Delegated Authenticated Authorization Framework (DCAF)

draft-gerdes-ace-dcaf-authorize

Stefanie Gerdes, Olaf Bergmann **Carsten Bormann**

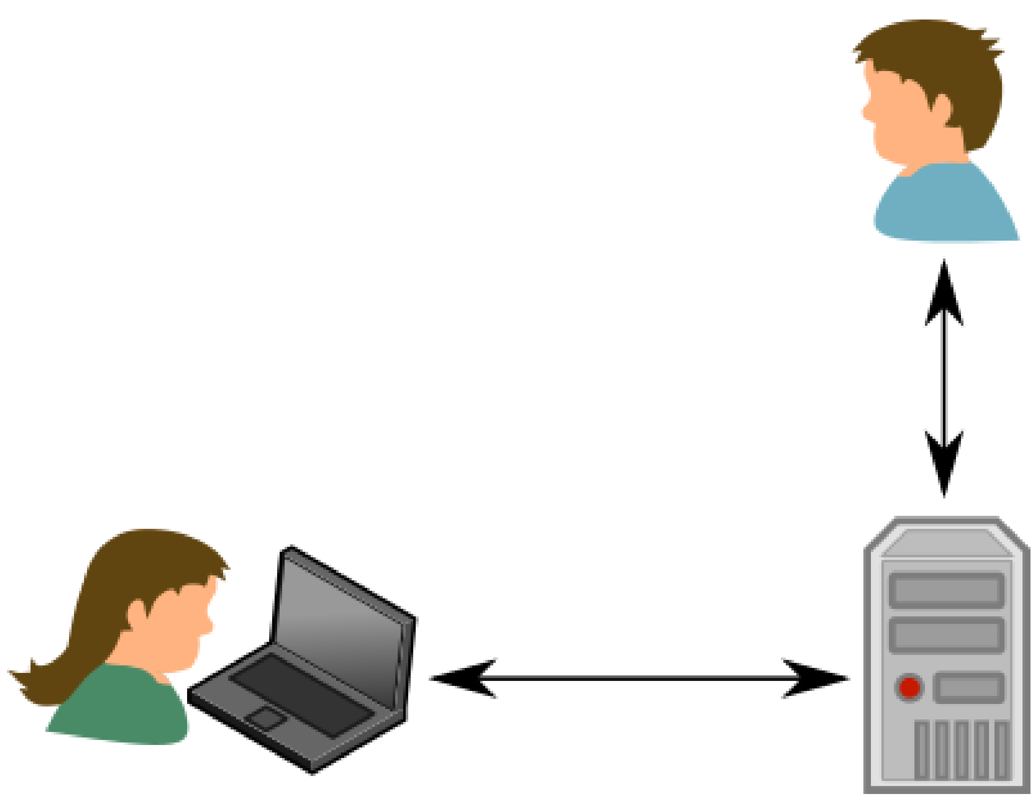
T2TRG Meeting, 2016-03-15, San Jose

The Problem

- C and S may not previously know each other
- C and / or S may be constrained
- C and S may belong to different owners / principals
- How can C and S communicate securely?
  - (Not just about communications security!)

