ACE Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 10, 2019

S. Gerdes
O. Bergmann
C. Bormann
Universitaet Bremen TZI
G. Selander
Ericsson
L. Seitz
RISE SICS

September 06, 2018

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

draft-ietf-ace-dtls-authorize-04

## Abstract

This specification defines a profile for delegating client authentication and authorization in a constrained environment by establishing a Datagram Transport Layer Security (DTLS) channel between resource-constrained nodes. The protocol relies on DTLS for communication security between entities in a constrained network using either raw public keys or pre-shared keys. A resource-constrained node can use this protocol to delegate management of authorization information to a trusted host with less severe limitations regarding processing power and memory.

Table of Contents



## 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 [RFC6347] to communicate. The client uses an access token, bound to a key (the proof-of-possession key) to authorize its access to protected resources hosted by the resource server. DTLS provides communication security, proof of possession, and server authentication. Optionally the client and the resource server may also use CoAP over DTLS to communicate with the authorization server. This specification supports the DTLS handshake

with Raw Public Keys (RPK) [RFC7250] and the DTLS handshake with Pre-Shared Keys (PSK) [RFC4279].

The DTLS RPK handshake [RFC7250] requires client authentication to provide proof-of-possession for the key tied to the access token. Here the access token needs to be transferred to the resource server before the handshake is initiated, as described in section 5.8.1 of draft-ietf-ace-oauth-authz [1].

The DTLS PSK handshake [RFC4279] provides the proof-of-possession for the key tied to the access token. Furthermore the psk_identity parameter in the DTLS PSK handshake is used to transfer the access token from the client to the resource server.

## 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].

## 2. Protocol Overview

The CoAP-DTLS profile for ACE specifies the transfer of authentication 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 a 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 RS.

This profile requires a Client (C) to retrieve an Access Token for the resource(s) it wants to access on a Resource Server (RS) 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):

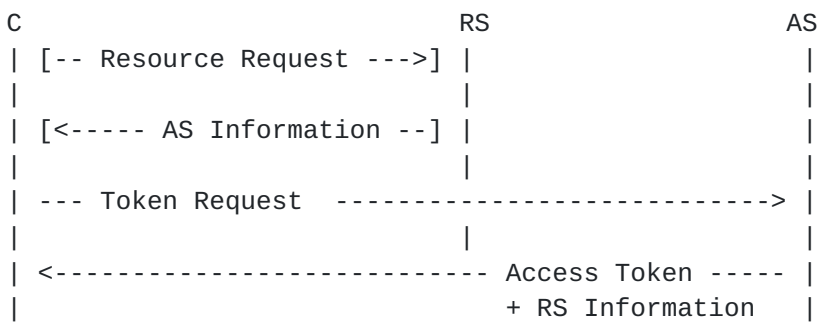


Figure 1: Retrieving an Access Token

To determine the AS in charge of a resource hosted at the RS, the client C MAY send an initial Unauthorized Resource Request message to the RS. The RS then denies the request and sends the address of its AS back to the client C as specified in section 5.1.2 of draft-ietf-ace-oauth-authz [2].

Once the client C knows the authorization server's address, it can send an Access Token request to the token endpoint at the AS 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. How the mutual authentication between the client and the authorization server is achieved is out of scope for this document; the client may have been configured with a public key of the authorization server and have been registered at the AS via the OAuth client registration mechanism as outlined in section 5.3 of draft-ietf-ace-oauth-authz [3].

If C wants to use the CoAP RawPublicKey mode as described in Section 9 of RFC 7252 [4] it MUST provide a key or key identifier within a "cnf" object in the token request. If the authorization server AS decides that the request is to be authorized it generates an access token response for the client C containing a "profile" parameter with the value "coap_dtls" to indicate that this profile MUST be used for communication between the client C and the resource server.

For RPK mode, the authorization server also adds a "rs_cnf" parameter containing information about the public that is used by the resource server (see Section 3).

For PSK mode, the authorization server adds a "cnf" parameter containing information about the shared secret that C can use to setup a DTLS session with the resource server (see Section 4).

The Access Token returned by the authorization server then can be used by the client to establish a new DTLS session with the resource server. When the client intends to use asymmetric cryptography in the DTLS handshake with the resource server, the client MUST upload the Access Token to the authz-info resource on the resource server before starting the DTLS handshake, as described in section 5.8.1 of draft-ietf-ace-oauth-authz [5]. If only symmetric cryptography is used between the client and the resource server, the Access Token MAY instead be transferred in the DTLS ClientKeyExchange message (see Section 4.1).

Figure 2 depicts the common protocol flow for the DTLS profile after the client C has retrieved the Access Token from the authorization server AS.


Figure 2: Protocol overview

The following sections specify how CoAP is used to interchange access-related data between the resource server 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.

