ACE Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 12, 2021

S. Gerdes
O. Bergmann
C. Bormann
Universitaet Bremen TZI
G. Selander
Ericsson AB
L. Seitz
Combitech

September 08, 2020

Datagram Transport Layer Security (DTLS) Profile for Authentication and Authorization for Constrained Environments (ACE)

draft-ietf-ace-dtls-authorize-13

Abstract

This specification defines a profile of the ACE framework that allows constrained servers to delegate client authentication and authorization. The protocol relies on DTLS version 1.2 for communication security between entities in a constrained network using either raw public keys or pre-shared keys. A resource-constrained server can use this protocol to delegate management of authorization information to a trusted host with less severe limitations regarding processing power and memory.

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 March 12, 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 2
1.1. Terminology 3
2. Protocol Overview 4
3. Protocol Flow 5
3.1. Communication Between the Client and the Authorization Server 6
3.2. RawPublicKey Mode 7
3.2.1. Access Token Retrieval from the Authorization Server 7
3.2.2. DTLS Channel Setup Between Client and Resource Server 9
3.3. PreSharedKey Mode 10
3.3.1. Access Token Retrieval from the Authorization Server 10
3.3.2. DTLS Channel Setup Between Client and Resource Server 14
3.4. Resource Access 16
4. Dynamic Update of Authorization Information 18
5. Token Expiration 19
6. Secure Communication with an Authorization Server 19
7. Security Considerations 20
7.1. Reuse of Existing Sessions 21
7.2. Multiple Access Tokens 22
7.3. Out-of-Band Configuration 22
8. Privacy Considerations 23
9. IANA Considerations 23
10. Acknowledgments 24
11. References 24
11.1. Normative References 24
11.2. Informative References 25
Authors' Addresses 26

1. Introduction

This specification defines a profile of the ACE framework [I-D.ietf-ace-oauth-authz]. In this profile, a client and a resource server use CoAP [RFC7252] over DTLS version 1.2 [RFC6347] to communicate. The client obtains an access token, bound to a key (the proof-of-possession key), from an authorization server to prove its authorization to access protected resources hosted by the resource

server. Also, the client and the resource server are provided by the authorization server with the necessary keying material to establish a DTLS session. The communication between client and authorization server may also be secured with DTLS. This specification supports DTLS with Raw Public Keys (RPK) [RFC7250] and with Pre-Shared Keys (PSK) [RFC4279].

The ACE framework requires that client and server mutually authenticate each other before any application data is exchanged. DTLS enables mutual authentication if both client and server prove their ability to use certain keying material in the DTLS handshake. The authorization server assists in this process on the server side by incorporating keying material (or information about keying material) into the access token, which is considered a "proof of possession" token.

In the RPK mode, the client proves that it can use the RPK bound to the token and the server shows that it can use a certain RPK.

The resource server needs access to the token in order to complete this exchange. For the RPK mode, the client must upload the access token to the resource server before initiating the handshake, as described in Section 5.8.1 of the ACE framework [I-D.ietf-ace-oauth-authz].

In the PSK mode, client and server show with the DTLS handshake that they can use the keying material that is bound to the access token. To transfer the access token from the client to the resource server, the "psk_identity" parameter in the DTLS PSK handshake may be used instead of uploading the token prior to the handshake.

As recommended in Section 5.8 of [I-D.ietf-ace-oauth-authz], this specification uses CBOR web tokens to convey claims within an access token issued by the server. While other formats could be used as well, those are out of scope for this document.

1.1. Terminology

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.

Readers are expected to be familiar with the terms and concepts described in [I-D.ietf-ace-oauth-authz] and in [I-D.ietf-ace-oauth-params].

The authorization information (authz-info) resource refers to the authorization information endpoint as specified in [I-D.ietf-ace-oauth-authz]. The term "claim" is used in this document with the same semantics as in [I-D.ietf-ace-oauth-authz], i.e., it denotes information carried in the access token or returned from introspection.

2. Protocol Overview

The CoAP-DTLS profile for ACE specifies the transfer of authentication information and, if necessary, authorization information between the client (C) and the resource server (RS) during setup of a DTLS session for CoAP messaging. It also specifies how the client can use CoAP over DTLS to retrieve an access token from the authorization server (AS) for a protected resource hosted on the resource server. As specified in Section 6.7 of [I-D.ietf-ace-oauth-authz], use of DTLS for one or both of these interactions is completely independent

This profile requires the client to retrieve an access token for protected resource(s) it wants to access on the resource server as specified in [I-D.ietf-ace-oauth-authz]. Figure 1 shows the typical message flow in this scenario (messages in square brackets are optional):

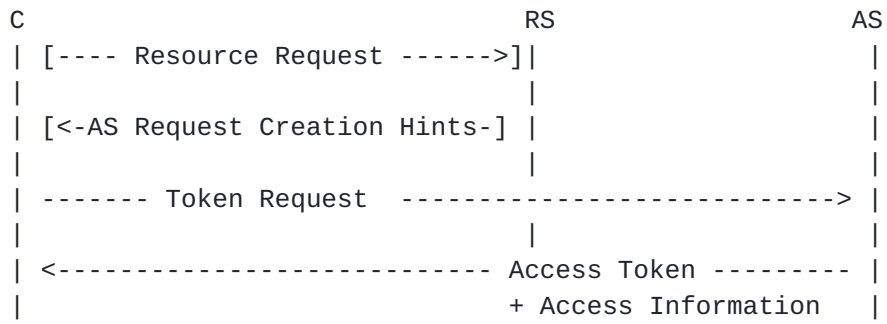


Figure 1: Retrieving an Access Token

To determine the authorization server in charge of a resource hosted at the resource server, the client can send an initial Unauthorized Resource Request message to the resource server. The resource server then denies the request and sends an AS Request Creation Hints message containing the address of its authorization server back to the client as specified in Section 5.1.2 of [I-D.ietf-ace-oauth-authz].