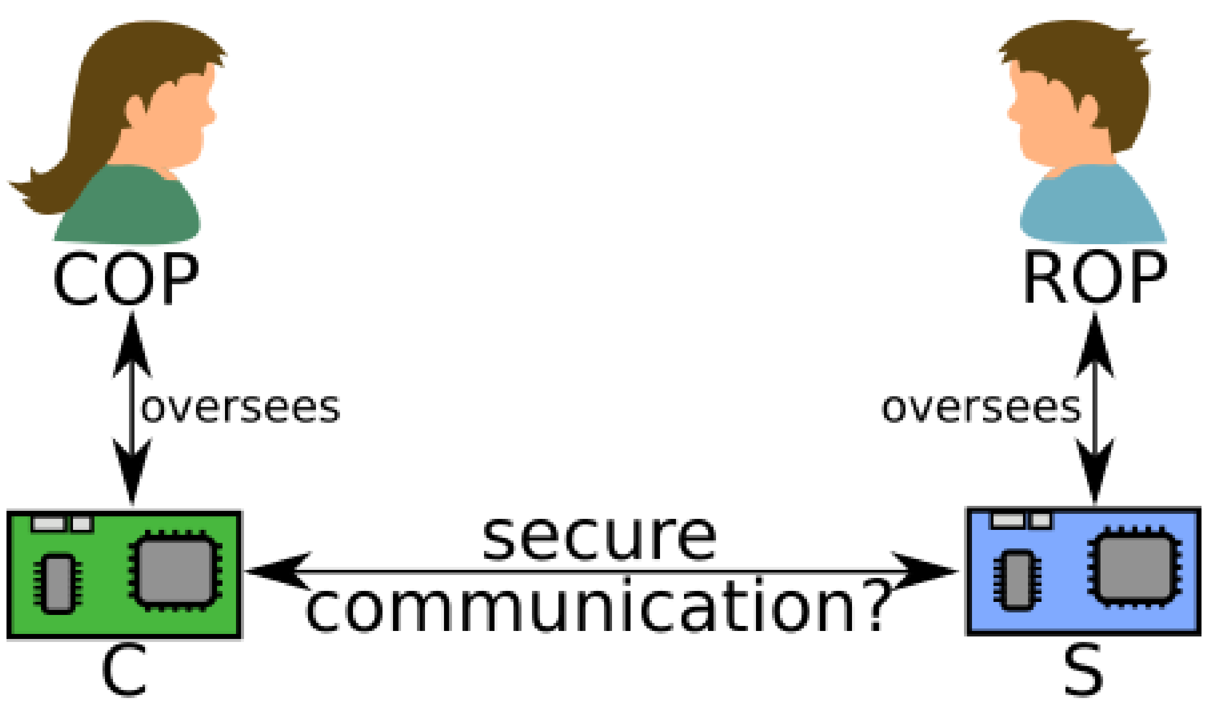
Web World (Browser Web)

- Servers act autonomously, need to enforce the security policies of their principals on their own
- Clients are controlled by their principals.
- In the web, only servers are required to validate the authorization of their peers
- Authorization is usually granted because of the principal's unique credentials
- Endpoint identities correspond to their principal's identity



Web of Things

- Servers AND clients may act autonomously
  - ($\rightarrow$ Thing Descriptions, HATEOAS, ...)
- There may be no active user present at the time of the communication
- Autonomous C and S must enforce the security intentions of their principals on their own
- Authorization is granted because of the principals' relationship
- Principals may control hundreds of endpoints
- The endpoint's own identity is mostly **not** meaningful for the authorization



Authenticated Authorization (simplified)

An endpoint needs to:

- Obtain the principal's authorization policies
- Make sure that they indeed originate from the principal and are fresh
- Validate that the peer actually has certain characteristics (e.g., name, affiliation) $\rightarrow$ authentication
- Validate that these characteristics (and the quality of the authentication) match the requirements in the authorization information
- Enforce the policies for every piece of information that is sent and received (always ascertain the authorization of the sender/receiver)

