ACE Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 4, 2017

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

October 31, 2016

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

draft-gerdes-ace-dtls-authorize-00

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. 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.

Status of This Memo

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

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

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

This Internet-Draft will expire on May 4, 2017.

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

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.

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 8.1 of draft-ietf-ace-oauth-authz. [1]

Note: While the scope of this draft is on client and resource server

communicating using CoAP over DTLS, it is expected that it applies also to CoAP over TLS, possibly with minor modifications. However, that is out of scope for this version of the draft.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

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

2. Protocol Overview

The CoAP-DTLS profile for ACE specifies the transfer of authentication and, if necessary, authorization information between C and 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 AS for a protected resource hosted on 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):

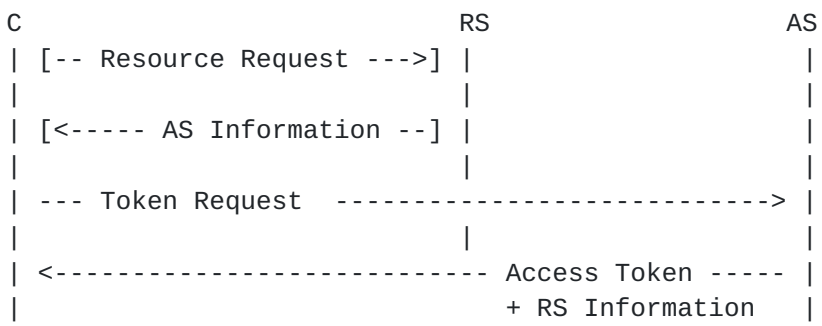

Figure 1: Retrieving an Access Token

To determine the AS in charge of a resource hosted at the RS, C MAY send an initial Unauthorized Resource Request message to RS. RS then denies the request and sends the address of its AS back to C.

Instead of the initial Unauthorized Resource Request message, C MAY look up the desired resource in a resource directory (cf. [I-D.ietf-core-resource-directory]).

Once C knows AS'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]. If C wants to use the CoAP RawPublicKey mode as described in Section 9 of RFC 7252 [2] it MUST provide a key or key identifier within a "cnf" object in the token request. If AS decides that the request is to be authorized it generates an access token response for C containing a "profile" parameter with the value "coap_dtls" to indicate that this profile MUST be used for communication between C and RS. Is also adds a "cnf" parameter with additional data for the establishment of a secure DTLS channel between C and RS. The semantics of the 'cnf' parameter depend on the type of key used between C and RS, see Section 3 and Section 4.

The Access Token returned by AS then can be used by C to establish a new DTLS session with RS. When C intends to use asymmetric cryptography in the DTLS handshake with RS, C MUST upload the Access Token to the "/authz-info" resource on RS before starting the DTLS handshake, as described in section 8.1 of draft-ietf-ace-oauth-authz [3]. If only symmetric cryptography is used between C and RS, 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 C has retrieved the Access Token from AS.

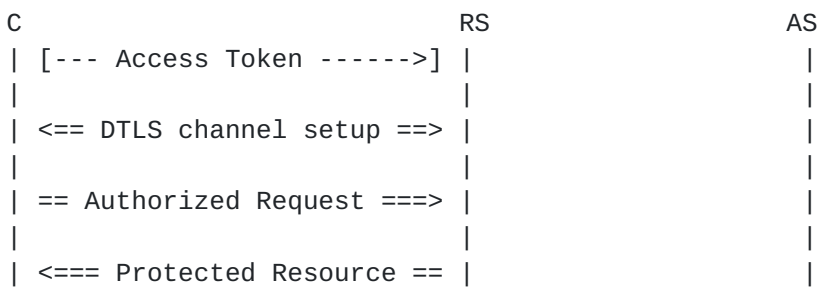

Figure 2: Protocol overview

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

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. Unauthorized Resource Request Message

The optional Unauthorized Resource Request message is a request for a resource hosted by RS for which no proper authorization is granted. RS MUST treat any CoAP request for a resource other than "/authz-info" as Unauthorized Resource Request message when any of the following holds:

- The request has been received on an unprotected channel.
- RS has no valid access token for the sender of the request regarding the requested action on that resource.
- RS has a valid access token for the sender of the request, but this does not allow the requested action on the requested resource.

Note: These conditions ensure that RS can handle requests autonomously once access was granted and a secure channel has been established between C and RS. The resource "/authz-info" is publicly accessible to be able to upload new access tokens to RS (cf. [I-D.ietf-ace-oauth-authz]).