Once the client knows the authorization server's address, it can send an access token request to the token endpoint at the authorization

server as specified in [I-D.ietf-ace-oauth-authz]. As the access token request as well as the response may contain confidential data, the communication between the client and the authorization server must be confidentiality-protected and ensure authenticity. The client may have been registered at the authorization server via the OAuth 2.0 client registration mechanism as outlined in Section 5.3 of [I-D.ietf-ace-oauth-authz].

The access token returned by the authorization server can then be used by the client to establish a new DTLS session with the resource server. When the client intends to use an asymmetric proof-of-possession key in the DTLS handshake with the resource server, the client MUST upload the access token to the authz-info resource, i.e. the authz-info endpoint, on the resource server before starting the DTLS handshake, as described in Section 5.8.1 of [I-D.ietf-ace-oauth-authz]. In case the client uses a symmetric proof-of-possession key in the DTLS handshake, the procedure as above MAY be used, or alternatively, the access token MAY instead be transferred in the DTLS ClientKeyExchange message (see Section 3.3.2). In any case, DTLS MUST be used in a mode that provides replay protection.

Figure 2 depicts the common protocol flow for the DTLS profile after the client has retrieved the access token from the authorization server, AS.



Figure 2: Protocol overview

3. Protocol Flow

The following sections specify how CoAP is used to interchange access-related data between the resource server, the client and the authorization server so that the authorization server can provide the client and the resource server with sufficient information to establish a secure channel, and convey authorization information specific for this communication relationship to the resource server.

Section 3.1 describes how the communication between the client (C) and the authorization server (AS) must be secured. Depending on the used CoAP security mode (see also Section 9 of [RFC7252], the Client-to-AS request, AS-to-Client response (see Section 5.6 of [I-D.ietf-ace-oauth-authz]) and DTLS session establishment carry slightly different information. Section 3.2 addresses the use of raw public keys while Section 3.3 defines how pre-shared keys are used in this profile.

3.1. Communication Between the Client and the Authorization Server

To retrieve an access token for the resource that the client wants to access, the client requests an access token from the authorization server. Before the client can request the access token, the client and the authorization server MUST establish a secure communication channel. This profile assumes that the keying material to secure this communication channel has securely been obtained either by manual configuration or in an automated provisioning process. The following requirements in alignment with Section 6.5 of [I-D.ietf-ace-oauth-authz] therefore must be met:

- The client MUST securely have obtained keying material to communicate with the authorization server.

- Furthermore, the client MUST verify that the authorization server is authorized to provide access tokens (including authorization information) about the resource server to the client, and that this authorization information about the authorization server is still valid.

- Also, the authorization server MUST securely have obtained keying material for the client, and obtained authorization rules approved by the resource owner (RO) concerning the client and the resource server that relate to this keying material.

The client and the authorization server MUST use their respective keying material for all exchanged messages. How the security association between the client and the authorization server is bootstrapped is not part of this document. The client and the authorization server must ensure the confidentiality, integrity and authenticity of all exchanged messages within the ACE protocol.

Section 6 specifies how communication with the authorization server is secured.

3.2. RawPublicKey Mode

When the client uses RawPublicKey authentication, the procedure is as described in the following.

3.2.1. Access Token Retrieval from the Authorization Server

After the client and the authorization server mutually authenticated each other and validated each other's authorization, the client sends a token request to the authorization server's token endpoint. The client MUST add a "req_cnf" object carrying either its raw public key or a unique identifier for a public key that it has previously made known to the authorization server. It is RECOMMENDED that the client uses DTLS with the same keying material to secure the communication with the authorization server, proving possession of the key as part of the token request. Other mechanisms for proving possession of the key may be defined in the future.

An example access token request from the client to the authorization server is depicted in Figure 3.

```
POST coaps://as.example.com/token
Content-Format: application/ace+cbor
Payload:
{
  grant_type : client_credentials,
  req_aud    : "tempSensor4711",
  req_cnf    : {
    COSE_Key : {
      kty : EC2,
      crv : P-256,
      x   : h'e866c35f4c3c81bb96a1...',
      y   : h'2e25556be097c8778a20...'
    }
  }
}
```

Figure 3: Access Token Request Example for RPK Mode

The example shows an access token request for the resource identified by the string "tempSensor4711" on the authorization server using a raw public key.

The authorization server MUST check if the client that it communicates with is associated with the RPK in the "req_cnf" parameter before issuing an access token to it. If the authorization server determines that the request is to be authorized according to the respective authorization rules, it generates an access token

response for the client. The access token MUST be bound to the RPK of the client by means of the "cnf" claim.

