ACE Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 5, 2015

S. Gerdes
O. Bergmann
C. Bormann
Universitaet Bremen TZI
July 04, 2014

Delegated CoAP Authentication and Authorization Framework (DCAF) draft-gerdes-ace-dcaf-authorize-00

Abstract

This specification defines a protocol for delegating client authentication and authorization in a constrained environment for establishing a Datagram Transport Layer Security (DTLS) channel between resource-constrained nodes. The protocol relies on DTLS to transfer authorization information and shared secrets for symmetric cryptography between entities in a constrained network. A resource-constrained node can use this protocol to delegate authentication of communication peers and management of authorization information to a trusted host with less severe limitations regarding processing power and memory.

Table of Contents

1. Introduction

The Constrained Application Protocol (CoAP) [RFC7252] is a transfer protocol similar to HTTP which is designed for the special requirements of constrained environments. A serious problem with constrained devices is the realization of secure communication. The devices only have limited resources such as memory, stable storage (such as disk space) and transmission capacity and often lack input/output devices such as keyboards or displays. Therefore, they are not readily capable of using common protocols. Especially authentication mechanisms are difficult to realize, because the lack of stable storage severely limits the number of keys the system can store. Moreover, CoAP has no mechanism to distinguish access rights for different clients (authorization).

The DCAF architecture is designed to relieve the constrained nodes from managing keys for numerous devices by introducing authorization servers which conduct the authentication and authorization for their nodes. To achieve this, access tokens are used. A device which wants to access a constrained node's resource first has to gain permission in the form of a token from the node's authorization server.

As fine-grained authorization is not always needed on constrained devices, DCAF supports an implicit authorization mode where no authorization information is exchanged.

The main goals of DCAF are the setup of a Datagram Transport Layer Security (DTLS) [RFC6347] channel with symmetric pre-shared keys (PSK) [RFC4279] and to securely transmit authorization tickets.

1.1. Features

- Utilize DTLS communication with pre-shared keys.
- Authenticated exchange of authorization information.
- Simplified authentication on constrained nodes by handing the more sophisticated authentication over to less-constrained devices.
- Simplified authorization mechanism for cases where implicit authorization is sufficient.
- Using only symmetric encryption on constrained nodes.

1.2. 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].

1.2.1. Roles

Resource Server (RS): A constrained device that hosts resources the Client wants to access.

Client (C): A device that wants to access resources on the Resource Server.

Authorization Server (AS): The node that conducts authentication and authorization for a Resource Server. An Authorization Server can be responsible for a single or multiple devices or even for a whole network. A Resource Server can have multiple Authorization Servers.

Authentication Manager (AM): The node that conducts authentication on behalf of the Client.

Resource Owner: The principal that owns the resource and controls its access permissions.

1.2.2. Other Terms

Access ticket: Contains the authentication and, if necessary, the authorization information needed to access a resource. A Ticket consists of the Ticket Face and the Ticket Verifier

Authorization information: Contains all information needed by RS to decide if C is privileged to access a resource in a specific way.

Authentication information: Contains all information needed by RS to decide if the entity in possession of a certain key is verified by the authorization server.

Access information: Contains authentication information and, if necessary, authorization information.

Ticket Face: The part of the ticket which is generated for the Resource Server. It contains the authorization information and all information needed by the Resource Server to verify that it was granted by AS.

Ticket Verifier: The part of the ticket which is generated for the Client. It enables the client to verify that it is communicating with an appropriate RS.

Explicit authorization: The Authorization Server informs the Resource Server in detail which privileges are granted to the Client.

Implicit authorization: The Authorization Server informs the Resource Server that the Client is authorized to access any resource on RS in any way, without specifying the privileges in detail.

2. System Overview

Within the DCAF Architecture each Resource Server (RS) has one or more Authorization Servers (AS) which conduct the authentication and authorization for RS. RS and AS share a symmetric key which has to be exchanged initially to provide for a secure channel. The mechanism used for this is not in the scope of this document.

To gain access to a specific resource on a Resource Server, a client (C) has to request an access ticket from one of the Authorization Servers serving RS either directly or, if it is a constrained device, using its Authentication Manager (AM). In the following, we always discuss the AM role separately, even if that is co-located within a (more powerful) C.

If AS decides that C is allowed to access the resource, it generates a DTLS pre-shared key (PSK) for the communication between C and RS and wraps it into an access ticket. For explicit access control, AS adds the detailed access permissions to the ticket in a way that RS can interpret. After presenting the ticket to RS, C and RS can communicate securely.

To be able to provide for the authentication and authorization services, the Authorization Servers have to fulfill several requirements:

- An AS must have enough stable storage (such as disk space) to store the necessary number of credentials (matching the number of clients and Resource Servers).

- An AS must possess means for user interaction, for example directly or indirectly connected input/output devices like keyboard and display, to allow for configuration of authorization information by the Resource Owner.

- An AS must have enough processing power to handle the authorization requests for all RS devices it is responsible for.

3. Protocol

The DCAF protocol comprises three parts:

1. transfer of authentication and, if necessary, authorization information between C and RS;

2. transfer of access requests and the respective ticket grants between C and AM; and

3. transfer of access requests and the respective ticket grants between AS and AM.

3.1. Overview

In Figure 1, a DCAF protocol flow is depicted (messages in square brackets are optional):

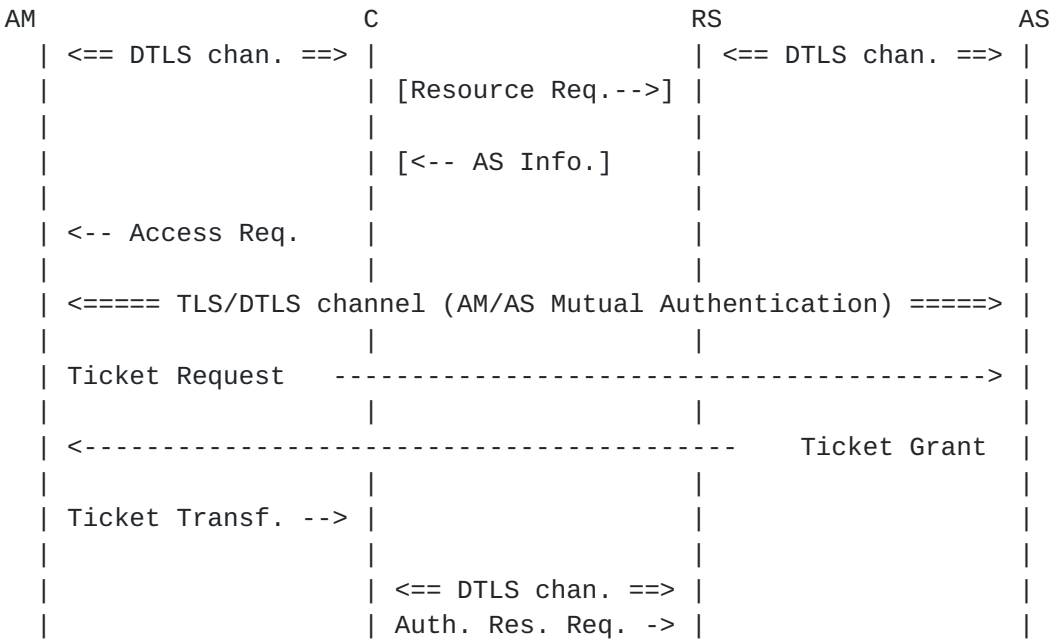

Figure 1: Protocol Overview

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

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]) that lists RS's resources as discussed in Section 9.

