (Continue Authentication)'.
   Similarly, the Client packet sets the Authentication Method to 'ace'.
   The Authentication Data in the Client's response contains a client-
   generated 8-byte nonce, and the signature or MAC computed over the RS
   nonce concatenated with the client nonce. Next, the token is
   validated as described in Section 2.2.5.

```
                               Resource
                   Client       Server
                     |            |
                     |<==========>| TLS connection set-up
                     |            |
                     |            |
                     +----------->| CONNECT with Authentication Data
                     |            | contains only token
                     |            |
                     <-----------+ AUTH '0x18 (Continue Authentication)'
                     |            | 8-byte nonce as RS challenge
                     |            |
                     |----------->| AUTH '0x18 (Continue Authentication)'
                     |            | 8-byte client nonce + signature/MAC
                     |            |
                     |            |-----+ Token validation (may involve
  introspection)
                     |            |     |
                     |            |<----+
                     |            |
                     |<-----------+ CONNACK '0x00 (Success)'
```

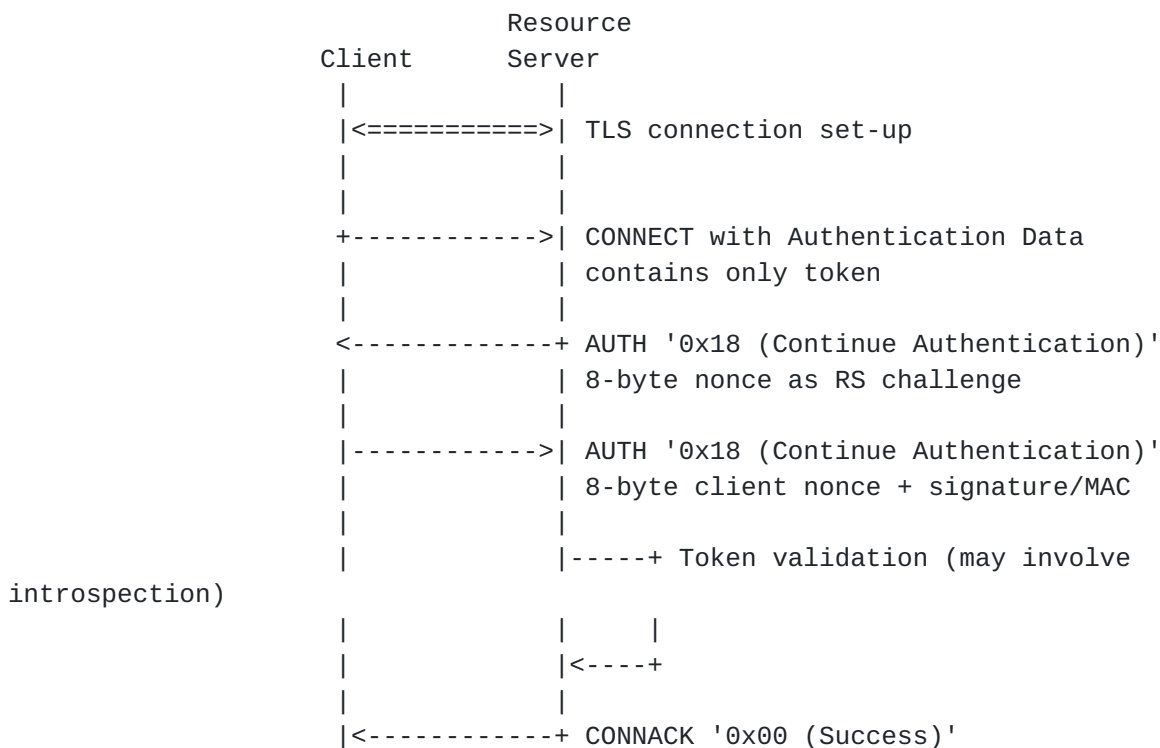   Figure 4: PoP Challenge/Response Protocol Flow - Success

2.2.4.3. Unauthorised Request: Authorisation Server Discovery

   Finally, this document allows the CONNECT message to have the
   Authentication Method set to 'ace' followed by an empty
   Authentication Data field. This is the AS discovery option and the
   RS responds with the CONNACK reason code '0x87 (Not Authorized)' and
   includes a User Property (identified by 38 (0x26)) for the AS
   creation hints as defined in the Section 5.1.2 of the ACE framework
   [I-D.ietf-ace-oauth-authz].