The response MAY contain a "profile" parameter with the value "coap_dtls" to indicate that this profile MUST be used for communication between the client and the resource server. The "profile" may be specified out-of-band, in which case it does not have to be sent. The response also contains an access token with information for the resource server about the client's public key. The authorization server MUST return in its response the parameter "rs_cnf" unless it is certain that the client already knows the public key of the resource server. The authorization server MUST ascertain that the RPK specified in "rs_cnf" belongs to the resource server that the client wants to communicate with. The authorization server MUST protect the integrity of the access token such that the resource server can detect unauthorized changes. If the access token contains confidential data, the authorization server MUST also protect the confidentiality of the access token.

The client MUST ascertain that the access token response belongs to a certain previously sent access token request, as the request may specify the resource server with which the client wants to communicate.

An example access token response from the authorization server to the client is depicted in Figure 4. Here, the contents of the "access_token" claim have been truncated to improve readability. Caching proxies process the Max-Age option in the CoAP response which has a default value of 60 seconds (Section 5.6.1 of [RFC7252]). The authorization server SHOULD adjust the Max-Age option such that it does not exceed the "expires_in" parameter to avoid stale responses.

```
2.01 Created
Content-Format: application/ace+cbor
Max-Age: 3560
Payload:
{
  access_token : b64'SlAV32hkKG...
   (remainder of CWT omitted for brevity;
   CWT contains the client's RPK in the cnf claim)',
  expires_in : 3600,
  rs_cnf     : {
    COSE_Key : {
      kty : EC2,
      crv : P-256,
      x   : h'd7cc072de2205bdc1537...',
      y   : h'f95e1d4b851a2cc80fff...'
    }
  }
}
```

Figure 4: Access Token Response Example for RPK Mode

3.2.2. DTLS Channel Setup Between Client and Resource Server

Before the client initiates the DTLS handshake with the resource server, the client MUST send a "POST" request containing the obtained access token to the authz-info resource hosted by the resource server. After the client receives a confirmation that the resource server has accepted the access token, it SHOULD proceed to establish a new DTLS channel with the resource server. The client MUST use its correct public key in the DTLS handshake. If the authorization server has specified a "cnf" field in the access token response, the client MUST use this key. Otherwise, the client MUST use the public key that it specified in the "req_cnf" of the access token request. The client MUST specify this public key in the SubjectPublicKeyInfo structure of the DTLS handshake as described in [RFC7250].

To be consistent with [RFC7252] which allows for shortened MAC tags in constrained environments, an implementation that supports the RPK mode of this profile MUST at least support the ciphersuite TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8 [RFC7251]. As discussed in [RFC7748], new ECC curves have been defined recently that are considered superior to the so-called NIST curves. This specification therefore mandates implementation support for curve25519 (cf. [RFC8032], [RFC8422]) as this curve said to be efficient and less dangerous regarding implementation errors than the secp256r1 curve mandated in [RFC7252].

The resource server MUST check if the access token is still valid, if the resource server is the intended destination (i.e., the audience) of the token, and if the token was issued by an authorized authorization server. The access token is constructed by the authorization server such that the resource server can associate the access token with the Client's public key. The "cnf" claim MUST contain either the client's RPK or, if the key is already known by the resource server (e.g., from previous communication), a reference to this key. If the authorization server has no certain knowledge that the Client's key is already known to the resource server, the Client's public key MUST be included in the access token's "cnf" parameter. If CBOR web tokens [RFC8392] are used (as recommended in [I-D.ietf-ace-oauth-authz]), keys MUST be encoded as specified in [RFC8747]. A resource server MUST have the capacity to store one access token for every proof-of-possession key of every authorized client.

The raw public key used in the DTLS handshake with the client MUST belong to the resource server. If the resource server has several raw public keys, it needs to determine which key to use. The authorization server can help with this decision by including a "cnf" parameter in the access token that is associated with this communication. In this case, the resource server MUST use the information from the "cnf" field to select the proper keying material.

Thus, the handshake only finishes if the client and the resource server are able to use their respective keying material.

3.3. PreSharedKey Mode

When the client uses pre-shared key authentication, the procedure is as described in the following.

3.3.1. Access Token Retrieval from the Authorization Server

To retrieve an access token for the resource that the client wants to access, the client MAY include a "cnf" object carrying an identifier for a symmetric key in its access token request to the authorization server. This identifier can be used by the authorization server to determine the shared secret to construct the proof-of-possession token. The authorization server MUST check if the identifier refers to a symmetric key that was previously generated by the authorization server as a shared secret for the communication between this client and the resource server. If no such symmetric key was found, the authorization server MUST generate a new symmetric key that is returned in its response to the client.

