ACE Working Group
Internet Draft
Intended status: Informational
Expires: January 2015

S. Gerdes
Universität Bremen TZI
July 04, 2014

Actors in the ACE Architecture
draft-gerdes-ace-actors-01

Abstract

Constrained nodes are small devices which are limited in terms of processing power, memory, non-volatile storage and transmission capacity. Due to these constraints, commonly used security protocols are not easily applicable. Nevertheless, an authentication and authorization solution is needed to ensure the security of these devices.

Due to the limitations of the constrained nodes it is especially important to develop a light-weight security solution which is adjusted to the relevant security objectives of each participating party in this environment. Necessary security measures must be identified and applied where needed.

In this document, the required security related tasks are identified as guidance for the development of authentication and authorization solutions for constrained environments. Based on the tasks, an architecture is developed to represent the relationships between the logical functional entities involved.

Status of this Memo

Copyright Notice

1. Introduction

Constrained nodes are small devices with limited abilities which in many cases are made to fulfill a single simple task. They have limited system resources such as processing power, memory, non-volatile storage and transmission capacity and additionally in most cases do not have user interfaces and displays. Due to these constraints, commonly used security protocols are not always easily applicable.

Constrained nodes are expected to be integrated in all aspects of everyday life and thus will be trusted with a lot of personal data. Without appropriate security mechanisms attackers might gain control over things relevant to our lives. Authentication and authorization mechanisms are therefore prerequisites for a secure Internet of Things.

The Authentication and Authorization in Constrained Environments (ACE) Working Group aims at defining a solution for authenticated and authorized access to resources. To achieve this, it is necessary to develop a deep understanding of the problem to be solved. An essential part of this is to identify the tasks which must be performed to meet the security requirements in this scenario. Moreover, these tasks need to be assigned to logical functional entities which perform the tasks: the actors in the architecture. Thus, relations between the actors and requirements for protocols can be identified.

In this document, the required security related tasks are identified as guidance for the development of authentication and authorization solutions for constrained environments. Based on the tasks, an architecture is developed to represent the relationships between the logical functional entities involved.

1.1 Terminology

This document uses the following terminology:

Resource: an item of interest. It might contain sensor or actuator values or other information. The author had resources in the sense of RFC7231 [RFC7231] in mind, but for the considerations in this document the kind of representation of the item is not relevant.

Constrained node: a constrained device in the sense of [RFC7228].

Actor: A logical functional entity within a device that performs one or more tasks. Depending on the tasks, the device may need to have certain system resources available. Multiple actors may share, i.e. be present within, a device or even a piece of software.

Resource Server (RS): An entity which hosts a Resource.

Client (C): An entity which attempts to access a resource on a Resource Server.

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

Client Owner (CO): The principal that owns the Client and controls permissions concerning authorized sources for R.

2. Problem Statement

The scenario the ACE Working Group addresses can be summarized as follows:

- A Client (C) wants to access a Resource (R) on a Resource Server (RS).
- A priori, C and RS do not necessarily know each other and have no security relationship.
- C and / or RS are constrained.

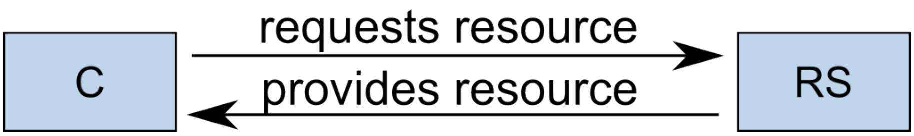

Figure 1: Basic Scenario

There are some security requirements for this scenario including one or more of:

- Rq0.1: No unauthorized entity has access to (or otherwise gains knowledge of) R.
- Rq0.2: When C attempts to access R, that access reaches the proper R.

3. Tasks

This section gives an overview of the tasks which must be performed in the given scenario (see Section 2) to meet the security requirements.

As described in the problem statement, either C or RS or both of them are constrained. Therefore tasks which must be conducted by either C or RS must be performable by constrained nodes.

3.1 Basic Scenario Tasks

This document does not assume a specific solution. We assume however, that at least the following information is exchanged between the client and the server:

- C transmits to RS which resource it requests to access, the kind of action it wants to perform on the resource and the parameters needed for the action.
- RS transmits to C the result of the attempted access.

3.2 Authentication-Related Tasks

According to the Internet Security Glossary [RFC4949], authentication is “the process of verifying a claim that a system entity or system resource has a certain attribute value.” Examples for attribute values are the ID of a device, the type of the device or the name of its owner. Authentication attributes might be (but not necessarily are) suitable to uniquely identify an individual entity.