2.2.5. Token Validation

   The RS MUST verify the validity of the token either locally (e.g., in
   the case of a self-contained token) or the RS MAY send an
   introspection request to the AS. RS MUST verify the claims according
   to the rules set in the Section 5.8.1.1 of the ACE framework
   [I-D.ietf-ace-oauth-authz].

   To authenticate the Client, the RS validates the signature or the
   MAC, depending on how the PoP protocol is implemented. Validation of
   the signature or MAC MUST fail if the signature algorithm is set to
   "none", when the key used for the signature algorithm cannot be
   determined, or the computed and received signature/MAC do not match.

2.2.6. **The Broker's Response to Client Connection Request**

   Based on the validation result (obtained either via local inspection
   or using the /introspection interface of the AS), the Broker MUST
   send a CONNACK message to the Client. The reason code of the CONNACK
   is '0x00 (Success)' if the token validation is successful. The
   Broker MUST also set Session Present to 0 in the CONNACK packet to
   signal a clean session to the Client. In case of an invalid PoP
   token, the CONNACK reason code is '0x87 (Not Authorized)'.

   If the Broker accepts the connection, it MUST store the token until
   the end of the connection. On Client or Broker disconnection, the
   Client is expected to provide a token again inside the next CONNECT
   message.

   If the token is not self-contained and the Broker uses token
   introspection, it MAY cache the validation result to authorize the
   subsequent PUBLISH and SUBSCRIBE messages. PUBLISH and SUBSCRIBE
   messages, which are sent after a connection set-up, do not contain
   access tokens. If the introspection result is not cached, then the
   RS needs to introspect the saved token for each request. The Broker
   SHOULD also use a cache time out to introspect tokens regularly.

3. **Authorizing PUBLISH and SUBSCRIBE Messages**

   To authorize a Client's PUBLISH and SUBSCRIBE messages, the Broker
   needs to use the scope field in the token (or in the introspection
   result). The scope field contains the publish and subscribe
   permissions for the Client. Scope strings SHOULD be encoded as a
   permission, followed by an underscore, followed by a topic filter.
   Two permissions apply to topic filters: 'publish' and 'subscribe'.
   Topic filters are implemented as described in MQTT specification and
   includes special wildcard characters. The multi-level wildcard, '#',
   matches any number of levels within a topic, and the single-level
   wildcard, '+', matches one topic level.

   An example scope field may contain multiple such strings, space
   delimited, e.g., 'publish_topic1 subscribe_topic2/#'
   publish_+/topic3. This access token gives 'publish' permission to
   the 'topic1', 'subscribe' permission to all the subtopics of
   'topic2', and 'publish' permission to all topic3, skipping one level.
   If the Will Flag is set,then the Broker MUST check that the token
   allows the publication of the Will message (i.e., the scope is
   "publish_" followed by the Will Topic).

3.1. PUBLISH Messages from the Publisher Client to the Broker

On receiving the PUBLISH message, the Broker MUST use the type of
message (i.e., PUBLISH) and the Topic name in the message header to
match against the scope string in the cached token or its
introspection result. Following the example above, a client sending
a PUBLISH message to 'a/topic3' would be allowed to publish, as the
scope includes the string 'publish_+/topic3'.

If the Client is allowed to publish to the topic, the RS must publish
the message to all valid subscribers of the topic. In the case of an
authorization failure, an error MAY be returned to the Client. For
this the QoS level of the PUBLISH message MUST be set to greater than
or equal to 1. This guarantees that RS responds with either a PUBACK
or PUBREC packet with reason code '0x87 (Not authorized)'.

On receiving a PUBACK with '0x87 (Not authorized)', the Client MAY
reauthenticate as described in Section 4, and pass a new token
following the same PoP methods as described in Figure 2.