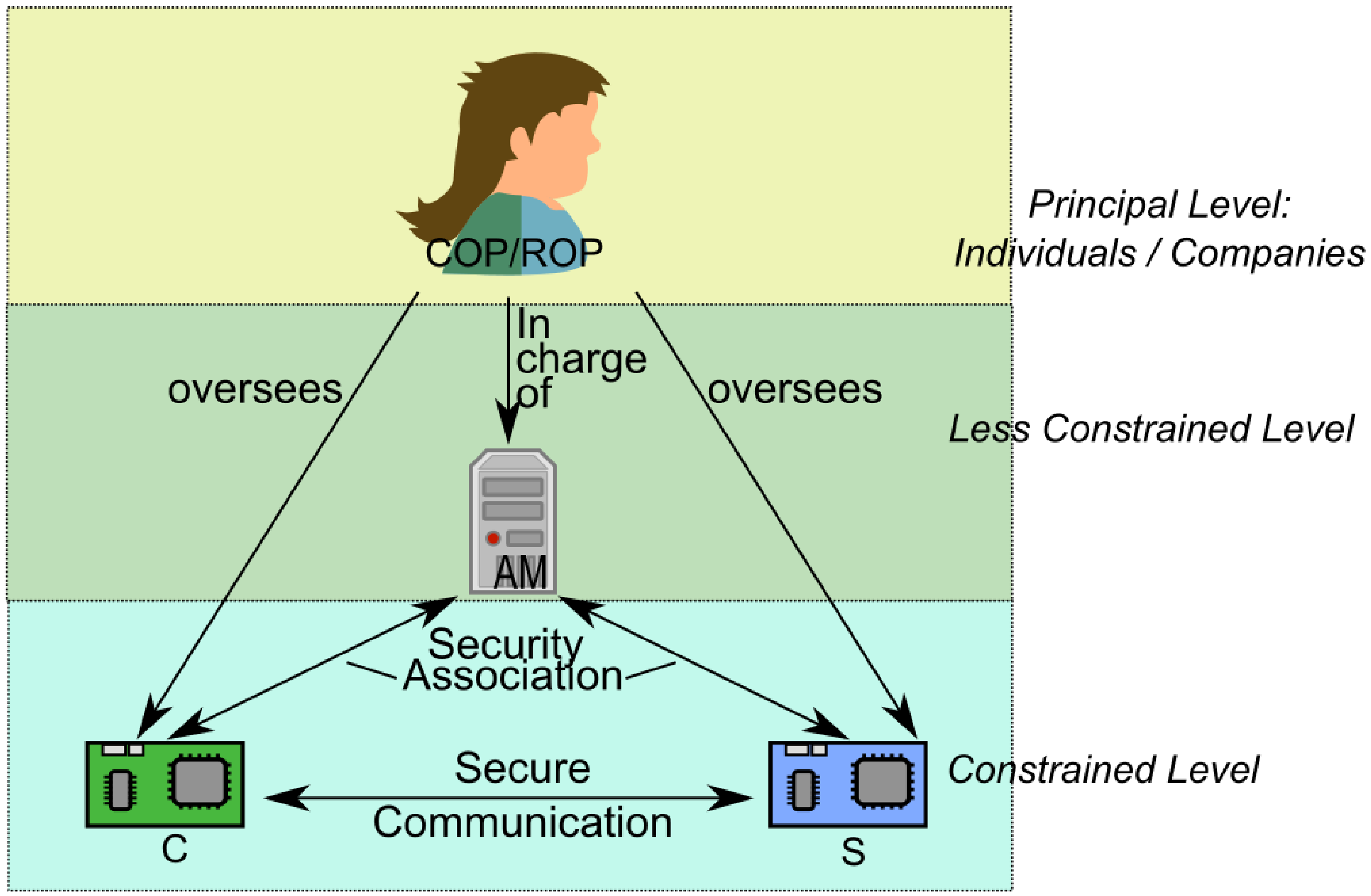
The Difficult Part: Authentication

- Storage space: things may not be able to store a large number of Credentials (Certificates likely are too large for many devices)
- Transmission capacity: things (or network) may not be able to transmit large keys
- Information obtained by authentication must be meaningful for authorization (Raw public keys do not contain information about their owners)
- Compromised keying material must not be used any more
- Third parties that help with authentication (e.g., CAs) must be authorized by the principals (and are entrusted with the authorization)