Several steps must be conducted for authenticating certain attributes of an entity and validating the authenticity of an information:

1. Attribute binding: The attribute that shall be verifiable must be bound to a verifier, e.g. a key. To achieve this, an attribute binding authority has to check if the entity in possession of a certain verifier really possesses the attributes it claims to have. The authority must provide some kind of endorsement information which enables other entities to validate the binding.
2. Authentication: The entity which wants to use the verifier for authenticating an entity checks the attribute-verifier-binding using the endorsement of the claim validation authority and uses the verifier for authenticating an entity or the source of an information.

Step 1 is addressed in Appendix A.2.5. Two types of tasks were defined for step 2: Information authenticity (see Appendix A.2.1) and secure communication (see Appendix A.2.3).

3.3 Authorization-Related Tasks

Several steps must be conducted for authorization:

1. Configuration of authorization information: The owner must configure the authorization information.
2. Obtaining authorization information: Authorization information must be made available to the entity which enforces the authorization.
3. Authorization validation: The authorization of an entity with certain attributes must be checked by mapping the attributes (which must be validated by authentication) to the authorization information.

Tasks for step 1 are defined in Appendix A.2.6. Appendix A.2.4 addresses step 2. Appendix A.2.2 introduces tasks for step 3.

4. Actors

This section describes the various actors in the architecture. An actor is identified by the tasks it has to fulfill. Several actors might share a single device or even be combined in a single piece of software. Interfaces between actors may be realized as protocols or be internal to such a piece of software.

The concept of actors is used to assign the tasks defined in Appendix A to logical functional entities.

4.1 Constrained Level Actors

As described in the problem statement (see Section 2), either C or RS or both of them may be located on a constrained node. We therefore define that C and RS must be able to perform their tasks even if they are located on a constrained node. Thus, C and RS are considered to be Constrained Level Actors.

C performs the following tasks:

- Negotiate means for secure communication (Task TSecureComm, see Appendix A.2.3).
- Validate that an entity is an authorized source for R (Task TValSourceAuthz, see Appendix A.2.2).
- Securely transmit an access request (Task TSendReq, see Appendix A.1.2).
- Validate that the response to an access request is authentic (Task TAuthnResp, see Appendix A.2.1).
- Process the response to an access request (Task TProcResp, see Appendix A.1.1).

RS performs the following tasks:

- Negotiate means for secure communication (Task TSecureComm, see Appendix A.2.3).
- Validate the authenticity of an access request (Task TAuthnReq, see Appendix A.2.1).
- Validate the authorization of the requester to access the requested resource as requested (Task TValAccessAuthZ, see Appendix A.2.2).
- Process an access request (Task TProcReq, see Appendix A.1.1).
- Securely transmit a response to an access request (Task TSendResp, see Appendix A.1.2).

R is an item of interest such as a sensor or actuator value. R is considered to be part of RS and not a separate actor. The device on which RS is located might contain several resources of different resource owners. For simplicity of exposition, these resources are described as if they had separate RS.

As C and RS do not necessarily know each other they might belong to different security domains.

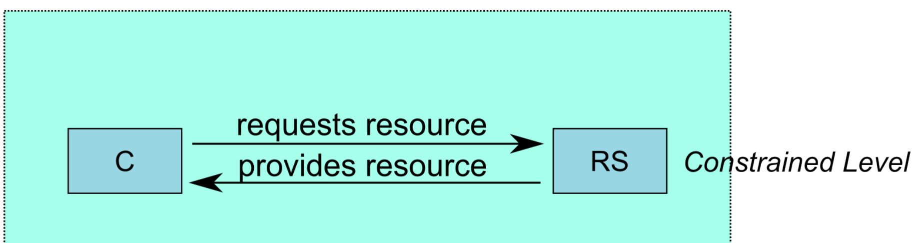

Figure 2: Constrained Level Actors

4.2 Principal Level Actors

Our objective is that C and RS are under control of principals in the physical world, the Client Owner (CO) and the Resource Owner (RO) respectively. The owners decide about the security policies of their respective devices and belong to the same security domain.

CO is in charge of C, i.e. CO specifies security policies for C, e.g. with whom C is allowed to communicate. By definition, C and CO belong to the same security domain.

CO must fulfill the following task:

- Configure for C authorization information for sources for R (Task TConfigSourceAuthz, see Appendix A.2.6).

RO is in charge of R and RS. RO specifies authorization policies for R and decides with whom RS is allowed to communicate. By definition, R, RS and RO belong to the same security domain.

RO must fulfill the following task:

- Configure for RS authorization information for accessing R (Task TConfigAccessAuthz, see Appendix A.2.6).