3.2. PUBLISH Messages from the Broker to the Subscriber Clients

To forward PUBLISH messages to the subscribing Clients, the Broker
identifies all the subscribers that have valid matching topic
subscriptions (i.e., the tokens are valid, and token scopes allow a
subscription to the particular topic). The Broker sends a PUBLISH
message with the Topic name to all the valid subscribers.

RS MUST stop forwarding messages to the unauthorized subscribers.
There is no way to inform the Clients with invalid tokens that an
authorization error has occurred other than sending a DISCONNECT
message. The RS SHOULD send a DISCONNECT message with the reason
code '0x87 (Not authorized)'. Note that the server-side DISCONNECT
is a new feature of MQTT v5.0 (in MQTT v3.1.1, the server needs to
drop the connection).

3.3. Authorizing SUBSCRIBE Messages

In MQTT, a SUBSCRIBE message is sent from a Client to the Broker to
create one or more subscriptions to one or more topics. The
SUBSCRIBE message may contain multiple Topic Filters. The Topic
Filters may include wildcard characters.

On receiving the SUBSCRIBE message, the Broker MUST use the type of
message (i.e., SUBSCRIBE) and the Topic Filter in the message header
to match against a scope string of the stored token or introspection
result.

As a response to the SUBSCRIBE message, the Broker issues a SUBACK message. For each Topic Filter, the SUBACK packet includes a return code matching the QoS level for the corresponding Topic Filter. In the case of failure, the return code is 0x87, indicating that the Client is 'Not authorized'. A reason code is returned for each Topic Filter. Therefore, the Client may receive success codes for a subset of its Topic Filters while being unauthorized for the rest.

4. Token Expiration and Reauthentication

The Broker MUST check for token expiration whenever a CONNECT, PUBLISH or SUBSCRIBE message is received or sent. The Broker SHOULD check for token expiration on receiving a PINGREQUEST message. The Broker MAY also check for token expiration periodically e.g., every hour. This may allow for early detection of a token expiry.

The token expiration is checked by checking the 'exp' claim of a JWT or introspection response, or via performing an introspection request with the AS as described in Section 5.7 of the ACE framework [I-D.ietf-ace-oauth-authz]. Token expirations may trigger the RS to send PUBACK, SUBACK and DISCONNECT messages with return code set to 'Not authorised'. After sending a DISCONNECT message, the network connection is closed, and no more messages can be sent. However, as a response to the PUBACK and SUBACK, the Client MAY re-authenticate by sending an AUTH packet with a Reason Code of '0x19 (Re-authentication)'.

To re-authenticate, the Client sends an AUTH packet with reason code '0x19 (Re-authentication)'. The Client MUST set the Authentication Method as 'ace' and transport the new token in the Authentication Data. The Client and the RS go through the same steps for proof of possession validation as described in Section 2.2. The Client SHOULD use the same method used for the first connection request. If the re-authentication fails, the server MUST send a DISCONNECT with the reason code '0x87 (Not Authorized)'. The Clients can also proactively update their tokens i.e., before they receive a message with 'Not authorized' return code.

5. Handling Disconnections and Retained Messages

In the case of a Client DISCONNECT, the Broker deletes all the session state but MUST keep the retained messages. By setting a RETAIN flag in a PUBLISH message, the publisher indicates to the Broker that it should store the most recent message for the associated topic. Hence, the new subscribers can receive the last sent message from the publisher of that particular topic without waiting for the next PUBLISH message. The Broker MUST continue

publishing the retained messages as long as the associated tokens are valid.

In case of disconnections due to network errors or server disconnection due to a protocol error (which includes authorization errors), the Will message must be sent if the Client supplied a Will in the CONNECT message. The Client's token scopes MUST include the Will Topic. The Will message MUST be published to the Will Topic regardless of whether the corresponding token has expired. In the case of a server-side DISCONNECT, the server returns the '0x87 Not Authorized' return code to the Client.

[6](#). Reduced Protocol Interactions for MQTT v3.1.1

This section describes a reduced set of protocol interactions for the MQTT v3.1.1 Client.

[6.1](#). Token Transport

As in MQTT v5, The Token MAY either be transported before the TLS session publishing to the "authz-info" topic, or inside the CONNECT message.