The authorization server MUST determine the authorization rules for the client it communicates with as defined by the resource owner and generate the access token accordingly. If the authorization server authorizes the client, it returns an AS-to-Client response. If the profile parameter is present, it is set to "coap_dtls". The authorization server MUST ascertain that the access token is generated for the resource server that the client wants to communicate with. Also, the authorization server MUST protect the integrity of the access token to ensure that the resource server can detect unauthorized changes. If the token contains confidential data such as the symmetric key, the confidentiality of the token MUST also be protected. Depending on the requested token type and algorithm in the access token request, the authorization server adds access Information to the response that provides the client with sufficient information to setup a DTLS channel with the resource server. The authorization server adds a "cnf" parameter to the access information carrying a "COSE_Key" object that informs the client about the shared secret that is to be used between the client and the resource server. To convey the same secret to the resource server, the authorization server can include it directly in the access token by means of the "cnf" claim or provide sufficient information to enable the resource server to derive the shared secret from the access token. As an alternative, the resource server MAY use token introspection to retrieve the keying material for this access token directly from the authorization server.

An example access token request for an access token with a symmetric proof-of-possession key is illustrated in Figure 5.

```
POST coaps://as.example.com/token
Content-Format: application/ace+cbor
Payload:
{
  audience    : "smokeSensor1807",
}
```

Figure 5: Example Access Token Request, (implicit) symmetric PoP-key

A corresponding example access token response is illustrated in Figure 6. In this example, the authorization server returns a 2.01 response containing a new access token (truncated to improve readability) and information for the client, including the symmetric key in the cnf claim. The information is transferred as a CBOR data structure as specified in [I-D.ietf-ace-oauth-authz].

```
   2.01 Created
   Content-Format: application/ace+cbor
   Max-Age: 85800
   Payload:
   {
      access_token : h'd08343a10...
      (remainder of CWT omitted for brevity)
      token_type : PoP,
      expires_in : 86400,
      profile    : coap_dtls,
      cnf        : {
        COSE_Key : {
          kty : symmetric,
          kid : h'3d027833fc6267ce',
          k   : h'73657373696f6e6b6579'
        }
      }
   }
```

Figure 6: Example Access Token Response, symmetric PoP-key

The access token also comprises a "cnf" claim. This claim usually contains a "COSE_Key" object that carries either the symmetric key itself or a key identifier that can be used by the resource server to determine the secret key it shares with the client. If the access token carries a symmetric key, the access token MUST be encrypted using a "COSE_Encrypt0" structure. The authorization server MUST use the keying material shared with the resource server to encrypt the token.

The "cnf" structure in the access token is provided in Figure 7.

```
cnf : {
  COSE_Key : {
    kty : symmetric,
    kid : h'3d027833fc6267ce'
  }
}
```

Figure 7: Access Token without Keying Material

A response that declines any operation on the requested resource is constructed according to Section 5.2 of [RFC6749], (cf. Section 5.6.3. of [I-D.ietf-ace-oauth-authz]). Figure 8 shows an example for a request that has been rejected due to invalid request parameters.

```
4.00 Bad Request
Content-Format: application/ace+cbor
Payload:
{
  error : invalid_request
}
```

Figure 8: Example Access Token Response With Reject

The method for how the resource server determines the symmetric key from an access token containing only a key identifier is application-specific; the remainder of this section provides one example.

The authorization server and the resource server are assumed to share a key derivation key used to derive the symmetric key shared with the client from the key identifier in the access token. The key derivation key may be derived from some other secret key shared between the authorization server and the resource server. This key needs to be securely stored and processed in the same way as the key used to protect the communication between the authorization server and the resource server.

Knowledge of the symmetric key shared with the client must not reveal any information about the key derivation key or other secret keys shared between the authorization server and resource server.

In order to generate a new symmetric key to be used by client and resource server, the authorization server generates a new key identifier which MUST be unique among all key identifiers used by the authorization server for this resource server. The authorization server then uses the key derivation key shared with the resource server to derive the symmetric key as specified below. Instead of providing the keying material in the access token, the authorization server includes the key identifier in the "kid" parameter, see Figure 7. This key identifier enables the resource server to calculate the symmetric key used for the communication with the client using the key derivation key and a KDF to be defined by the application, for example HKDF-SHA-256. The key identifier picked by the authorization server MUST be unique for each access token where a unique symmetric key is required.

In this example, HKDF consists of the composition of the HKDF-Extract and HKDF-Expand steps [RFC5869]. The symmetric key is derived from the key identifier, the key derivation key and other data:

OKM = HKDF(salt, IKM, info, L),

where:

- OKM, the output keying material, is the derived symmetric key
- salt is the empty byte string
- IKM, the input keying material, is the key derivation key as defined above
- info is the serialization of a CBOR array consisting of ([RFC8610]):

```
info = [
  type : tstr,
  L : uint,
  access_token: bytes
]
```

where:

- type is set to the constant text string "ACE-CoAP-DTLS-key-derivation",
- L is the size of the symmetric key in bytes,
- access_token is the content of the "access_token" field as transferred from the authorization server to the resource server.

All CBOR data types are encoded in CBOR using preferred serialization and deterministic encoding as specified in Section 4 of [I-D.ietf-cbor-7049bis]. This implies in particular that the "type" and "L" components use the minimum length encoding. The content of the "access_token" field is treated as opaque data for the purpose of key derivation.

Use of a unique (per resource server) "kid" and the use of a key derivation IKM that MUST be unique per authorization server/resource server pair as specified above will ensure that the derived key is not shared across multiple clients. However, to additionally provide variation in the derived key across different tokens used by the same client, it is additionally RECOMMENDED to include the "iat" claim and either the "exp" or "exi" claims in the access token.