Obtaining Authentication Information

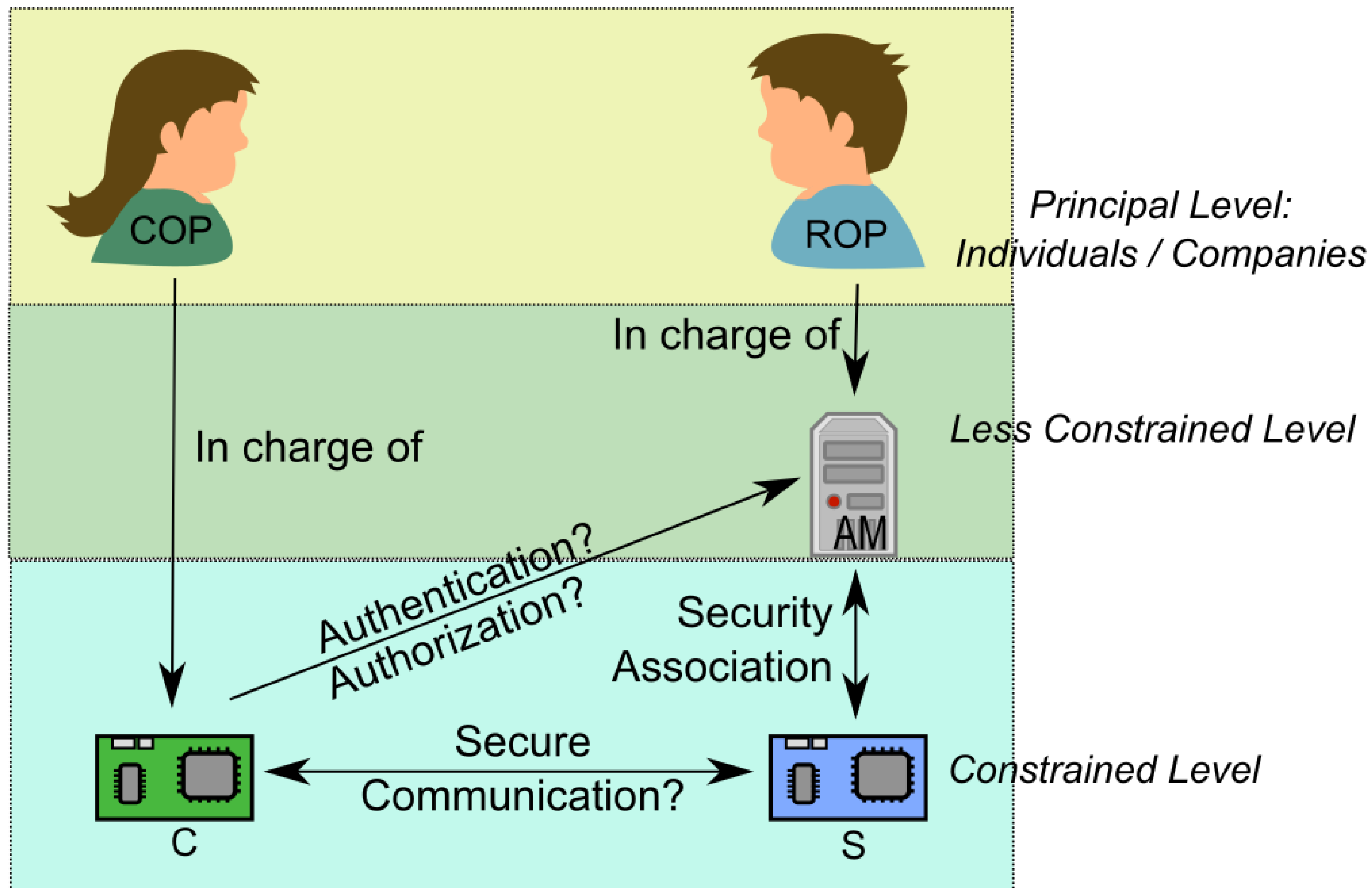
- Every keying material must either be pre-provisioned or obtained from a party whose keying material is pre-provisioned ("trusted third party").
- Hundreds of root certificates included in current browsers. Storage space?
- How to limit the number of certificates without losing flexibility?
- Optimum: use a single security association and obtain every other key with its help.