In MQTT v3.1.1, after the Client published to the "authz-info" topic, it is not possible for the Broker to communicate the result of the token validation as PUBACK reason codes or server-side DISCONNECT messages are not supported. In any case, an invalid token would fail the subsequent TLS handshake, which can prompt the Client to obtain a valid token.

To transport the token to the Broker inside the CONNECT message, the Client uses the username and password fields of the CONNECT message. Figure 5 shows the structure of the MQTT CONNECT message.

```
          0             8            16            24            32
          +---------------------------------------------------+
          |CPT=1 | Rsvd.|Remaining len.| Protocol  name len. = 4 |
          +---------------------------------------------------+
          |                   'M' 'Q' 'T' 'T'                 |
          +---------------------------------------------------+
          | Proto.level=4|Connect flags|        Keep alive     |
          +---------------------------------------------------+
          | Payload                                           |
          |     Client Identifier                             |
          |     Username as access token (UTF-8)              |
          |     Password length (2 Bytes)                     |
          |     Password data as signature/MAC (binary)       |
          +---------------------------------------------------+
```
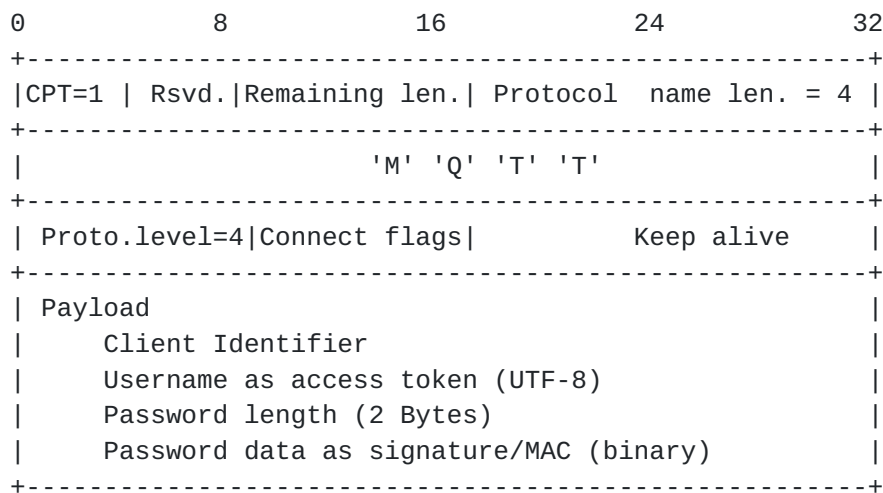
Figure 5: MQTT CONNECT control message. (CPT=Control Packet Type,
          Rsvd=Reserved, len.=length, Proto.=Protocol)

Figure 6 shows how the MQTT connect flags MUST be set to initiate a
connection with the Broker.

```
+----------------------------------------------------------------+
|User name|Pass.|Will retain|Will QoS|Will Flag|Clean| Rsvd.|
| flag    |flag |           |        |         |     |      |
+----------------------------------------------------------------+
| 1       | 1   |    X      |  X X   |    X    |  X  |  0   |
+----------------------------------------------------------------+
```
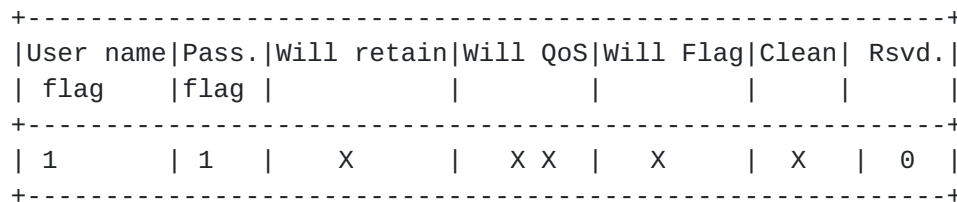
Figure 6: MQTT CONNECT flags. (Rsvd=Reserved)

The Clean Session Flag is ignored, and the Broker always sets up a
clean session. On connection success, the Broker MUST set the
Session Present flag to 0 in the CONNACK packet.