Once C knows AS' address, it can send a request for authorization to AS using its own Authentication Manager (AM). AS authenticates AM, who serves as a trusted introducer for C, and decides if C is allowed to communicate with RS and access the requested resource. If it is, AS generates an access ticket for C. The ticket contains keying material for the establishment of a secure channel and, if necessary, a representation of the permissions C has for the resource. C keeps one part of the access ticket and presents the other part to RS to prove its right to access. With their respective parts of the ticket, C and RS are able to establish a secure channel.

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 simultaneously convey authorization information specific for this communication relationship to RS.

Note: Special implementation considerations apply when one single entity takes the role of more than one actors. Section 11 gives additional advice on some of these usage scenarios.

This document uses Concise Binary Object Representation (CBOR, [RFC7049]) to express authorization information as set of attributes passed in CoAP payloads. Notation and encoding options are discussed in Section 5.

3.2. 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 as Unauthorized Resource Request message when any of the following holds:

- The request has been received on an insecure channel.

- RS has no valid access information for the sender of the request regarding the requested action on that resource.

- RS has valid access information 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.

Unauthorized Resource Request messages MUST be denied with a client error response. In this response, the Resource Server MUST provide proper AS Information to enable the Client to request an access ticket from RS's Authorization Server as described in Section 3.3.

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 ticket for C. If RS has authorization information for C but not for the resource that C has requested, RS MUST reject the request with a 4.03 (Forbidden). If RS has authorization information for C but they do 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 the 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.

3.3. AS Information Message

The AS Information Message is sent by RS as a response to an Unauthorized Resource Request message (see Section 3.2) to point the sender of the Unauthorized Resource Request message to RS's Authorization Server. The AS information is a set of attributes containing an absolute URI (see Section 4.3 of [RFC3986]) that specifies the Authorization Server in charge of RS.

The message MAY also contain a timestamp generated by RS.

Figure 2 shows an example for an AS Information message payload using CBOR diagnostic notation. (Refer to Section 5 for a detailed description of the available attributes and their semantics.)

```
4.01 Unauthorized
Content-Format: application/dcaf+cbor
{AS: "coaps://as-rs.example.com/authorize", TS: 168537}
```

Figure 2: AS Information Payload Example

In this example, the attribute AS points the receiver of this message to the URI "coaps://as-rs.example.com/authorize" 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 time stamp on its own time scale that is used as a nonce for replay attack prevention. Refer to Section 4.1 for more details concerning the usage of time stamps to ensure freshness of access tickets.

The examples in this document are written in CBOR diagnostic notation to improve readability. Figure 3 illustrates the binary encoding of the message payload shown in Figure 2.

```
a2                                      # map(2)
      00                                   # unsigned(0) (=AS)
      78 23                                # text(35)
         636f6170733a2f2f61732d72732e6578
         616d706c652e636f6d2f617574686f72
         697a65            # "coaps://as-rs.example.com/authorize"
      05                                   # unsigned(5) (=TS)
      1a 00029259                          # unsigned(168537)
```

Figure 3: AS Information Payload Example encoded in CBOR

3.4. Access Request

To retrieve an access ticket for the resource that C wants to access, C sends an Access Request to its authentication manager AM. The Access Request is constructed as follows:

1. The request method is POST.

2. The request URI is set as described below.

3. The message payload contains a data structure that describes the action and resource for which C requests an access ticket.

The request URI identifies a resource at AM for handling authorization requests from C. The URI SHOULD be announced by AM in its resource directory as described in Section 9.

Note: Where capacity limitations of C do not allow for resource directory lookups, the request URI in Access Requests could be hard-coded during provisioning or set in a specific device configuration profile.

The message payload is constructed from the AS information that RS has returned in its AS Information message (see Section 3.3) and information that C provides to describe its intended request(s). The Access Request MUST contain the following attributes:

1. Contact information for the AS to use.

2. An absolute URI of the resource that C wants to access.

3. The actions that C wants to perform on the resource.

4. Any time stamp generated by RS.

An example Access Request from C to AM is depicted in Figure 4. (Refer to Section 5 for a detailed description of the available attributes and their semantics.)

```
POST client-authorize
Content-Format: application/dcaf+cbor
{
  AS: "coaps://as-rs.example.com/authorize",
  AI: ["coaps://temp451.example.com/s/tempC", 5],
  TS: 168537
}
```

Figure 4: Access Request Message Example

The example shows an Access Request message payload for the resource "/s/tempC" on the Resource Server "temp451.example.com". Requested operations in attribute AR are GET and PUT.

The attributes AS (that denotes the Authorization Server to use) and TS (a nonce generated by RS) are taken from the AS Information message from RS.

The response to an Authorization Request is delivered by AM back to C in a Ticket Transfer message.

3.5. Ticket Request Message

When AM receives an Access Request message from C it MAY return a cached response if it is known to be fresh. Otherwise, it checks whether the request payload is of type "application/dcaf+cbor and contains at least the fields AS and AI. AM MUST respond with 4.00 (Bad Request) if the type is "application/dcaf+cbor and any of these fields is missing or does not conform to the format described in Section 5. Content formats other than application/dcaf+cbor are out of scope of this specification.

When the payload is correct, AM creates a Ticket Request message from the Access Request received from C as follows:

1. The destination of the Ticket Request message is derived from the authority information in the URI contained in field "AS" of the Access Request message payload.

2. The request method is POST.

3. The request URI is constructed from the AS field received in the Access Request message payload.

4. The payload is copied from the Access Request sent by C.

5. A label that describes the Client is added to the payload

To send the Ticket Request message to AS a secure channel between AM and AS MUST be used. Depending on the URI scheme used in the AS field of the Access Request message payload (the less-constrained devices AM and AS do not necessarily use coap to communicate with each other), this could be, e.g., a DTLS channel (for "coaps") or a TLS connection (for "https"). AM and AS MUST be able to mutually authenticate each other, e.g. based on a public key infrastructure. (Refer to Section 8 for a detailed discussion of the trust relationship between authentication managers and authorization servers.)

The descriptive label of C included in the Ticket Request is used to distinguish the clients within AS's namespace and MUST NOT be used for authenticating the client.