Depending on the desired CoAP security mode, the Client-to-AS request, AS-to-Client response and DTLS session establishment carry slightly different information. Section 3 addresses the use of raw public keys while Section 4 defines how pre-shared keys are used in this profile.

## 2.1. Resource Access

Once a DTLS channel has been established as described in Section 3 and Section 4, 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.

On the resource server side, successful establishment of the DTLS channel binds the client to the access token, functioning as a proof-of-possession associated key. Any request that the resource server receives on this channel MUST be checked against these authorization rules that are associated with the identity of the client. 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 as described in Section 5.1.1 of draft-ietf-ace-oauth-authz [6].

Note: The identity of the client is determined by the authentication process
during the DTLS handshake. In the asymmetric case, the public key will define the client's identity, while in the PSK case, the client's identity is defined by the shared secret generated by the authorization server for this communication.

The resource server SHOULD treat 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 peer 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 MUST be rejected according to [Section 5.1.1 of draft-ietf-ace-oauth-authz](https://tools.ietf.org/html/draft-ietf-ace-oauth-authz-13#section-5.1.1

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 cannot always know a priori if an Authorized Resource Request will succeed. If the client repeatedly gets error responses

containing AS Information (cf. Section 5.1.1 of draft-ietf-ace-oauth-authz [7] 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.

## 2.2. Dynamic Update of Authorization Information

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

Figure 3 depicts the message flow where the client C requests a new Access Token after a security association between the client and the resource server RS has been established using this protocol. The token request MUST specify the key identifier of 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 ticket that has previously been issued to the requesting client. Otherwise, the Client-to-AS request MUST be declined with a the error code "unsupported_pop_key" as defined in Section 5.6.3 of draft-ietf-ace-oauth-authz [8].

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 associate the updated authorization information with any existing DTLS session that is identified by this key identifier.

Note: 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.

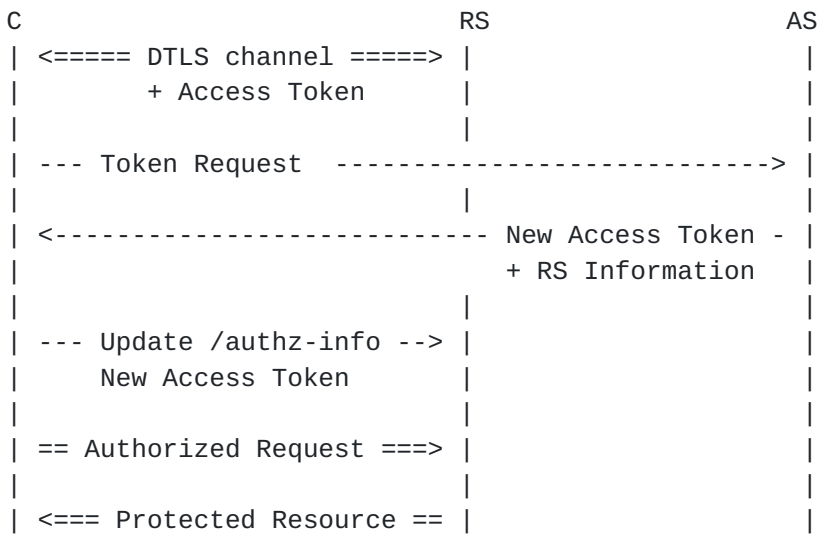


Figure 3: Overview of Dynamic Update Operation

## 2.3. Token Expiration

DTLS sessions that have been established in accordance with this profile are always tied to a specific set of access tokens. As these tokens may become invalid at any time (either because the token has expired or the responsible authorization server has revoked the token), the session may become useless at some point. A resource server therefore may decide to terminate existing DTLS sessions after the last valid access token for this session has been deleted.

As specified in section 5.8.3 of draft-ietf-ace-oauth-authz [9], the resource server MUST notify the client with an error response with code 4.01 (Unauthorized) for any long running request before terminating the session.

The resource server MAY also keep the session alive for some time and respond to incoming requests with a 4.01 (Unauthorized) error message including AS Information to signal that the client needs to upload a new access token before it can continue using this DTLS session. The AS Information is created as specified in section 5.1.2 of draft-ietf-ace-oauth-authz [10]. The resource server SHOULD add a "kid" parameter to the AS Information denoting the identifier of the key that it uses internally for this DTLS session. The client then includes this "kid" parameter in a Client-to-AS request used to retrieve a new access token to be used with this DTLS session. In case the key identifier is already known by the client (e.g. because it was included in the RS Information in an AS-to-Client response), the "kid" parameter MAY be elided from the AS Information.