Single-Domain with Single AM



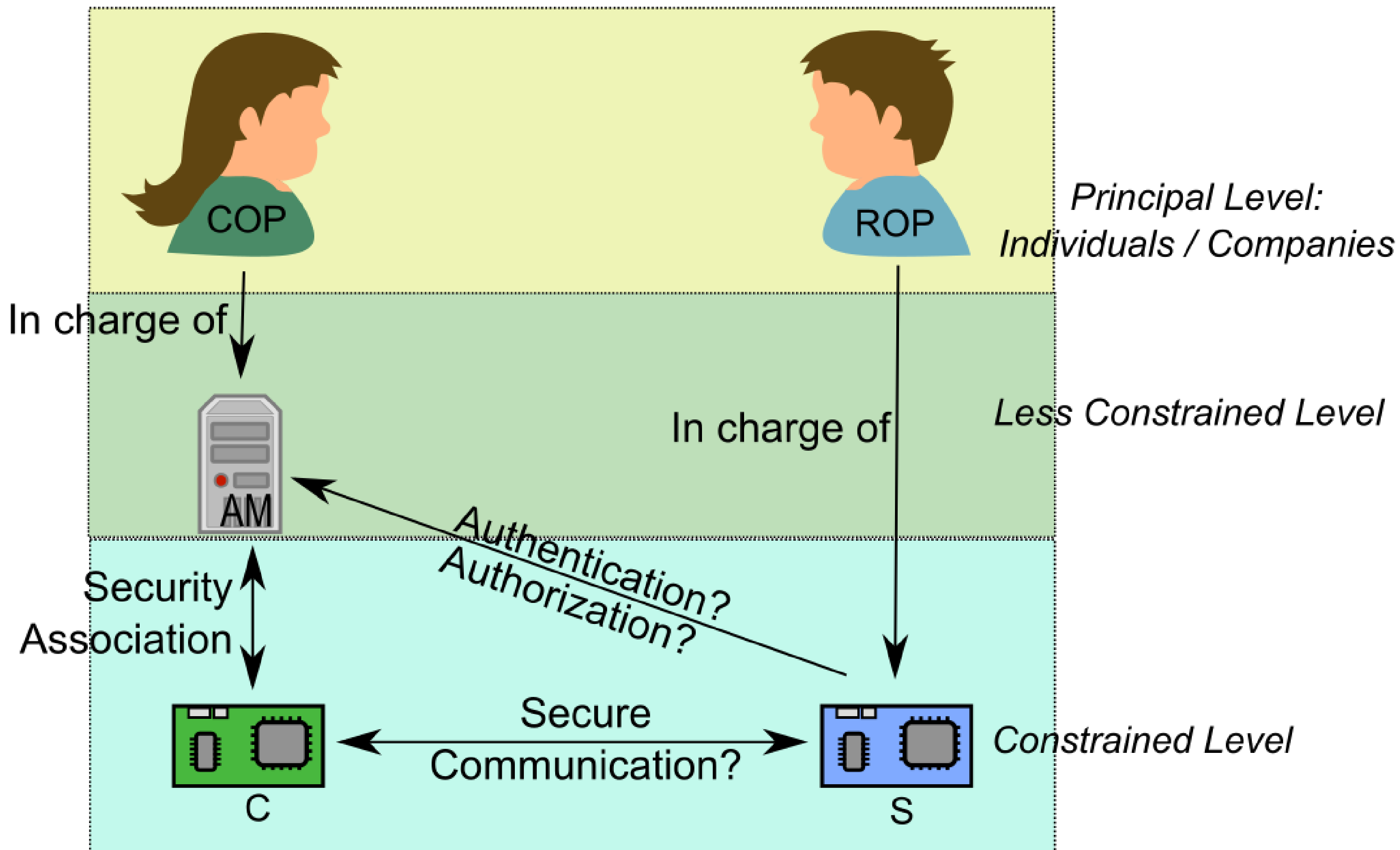
- Simplest case: C and S have the same principal
- C and S have a security association with the same Authorization Manager (AM)
- AM helps C and S with authentication and authorization