3.6. Ticket Grant Message

When AS has received a Ticket Request message it has to evaluate the access request information contained therein. First, it checks whether the request payload is of type "application/dcaf+cbor" and contains at least the fields AS, D, and AI. AS MUST respond with

4.00 (Bad Request) for CoAP (or 400 for HTTP) if the type is "application/dcaf+cbor" and any of these fields is missing or does not conform to the format described in Section 5.

AS decides whether or not access is granted to the requested resource and then creates a Ticket Grant message that reflects the result. To grant access to the requested resource, AS creates an access ticket comprised of a Face and a Verifier as described in Section 4.1.

The Ticket Grant message then is constructed as a success response indicating attached content, i.e. 2.05 for CoAP, or 200 for HTTP, respectively. The payload of the Ticket Grant message is a data structure that contains the result of the access request. When access is granted, the data structure contains the ticket's Face, the Verifier and the Session Key Generation Method.

The Ticket Grant message MAY provide cache-control options to enable intermediaries to cache the response. The message MAY be cached according to the rules defined in [RFC7252] to facilitate ticket retrieval when C has crashed and wants to recover the DTLS session with RS.

AS sets Max-Age according to the ticket lifetime in its response (Ticket Grant Message).

Figure 5 shows an example Ticket Grant message using CoAP. The Face/ Verifier information is transferred as a CBOR data structure as specified in Section 5. The Max-Age option tells the receiving AM how long this ticket will be valid.

```
2.05 Content
Content-Format: application/dcaf+cbor
Max-Age: 86400
{ F: {
        AI: [ "/s/tempC", 7 ],
        D: "2001:db8:ab9:1234:7920:3133:ae5f:87",
        TS: 0("2013-07-10T10:04:12.391"),
        L:  86400,
        G: hmac_sha256
  },
  V: h'b2dd4e409c2d36a7423da3c87e571999
        0b778ebd2c7d3730729a7fcde26c7201'
}
```

Figure 5: Example Ticket Grant Message

A Ticket Grant message that declines any operation on the requested resource is illustrated in Figure 6. As no ticket needs to be issued, an empty payload is included with the response.

```
2.05 Content
Content-Format: application/dcaf+cbor
```

Figure 6: Example Ticket Grant Message With Reject

3.7. Ticket Transfer Message

A Ticket Transfer message delivers the access information sent by AS in a Ticket Grant message to the requesting client C. The Ticket Transfer message is the response to the Access Request message sent from C to AM and includes any access information from AS contained in the Ticket Grant message.

3.8. DTLS Channel Setup Between C and RS

Using the information contained in a positive response to its Access Request (i.e. a Ticket Transfer message that contains a Face and a Verifier), 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], with the following additional requirements:

1. C sets the psk_identity field of the ClientKeyExchange message to the ticket Face received in the Ticket Transfer message.

2. C uses the ticket Verifier as PSK when constructing the premaster secret.

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.

Note3: The ticket is constructed by AS such that RS can derive the authorization information as well as the PSK (refer to Section 6 for details).

3.9. Authorized Resource Request Message

Successful establishment of the DTLS channel between C and RS ties the authorization information contained in the psk_identity field to

this channel. Any request that RS receives on this channel is checked against these authorization rules. Incoming CoAP requests that are not Authorized Resource Requests MUST be rejected by RS with 4.01 response as described in Section 3.2.

RS SHOULD treat an incoming CoAP request as Authorized Resource Request if the following holds:

1. The message was received on a secure channel that has been established using the procedure defined in Section 3.8.

2. The authorization information tied to the secure channel 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.

Note that the authorization information is not restricted to a single resource URI. For example, role-based authorization can be used to authorize a collection of semantically connected resources simultaneously. Implicit authorization also provides access rights to authenticated clients for all actions on all resources that RS offers. As a result, C can use the same DTLS channel not only for subsequent requests for the same resource (e.g. for block-wise transfer as defined in [I-D.ietf-core-block] or refreshing observe-relationships [I-D.ietf-core-observe]) but also for requests to distinct resources.

Incoming CoAP requests received on a secure channel according to the procedure defined in Section 3.8 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.

Since AS may limit the set of requested actions in its Ticket Grant message, C cannot know a priori if an Authorized Resource Request will succeed.

3.10. Dynamic Update of Authorization Information

Once a security association exists between a Client and a Resource Server, the Client can update the Authorization Information stored at the Resource Server at any time. To do so, the Client creates a new Access Request for the intended action on the respective resource and sends this request to its Authentication Manager which relays this request to the Resource Server's Authorization Server as described in Section 3.4.

Note: Requesting a new Access Ticket also can be a Client's reaction on a 4.03 or 4.05 error that it has received in response to an Authorized Resource Request.

Figure 7 depicts the message flow where C requests a new Access Tickets after a security association between C and RS has been established using this protocol.

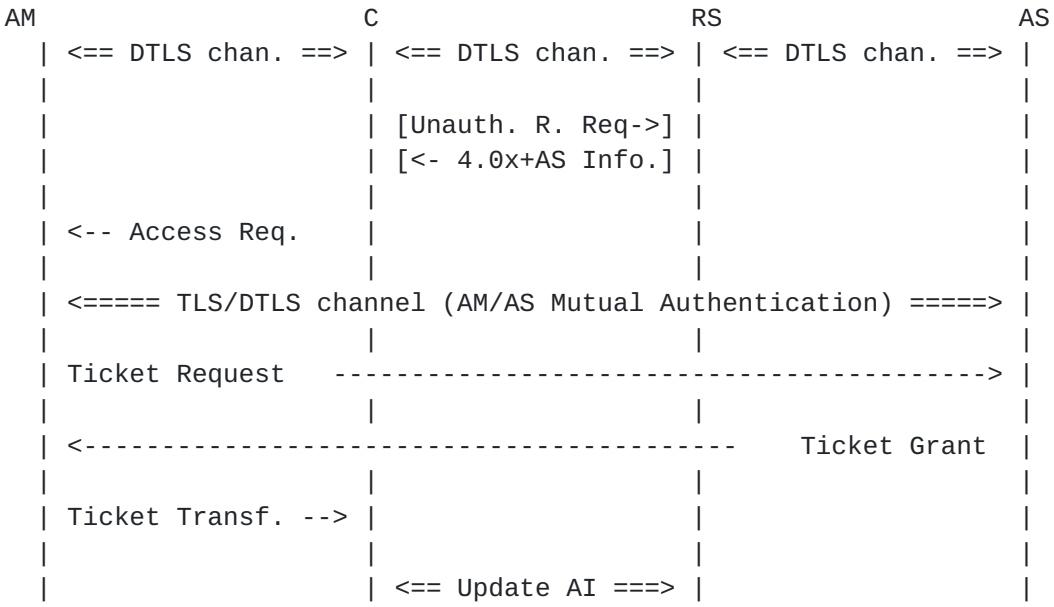

Figure 7: Overview of Dynamic Update Operation

Processing the Ticket Request is done at the Authorization Server as specified in Section 3.6, i.e. the AS checks whether or not the requested operation is permitted by the Resource Owner's policy, and then return a Ticket Grant message with the result of this check. If access is granted, the Ticket Grant message contains an Access Ticket comprised of a public Ticket Face and a private Ticket Verifier. This authorization payload is relayed by the Authorization Manager to the Client in a Ticket Transfer Message as defined in Section 3.7.