Table 1 updates Figure 2 in section 5.1.2 of draft-ietf-ace-oauth-authz [11] with the new "kid" parameter in accordance with [RFC8152].

| Parameter name | CBOR Key | Major Type |
|---|---|---|
| kid | 4 | 2 (byte string) |

Table 1: Updated AS Information parameters

## 3. RawPublicKey Mode

To retrieve an access token for the resource that the client wants to access, the client requests an Access Token from the authorization server. The client MUST add a "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. To prove that the client is in possession of this key, it MUST use the same public key as in certificate message that is used to establish the DTLS session with the authorization server.

An example Access Token request from the client to the resource server is depicted in Figure 4.

```
POST coaps://as.example.com/token
Content-Format: application/cbor
{
  grant_type:    client_credentials,
  aud:           "tempSensor4711",
  cnf: {
    COSE_Key: {
      kty: EC2,
      crv: P-256,
      x:   h'TODOX',
      y:   h'TODOY'
    }
  }
}
```

Figure 4: Access Token Request Example for RPK Mode

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

When the authorization server authorizes a request, it will return an Access Token and a "cnf" object in the AS-to-Client response. Before

the client initiates the DTLS handshake with the resource server, it MUST send a "POST" request containing the new Access Token to the authz-info resource hosted by the resource server. If this operation yields a positive response, the client SHOULD proceed to establish a new DTLS channel with the resource server. To use raw public key mode, the client MUST pass the same public key that was used for constructing the Access Token with the SubjectPublicKeyInfo structure in the DTLS handshake as specified in [RFC7250].

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] with the ed25519 curve (cf. [RFC8032], [RFC8422]).

Note: According to [RFC7252], CoAP implementations MUST support the ciphersuite TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8 [RFC7251] and the NIST P-256 curve. As discussed in [RFC7748], new ECC curves have been defined recently that are considered superior to the so-called NIST curves. The curve that is mandatory to implement in this specification is said to be efficient and less dangerous regarding implementation errors than the secp256r1 curve mandated in [RFC7252].

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. If CBOR web tokens [RFC8392] are used as recommended in [I-D.ietf-ace-oauth-authz], the authorization server MUST include a "COSE_Key" object in the "cnf" claim of the Access Token. This "COSE_Key" object MAY contain a reference to a key for the client that is already known by the resource server (e.g., from previous communication). 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.

## 4. PreSharedKey Mode

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 and therefore MUST specify a symmetric key that was previously generated by the authorization server as a shared secret for the communication between the client and the resource server.

Depending on the requested token type and algorithm in the Access Token request, the authorization server adds RS Information to the response that provides the client with sufficient information to

setup a DTLS channel with the resource server. For symmetric proof-of-possession keys (c.f. [I-D.ietf-ace-oauth-authz]), the client must ensure that the Access Token request is sent over a secure channel that guarantees authentication, message integrity and confidentiality.

When the authorization server authorizes the client it returns an AS-to-Client response with the profile parameter set to "coap_dtls" and a "cnf" parameter carrying a "COSE_Key" object that contains the symmetric key to be used between the client and the resource server as illustrated in Figure 5.

```
2.01 Created
Content-Format: application/cbor
Location-Path: /token/asdjbaskd
{
   access_token: h'd08343a10...
   (remainder of CWT omitted for brevity)
   token_type:   pop,
   alg:          HS256,
   expires_in:   86400,
   profile:      coap_dtls,
   cnf: {
     COSE_Key: {
       kty: symmetric,
       k: h'73657373696f6e6b6579'
     }
   }
}
```

Figure 5: Example Access Token response

In this example, the authorization server returns a 2.01 response containing a new Access Token. The information is transferred as a CBOR data structure as specified in [I-D.ietf-ace-oauth-authz].

A response that declines any operation on the requested resource is constructed according to Section 5.2 of RFC 6749 [12], (cf. Section 5.7.3 of [I-D.ietf-ace-oauth-authz]).

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

Figure 6: Example Access Token response with reject

### 4.1. DTLS Channel Setup Between C and RS

When a client receives an Access Token from an authorization server, it checks if the payload contains an "access_token" parameter and a "cnf" parameter. With this information the client can initiate 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.

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. Alternatively, the client MAY 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 and not perform any re-coding.

Note: As stated in section 4.2 of [RFC7925], the PSK identity should be treated as binary data in the Internet of Things space and not assumed to have a human-readable form of any sort.

If a resource server receives a ClientKeyExchange message that contains a "psk_identity" with a length greater zero, it uses the contents as index for its key store (i.e., treat the contents as key identifier). The resource server MUST check if it has one or more Access Tokens that are associated with the specified key. If no valid Access Token is available for this key, the DTLS session setup is terminated with an "illegal_parameter" DTLS alert message.