Unauthorized Resource Request messages MUST be denied with a client error response. In this response, the Resource Server SHOULD provide proper AS Information to enable the Client to request an access token from RS's Authorization Server as described in Section 2.2.

The response code MUST be 4.01 (Unauthorized) in case the sender of the Unauthorized Resource Request message is not authenticated, or if RS has no valid access token for C. If RS has an access token for C but not for the resource that C has requested, RS MUST reject the request with a 4.03 (Forbidden). If RS has an access token for C but it does not cover the action C requested on the resource, RS MUST reject the request with a 4.05 (Method Not Allowed).

Note: The use of the response codes 4.03 and 4.05 is intended to prevent infinite loops where a dumb Client optimistically tries to access a requested resource with any access token received from AS. As malicious clients could pretend to be C to determine C's privileges, these detailed response codes must be used only when a certain level of security is already available which can be achieved only when the Client is authenticated.

2.2. AS Information

The AS Information is sent by RS as a response to an Unauthorized Resource Request message (see Section 2.1) to point the sender of the Unauthorized Resource Request message to RS's AS. The AS information is a set of attributes containing an absolute URI (see Section 4.3 of [RFC3986]) that specifies the AS in charge of RS.

TBD: We might not want to add more parameters in the AS information because
this would not only reveal too much information about RS's capabilities to unauthorized peers but also be of little value as C cannot really trust that information anyway.

The message MAY also contain a nonce generated by RS to ensure freshness in case that the RS and AS do not have synchronized clocks.

Figure 3 shows an example for an AS Information message payload using CBOR [RFC7049] diagnostic notation.

```
4.01 Unauthorized
Content-Format: application/ace+cbor
{AS: "coaps://as.example.com/token",
 nonce: h'e0a156bb3f'}
```

Figure 3: AS Information payload example

In this example, the attribute AS points the receiver of this message to the URI "coaps://as.example.com/token" to request access permissions. The originator of the AS Information payload (i.e., RS) uses a local clock that is loosely synchronized with a time scale common between RS and AS (e.g., wall clock time). Therefore, it has included a parameter "nonce" for replay attack prevention (c.f. Section 5.2).

Note: There is an ongoing discussion how freshness of access tokens can be achieved in constrained environments. This specification for now assumes that RS and AS do not have a common understanding of time that allows RS to achieve its security objectives without explicitly adding a nonce.

The examples in this document are written in CBOR diagnostic notation to improve readability. Figure 4 illustrates the binary encoding of the message payload shown in Figure 3.

```
a2                                  # map(2)
    00                              # unsigned(0) (=AS)
    78 1c                           # text(28)
       636f6170733a2f2f61732e657861
       6d706c652e636f6d2f746f6b656e  # "coaps://as.example.com/token"
    05                              # unsigned(5) (=nonce)
    45                              # bytes(5)
       e0a156bb3f
```

Figure 4: AS Information example encoded in CBOR

2.3. Resource Access

Once a DTLS channel has been established as described in Section 3 and Section 4, respectively, C is authorized to access resources covered by the Access Token it has uploaded to the "/authz-info" resource hosted by RS.

On the server side (i.e., RS), successful establishment of the DTLS channel binds C to the access token, functioning as a proof-of-possession associated key. Any request that RS receives on this channel MUST be checked against these authorization rules that are associated with the identity of C. Incoming CoAP requests that are not authorized with respect to any Access Token that is associated with C MUST be rejected by RS with 4.01 response as described in Section 2.1.

Note: The identity of C is determined by the authentication process

during the DTLS handshake. In the asymmetric case, the public key will define C's identity, while in the PSK case, C's identity is defined by the session key generated by AS for this communication.

RS 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 RS.

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

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.

C cannot always know a priori if a Authorized Resource Request will succeed. If C repeatedly gets AS Information messages (cf. Section 2.2) as response to its requests, it SHOULD request a new Access Token from AS in order to continue communication with RS.

2.4. Dynamic Update of Authorization Information

The Client can update the authorization information stored at RS at any time. To do so, the Client requests from AS a new Access Token for the intended action on the respective resource and uploads this Access Token to the "/authz-info" resource on RS.

Figure 5 depicts the message flow where C requests a new Access Token after a security association between C and RS has been established using this protocol.