The major difference between dynamic update of Authorization Information and the initial handshake is the handling of a Ticket Transfer message by the Client that is described in Section 3.10.1.

3.10.1. Handling of Ticket Transfer Messages

If the security association with RS still exists and RS has indicated support for session renegotiation according to [RFC5746], the ticket Face SHOULD be used to renegotiate the existing DTLS session. In this case, the ticket Face is used as psk_identity as defined in Section 3.8. Otherwise, the Client MUST perform a new DTLS handshake according to Section 3.8 that replaces the existing DTLS session.

After successful completion of the DTLS handshake RS updates the existing Authorization Information for C according to the contents of the ticket Face.

Note: No mutual authentication between C and RS is required for dynamic updates when a DTLS channel exists that has been established as defined in Section 3.8. RS only needs to verify the authenticity and integrity of the ticket Face issued by AS which is achieved by having performed a successful DTLS handshake with the ticket Face as psk_identity. This could even be done within the existing DTLS session by tunneling a CoDTLS [I-D.schmertmann-dice-codtls] handshake.

4. Ticket

Access tokens in DCAF are tickets that consist of two parts, namely the Face and the Verifier. The Face goes to RS, the Verifier goes to the Client. The Face and the Verifier are parts of the same ticket.

RS only needs the information contained in the Ticket Face to authorize the client and make sure that AS generated the Ticket Face (RS cannot make authorization decisions by itself and hence needs AS to do it). No additional information about the Client is needed. RS keeps the Ticket Face as long as it is valid.

4.1. Face

Face is the part of the ticket generated for RS. Face MUST contain all information needed for authorized access to a resource:

- Authorization Information
- Descriptive label
- A timestamp generated by AS

Optionally, Face MAY also contain:

- A lifetime (optional)
- A DTLS pre-shared key (optional)

RS MUST verify the integrity of Face, i.e. the information contained in Face stems from AS and was not manipulated by anyone else.

Face MUST contain a timestamp to verify that the contained information is fresh. As constrained devices may not have a clock, timestamps MAY be generated using the clock ticks since the last reboot. To circumvent synchronization problems the timestamp MAY be generated by RS and included in the first AS Information message. Alternatively, AS MAY generate the timestamp. In this case, AS and RS MUST use a time synchronization mechanism to make sure that RS interprets the timestamp correctly.

Face MAY be encrypted. If Face contains a DTLS PSK, the whole content of Face MUST be encrypted.

Note: The integrity of Face can be ensured by various means. Face may be encrypted by AS with a key it shares with RS. Alternatively, RS can use a mechanism to generate the DTLS PSK which includes Face and is only able to calculate the correct key with the correct Face (refer to Section 6 for details).

4.2. Verifier

The Verifier part of the ticket is generated for C. It contains the DTLS PSK for C. The Verifier MUST NOT be transmitted over insecure channels.

4.3. Revocation

The existence of access tickets SHOULD be limited in time. This can be achieved either by explicit Revocation Messages to invalidate a ticket or implicitly by attaching a lifetime to the ticket.

4.3.1. Lifetime

Tickets MAY have a lifetime. AS is responsible for defining the ticket lifetime. If AS sets a lifetime for a ticket, AS and RS MUST use a time synchronization method to ensure that RS is able to interpret the lifetime correctly. RS SHOULD end the DTLS connection to C if the lifetime of a ticket has run out and it MUST NOT accept new requests. RS MUST NOT accept tickets with an invalid lifetime.

Note: Defining reasonable ticket lifetimes is difficult to accomplish. How long a client needs to access a resource depends heavily on the application scenario and may be difficult to decide for AS.

4.3.2. Revocation Messages

AS MAY revoke tickets by sending a ticket revocation message to RS. If RS receives a ticket revocation message, it MUST end the DTLS connection to C and MUST NOT accept any further requests from C.

If ticket revocation messages are used, RS MUST check regularly if AS is still available. If RS cannot contact AS, it MUST end all DTLS connections and reject any further requests from C.

Note: The loss of the connection between RS and AS prevents all access to RS. This might especially be a severe problem if AS is responsible for several Resource Servers or even a whole network.

5. Payload Format and Encoding (application/dcaf+cbor)

Various messages types of the DCAF protocol carry payloads to express authorization information and parameters for generating the DTLS PSK to be used by C and RS. In this section, a representation in Concise Binary Object Representation (CBOR, [RFC7049]) is defined.

DCAF data structures are defined as CBOR maps that contain key value pairs. For efficient encoding, the keys defined in this document are represented as unsigned integers in CBOR, i. e. major type 0. For improved reading, we use symbolic identifiers to represent the corresponding encoded values as defined in Table 1.

Encoded Value	Key
0b000_00000	AS
0b000_00001	AI
0b000_00010	D
0b000_00011	E
0b000_00100	K
0b000_00101	TS
0b000_00110	L

0b000_00111	G
0b000_01000	F
0b000_01001	V

Table 1: DCAF field identifiers encoded in CBOR

The following list describes the semantics of the keys defined in DCAF.

AS: Authorization Server. This attribute denotes the authorization server that is in charge of the resource specified in attribute R. The attribute's value is a string that contains an absolute URI according to Section 4.3 of [RFC3986].

AI: Authorization Information. A data structure used to convey authorization information from AS to RS and to describe the permissions requested from AS in a Ticket Request. The AI attribute contains an AIF object as defined in [I-D.bormann-core-ace-aif].

D: Description. A descriptive label of the initiator of the authorization request. This label MAY be a fully qualified domain name, an IP address, or any other character literal that is used by the Authorization Server to decide whether or not access is granted to the requesting entity.

E: Encrypted Ticket Face. A binary string containing an encrypted ticket Face.

K: Key. A string that identifies the shared key between RS and AS that can be used to decrypt the contents of E. If the attribute E is present and no attribute K has been specified, the default is to use the current session key for the secured channel between RS and AS.

TS: Time Stamp. An optional time stamp that indicates the instant when the access ticket request was formed. This attribute can be used by the resource server in an AS Information message to convey a time stamp in its local time scale (e.g. when it does not have a real time clock with synchronized global time). When the attribute's value is encoded as a string, it MUST contain a valid UTC timestamp without time zone information. When encoded as integer, TS contains a system timestamp relative to the local time scale of its generator, usually RS.