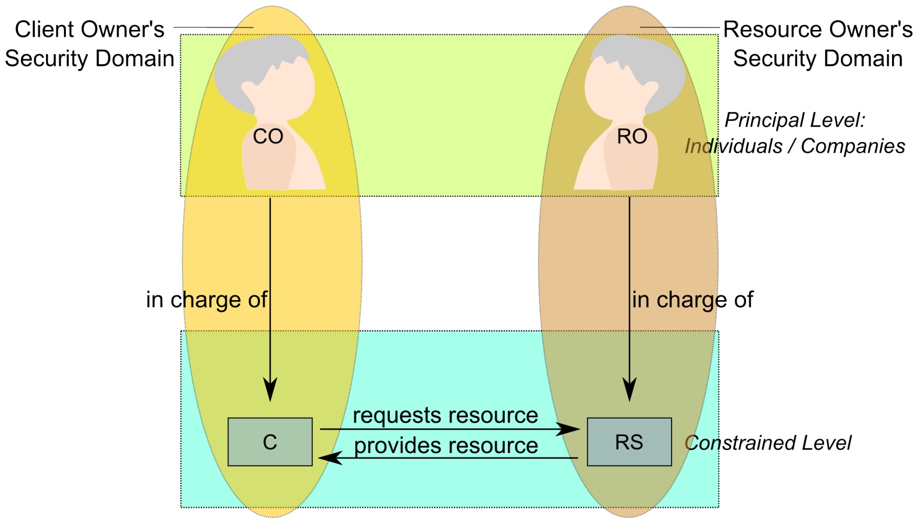

Figure 3: Constrained Level Actors and Principal Level Actors

4.3 Less-Constrained Level Actors

Constrained level actors can only fulfill a limited number of tasks and may not have network connectivity all the time. To relieve them from having to manage keys for numerous devices and conducting computationally intensive tasks, another complexity level for actors is introduced. An actor on the less-constrained level belongs to the same security domain as its respective constrained level actor. They also have the same principal.

The Authentication Manager (AM) belongs to the same security domain as C and CO. AM acts on behalf of CO. It assists C in authenticating RS and determining if RS an authorized source for R. AM can do that because for C, AM is the authority for claims about RS.

AM performs the following tasks:

- Validate on the client side that an entity has certain attributes (Task TValSourceAttr, see Appendix A.2.5).
- Obtain authorization information about an entity from C's owner and provide it to C. (Task TObtainSourceAuthz, see Appendix A.2.4).
- Negotiate means for secure communication to communicate with C (Task TSecureComm, see Appendix A.2.3).

The Authorization Server (AS) belongs to the same security domain as R, RS and RO. AS acts on behalf of RO. It supports RS by authenticating C and determining C's permissions on R. AS can do that because for RS, AS is the authority for claims about C.

AS performs the following tasks:

- Validate on the server side that an entity has certain attributes (Task TValReqAttr, see Appendix A.2.5).
- Obtain authorization information about an entity from RS' owner and provide it to RS (Task TObtainAccessAuthz, see Appendix A.2.4).

- Negotiate means for secure communication to communicate with RS (Task TSecureComm, see Appendix A.2.3).

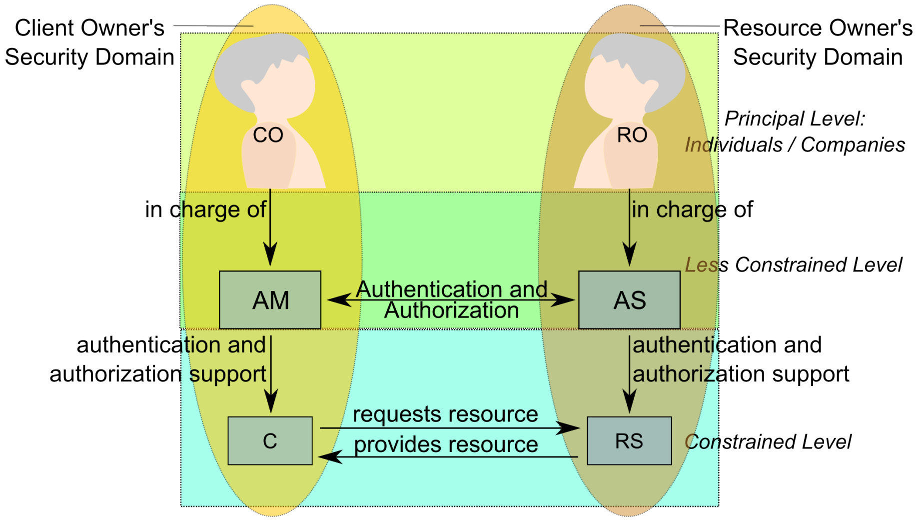

Figure 4: Overview of all Complexity Levels

5. Protocol Requirements

Devices on the less-constrained level potentially are more powerful than constrained level devices in terms of processing power, memory, non-volatile storage. This results in different requirements for the protocols used on these levels.