The Client may set the Will Flag as desired (marked as 'X' in
Figure 6). Username and Password flags MUST be set to 1 to ensure
that the Payload of the CONNECT message includes both Username and
Password fields.

The CONNECT in MQTT v3.1.1 does not have a field to indicate the
authentication method. To signal that the Username field contains an
ACE token, this field MUST be prefixed with 'ace' keyword, which is
followed by the access token. The Password field MUST be set to the
keyed message digest (MAC) or signature associated with the access
token for proof-of-possession. The Client MUST apply the PoP key on
the challenge derived from the TLS session as described in
Section 2.2.4.1.

In MQTT v3.1.1, the MQTT Username as a UTF-8 encoded string (i.e., is
prefixed by a 2-byte length field followed by UTF-8 encoded character
data) and may be up to 65535 bytes. Therefore, an access token that
is not a valid UTF-8 MUST be Base64 [RFC4648] encoded. (The MQTT
Password allows binary data up to 65535 bytes.)

6.2. Handling Authorization Errors

Handling errors are more primitive in MQTT v3.1.1 due to not having
appropriate error fields, error codes, and server-side DISCONNECTS.
In the following, we list how errors are handled without such
protocol support.

o CONNECT without a token: It is not possible to support AS
   discovery via sending a tokenless CONNECT message to the Broker.
   This is because a CONNACK packet in MQTT v3.1.1 does not include a
   means to provide additional information to the Client. Therefore,
   AS discovery needs to take place out-of-band. CONNECT attempt
   MUST fail.

o Client-RS PUBLISH authorization failure: In the case of a failure,
   it is not possible to return an error in MQTT v3.1.1.
   Acknowledgement messages only indicate success. In the case of an
   authorization error, the Broker SHOULD disconnect the Client.
   Otherwise, it MUST ignore the PUBLISH message. Also, as
   DISCONNECT messages are only sent from a Client to the Broker, the
   server disconnection needs to take place below the application
   layer.

o SUBSCRIBE authorization failure: In the SUBACK packet, the return
   code must be 0x80 indicating 'Failure' for the unauthorized
   topic(s). Note that, in both MQTT versions, a reason code is
   returned for each Topic Filter.

o RS-Client PUBLISH authorization failure: When RS is forwarding
   PUBLISH messages to the subscribed Clients, it may discover that
   some of the subscribers are no more authorized due to expired
   tokens. These token expirations SHOULD lead to disconnecting the
   Client rather than silently dropping messages.

7. IANA Considerations

This document registers 'EXPORTER-ACE-Sign-Challenge from
Section 2.2.4.1 in the TLS Exporter Label Registry TLS-REGISTRIES
[RFC8447].

In addition, the following registrations are done for the ACE OAuth
Profile Registry following the procedure specified in
[I-D.ietf-ace-oauth-authz].

Note to the RFC editor: Please replace all occurrences of "[RFC-
XXXX]" with the RFC number of this specification and delete this
paragraph.

Name: mqtt_tls

Description: Profile for delegating Client authentication and
authorization using MQTT as the application protocol and TLS For
transport layer security.

CBOR Value:

Reference: [RFC-XXXX]

8. Security Considerations

This document specifies a profile for the Authentication and
Authorization for Constrained Environments (ACE) framework
[I-D.ietf-ace-oauth-authz]. Therefore, the security considerations
outlined in [I-D.ietf-ace-oauth-authz] apply to this work.

In addition, the security considerations outlined in MQTT v5.0 - the
OASIS Standard [MQTT-OASIS-Standard-v5] and MQTT v3.1.1 - the OASIS
Standard [MQTT-OASIS-Standard] apply. Mainly, this document provides
an authorization solution for MQTT, the responsibility of which is
left to the specific implementation in the MQTT standards. In the
following, we comment on a few relevant issues based on the current
MQTT specifications.

To authorize a Client's publish and subscribe requests in an ongoing
session, the RS caches the access token after accepting the
connection from the Client. However, if some permissions are revoked
in the meantime, the RS may still grant publish/subscribe to revoked
topics. If the RS caches the token introspection responses, then the
RS should use a reasonable cache timeout to introspect tokens
regularly. When permissions change dynamically, it is expected that
AS also follows a reasonable expiration strategy for the access
tokens.