L: Lifetime. A lifetime of the ticket. When encoded as a string, L MUST denote the ticket's expiry time as a valid UTC timestamp without time zone information. When encoded as an integer, L MUST denote the ticket's validity period in seconds relative to TS.

G: DTLS PSK Generation Method. A numeric identifier for the method that RS MUST use to derive the DTLS PSK from the ticket Face. This attribute MUST NOT be used when attribute V is present within the contents of F. This specification uses symbolic identifiers for improved readability. The corresponding numeric values encoded in CBOR are defined in Table 2. A registry for these codes is defined in Section 13.1.

F: Ticket Face. An object containing the fields AI, D, TS, and optionally G, L and V.

V: Ticket Verifier. A binary string containing the shared secret between C and RS.

Encoded Value	Mnemonic	Support
0b000_00000	hmac_sha256	mandatory
0b000_00001	hmac_sha384	optional
0b000_00010	hmac_sha512	optional

Table 2: CBOR encoding for DTLS PSK Key Generation Methods

5.1. Examples

The following example specifies an Authorization Server that will be accessed using HTTP over TLS. The request URI is set to "/a?ep=%5B2001:DB8::dcaf:1234%5D" (hence denoting the endpoint address to authorize). TS denotes a local timestamp in UTC.

```
POST /a?ep=%5B2001:DB8::dcaf:1234%5D HTTP/1.1
Host: as-rs.example.com
Content-Type: application/dcaf+cbor
{AS: "https://as-rs.example.com/a?ep=%5B2001:DB8::dcaf:1234%5D",
 D: "2001:DB8::dcaf:1234",
 AI: ["coaps://temp451.example.com/s/tempC", 1],
 TS: 0("2013-07-14T11:58:22.923")}
```

The following example shows a ticket for the distributed key generation method (cf. Section 6.2), comprised of a Face (F) and a

Verifier (V). The Face data structure contains authorization information AI, a client descriptor, a timestamp using the local time scale of RS, and a lifetime relative to RS's time scale.

The DTLS PSK Generation Method is set to hmac_sha256 denoting that the distributed key derivation is used as defined in Section 6.2 with SHA-256 as HMAC function.

The Verifier V contains a shared secret to be used as DTLS PSK between C and RS.

```
HTTP/1.1 200 OK
Content-Type: application/dcaf+cbor
{
  F: {
       AI: [ "/s/tempC", 1 ],
       D: "2001:db8:ab9:1234:7920:3133:ae5f:87",
       TS: 2938749,
       L:  3600,
       G: hmac_sha256
     },
  V: h'93b9448d4380304d5a574fc50b944958
       55bbd5ba1422cc09fde61665aa519cf9'
}
```

The Face may be encrypted as illustrated in the following example. Here, the field E carries an encrypted Face data structure that contains the same information as the previous example, and an additional Verifier. Encryption was done with a secret shared by AS and RS. (This example uses AES128_CCM with the secret { 0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f } and RS's timestamp { 0x00, 0x2C, 0xD7, 0x7D } as nonce.) Line breaks have been inserted to improve readability.

The attribute K describes the identity of the key to be used by RS to decrypt the contents of attribute E. Here, The value "key0" in this example is used to indicate that the shared session key between RS and AS was used for encrypting E.

```
{
 E: h'2e1c0c0ae1915711f1073f34e44bfc81
      12167f5bdbd8801d07686615b0b434
      cdca7a5453d0d582565e2f236948235d
      d353cef1114d64d138949f7ab01b92f0
      b6f2caccce3a43cb0a32f270a82cde0a
      98250e6ac2b79a26fb47c09ef4cb366f
      1aa38017cd8b891a6d796fa684294a60
      64f3665527c5890b65a33af73a5c66ef
```

```
        66cbb9e28ea30c89',
    K: "key0",
    V: h'93b9448d4380304d5a574fc50b944958
         55bbd5ba1422cc09fde61665aa519cf9'
}
```

The decrypted contents of E are depicted below (whitespace has been added to improve readability). The presence of the attribute V indicates that the DTLS PSK Transfer is used to convey the session key (cf. Section 6.1).

```
{
  F: {
       AI: [ "/s/tempC", 1 ],
       D: "2001:db8:ab9:1234:7920:3133:ae5f:87",
       TS: 2938749,
       L:  3600,
       G: hmac_sha256
     },
  V: h'93b9448d4380304d5a574fc50b944958
       55bbd5ba1422cc09fde61665aa519cf9'
}
```

6. DTLS PSK Generation Methods

One goal of the DCAF protocol is to provide for a DTLS PSK shared between C and RS. AS and RS MUST negotiate the method for the DTLS PSK generation.

6.1. DTLS PSK Transfer

The DTLS PSK is generated by AS and transmitted to C and RS using a secure channel.

The DTLS PSK transfer method is defined as follows:

- AS generates the DTLS PSK using an algorithm of its choice
- AS MUST include a representation of the DTLS PSK in Face and encrypt it together with all other information in Face with a key K(AS,RS) it shares with RS. How AS and RS exchange K(AS,RS) is not in the scope of this document. AS and RS MAY use their preshared key as K(AS,RS).
- AS MUST include a representation of the DTLS PSK in the Verifier.
- As AS and C do not have a shared secret, the Verifier MUST be transmitted to C using encrypted channels.

- RS MUST decrypt Face using K(AS,RS)

6.2. Distributed Key Derivation

AS generates a DTLS PSK for C which is transmitted using a secure channel. RS generates its own version of the DTLS PSK using the information contained in Face (see also Section 4.1).

The distributed key derivation method is defined as follows:

- AS and RS both generate the DTLS PSK using the information. included in Face. They use an HMAC algorithm on Face with a shared key. The result serves as the DTLS PSK. How AS and RS negotiate the used HMAC algorithm is not in the scope of this document. They MAY however use the HMAC algorithm they use for their DTLS connection.

- AS MUST include a representation of the DTLS PSK in the Verifier.

- As AS and C do not have a shared secret, the Verifier MUST be transmitted to C using encrypted channels.

- AS MUST NOT include a representation of the DTLS PSK in Face.

- AS MUST NOT encrypt Face.

7. Authorization Configuration

For the protocol defined in this document, proper configuration of AS is crucial. The principal who owns the resources hosted by RS (i.e. the Resource Owner) needs to define permissions for the resources. The data representation of these permissions are not in the scope of this document.

8. Trust Relationships

C trusts AM, and RS trusts AS. Obviously, AM trusts C with the specific permissions it hands over to it. How this trust is established, is not in the scope of this document. It may be achieved by using a bootstrapping mechanism similar to [bergmann12].