5.1 Constrained Level Protocols

A protocol is considered to be on the constrained level if it is used between the actors C and RS which are considered to be constrained (see Section 4.1). C and RS might not belong to the same security domain. Therefore, constrained level protocols are required to work between different security domains.

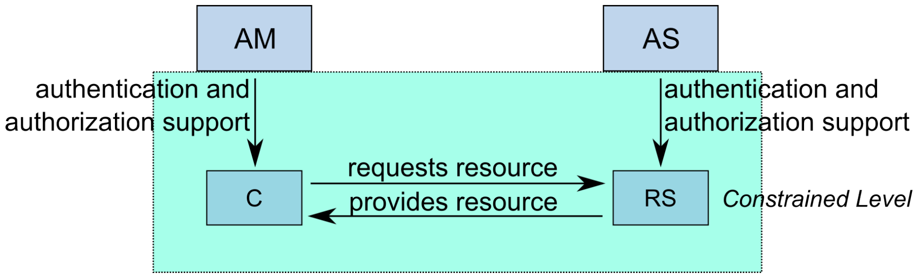

Figure 5: Constrained Level Tasks

Commonly used Internet protocols can not in every case be applied to constrained environments. In some cases, tweaking and profiling is required. In other cases it is beneficial to define new protocols which were designed with the special characteristics of constrained environments in mind.

On the constrained level, protocols must be used which address the specific requirements of constrained environments. The Constrained Application Protocol (CoAP) [RFC7252] should be used as transfer protocol if possible. CoAP defines a security binding to Datagram Transport Layer Security Protocol (DTLS) [RFC6347]. Thus, DTLS should be used for channel security.

Constrained devices have only limited storage space and thus cannot store large numbers of keys. This is especially important because constrained networks are expected to consist of thousands of nodes. Protocols on the constrained level should keep this limitation in mind.

5.1.1 Cross Level Support Protocols

Protocols which operate between a constrained device on one side and the corresponding less constrained device on the other are considered to be (cross level) support protocols. Protocols used between C and AM or RS and AS are therefore support protocols.

Support protocols must consider the limitations of their constrained endpoint and therefore belong to the constrained level protocols.

5.2 Less-Constrained Level Protocols

A protocol is considered to be on the less-constrained level if it is used between the actors AM and AS. AM and AS might belong to different security domains.

On the less-constrained level, HTTP [RFC7230] and Transport Layer Security (TLS) [RFC5246] can be used alongside or instead of CoAP and DTLS. Moreover, existing security solutions for authentication and authorization such as the Web Authorization Protocol (OAuth) [RFC6749] and Kerberos [RFC4120] can likely be used without modifications and there are no limitations for the use of a Public Key Infrastructure (PKI).

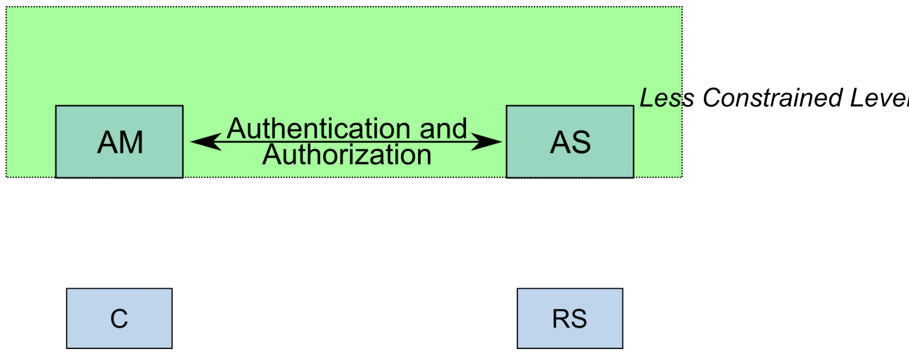

Figure 6: Less-constrained Level Tasks

6. IANA Considerations

None

7. Security Considerations

This document discusses security requirements for the ACE architecture.

8. Acknowledgments

The author would like to thank Carsten Bormann, Olaf Bergmann and Klaus Hartke for their valuable input and feedback.

9. References

9.1 Normative References

[RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, May 2014.

9.2 Informative References

[RFC7230] Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, June 2014.

[RFC7231] Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, June 2014.

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

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

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC4120] Neuman, C., Yu, T., Hartman, S., and K. Raeburn, "The Kerberos Network Authentication Service (V5)", RFC 4120, July 2005.

[RFC6749] Hardt, D., "The OAuth 2.0 Authorization Framework", RFC 6749, October 2012.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", RFC 4949, August 2007.

Author's Address

Stefanie Gerdes
Universität Bremen TZI
Postfach 330440
Bremen, D-28359
Germany
Phone: +49-421-218-63906
EMail: gerdes@tzi.org

A. List of Tasks

This section defines the tasks which must be performed in the given scenario (see Section 2) starting from communication related tasks and then deriving the required security-related tasks. An overview of the tasks can be found in Section 3.