The RS may monitor Client behaviour to detect potential security
problems, especially those affecting availability. These include
repeated token transfer attempts to the public "authz-info" topic,
repeated connection attempts, abnormal terminations, and Clients that
connect but do not send any data. If the RS supports the public

"authz-info" topic, described in Section 2.2.2, then this may be
vulnerable to a DDoS attack, where many Clients use the "authz-info"
public topic to transport fictitious tokens, which RS may need to
store indefinitely.

9. Privacy Considerations

The privacy considerations outlined in [I-D.ietf-ace-oauth-authz]
apply to this work.

In MQTT, the RS is a central trusted party and may forward
potentially sensitive information between Clients. This document
does not protect the contents of the PUBLISH message from the Broker,
and hence, the content of the the PUBLISH message is not signed or
encrypted separately for the subscribers. This functionality may be
implemented using the proposal outlined in the CoAP Pub-Sub Profile
[I-D.palombini-ace-coap-pubsub-profile]. However, this solution
would still not provide privacy for other properties of the message
such as Topic Name.

10. References

10.1. Normative References

[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)", draft-ietf-ace-dtls-
          authorize-09 (work in progress), December 2019.

[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)", draft-ietf-ace-oauth-authz-29
          (work in progress), December 2019.

[I-D.ietf-ace-oauth-params]
          Seitz, L., "Additional OAuth Parameters for Authorization
          in Constrained Environments (ACE)", draft-ietf-ace-oauth-
          params-07 (work in progress), December 2019.

[I-D.ietf-cose-x509]
          Schaad, J., "CBOR Object Signing and Encryption (COSE):
          Headers for carrying and referencing X.509 certificates",
          draft-ietf-cose-x509-05 (work in progress), November 2019.

   [I-D.palombini-ace-coap-pubsub-profile]
              Palombini, F., "CoAP Pub-Sub Profile for Authentication
              and Authorization for Constrained Environments (ACE)",
              draft-palombini-ace-coap-pubsub-profile-06 (work in
              progress), November 2019.

   [MQTT-OASIS-Standard]
              Banks, A., Ed. and R. Gupta, Ed., "OASIS Standard MQTT
              Version 3.1.1 Plus Errata 01", 2015, <http://docs.oasis-
              open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html>.

   [MQTT-OASIS-Standard-v5]
              Banks, A., Ed., Briggs, E., Ed., Borgendale, K., Ed., and
              R. Gupta, Ed., "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>.

   [RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data
              Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006,
              <https://www.rfc-editor.org/info/rfc4648>.

   [RFC5705] Rescorla, E., "Keying Material Exporters for Transport
              Layer Security (TLS)", RFC 5705, DOI 10.17487/RFC5705,
              March 2010, <https://www.rfc-editor.org/info/rfc5705>.

   [RFC7250] Wouters, P., Ed., Tschofenig, H., Ed., Gilmore, J.,
              Weiler, S., and T. Kivinen, "Using Raw Public Keys in
              Transport Layer Security (TLS) and Datagram Transport
              Layer Security (DTLS)", RFC 7250, DOI 10.17487/RFC7250,
              June 2014, <https://www.rfc-editor.org/info/rfc7250>.

   [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC
              2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174,
              May 2017, <https://www.rfc-editor.org/info/rfc8174>.

   [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol
              Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018,
              <https://www.rfc-editor.org/info/rfc8446>.

   [RFC8447] Salowey, J. and S. Turner, "IANA Registry Updates for TLS
              and DTLS", RFC 8447, DOI 10.17487/RFC8447, August 2018,
              <https://www.rfc-editor.org/info/rfc8447>.

10.2. Informative References

   [fremantle14]
             Fremantle, P., Aziz, B., Kopecky, J., and P. Scott,
             "Federated Identity and Access Management for the Internet
             of Things", research International Workshop on Secure
             Internet of Things, September 2014,
             <http://dx.doi.org/10.1109/SIoT.2014.8>.