Additionally, AS and AM need to have a trust relationship established. Its establishment is also not in the scope of this document. It fulfills the following conditions:

1. AS has means to authenticate AM (e.g. it has a certificate of AM or a PKI in which AM is included) and vice versa

2. As far as AS needs to rely on the different clients of AM to receive different permissions, it can be sure that AM correctly identifies these clients towards AS and does not leak tickets that have been generated for a specific client C to another client.

AS trusts C indirectly because it trusts AM and AM vouches for C. The DCAF Protocol does not provide any means for AS to validate that a resource requests stems from C.

C indirectly trusts AS with some potentially confidential information, and that AS correctly represents RS, because AM trusts AS.

AM trusts RS indirectly because it trusts AS and AS vouches for RS.

C implicitly trusts RS with some potentially confidential information because it trusts AM and because RS can prove that it shares a key with AS.

9. Listing Authorization Server Information in a Resource Directory

CoAP utilizes the Web Linking format [RFC5988] to facilitate discovery of services in an M2M environment. [RFC6690] defines specific link parameters that can be used to describe resources to be listed in a resource directory [I-D.ietf-core-resource-directory].

9.1. The "auth-request" Link Relation

This section defines a resource type "auth-request" that can be used by clients to retrieve the request URI for a server's authorization service. When used with the parameter rt in a web link, "auth-request" indicates that the corresponding target URI can be used in a POST message to request authorization for the resource and action that are described in the request payload.

The Content-Format "application/dcaf+cbor with numeric identifier TBD1 defined in this specification MAY be used to express access requests and their responses.

The following example shows the web link used by AM in this document to relay incoming Authorization Request messages to AS. (Whitespace is included only for readability.)

```
<client-authorize>;rt="auth-request";ct=TBD1
                  ;title="Contact Remote Authorization Server"
```

The resource directory that hosts the resource descriptions of RS could list the following description. In this example, the URI "ep/node138/a/switch2941" is relative to the resource context "coaps://as-rs.example.com/", i.e. the authorization server AS.

```
<ep/node138/a/switch2941>;rt="auth-request";ct=TBD1;ep="node138"
                         ;title="Request Client Authorization"
                         ;anchor="coaps://as-rs.example.com/"
```

10. Examples

This section gives a number of short examples with message flows for the initial Unauthorized Resource Request and the subsequent retrieval of a ticket from AS. The notation here follows the role conventions defined in Section 1.2.1. The payload format is encoded as proposed in Section 5. The IP address of AS is 2001:DB8::1, the IP address of RS is 2001:DB8::dcaf:1234, and C's IP address is 2001:DB8::c.

10.1. Access Granted

This example shows an Unauthorized PUT request from C to RS that is answered with an AS Information message. C then sends a POST request to AM with a description of its intended request. AM forwards this request to AS using CoAP over a DTLS-secured channel. The response from AS contains an access ticket that is relayed back to AM.

```
C --> RS
PUT a/switch2941 [Mid=1234]
Content-Format: application/senml+json
{"e": [{"bv": "1"}]}

C <-- RS
4.01 Unauthorized  [Mid=1234]
Content-Format: application/dcaf+cbor
{AS: "coaps://[2001:DB8::1]/ep/node138/a/switch2941"}

C --> AM
POST client-authorize [Mid=1235,Token="tok"]
Content-Format: application/dcaf+cbor
{
```

```
    AS: "coaps://[2001:DB8::1]/ep/node138/a/switch2941",
    AI: ["coaps://[2001:DB8::dcaf:1234]/a/switch2941", 4]
  }

  AM --> AS [Mid=23146]
  POST ep/node138/a/switch2941
  Content-Format: application/dcaf+cbor
  {
    AS: "coaps://[2001:DB8::1]/ep/node138/a/switch2941",
    D:  "2001:DB8::c",
    AI: ["coaps://[2001:DB8::dcaf:1234]/a/switch2941", 4]
  }

  AM <-- AS
  2.05 Content  [Mid=23146]
  Content-Format: application/dcaf+cbor
  { F: {
         AI: ["a/switch2941", 5],
         D:  "2001:DB8::c",
         TS: 0("2013-07-04T20:17:38.002"),
         G: hmac_sha256
       },
    V: h'50f18bf1d6f084eb0fd9d2ee6ec882d8
         a87ef66a332c86a45bff8f67fe19bc47'
  }

  C <-- AM
  2.05 Content  [Mid=1235,Token="tok"]
  Content-Format: application/dcaf+cbor
  { F: {
         AI: ["a/switch2941", 5],
         D:  "2001:DB8::c",
         TS: 0("2013-07-04T20:17:38.002"),
         G: hmac_sha256
       },
    V: h'50f18bf1d6f084eb0fd9d2ee6ec882d8
         a87ef66a332c86a45bff8f67fe19bc47'
  }

  C --> RS
  ClientHello (TLS_PSK_WITH_AES_128_CCM_8)

  C <-- RS
  ServerHello (TLS_PSK_WITH_AES_128_CCM_8)
  ServerHelloDone

  C --> RS
  ClientKeyExchange
```

```
  psk_identity=0x6146a4624149826c612f737769746368
               0x323934310561446b323030313a444238
               0x3a3a63625453c077323031332d30372d
               0x30345432303a31373a33382e30303261
               0x476b686d61635f736861323536

(C decodes the contents of V and uses the result as PSK)
ChangeCipherSpec
Finished

(RS calculates PSK from AI, D, TS and its session key
 HMAC_sha256(0x6146a4624149826c612f737769746368
             0x323934310561446b323030313a444238
             0x3a3a63625453c077323031332d30372d
             0x30345432303a31373a33382e30303261
             0x476b686d61635f736861323536,
             0x66736563726574)
= 0x0e70158e...
)

C <-- RS
ChangeCipherSpec
Finished
```

10.2. Access Denied

This example shows a denied Authorization request for the DELETE operation.

```
C --> RS
DELETE a/switch2941

C <-- RS
4.01 Unauthorized
Content-Format: application/dcaf+cbor
{AS: "coaps://[2001:DB8::1]/ep/node138/a/switch2941"}

C --> AM
POST client-authorize
Content-Format: application/dcaf+cbor
{
  AS: "coaps://[2001:DB8::1]/ep/node138/a/switch2941",
  AI: ["coaps://[2001:DB8::dcaf:1234]/a/switch2941", 8]
}

AM --> AS
POST ep/node138/a/switch2941
```

```
Content-Format: application/dcaf+cbor
{
  AS: "coaps://[2001:DB8::1]/ep/node138/a/switch2941",
  D:  "2001:DB8::c",
  AI: ["coaps://[2001:DB8::dcaf:1234]/a/switch2941", 8]
}

AM <-- AS
2.05 Content
Content-Format: application/dcaf+cbor

C <-- AM
2.05 Content
Content-Format: application/dcaf+cbor
```