3.3.2. DTLS Channel Setup Between Client and Resource Server

When a client receives an access token response from an authorization server, the client MUST check if the access token response is bound to a certain previously sent access token request, as the request may specify the resource server with which the client wants to communicate.

The client checks if the payload of the access token response contains an "access_token" parameter and a "cnf" parameter. With this information the client can initiate the establishment of a new DTLS channel with a resource server. To use DTLS with pre-shared keys, the client follows the PSK key exchange algorithm specified in Section 2 of [RFC4279] using the key conveyed in the "cnf" parameter of the AS response as PSK when constructing the premaster secret. To be consistent with the recommendations in [RFC7252] a client is expected to offer at least the ciphersuite TLS_PSK_WITH_AES_128_CCM_8 [RFC6655] to the resource server.

In PreSharedKey mode, the knowledge of the shared secret by the client and the resource server is used for mutual authentication between both peers. Therefore, the resource server must be able to determine the shared secret from the access token. Following the general ACE authorization framework, the client can upload the access token to the resource server's authz-info resource before starting the DTLS handshake. The client then needs to indicate during the DTLS handshake which previously uploaded access token it intends to use. To do so, it MUST create a "COSE_Key" structure with the "kid" that was conveyed in the "rs_cnf" claim in the token response from the authorization server and the key type "symmetric". This structure then is included as the only element in the "cnf" structure that is used as value for "psk_identity" as shown in Figure 9.

```
{ cnf : {
   COSE_Key : {
      kty: symmetric,
      kid : h'3d027833fc6267ce'
    }
  }
}
```

Figure 9: Access token containing a single kid parameter

As an alternative to the access token upload, the client can provide the most recent access token in the "psk_identity" field of the ClientKeyExchange message. To do so, the client MUST treat the contents of the "access_token" field from the AS-to-Client response as opaque data as specified in Section 4.2 of [RFC7925] and not perform any re-coding. This allows the resource server to retrieve the shared secret directly from the "cnf" claim of the access token.

If a resource server receives a ClientKeyExchange message that contains a "psk_identity" with a length greater than zero, it MUST parse the contents of the "psk_identity" field as CBOR data structure and process the contents as following:

- If the data contains a "cnf" field with a "COSE_Key" structure with a "kid", the resource server continues the DTLS handshake with the associated key that corresponds to this kid.

- If the data comprises additional CWT information, this information must be stored as an access token for this DTLS association before continuing with the DTLS handshake.

If the contents of the "psk_identity" do not yield sufficient information to select a valid access token for the requesting client, the resource server aborts the DTLS handshake with an "illegal_parameter" alert.

When the resource server receives an access token, it MUST check if the access token is still valid, if the resource server is the intended destination (i.e., the audience of the token), and if the token was issued by an authorized authorization server. This specification implements access tokens as proof-of-possession tokens. Therefore, the access token is bound to a symmetric PoP key that is used as shared secret between the client and the resource server. A resource server MUST have the capacity to store one access token for every proof-of-possession key of every authorized client. The resource server may use token introspection [RFC7662] on the access token to retrieve more information about the specific token. The use of introspection is out of scope for this specification.

While the client can retrieve the shared secret from the contents of the "cnf" parameter in the AS-to-Client response, the resource server uses the information contained in the "cnf" claim of the access token to determine the actual secret when no explicit "kid" was provided in the "psk_identity" field. If key derivation is used, the resource server uses the "COSE_KDF_Context" information as described above.

3.4. Resource Access

Once a DTLS channel has been established as described in Section 3.2 or Section 3.3, respectively, the client is authorized to access resources covered by the access token it has uploaded to the authz-info resource hosted by the resource server.

With the successful establishment of the DTLS channel, the client and the resource server have proven that they can use their respective keying material. An access token that is bound to the client's keying material is associated with the channel. According to Section 5.8.1 of [I-D.ietf-ace-oauth-authz], there should be only one access token for each client. New access tokens issued by the authorization server replace previously issued access tokens for the respective client. The resource server therefore must have a common

understanding with the authorization server how access tokens are ordered. The authorization server may, e.g., specify a "cti" claim for the access token (see Section 5.8.3 of [I-D.ietf-ace-oauth-authz]) to employ a strict order.

Any request that the resource server receives on a DTLS channel that is tied to an access token via its keying material MUST be checked against the authorization rules that can be determined with the access token. The resource server MUST check for every request if the access token is still valid. If the token has expired, the resource server MUST remove it. Incoming CoAP requests that are not authorized with respect to any access token that is associated with the client MUST be rejected by the resource server with 4.01 response. The response SHOULD include AS Request Creation Hints as described in Section 5.1.1 of [I-D.ietf-ace-oauth-authz].

The resource server MUST only accept an incoming CoAP request as authorized if the following holds:

1. The message was received on a secure channel that has been established using the procedure defined in this document.

2. The authorization information tied to the sending client is valid.

3. The request is destined for the resource server.

4. The resource URI specified in the request is covered by the authorization information.

5. The request method is an authorized action on the resource with respect to the authorization information.

Incoming CoAP requests received on a secure DTLS channel that are not thus authorized MUST be rejected according to Section 5.8.2 of [I-D.ietf-ace-oauth-authz]

1. with response code 4.03 (Forbidden) when the resource URI specified in the request is not covered by the authorization information, and

2. with response code 4.05 (Method Not Allowed) when the resource URI specified in the request covered by the authorization information but not the requested action.