A task has the following structure:

- The name of the task which has the form TXXX
- One or more Requirements (if applicable) of the form RqXXX
- One or more Preconditions (if applicable) of the form PreXXX
- One or more Postconditions (if applicable) of the form PostXXX

Requirements have to be met *while* performing the task. They derive directly from the scenario (see Section 2) or from the security requirements defined for the scenario. Preconditions have to be fulfilled *before* conducting the task. Postconditions are the *results* of the completed task.

We start our analysis with the processing tasks and define which preconditions need to be fulfilled before these tasks can be conducted. We then determine which tasks therefore need to be performed first (have postconditions which match the respective preconditions).

Note: Regarding the communication, C and RS are defined as entities each having their set of attributes and a verifier which is bound to these attributes. Attributes are not necessarily usable to identify an individual C or RS. Several entities might have the same attributes.

A.1 Basic Scenario

The intended result of the interaction between C and RS is that C has successfully accessed R. C gets to know that its access request was successful by receiving the answer from RS.

The transmission of information from C to RS comprises two parts: sending the information on one side and receiving and processing it on the other. Security has to be considered at each of these steps.

A.1.1 Processing Information

The purpose of the communication between C and RS is C’s intent to access R. To achieve this, RS must process the information about the requested access and C must process the information in the response to a requested access. The request and the response might both contain resource values.

The confidentiality and integrity of R require that only authorized entities are able to access R (see Rq0.1). Therefore, C and RS must check that the information is authentic and that the source of the information is authorized to provide it, before the information can be processed. C must validate that RS is an authorized source for R. RS must validate that C is authorized to access R as requested.

If proxies are used, it depends on the type of proxy how they are integrated into the communication and what kind of security relationships need to be established. A future version of this document will provide more details on this topic. At this point we assume that C and RS might receive the information either from RS or C directly or from a proxy which is authorized to speak for the respective communication partner.

- Task TProcResp: Process the response to an access request.
  Description: C processes the response to an access request according to the reason for requesting the resource in the first place. The response might include resource values or information about the results of a request.
  Requirements:
  * RqProcResp.1: Is performed by C (derives from the problem statement).
  * RqProcResp.2: Must be performable by a constrained device (derives from the problem statement: C and / or RS are constrained).
  Preconditions:
  * PreProcResp.1: A response to an access request was sent (see Appendix A.1.2).

* PreProcResp.2 (required for Rq0.2): C validated that the response to an access request is authentic, i.e. it stems from the entity requested in TSendReq (see Appendix A.1.2), i.e. RS or an entity which is authorized to speak for RS (see Appendix A.2.1).
* PreProcResp.3 (required for Rq0.2): C validated that RS or the entity which is authorized to speak for RS is an authorized source for R (see Appendix A.2.2).
Postcondition:
* PostProcResp.1: C processed the response.

- Task TProcReq: Process an access request.
  Description: RS either performs an action on the resource according to the information in the request, or determines the reason for not performing an action.
  Requirements:
  * RqProcReq.1: Is performed by RS.
  * RqProcReq.2: Must be performable by a constrained device (derives from the problem statement: C and / or RS are constrained).
  Preconditions:
  * PreProcReq.1: An access request was sent (see Appendix A.1.2).
  * PreProcReq.2 (needed for Rq0.1): RS validated that the request is authentic, i.e. it stems from C or an entity which is authorized to speak for C and is fresh. (see Appendix A.2.1).
  * PreProcReq.3 (needed for Rq0.1): RS validated the authorization of C or the entity which is authorized to speak for C to access the resource as requested (see Appendix A.2.2).
  Postconditions:
  * PostProcReq.1: The access request was processed (fulfills PreSendResp.1, see Appendix A.1.2).

Note: The preconditions PreProcReq.2 and PreProcReq.3 must be conducted together. RS must assure that the response is bound to a verifier, the verifier is bound to certain attributes and the authorization information refers to these attributes.

A.1.2 Sending Information

The information needed for processing has to be transmitted at some point. C has to transmit to RS which resource it wants to access with which actions and parameters. RS has to transmit to C the result of the request. The request and the response might both contain resource values. To fulfill Rq0.1, the confidentiality and integrity of the transmitted data has to be assured.

If proxies are used, it depends on the type of proxy how they need to be handled. A future version of this document will provide more details on this topic. At this point we assume that C and RS might transmit the message either to RS and C directly or to a proxy which is authorized to speak for the respective communication partner.