10.3. Access Restricted

This example shows a denied Authorization request for the operations GET, PUT, and DELETE. AS grants access for PUT only.

```
AM --> AS
POST ep/node138/a/switch2941
Content-Format: application/dcaf+cbor
{
  AS: "coaps://[2001:DB8::1]/ep/node138/a/switch2941",
  D:  "2001:DB8::c",
  AI: ["coaps://[2001:DB8::dcaf:1234]/a/switch2941", 13]
}

AM <-- AS
2.05 Content
Content-Format: application/dcaf+cbor
{ F: {
       AI: ["a/switch2941", 5],
       D:  "2001:DB8::c",
       TS: 0("2013-07-04T21:33:11.930"),
       G: hmac_sha256
     },
  V: h'f5628265ec99349d2b1f3a1020223793
       7098512d555f085a775f1ae6a9c66950'
}
```

10.4. Implicit Authorization

This example shows an Authorization request using implicit authorization. AM initially requests the actions GET and POST on the resource "coaps://[2001:DB8::dcaf:1234]/a/switch2941". AS returns a ticket that has no AI field in its ticket Face, hence implicitly authorizing C.

```
AM --> AS
POST ep/node138/a/switch2941
Content-Format: application/dcaf+cbor
{
   AS: "coaps://[2001:DB8::1]/ep/node138/a/switch2941",
   D:  "2001:DB8::c",
   AI: ["coaps://[2001:DB8::dcaf:1234]/a/switch2941", 3]
}

AM <-- AS
2.05 Content
Content-Format: application/dcaf+cbor
{ F: {
       D:  "2001:DB8::c",
       TS: 0("2013-07-16T10:15:43.663"),
       G: hmac_sha256
      },
  V: h'6d30f6162b54cd50c8b7421674d46150
       1baba2a34c0a86a7aacc0cfe3c2f2643'
}
```

11. Specific Usage Scenarios

The general DCAF architure outlined in Section 3.1 illustrates the various actors who participate in the message exchange for authenticated authorization. The message types defined in this document cover the most general case where all four actors are separate entities that may or may not reside on the same device.

Special implementation considerations apply when one single entity takes the role of more than one actor. This section gives advice on the most common usage scenarios where the Authentication Manager and Client, the Authorization Server and Resource Server or the Authentication Manager and Authorization Server reside on the same (less-constrained) device and have a means of secure communication outside the scope of this document.

11.1. Combined Authentication Manager and Client

When AM and C reside on the same (less-constrained) device, the Access Request and Ticket Transfer messages can be substituted by other means of secure communication. Figure 8 shows a simplified message exchange for a combined AM+C device.

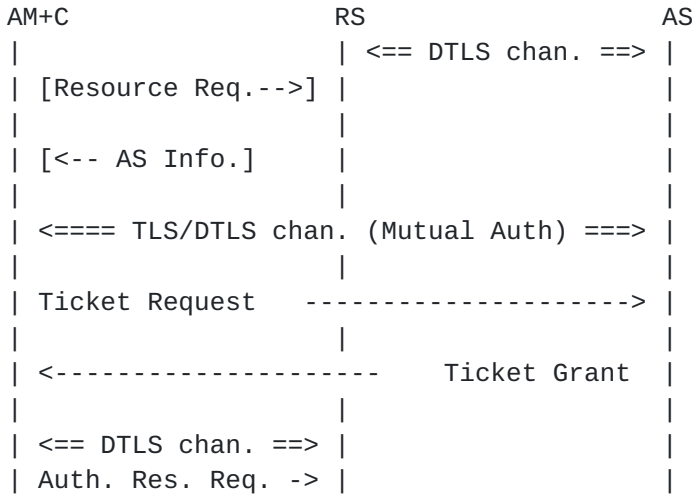

Figure 8: Combined Authentication Manager and Client

11.1.1. Creating the Ticket Request Message

When AM+C receives an AS Information message as a reaction to an Unauthorized Request message, it creates a Ticket Request message as follows:

1. The destination of the Ticket Request message is derived from the authority information in the URI contained in field "AS" of the AS Information message payload.

2. The request method is POST.

3. The request URI is constructed from the AS field received in the AS Information message payload.

4. The payload contains the AS field from the AS Information message, an absolute URI of the resource that AM+C wants to access, the the actions that AM+C wants to perform on the resource, and any time stamp generated by RS that was transferred with the AS Information message.

5. A label that describes AM+C is added to the payload.

11.1.2. Processing the Ticket Grant Message

Based on the Ticket Grant message, AM+C is able to establish a DTLS channel with RS. To do so, AM+C sets the psk_identity field of the DTLS ClientKeyExchange message to the ticket Face received in the Ticket Grant message and uses the ticket Verifier as PSK when constructing the premaster secret.

11.2. Combined Authentication Manager and Authorization Server

In certain scenarios, AM and AS may be combined to a single entity that knows both, C and RS, and decides if their actions are authorized. Therefore, no explicit communication between AM and AS is necessary, resulting in omission of the Ticket Request and Ticket Grant messages. Figure 9 depicts the resulting message sequence in this simplified architecture.

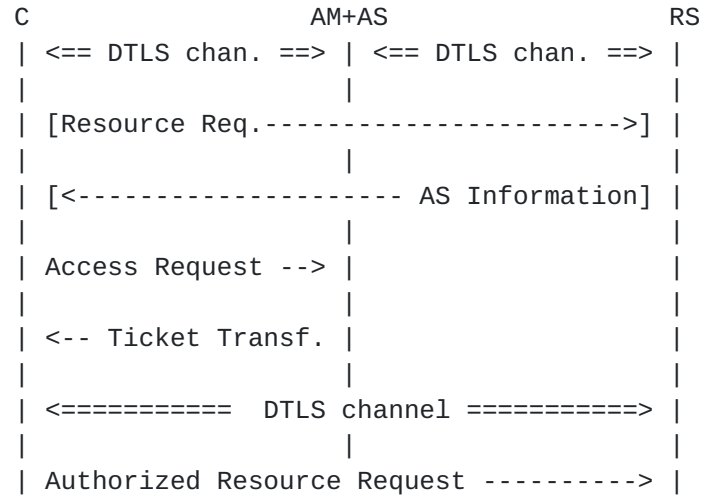

Figure 9: Combined Authentication Manager and Authorization Server

11.2.1. Processing the Access Request Message

When receiving an Access Request message, AM+AS performs the checks specified in Section 3.5 and returns a 4.00 (Bad Request) response in case of failure. Otherwise, if the checks have succeeded, AM+AS evaluates the contents of Access Request message as described in Section 3.6.

The decision on the access request is performed by AM+AS with respect to the stored policies. When the requested action is permitted on the respective resource, AM+AS generates an access ticket as outlined in Section 4.1 and creates a Ticket Transfer message to convey the access ticket to the Client.