The client MUST ascertain that its keying material is still valid before sending a request or processing a response. If the client recently has updated the access token (see Section 4), it must be

prepared that its request is still handled according to the previous authorization rules as there is no strict ordering between access token uploads and resource access messages. See also Section 7.2 for a discussion of access token processing.

If the client gets an error response containing AS Request Creation Hints (cf. Section 5.1.2 of [I-D.ietf-ace-oauth-authz] as response to its requests, it SHOULD request a new access token from the authorization server in order to continue communication with the resource server.

Unauthorized requests that have been received over a DTLS session SHOULD be treated as non-fatal by the resource server, i.e., the DTLS session SHOULD be kept alive until the associated access token has expired.

4. Dynamic Update of Authorization Information

Resource servers must only use a new access token to update the authorization information for a DTLS session if the keying material that is bound to the token is the same that was used in the DTLS handshake. By associating the access tokens with the identifier of an existing DTLS session, the authorization information can be updated without changing the cryptographic keys for the DTLS communication between the client and the resource server, i.e. an existing session can be used with updated permissions.

The client can therefore update the authorization information stored at the resource server at any time without changing an established DTLS session. To do so, the client requests a new access token from the authorization server for the intended action on the respective resource and uploads this access token to the authz-info resource on the resource server.

Figure 10 depicts the message flow where the client requests a new access token after a security association between the client and the resource server has been established using this protocol. If the client wants to update the authorization information, the token request MUST specify the key identifier of the proof-of-possession key used for the existing DTLS channel between the client and the resource server in the "kid" parameter of the Client-to-AS request. The authorization server MUST verify that the specified "kid" denotes a valid verifier for a proof-of-possession token that has previously been issued to the requesting client. Otherwise, the Client-to-AS request MUST be declined with the error code "unsupported_pop_key" as defined in Section 5.6.3 of [I-D.ietf-ace-oauth-authz].

When the authorization server issues a new access token to update existing authorization information, it MUST include the specified "kid" parameter in this access token. A resource server MUST replace the authorization information of any existing DTLS session that is identified by this key identifier with the updated authorization information.

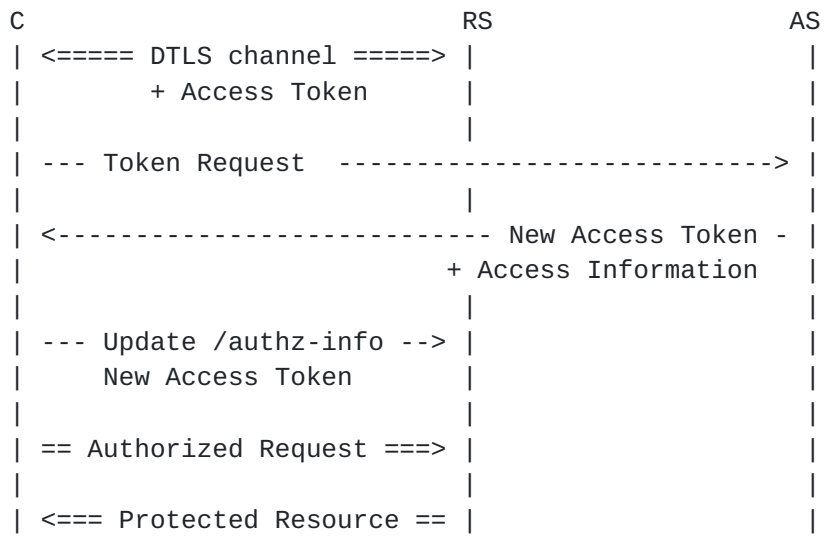


Figure 10: Overview of Dynamic Update Operation

5. Token Expiration

The resource server MUST delete access tokens that are no longer valid. DTLS associations that have been setup in accordance with this profile are always tied to specific tokens (which may be exchanged with a dynamic update as described in Section 4). As tokens may become invalid at any time (e.g., because they have expired), the association may become useless at some point. A resource server therefore MUST terminate existing DTLS association after the last access token associated with this association has expired.

As specified in Section 5.8.3 of [I-D.ietf-ace-oauth-authz], the resource server MUST notify the client with an error response with code 4.01 (Unauthorized) for any long running request before terminating the association.

6. Secure Communication with an Authorization Server

As specified in the ACE framework (Sections 5.6 and 5.7 of [I-D.ietf-ace-oauth-authz]), the requesting entity (the resource server and/or the client) and the authorization server communicate

via the token endpoint or introspection endpoint. The use of CoAP and DTLS for this communication is RECOMMENDED in this profile, other protocols (such as HTTP and TLS, or CoAP and OSCORE [RFC8613]) MAY be used instead.

How credentials (e.g., PSK, RPK, X.509 cert) for using DTLS with the authorization server are established is out of scope for this profile.

If other means of securing the communication with the authorization server are used, the communication security requirements from Section 6.2 of [I-D.ietf-ace-oauth-authz] remain applicable.