- Task TSendReq: Securely transmit an access request.
  Description: C wants to access a resource R hosted by the resource server RS. To achieve this, it has to transmit some information to RS such as the resource to be accessed, the action to be performed on the resource and, if a writing access is requested, the value to write. C might send the request directly to RS or to an entity which is authorized to speak for RS. C assures that the request reaches the proper R. C binds the request to C's verifier to ensure the integrity of the message. C uses means to assure that no unauthorized entity is able to access the information in the request.
  Requirements:
  * RqSendReq.1: Is performed by C (derives from problem statement).
  * RqSendReq.2: Must be performable by a constrained device (derives from the problem statement: C and / or RS are constrained).
  * RqSendReq.3: As the request might contain resource values, the confidentiality and integrity of the request must be ensured during transmission. Only authorized parties must be able to read or modify the request (derives from Rq0.1).
  Preconditions:
  * PreSendReq.1: Validate that the receiver is an authorized source for R (see Appendix A.2.2).

* PreSendReq.2: To assure that the request reaches the proper RS, that no unauthorized party is able to access the request, and that the information in the request is bound to C’s verifier it is necessary to negotiate means for secure communication with RS (see Appendix A.2.3).
Postconditions:
* PostSendReq.1: The request was sent securely to RS (necessary for Rq0.1) (fulfills PreProcReq.1, see Appendix A.1.1).

Note: The preconditions PreSendReq.1 and PreSendReq.2 must be conducted together. C must assure that the request reaches an entity with certain attributes and that the authorization information refers to these attributes.

- Task TSendResp: Securely transmit a response to an access request.
Description: RS sends a response to an access request to inform C about the result of the request. RS must assure that response reaches the requesting C. RS might send the response to C or to an entity which is authorized to speak for C. The response might contain resource values. RS binds the request to RS’s verifier to ensure the integrity of the message. RS uses means to assure that no unauthorized entity is able to access the information in the response.
Requirements:
* RqSendResp.1: Is performed by RS (derives from the problem statement).
* RqSendResp.2: Must be performable by a constrained device (derives from the problem statement: C and / or RS are constrained).
* RqSendResp.3: As the response might contain resource values, the confidentiality and integrity of the response must be ensured during transmission. Only authorized parties must be able to read or modify the response (derives from Rq0.1).
Preconditions:
* PreSendResp.1: An access request was processed (see Appendix A.1.1).
* PreSendResp.2: If information about R is transmitted, validate that the receiver is authorized to access R (see Appendix A.2.2).
* PreSendResp.3: RS must assure that the response reaches the requesting C, no unauthorized party is able to access the response and the information in the response is bound to RS’ verifier: Means for secure communication were negotiated (see Appendix A.2.3).
Postconditions:
* PostSendResp.1: A response to an access request was sent (fulfills PreProcResp.1, see Appendix A.1.1).

A.2 Security-Related Tasks

A.2.1 Information Authenticity

This section addresses information authentication, i.e. using the verifier to validate the source of an information. Information authentication must be conducted before processing received information. C must validate that a response to an access request is fresh, really stems from the queried RS (or an entity which is authorized to speak for RS) and was not modified during transmission. RS must validate that the information in the access request is fresh, really stems from C (or an entity which is authorized to speak for C) and was not modified during transmission.

The entity which processes the information must be the entity which is validating the source of the information.

C and RS must assure that the authenticated source of the information is authorized to provide the information.

- Task TAuthnResp: Validate that the response to an access request is authentic.
Description: C checks if the response to an access request stems from an entity in possession of the respective verifier and is fresh. Thus, C validates that the response stems from the queried RS or an entity which is authorized to speak for RS.
Requirements:
* RqAuthnResp.1: Must be performed by C.
* RqAuthnResp.2: Must be performable by a constrained device (derives from the problem statement: C and / or RS are constrained).
Preconditions:

* PreAuthnResp.1: Means for secure communication were negotiated (see Appendix A.2.3).
Postconditions:
* PostAuthnResp.1: C knows that the response came from RS (fulfills PreProcResp.2, see Appendix A.1.1).

- Task TAuthnReq: Validate the authenticity of a request.
  Description: RS checks if the request stems from an entity in possession of the respective verifier and is fresh. Thus, RS validates that the request stems from C or an entity which is authorized to speak for C.
  Requirements:
  * RqAuthnReq.1: Must be performed by RS.
  * RqAuthnReq.2: Must be performable by a constrained device (derives from the problem statement: C and / or RS are constrained).
  Preconditions:
  * PreAuthnReq.1: Means for secure communication were negotiated (see Appendix A.2.3).
  Postconditions:
  * PostAuthnReq.1: RS knows that the request is authentic (fulfills PreProcReq.2, see Appendix A.1.1).

A.2.2 Authorization Validation

This section addresses the validation of the authorization of an entity. The entity which processes the information must validate that the source of the information is authorized to provide it. The processing entity has to verify that the source of the information has certain attributes which authorize it to provide the information: C must validate that RS (or the entity which speaks for RS) is in possession of attributes which are necessary for being an authorized source for R. RS must validate that C (or the entity which speaks for C) has attributes which are necessary for a permission to access R as requested.

