Actors in the ACE Architecture

draft-ietf-ace-actors-02

Stefanie Gerdes, Ludwig Seitz, Goeran Selander, **Carsten Bormann**

IETF-94, ACE Meeting, 2015-11-02

Purpose of the Actors Draft

- Provide terminology, the architectural elements and describe the authentication and authorization problems in constrained node networks

Changes in the 02-Version

- Addressed Jim's Comments

Scenario

- RESTful architecture: a client (C) attempts to access a resource (R) which is hosted by a resource server (RS).
- C and/or RS are constrained.
- C and RS may not know each other, have no trust relationship.
- C and RS may not have the same principal (belong to the same person / company).
- How can principals keep the control over their data and devices?

Lessons Learned from the Use Cases: Security Objectives

- Devices handle sensitive data that needs to be protected.
- Different stakeholders have different security objectives.
- Authorization policies might change any time.

Consequences:

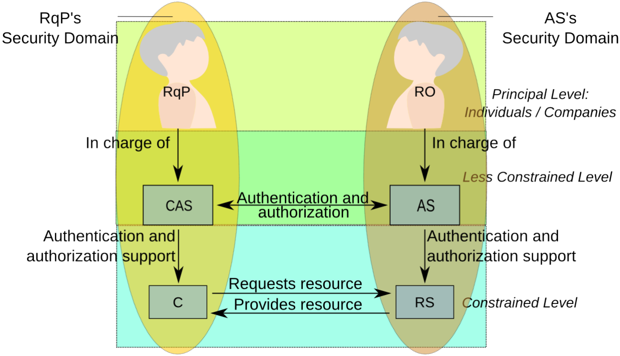
- Authorization policies must be enforced by devices that send or receive sensitive data.
- The authorization policies must be made available to the devices to make them enforceable (in some cases dynamically).

Actors

- Actors are **model**-level
  - defined by their tasks and characteristics
- Several actors **MAY** share a single device.
- Several actors **MAY** be combined in a single piece of software.
  - for a specific application
  - for a specific protocol
- Do not prematurely reduce model to one application/protocol

Actors in the Architecture

- C and RS are constrained level actors: must be able to operate on a constrained node.
- C and RS are controlled by principals in the physical world who specifiy security policies. C and RS must enact these policies.
- The less constrained nodes CAS and AS help their constrained node with authentication and authorization.



Lessons Learned from the Use Cases: Absent Users

- Often no active user at the time of access.
- Authorization policies cannot always be configured manually for each device.
- Devices often have no user interfaces and displays.

Consequences:

- Principals will not intervene in the communication (e.g., not control the client).
- Principals cannot make authorization decisions at the time of access (e.g., no authorization via pop-ups).
- Devices must be able to enforce authorization policies on their own.

Benefits of Offloading Tasks

- There might not be an active user at the time of access.
- Devices often don't have user interfaces and displays and thus cannot be controlled by the user at the time of access.
- One or both of C and RS are "constrained"
  - in terms of power, memory, storage space.
  - can only fulfill a limited number of tasks.
  - may not have network connectivity all the time.
  - may not be able to manage complex authorization policies.
  - may not be able to manage a large number of keys.
  - may not be able to precisely measure time.
- Address this by associating a *less-constrained device* to each constrained device for one or more of those difficult tasks -> Devices still have to enforce the principal's policies on their own.

Lessons Learned from the Use Cases: Constrained vs Less-Constrained

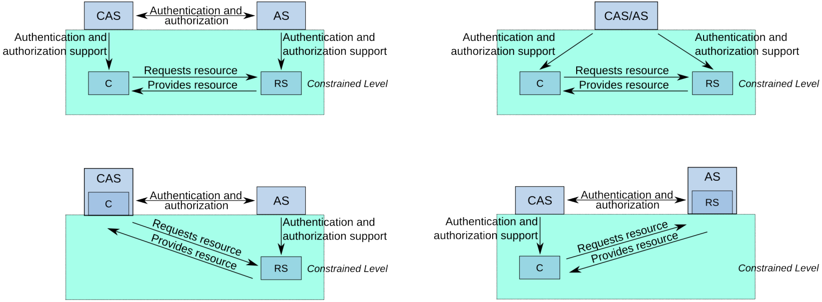
- Limitations of the communicating devices may vary.
  - Devices might have only some constraints (e.g., no user interface).
- Constrained device to less-constrained device communication is useful.
- Constrained to constrained communication allows for additional benefits (e.g., direct communication between the sensor and the cooling unit in the container monitoring use case enables more efficient cooling).