7. Security Considerations

This document specifies a profile for the Authentication and Authorization for Constrained Environments (ACE) framework [I-D.ietf-ace-oauth-authz]. As it follows this framework's general approach, the general security considerations from Section 6 of [I-D.ietf-ace-oauth-authz] also apply to this profile.

The authorization server must ascertain that the keying material for the client that it provides to the resource server actually is associated with this client. Malicious clients may hand over access tokens containing their own access permissions to other entities. This problem cannot be completely eliminated. Nevertheless, in RPK mode it should not be possible for clients to request access tokens for arbitrary public keys: if the client can cause the authorization server to issue a token for a public key without proving possession of the corresponding private key, this allows for identity misbinding attacks where the issued token is usable by an entity other than the intended one. The authorization server therefore at some point needs to validate that the client can actually use the private key corresponding to the client's public key.

When using pre-shared keys provisioned by the authorization server, the security level depends on the randomness of PSK, and the security of the TLS cipher suite and key exchange algorithm. As this specification targets at constrained environments, message payloads exchanged between the client and the resource server are expected to be small and rare. CoAP [RFC7252] mandates the implementation of cipher suites with abbreviated, 8-byte tags for message integrity protection. For consistency, this profile requires implementation of the same cipher suites. For application scenarios where the cost of full-width authentication tags is low compared to the overall amount of data being transmitted, the use of cipher suites with 16-byte integrity protection tags is preferred.

The PSK mode of this profile offers a distribution mechanism to convey authorization tokens together with a shared secret to a client and a server. As this specification aims at constrained devices and uses CoAP [RFC7252] as transfer protocol, at least the ciphersuite TLS_PSK_WITH_AES_128_CCM_8 [RFC6655] should be supported. The access tokens and the corresponding shared secrets generated by the authorization server are expected to be sufficiently short-lived to provide similar forward-secrecy properties to using ephemeral Diffie-Hellman (DHE) key exchange mechanisms. For longer-lived access tokens, DHE ciphersuites should be used.

Constrained devices that use DTLS [RFC6347] are inherently vulnerable to Denial of Service (DoS) attacks as the handshake protocol requires creation of internal state within the device. This is specifically of concern where an adversary is able to intercept the initial cookie exchange and interject forged messages with a valid cookie to continue with the handshake. A similar issue exists with the unprotected authorization information endpoint when the resource server needs to keep valid access tokens for a long time. Adversaries could fill up the constrained resource server's internal storage for a very long time with interjected or otherwise retrieved valid access tokens. To mitigate against this, the resource server should set a time boundary until an access token that has not been used until then will be deleted.

The protection of access tokens that are stored in the authorization information endpoint depends on the keying material that is used between the authorization server and the resource server: The resource server must ensure that it processes only access tokens that are (encrypted and) integrity-protected by an authorization server that is authorized to provide access tokens for the resource server.

7.1. Reuse of Existing Sessions

To avoid the overhead of a repeated DTLS handshake, [RFC7925] recommends session resumption [RFC5077] to reuse session state from an earlier DTLS association and thus requires client side implementation. In this specification, the DTLS session is subject to the authorization rules denoted by the access token that was used for the initial setup of the DTLS association. Enabling session resumption would require the server to transfer the authorization information with the session state in an encrypted SessionTicket to the client. Assuming that the server uses long-lived keying material, this could open up attacks due to the lack of forward secrecy. Moreover, using this mechanism, a client can resume a DTLS session without proving the possession of the PoP key again. Therefore, the use of session resumption is NOT RECOMMENDED for resource servers.

Since renegotiation of DTLS associations is prone to attacks as well, [RFC7925] requires clients to decline any renogiation attempt. A server that wants to initiate re-keying therefore SHOULD periodically force a full handshake.

7.2. Multiple Access Tokens

The use of multiple access tokens for a single client increases the strain on the resource server as it must consider every access token and calculate the actual permissions of the client. Also, tokens may contradict each other which may lead the server to enforce wrong permissions. If one of the access tokens expires earlier than others, the resulting permissions may offer insufficient protection. Developers SHOULD avoid using multiple access tokens for a client.

Even when a single access token per client is used, an attacker could compromise the dynamic update mechanism for existing DTLS connections by delaying or reordering packets destined for the authz-info endpoint. Thus, the order in which operations occur at the resource server (and thus which authorization info is used to process a given client request) cannot be guaranteed. Especially in the presence of later-issued access tokens that reduce the client's permissions from the initial access token, it is impossible to guarantee that the reduction in authorization will take effect prior to the expiration of the original token.

7.3. Out-of-Band Configuration

To communicate securely, the authorization server, the client and the resource server require certain information that must be exchanged outside the protocol flow described in this document. The authorization server must have obtained authorization information concerning the client and the resource server that is approved by the resource owner as well as corresponding keying material. The resource server must have received authorization information approved by the resource owner concerning its authorization managers and the respective keying material. The client must have obtained authorization information concerning the authorization server approved by its owner as well as the corresponding keying material. Also, the client's owner must have approved of the client's communication with the resource server. The client and the authorization server must have obtained a common understanding how this resource server is identified to ensure that the client obtains access token and keying material for the correct resource server. If the client is provided with a raw public key for the resource server, it must be ascertained to which resource server (which identifier and authorization information) the key is associated. All authorization information and keying material must be kept up to date.