- Task TValSourceAuthz: Validate that an entity is an authorized source for R.
  Description: C checks if according to CO's authorization policy and the authentication endorsement provided by the attribute binding authority, RS (or an entity which speaks for RS) is authorized to be a source for R. RS assures that the entity's verifier is bound to certain attributes and the authorization information refers to these attributes.
  Requirements:
  * RqValSourceAuthz.1: Is performed by C
  * RqValSourceAuthz.2: Must be performable by a constrained device (derives from the problem statement: C and / or RS are constrained).
  Preconditions:
  * PreValSourceAuthz.1: Authorization information about the entity are available. Requires obtaining authorization information about the entity from C's owner (see Appendix A.2.4).
  * PreValSourceAuthz.2: Means to validate that the entity has certain attributes which are relevant for the authorization: Requires validation of claims about RS (see Appendix A.2.5).
  Postconditions:
  * PostValSourceAuthz.1: The entity which performs the task knows that an entity is an authorized source for R (fulfills PreProcResp.3, see Appendix A.1.1 and PreSendReq.1, see Appendix A.1.2).
- Task TValAccessAuthZ: Validate the authorization of the requester to access the requested resource as requested.
  Description: R's owner configures which clients are authorized to perform which action on R. RS has to check if according to RO's authorization policy and the authentication endorsement provided by the attribute binding authority, C (or an entity which speaks for C) is authorized to access R as requested. RS assures that requester's verifier is bound to certain attributes and the authorization information refers to these attributes.
  Requirements:
  * RqValAccessAuthz.1: Is performed by RS
  * RqValAccessAuthz.2: Must be performable by a constrained device (derives from the problem statement: C and / or RS are constrained).
  Preconditions:

* PreValAccessAuthz.1: Authorization information about the entity are available. Requires obtaining authorization information about the entity from RS's owner (see Appendix A.2.4).
* PreValAccessAuthz.2: Means to validate that the entity has certain attributes which are relevant for the authorization: Requires validation of claims about C or the entity which speaks for C (see Appendix A.2.5).
Postconditions:
* PostValAccessAuthz.1: The entity which performs the task knows that an entity is authorized to access R with the requested action (fulfills PreProcReq.3, see Appendix A.1.1).

A.2.3 Transmission Security

To ensure the confidentiality and integrity of information during transmission means for secure communication have to be negotiated between the communicating parties.

- Task TSecureComm: Negotiate means for secure communication.
  Description: To ensure the confidentiality and integrity of transmitted information, means for secure communication have to be negotiated. Channel security as well as object security solutions are possible. Details depend on the used solution and are not in the scope of this document.
  Requirements:
  * RqSecureComm.1: Must be performable by a constrained device (derives from the problem statement: C and / or RS are constrained).
  Preconditions:
  * PreSecureComm.1: Sender and receiver must be able to validate that the entity in possession of a certain verifier has the claimed attributes. (see Appendix A.2.5).
  Postconditions:
  * PostSecureComm.1: C and RS can communicate securely: The integrity and confidentiality of information is ensured during transmission. The sending entity can use means to assure that the information reaches the intended receiver so that no unauthorized party is able to access the information. The sending entity can bind the information to the entity's verifier (fulfills PreSendResp.3 and PreSendReq.2, see Appendix A.1.2 as well as PreAuthnResp.1 and PreAuthnReq.1, see Appendix A.2.1).

A.2.4 Obtain Authorization information

As described in Section 3.3, the authorization of an entity requires several steps. The authorization information must be configured by the owner and provided to the enforcing entity.

- Task TObtainSourceAuthz: Obtain authorization information about an entity from C's owner.
  Description: C's owner defines authorized sources for R. The authorization information must be made available to C to enable it to enforce CO's authorization information. To facilitate the configuration for the owner this device should have a user interface. The authorization information has to be made available to C in a secure way.
  Requirements:
  * RqObtainSourceAuthz.1: Must be performed by an entity which belongs to C's security domain.
  * RqObtainSourceAuthz.2: Must be performed by an entity which is authorized to speak for C's owner concerning authorized sources for R.
  * RqObtainSourceAuthz.3: Should be performed by a device which can provide some sort of user interface to facilitate the configuration of authorization information for C's owner.
  Preconditions:
  * PreObtainSourceAuthz.1: C's owner configured authorized sources for R (see Appendix A.2.6).
  Postconditions:
  * PostObtainSourceAuthz.1: C obtained RS' authorization to be a source for R (fulfills PreValSourceAuthz.1, see Appendix A.2.2).
- Task TObtainAccessAuthz: Obtain authorization information about an entity from RS' owner.
  Description: RS' owner defines if and how C is authorized to access R. The authorization information must be made available to RS to enable it to enforce RO's authorization policies. To facilitate the configuration for the owner this device should have a user interface. The authorization information has to be made available to RS in a secure way.