Consequences:

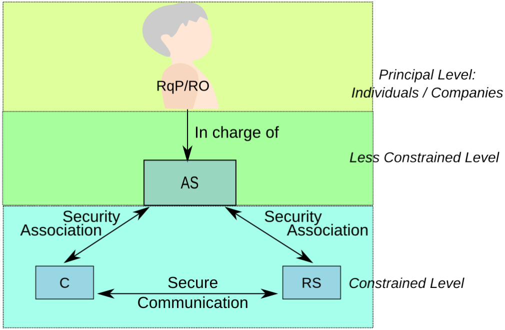
- Constrained devices communicate among themselves as well as with less-constrained devices.

Constrained Level Communication: Variants

- Protocols must consider the limitations of their constrained endpoints.
- Communication protocols are still constrained level protocols.

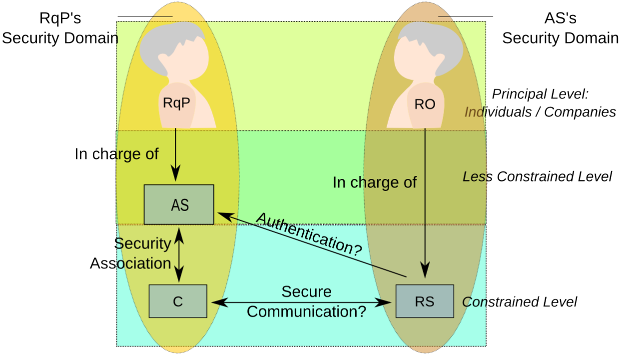


Single-Domain with Single AS



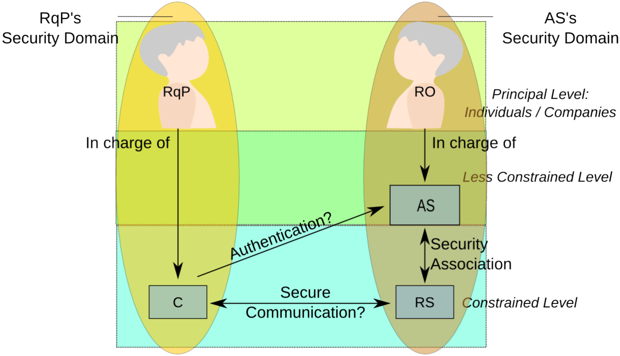
Cross-Domain with single AS: RqP in Charge

- Without (R)AS, a constrained RS cannot authenticate C and validate its authorization.



Cross-Domain with single AS: RO in Charge

- Without (C)AS, a constrained C cannot authenticate RS and cannot obtain authorization policies from RqP (COP).

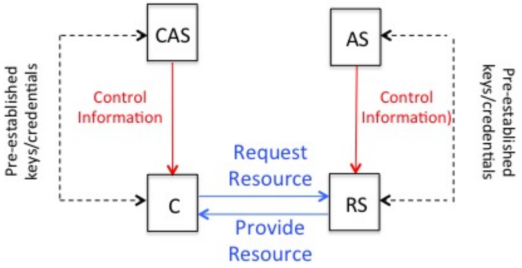


ACE Architecture

- Covers all variants including cross-domain settings.

Legend:

- Information flows in solid lines (actual message flow between the actors may be different).
  - Resource access (based on CoAP)
  - Control information (authorization information, keys, etc.)
  - Information flow may need to be secured end-to-end through intermediary devices



Information flows may be protected with session-based security (DTLS) or data object based security (COSE)

Questions the Actors Draft deals with

- How do we handle authorization without an active user?
- How do we cope with the lack of displays and user interfaces?
- How do we cope with dynamic changes in a setting (e.g., outage of the communication partner (server or client), need for a replacement)?
- How do we consider the different security objectives of the principals on both sides?
- How do we combine the constrained world with the less-constrained world?
- How do we manage the different possible client/server settings?
- How can we cope with cross-domain scenarios?

How to proceed?

- Provide a summary of tasks of the various actors in the draft
- Use the accompanying draft about tasks for a more detailed description (see draft-gerdes-ace-tasks: comments welcome)