8. Privacy Considerations

This privacy considerations from Section 7 of the [I-D.ietf-ace-oauth-authz] apply also to this profile.

An unprotected response to an unauthorized request may disclose information about the resource server and/or its existing relationship with the client. It is advisable to include as little information as possible in an unencrypted response. When a DTLS session between an authenticated client and the resource server already exists, more detailed information MAY be included with an error response to provide the client with sufficient information to react on that particular error.

Also, unprotected requests to the resource server may reveal information about the client, e.g., which resources the client attempts to request or the data that the client wants to provide to the resource server. The client SHOULD NOT send confidential data in an unprotected request.

Note that some information might still leak after DTLS session is established, due to observable message sizes, the source, and the destination addresses.

9. IANA Considerations

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

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

Profile name: coap_dtls

Profile Description: Profile for delegating client authentication and authorization in a constrained environment by establishing a Datagram Transport Layer Security (DTLS) channel between resource-constrained nodes.

Profile ID: TBD (suggested: 1)

Change Controller: IESG

Reference: [RFC-XXXX]

10. Acknowledgments

Thanks to Jim Schaad for his contributions and reviews of this document. Special thanks to Ben Kaduk for his thorough reviews of this document.

Ludwig Seitz worked on this document as part of the CelticNext projects CyberWI, and CRITISEC with funding from Vinnova.

11. References

11.1. Normative References

[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-35 (work in progress), June 2020.

[I-D.ietf-ace-oauth-params]
Seitz, L., "Additional OAuth Parameters for Authorization in Constrained Environments (ACE)", draft-ietf-ace-oauth-params-13 (work in progress), April 2020.

[I-D.ietf-cbor-7049bis]
Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", draft-ietf-cbor-7049bis-14 (work in progress), June 2020.

[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>.

[RFC4279] Eronen, P., Ed. and H. Tschofenig, Ed., "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)", RFC 4279, DOI 10.17487/RFC4279, December 2005, <https://www.rfc-editor.org/info/rfc4279>.

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

[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>.

[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>.

[RFC7251] McGrew, D., Bailey, D., Campagna, M., and R. Dugal, "AES-CCM Elliptic Curve Cryptography (ECC) Cipher Suites for TLS", RFC 7251, DOI 10.17487/RFC7251, June 2014, <https://www.rfc-editor.org/info/rfc7251>.

[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>.

[RFC8152] Schaad, J., "CBOR Object Signing and Encryption (COSE)", RFC 8152, DOI 10.17487/RFC8152, July 2017, <https://www.rfc-editor.org/info/rfc8152>.

[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>.

[RFC8422] Nir, Y., Josefsson, S., and M. Pegourie-Gonnard, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS) Versions 1.2 and Earlier", RFC 8422, DOI 10.17487/RFC8422, August 2018, <https://www.rfc-editor.org/info/rfc8422>.

[RFC8747] Jones, M., Seitz, L., Selander, G., Erdtman, S., and H. Tschofenig, "Proof-of-Possession Key Semantics for CBOR Web Tokens (CWTs)", RFC 8747, DOI 10.17487/RFC8747, March 2020, <https://www.rfc-editor.org/info/rfc8747>.

11.2. Informative References

[RFC5077] Salowey, J., Zhou, H., Eronen, P., and H. Tschofenig, "Transport Layer Security (TLS) Session Resumption without Server-Side State", RFC 5077, DOI 10.17487/RFC5077, January 2008, <https://www.rfc-editor.org/info/rfc5077>.

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

[RFC6655] McGrew, D. and D. Bailey, "AES-CCM Cipher Suites for Transport Layer Security (TLS)", RFC 6655, DOI 10.17487/RFC6655, July 2012, <https://www.rfc-editor.org/info/rfc6655>.

[RFC7662] Richer, J., Ed., "OAuth 2.0 Token Introspection", RFC 7662, DOI 10.17487/RFC7662, October 2015, <https://www.rfc-editor.org/info/rfc7662>.

[RFC7748] Langley, A., Hamburg, M., and S. Turner, "Elliptic Curves for Security", RFC 7748, DOI 10.17487/RFC7748, January 2016, <https://www.rfc-editor.org/info/rfc7748>.

[RFC8032] Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, DOI 10.17487/RFC8032, January 2017, <https://www.rfc-editor.org/info/rfc8032>.

[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>.

[RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <https://www.rfc-editor.org/info/rfc8610>.

[RFC8613] Selander, G., Mattsson, J., Palombini, F., and L. Seitz, "Object Security for Constrained RESTful Environments (OSCORE)", RFC 8613, DOI 10.17487/RFC8613, July 2019, <https://www.rfc-editor.org/info/rfc8613>.

Authors' Addresses

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

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

Olaf Bergmann
Universitaet Bremen TZI
Postfach 330440
Bremen D-28359
Germany

Phone: +49-421-218-63904
Email: bergmann@tzi.org

Carsten Bormann
Universitaet Bremen TZI
Postfach 330440
Bremen D-28359
Germany

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

Goeran Selander
Ericsson AB

Email: goran.selander@ericsson.com

Ludwig Seitz
Combitech
Djaeknegatan 31
Malmoe 211 35
Sweden

Email: ludwig.seitz@combitech.se