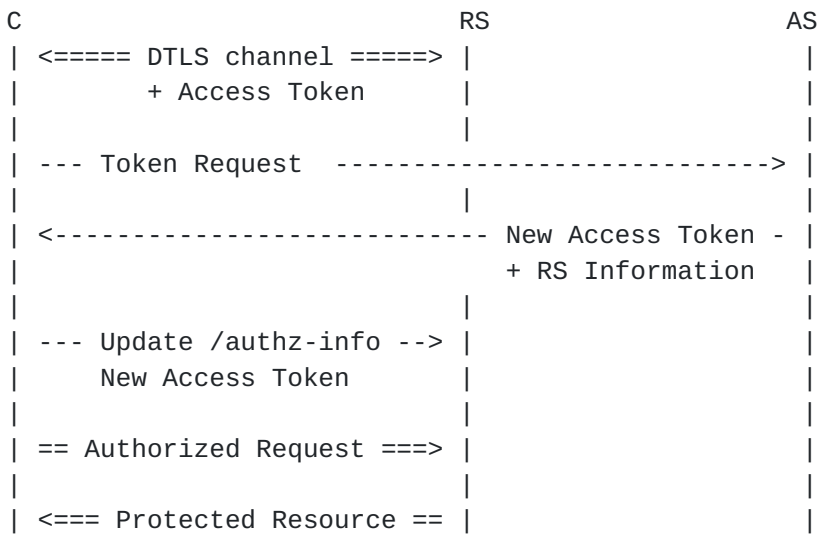

Figure 5: Overview of Dynamic Update Operation

3. RawPublicKey Mode

To retrieve an access token for the resource that C wants to access, C requests an Access Token from AS. C 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 AS.

An example Access Token request from C to RS is depicted in Figure 6.

```
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 6: 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 AS using a raw public key.

When AS authorizes a request, it will return an Access Token and a "cnf" object in the AS-to-Client response. Before C initiates the DTLS handshake with RS, it MUST send a "POST" request containing the new Access Token to the "/authz-info" resource hosted by RS. If this operation yields a positive response, C SHOULD proceed to establish a new DTLS channel with RS. To use raw public key mode, C 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].

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. C is therefore expected to offer at least this ciphersuite to RS.

The Access Token is constructed by AS such that RS can associate the Access Token with the Client's public key. If CBOR web tokens [I-D.ietf-ace-cbor-web-token] are used as recommended in [I-D.ietf-ace-oauth-authz], the AS 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 C that is already known by RS (e.g., from previous communication). If the AS has no certain knowledge that the Client's key is already known to RS, 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 C wants to access, C MAY include a "cnf" object carrying an identifier for a symmetric key in its Access Token request to AS. This identifier can be used by AS to determine the session key to construct the proof-of-possession token and therefore MUST specify a symmetric key that was previously generated by AS as a session key for the communication between C and RS.

Depending on the requested token type and algorithm in the Access Token request, AS adds RS Information to the response that provides C with sufficient information to setup a DTLS channel with RS. For symmetric proof-of-possession keys (c.f. [I-D.ietf-ace-oauth-authz]), C must ensure that the Access Token request is sent over a secure channel that guarantees authentication, message integrity and confidentiality. This could be, e.g., a DTLS channel (for "coaps") or an OSCOAP [I-D.ietf-core-object-security] exchange (for "coap").

When AS authorizes C 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 session key to be used between C and RS as illustrated in Figure 7.

```
2.01 Created
Content-Format: application/cbor
Location-Path: /token/asdjbaskd
Max-Age: 86400
{
   access_token: b64'SlAV32hkKG ...
   (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 7: Example Access Token response

In this example, AS 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]. The Max-Age option tells the receiving Client how long this token will be valid.

A response that declines any operation on the requested resource is constructed according to Section 5.2 of RFC 6749 [4], (cf. Section 6.3 of [I-D.ietf-ace-oauth-authz]).

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

Figure 8: Example Access Token response with reject

4.1. DTLS Channel Setup Between C and RS

When C receives an Access Token from AS, it checks if the payload contains an "access_token" parameter and a "cnf" parameter. With this information C can initiate establishment of a new DTLS channel with RS. To use DTLS with pre-shared keys, C 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 session key by C and RS is used for mutual authentication between both peers. Therefore, RS must be able to determine the session key from the Access Token. Following the general ACE authorization framework, C can upload the Access Token to RS's "/authz-info" resource before starting the DTLS handshake. Alternatively, C MAY provide the most recent base64-encoded Access Token in the "psk_identity" field of the ClientKeyExchange message.

If RS receives a ClientKeyExchange message that contains a "psk_identity" with a length greater zero, it MUST base64-decode its contents and process it in the same way as an Access Token that has been uploaded to its "/authz-info" resource. If the contents of the "psk_identity" contained a syntactically valid Access Token, RS continues processing the ClientKeyExchange message. Otherwise, the DTLS session setup is terminated with an "illegal_parameter" DTLS alert message.