   [I-D.ietf-ace-cwt-proof-of-possession]
             Jones, M., Seitz, L., Selander, G., Erdtman, S., and H.
             Tschofenig, "Proof-of-Possession Key Semantics for CBOR
             Web Tokens (CWTs)", draft-ietf-ace-cwt-proof-of-
             possession-11 (work in progress), October 2019.

   [RFC4949] Shirey, R., "Internet Security Glossary, Version 2",
             FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007,
             <https://www.rfc-editor.org/info/rfc4949>.

   [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>.

   [RFC7800] Jones, M., Bradley, J., and H. Tschofenig, "Proof-of-
             Possession Key Semantics for JSON Web Tokens (JWTs)",
             RFC 7800, DOI 10.17487/RFC7800, April 2016,
             <https://www.rfc-editor.org/info/rfc7800>.

Appendix A. Checklist for profile requirements

   o AS discovery: AS discovery is possible with the MQTT v5.0
      described in Section 2.2.

   o The communication protocol between the Client and RS: MQTT

   o The security protocol between the Client and RS: TLS

   o Client and RS mutual authentication: Several options are possible
      and descibed in Section 2.2.1.

   o Content format: For the HTTPS interactions with AS, "application/
      ace+json".

   o PoP protocols: Either symmetric or asymmetric keys can be
      supported.

   o Unique profile identifier: mqtt_tls

o Token introspection: RS uses HTTPS /introspect interface of AS.

o Token request: Client or its Client AS uses HTTPS /token interface
   of AS.

o /authz-info endpoint: It MAY be supported using the method
   described in Section 2.2.2, but is not protected.

o Token transport: Via "authz-info" topic, or in MQTT CONNECT
   message for both versions of MQTT. AUTH extensions also used for
   authentication and re-authentication for MQTT v5.0 as described in
   Section 2.2 and in Section 4.

Appendix B. Document Updates

Version 02 to 03:

o Added the option of Broker certificate thumbprint in the 'rs_cnf'
   sent to the Client.

o Clarified the use of a random nonce from the TLS Exporter for PoP,
   added to the IANA requirements that the label should be
   registered.

o Added a client nonce, when Challenge/Response Authentication is
   used between Client and Broker.

o Clarified the use of the "authz-info" topic and the error response
   if token validation fails.

o Added clarification on wildcard use in scopes for publish/
   subscribe permissions

o Reorganised sections so that token authorisation for publish/
   subscribe messages are better placed.

Version 01 to 02:

o Clarified protection of Application Message payload as out of
   scope, and cited draft-palombini-ace-coap-pubsub-profile for a
   potential solution

o Expanded Client connection authorization to capture different
   options for Client and Broker authentication over TLS and MQTT

o Removed Payload (and specifically Client Identifier) from proof-
   of-possesion in favor of using tls-exporter for a TLS-session
   based challenge.

   o Moved token transport via "authz-info" topic from the Appendix to
      the main text.

   o Clarified Will scope.

   o Added MQTT AUTH to terminology.

   o Typo fixes, and simplification of figures.

   Version 00 to 01:

   o Present the MQTTv5 as the RECOMMENDED version, and MQTT v3.1.1 for
      backward compatibility.

   o Clarified Will message.

   o Improved consistency in the use of terminology, and upper/lower
      case.

   o Defined Broker and MQTTS.

   o Clarified HTTPS use for C-AS and RS-AS communication. Removed
      reference to actors document, and clarified the use of client
      authorization server.

   o Clarified the Connect message payload and Client Identifier.

   o Presented different methods for passing the token, and PoP.

   o Added new figures to explain AUTH packets exchang, updated CONNECT
      message figure.

Acknowledgements

   The authors would like to thank Ludwig Seitz for his review and his
   input on the authorization information endpoint, presented in the
   appendix.

Authors' Addresses

   Cigdem Sengul
   Nominet
   4 Kingdom Street
   London W2 6BD
   UK

   Email: csengul@acm.org

Anthony Kirby
Oxbotica
1a Milford House, Mayfield Road, Summertown
Oxford OX2 7EL
UK

Email: anthony@anthony.org

Paul Fremantle
University of Portsmouth
School of Computing, Buckingham House
Portsmouth PO1 3HE
UK

Email: paul.fremantle@port.ac.uk