If no key with a matching identifier is found the resource server the resource server MAY process the decoded contents of the "psk_identity" field as access token that is stored with the authorization information endpoint before continuing the DTLS handshake. If the decoded contents of the "psk_identity" do not yield a valid access token for the requesting client, the DTLS session setup is terminated with an "illegal_parameter" DTLS alert message.

Note1: As a resource server cannot provide a client with a meaningful PSK identity hint in
response to the client's ClientHello message, the resource server SHOULD NOT send a ServerKeyExchange message.

Note2: According to [RFC7252], CoAP implementations MUST support the ciphersuite TLS_PSK_WITH_AES_128_CCM_8 [RFC6655]. A client is therefore expected to offer at least this ciphersuite to the resource server.

This specification assumes that the Access Token is a PoP token as described in [I-D.ietf-ace-oauth-authz] unless specifically stated otherwise. Therefore, the Access Token is bound to a symmetric PoP key that is used as shared secret between the client and the resource server.

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. Usually, this is done by including a "COSE_Key" object carrying either a key that has been encrypted with a shared secret between the authorization server and the resource server, or a key identifier that can be used by the resource server to lookup the shared secret.

Instead of the "COSE_Key" object, the authorization server MAY include a "COSE_Encrypt" structure to enable the resource server to calculate the shared key from the Access Token. The "COSE_Encrypt" structure MUST use the _Direct Key with KDF_ method as described in Section 12.1.2 of RFC 8152 [13]. The authorization server MUST include a Context information structure carrying a PartyU "nonce" parameter carrying the nonce that has been used by the authorization server to construct the shared key.

This specification mandates that at least the key derivation algorithm "HKDF SHA-256" as defined in [RFC8152] MUST be supported. This key derivation function is the default when no "alg" field is included in the "COSE_Encrypt" structure for the resource server.

## 4.2. Updating Authorization Information

Usually, the authorization information that the resource server keeps for a client is updated by uploading a new Access Token as described in Section 2.2.

The Client MAY also perform a new DTLS handshake according to Section 4.1 that replaces the existing DTLS session. After successful completion of the DTLS handshake the resource server updates the existing authorization information for the client according to the new Access Token.

## 5. 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 and privacy considerations from section 6 and section 7 also apply to this profile.

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.

[I-D.tiloca-tls-dos-handshake] specifies a TLS extension to prevent this type of attack which is applicable especially for constrained environments where the authorization server can act as trust anchor.

## 6. Privacy Considerations

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 the 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.

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

## 7. 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: 1

Change Controller: IESG

Reference: [RFC-XXXX]

## 8. References

### 8.1. 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-13 (work in progress), July 2018.

[I-D.tiloca-tls-dos-handshake]
Tiloca, M., Seitz, L., Hoeve, M., and O. Bergmann, "Extension for protecting (D)TLS handshakes against Denial of Service", draft-tiloca-tls-dos-handshake-02 (work in progress), March 2018.

[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>.

[RFC5746] Rescorla, E., Ray, M., Dispensa, S., and N. Oskov, "Transport Layer Security (TLS) Renegotiation Indication Extension", RFC 5746, DOI 10.17487/RFC5746, February 2010, <https://www.rfc-editor.org/info/rfc5746>.

[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>.

[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>.

## 8.2. Informative References

[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>.

[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>.

[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>.

[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>.

### 8.3. URIs

[1] https://tools.ietf.org/html/draft-ietf-ace-oauth-authz-13#section-5.8.1

[2] https://tools.ietf.org/html/draft-ietf-ace-oauth-authz-13#section-5.1.2

[3] https://tools.ietf.org/html/draft-ietf-ace-oauth-authz-13#section-5.3

[4] https://tools.ietf.org/html/rfc7252#section-9

[5] https://tools.ietf.org/html/draft-ietf-ace-oauth-authz-13#section-5.8.1

[6] https://tools.ietf.org/html/draft-ietf-ace-oauth-authz-13#section-5.1.1

[7] https://tools.ietf.org/html/draft-ietf-ace-oauth-authz-13#section-5.1.1

[8] https://tools.ietf.org/html/draft-ietf-ace-oauth-authz-13#section-5.6.3

[9] https://tools.ietf.org/html/draft-ietf-ace-oauth-authz-13#section-5.8.3

[10] https://tools.ietf.org/html/draft-ietf-ace-oauth-authz-13#section-5.1.2

[11] https://tools.ietf.org/html/draft-ietf-ace-oauth-authz-13#section-5.1.2

[12] https://tools.ietf.org/html/rfc6749#section-5.2

[13] https://tools.ietf.org/html/rfc8152#section-12.1.2

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
Faroegatan 6
Kista 164 80
Sweden

Email: goran.selander@ericsson.com

Ludwig Seitz
RISE SICS
Scheelevaegen 17
Lund 223 70
Sweden

Email: ludwig.seitz@ri.se