Note1: As RS cannot provide C with a meaningful PSK identity hint in

response to C's ClientHello message, RS SHOULD NOT send a ServerKeyExchange message.

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

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 session key between C and RS.

While C can retrieve the session key from the contents of the "cnf" parameter in the AS-to-Client response, RS uses the information contained in the "cnf" claim of the Access Token to determine the actual session key. Usually, this is done by including a "COSE_Key" object carrying either a key that has been encrypted with a shared secret between AS and RS, or a key identifier that can be used by RS to lookup the session key.

Instead of the "COSE_Key" object, AS MAY include a "COSE_Encrypt" structure to enable RS to calculate the session 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 draft-ietf-cose-msg [5]. The AS MUST include a Context information structure carrying a

PartyU "nonce" parameter carrying the nonce that has been used by AS to construct the session key.

This specification mandates that at least the key derivation algorithm "HKDF SHA-256" as defined in [I-D.ietf-cose-msg] MUST be supported. This key derivation function is the default when no "alg" field is included in the "COSE_Encrypt" structure for RS.

4.2. Updating Authorization Information

Usually, the authorization information that RS keeps for C is updated by uploading a new Access Token as described in Section 2.4.

If the security association with RS still exists and RS has indicated support for session renegotiation according to [RFC5746], the new Access Token MAY be used to renegotiate the existing DTLS session. In this case, the Access Token is used as "psk_identity" as defined in Section 4.1. 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 RS updates the existing authorization information for C according to the new Access Token.

5. Security Considerations

TODO

5.1. Unprotected AS Information

Initially, no secure channel exists to protect the communication between C and RS. Thus, C cannot determine if the AS information contained in an unprotected response from RS to an unauthorized request (c.f. Section 2.2) is authentic. It is therefore advisable to provide C with a (possibly hard-coded) list of trustworthy authorization servers. AS information responses referring to a URI not listed there would be ignored.

5.2. Use of Nonces for Replay Protection

RS may add a nonce to the AS Information message sent as a response to an unauthorized request to ensure freshness of an Access Token subsequently presented to RS. While a timestamp of some granularity would be sufficient to protect against replay attacks, using randomized nonce is preferred to prevent disclosure of information about RS's internal clock characteristics.

5.3. Privacy

An unprotected response to an unauthorized request (c.f. Section 2.2) may disclose information about RS and/or its existing relationship with C. It is advisable to include as little information as possible in an unencrypted response. When a DTLS session between C and RS already exists, more detailed information may be included with an error response to provide C with sufficient information to react on that particular error.

6. IANA Considerations

This document has no actions for IANA.

7. References

7.1. 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)", draft-ietf-ace-oauth-authz-04 (work in progress), October 2016.

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

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <http://www.rfc-editor.org/info/rfc3986>.

[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, <http://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, <http://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, <http://www.rfc-editor.org/info/rfc6347>.

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

7.2. Informative References

[I-D.ietf-ace-cbor-web-token]
Wahlstroem, E., Jones, M., Tschofenig, H., and S. Erdtman, "CBOR Web Token (CWT)", draft-ietf-ace-cbor-web-token-01 (work in progress), July 2016.

[I-D.ietf-core-object-security]
Selander, G., Mattsson, J., Palombini, F., and L. Seitz, "Object Security of CoAP (OSCOAP)", draft-ietf-core-object-security-00 (work in progress), October 2016.

[I-D.ietf-core-resource-directory]
Shelby, Z., Koster, M., Bormann, C., and P. Stok, "CoRE Resource Directory", draft-ietf-core-resource-directory-09 (work in progress), October 2016.

[I-D.ietf-cose-msg]
Schaad, J., "CBOR Object Signing and Encryption (COSE)", draft-ietf-cose-msg-23 (work in progress), October 2016.

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

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <http://www.rfc-editor.org/info/rfc7049>.

[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, <http://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, <http://www.rfc-editor.org/info/rfc7251>.

7.3. URIs

[1] https://tools.ietf.org/html/draft-ietf-ace-oauth-authz-03#section-8.1

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

[3] https://tools.ietf.org/html/draft-ietf-ace-oauth-authz-03#section-8.1

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

[5] https://tools.ietf.org/html/draft-ietf-cose-msg-23#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
SICS Swedish ICT AB
Scheelevaegen 17
Lund 223 70
Sweden

Email: ludwig@sics.se