11.2.2. Creating the Ticket Transfer Message

A Ticket Transfer message is constructed as a 2.05 response with the access ticket contained in its payload. The response MAY contain a Max-Age option to indicate the ticket's lifetime to the receiving Client.

This specification defines a CBOR data representation for the access ticket as illustrated in Section 3.6.

11.3. Combined Authorization Server and Resource Server

If AS and RS are colocated in one entity (AS+RS), the main objective is to allow AM to delegate access to C. Accordingly, the authorization information could be replaced by a nonce internal to AS+RS. (TBD.)

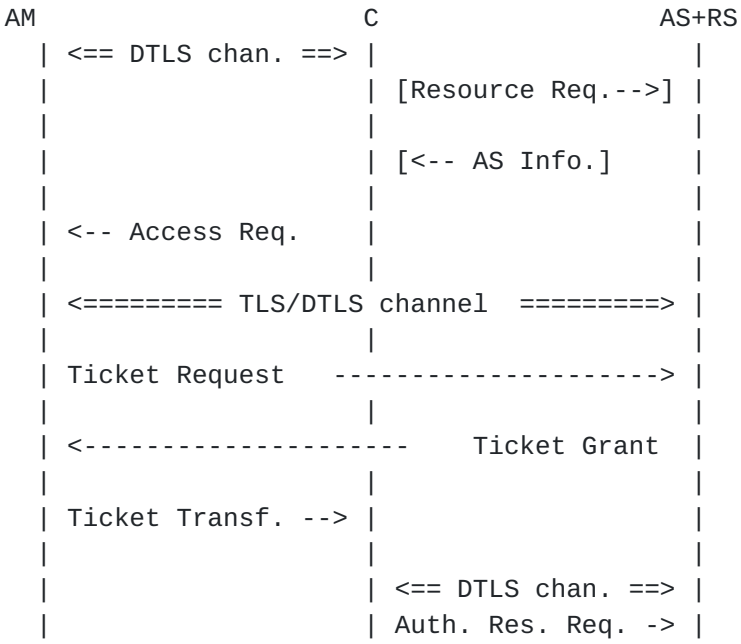

Figure 10: Combined Authorization Server and Resource Server

12. Security Considerations

As this protocol builds on transitive trust between authorization servers as mentioned in Section 8, AS has no direct means to validate that a resource request originates from C. It has to trust AM that it correctly vouches for C and that it does not give authorization tickets meant for C to another client nor disclose the contained session key.

The Authorization Server also could constitute a single point of failure. If the Authorization Server fails, the resources on all Resource Servers it is responsible for cannot be accessed any more. Thus, it is crucial for large networks to use Authorization Servers in a redundant setup.

13. IANA Considerations

The following registrations are done following the procedure specified in [RFC6838].

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

13.1. DTLS PSK Key Generation Methods

A sub-registry for the values indicating the PSK key generation method as contents of the field G in a payload of type application/ dcaf+cbor is defined. Values in this sub-registry are numeric integers encoded in Concise Binary Object Notation (CBOR, [RFC7049]). This document follows the notation of [RFC7049] for binary values, i.e. a number starts with the prefix "0b". The major type is separated from the actual numeric value by an underscore to emphasize the value's internal structure.

Initial entries in this sub-registry are as follows:

Encoded Value	Name	Reference
0b000_00000	hmac_sha256	[RFC-XXXX]
0b000_00001	hmac_sha384	[RFC-XXXX]
0b000_00010	hmac_sha512	[RFC-XXXX]

Table 3: DTLS PSK Key Generation Methods

New methods can be added to this registry based on designated expert review according to [RFC5226].

(TBD: criteria for expert review.)

13.2. dcaf+cbor Media Type Registration

Type name: application

Subtype name: dcaf+cbor

Required parameters: none

Optional parameters: none

Encoding considerations: Must be encoded as using a subset of the encoding allowed in [RFC7049]. Specifically, only the primitive data types String and Number are allowed. The type Number is restricted to unsigned integers (i.e., no negative numbers, fractions or exponents are allowed). Encoding MUST be UTF-8. These restrictions simplify implementations on devices that have very limited memory capacity.

Security considerations: TBD

Interoperability considerations: TBD

Published specification: [RFC-XXXX]

Applications that use this media type: TBD

Additional information:

Magic number(s): none

File extension(s): dcaf

Macintosh file type code(s): none

Person & email address to contact for further information: TBD

Intended usage: COMMON

Restrictions on usage: None

Author: TBD

Change controller: IESG

13.3. CoAP Content Format Registration

This document specifies a new media type application/dcaf+cbor (cf. Section 13.2). For use with CoAP, a numeric Content-Format identifier is to be registered in the "CoAP Content-Formats" sub-registry within the "CoRE Parameters" registry.

Note to RFC Editor: Please replace all occurrences of "RFC-XXXX" with the RFC number of this specification.

Media type	Encoding	Id.	Reference
application/dcaf+cbor	-	TBD1	[RFC-XXXX]

14. References

14.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC4279] Eronen, P. and H. Tschofenig, "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)", RFC 4279, December 2005.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC5746] Rescorla, E., Ray, M., Dispensa, S., and N. Oskov, "Transport Layer Security (TLS) Renegotiation Indication Extension", RFC 5746, February 2010.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, January 2013.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, October 2013.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, June 2014.

14.2. Informative References

[I-D.bormann-core-ace-aif]

Bormann, C., "An Authorization Information Format (AIF) for ACE", draft-bormann-core-ace-aif-00 (work in progress), January 2014.

[I-D.ietf-core-block]
Bormann, C. and Z. Shelby, "Blockwise transfers in CoAP", draft-ietf-core-block-15 (work in progress), July 2014.

[I-D.ietf-core-observe]
Hartke, K., "Observing Resources in CoAP", draft-ietf-core-observe-14 (work in progress), June 2014.

[I-D.ietf-core-resource-directory]
Shelby, Z., Bormann, C., and S. Krco, "CoRE Resource Directory", draft-ietf-core-resource-directory-01 (work in progress), December 2013.

[I-D.schmertmann-dice-codtls]
Schmertmann, L., Hartke, K., and C. Bormann, "CoDTLS: DTLS handshakes over CoAP", draft-schmertmann-dice-codtls-00 (work in progress), February 2014.

[RFC5988] Nottingham, M., "Web Linking", RFC 5988, October 2010.

[RFC6655] McGrew, D. and D. Bailey, "AES-CCM Cipher Suites for Transport Layer Security (TLS)", RFC 6655, July 2012.

[RFC6690] Shelby, Z., "Constrained RESTful Environments (CoRE) Link Format", RFC 6690, August 2012.

[bergmann12]
Bergmann, O., Gerdes, S., Schaefer, S., Junge, F., and C. Bormann, "Secure Bootstrapping of Nodes in a CoAP Network", IEEE Wireless Communications and Networking Conference Workshops (WCNCW), April 2012.

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