Requirements:
* RqObtainAccessAuthz.1: Must be performed by an entity which belongs to R's security domain.
* RqObtainAccessAuthz.2: Must be performed by an entity which is authorized to speak for R's owner concerning authorization of access to R.
* RqObtainAccessAuthz.3: Should be performed by a device which can provide some sort of user interface to facilitate the configuration of authorization information for R's owner.
Preconditions:
* PreObtainAccessAuthz.1: R's owner configured authorization information for the access to R (see Appendix A.2.6).
Postconditions:
* PostObtainAccessAuthz.1: RS obtained C's authorization for accessing R (fulfills PreValAccessAuthz.1, see Appendix A.2.2).

A.2.5 Attribute Binding

As described in Section 3.2, several steps must be conducted for authentication. This section addresses the binding of attributes to a verifier.

For authentication it is necessary to validate if an entity has certain attributes. An example for such an attribute in the physical world is the name of a person or her age. In constrained environments, attributes might be the name of the owner or the type of device. Authorizations are bound to such attributes.

The possession of attributes must be verifiable. For that purpose, attributes must be bound to a verifier. An example for a verifier in the physical world is a passport. In constrained environments, a verifier will likely be the knowledge of a secret.

At some point, an authority has to check if an entity in possession of the verifier really possesses the claimed attributes. In the physical world, government agencies check your name and age before they give you a passport.

The entity that validates the claims has to provide some kind of seal to make its endorsement verifiable for other entities and thus bind the attributes to the verifier. In the physical world passports are stamped by the issuing government agencies (and must only be provided by government agencies anyway).

- Task TValSourceAttr: Validate on the client side that an entity has certain attributes.
  Description: The claim that an entity has certain attributes has to be checked and made available for C in a secure way. The validating party states that an entity in possession of a certain key has certain attributes and provides C with means to validate this endorsement.
  Requirements:
  * RqValSourceAttr.1: Must be performed by an entity which belongs to C's security domain and is an authority for claims about RS.
  * RqValSourceAttr.2: The executing entity must have the means to fulfill the task (e.g. enough storage space, computational power, a user interface to facilitate the configuration of authentication policies).
  Postconditions:
  * PostValSourceAttr.1: Means for authenticating (validating the attribute-verifier-binding of) other entities were given to C in form of a verifiable endorsement (fulfills PreValSourceAuthz.2, see Appendix A.2.2 and PreSecureComm.1, see Appendix A.2.3).
- Task TValReqAttr: Validate on the server side that an entity has certain attributes.
  Description: The claim that an entity has certain attributes has to be checked and made available for RS in a secure way. The validating party states that an entity in possession of a certain key has certain attributes and provides RS with means to validate this endorsement.
  Requirements:
  * RqValReqAttr.1: Must be performed by an entity which belongs to RS' security domain and is an authority for claims about C.
  * RqValReqAttr.2: The executing entity must have the means to fulfill the task (e.g. enough storage space, computational power, a user interface to facilitate the configuration of authentication policies).
  Postconditions:

* PostValReqAttr.1: Means for authenticating (validating the attribute-verifier-binding of) other entities were given to RS in form of a verifiable endorsement (fulfills PreValSourceAuthz.2, see Appendix A.2.2 and PreSecureComm.1, see Appendix A.2.3).

A.2.6 Configuration of Authorization Information

As stated in Section 3.3, several steps have to be conducted for authorization. This section is about the configuration of authorization information.

The owner of a device or resource wants to be in control of her device and her data. For that purpose, she has to configure authorization information. C's owner might want to configure which attributes an entity must possess to be a source for R. R's owner might want to configure which attributes are required for accessing R with a certain action.

- Task TConfigSourceAuthz: Configure for C authorization information for sources for R.
  Description: C's owner has to define authorized sources for R.
  Requirements:
  * RqConfigSourceAuthz.1: Must be provided by C's owner.
  Postconditions:
  * PostConfigSourceAuthz.1: The authorization information are available to a device which performs TObtainSourceAuthz (fulfills PreObtainSourceAuthz.1 see Appendix A.2.4).
- Task TConfigAccessAuthz: Configure for RS authorization information for accessing R.
  Description: R's owner has to configure if and how an entity with certain attributes is allowed to access R.
  Requirements:
  * RqConfigAccessAuthz.1: Must be provided by R's owner.
  Postconditions:
  * PostConfigAccessAuthz.1: The authorization information are available to the device which performs TObtainAccessAuthz (fulfills PreObtainAccessAuthz.1, see Appendix A.2.4).