Cross-Domain with Single AM: ROP in Charge of AM



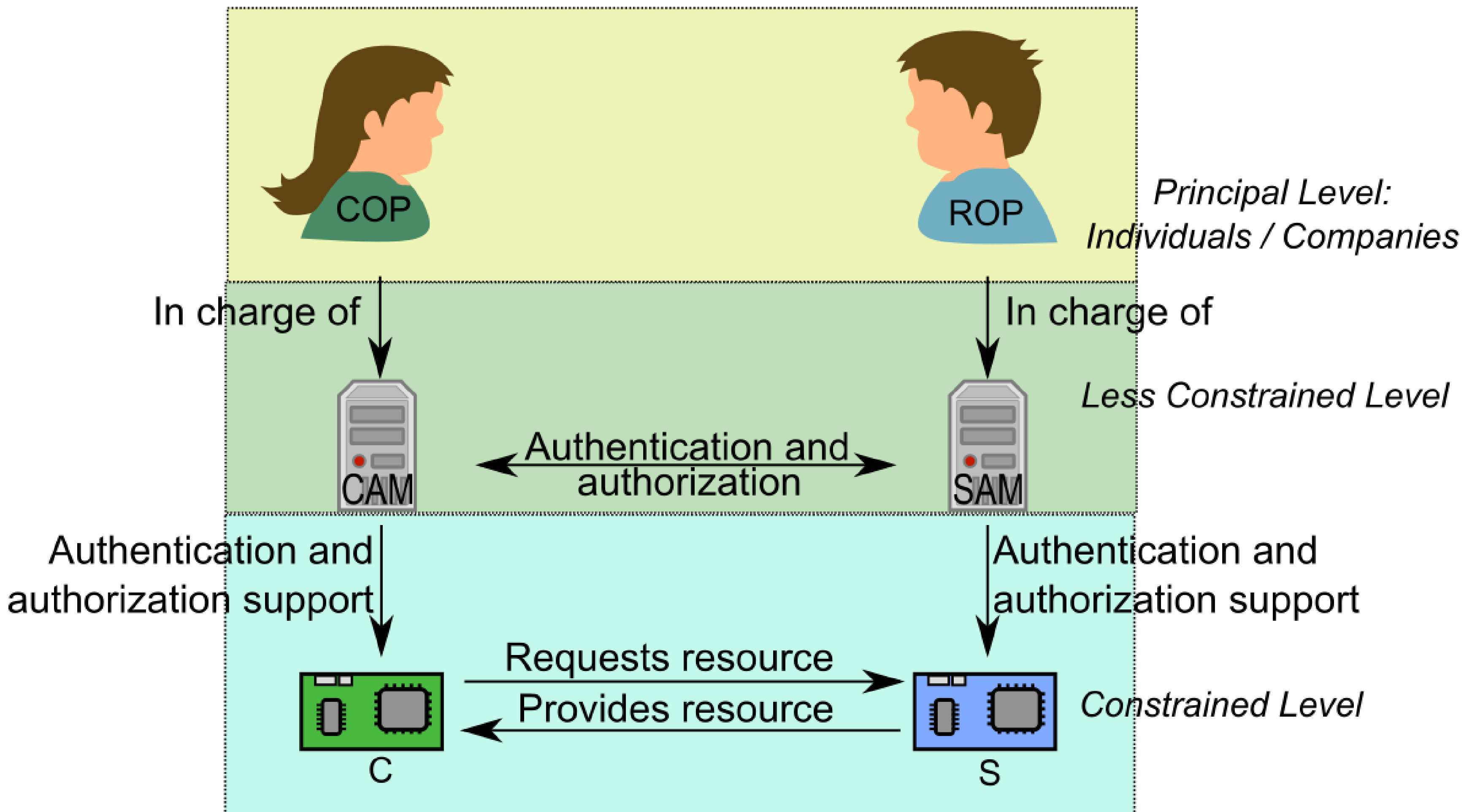
- Without (C)AM, a constrained C cannot authenticate S
- Without (C)AM, a constrained C cannot obtain COP's authorization policies

Cross-Domain with Single AM: COP in Charge of AM



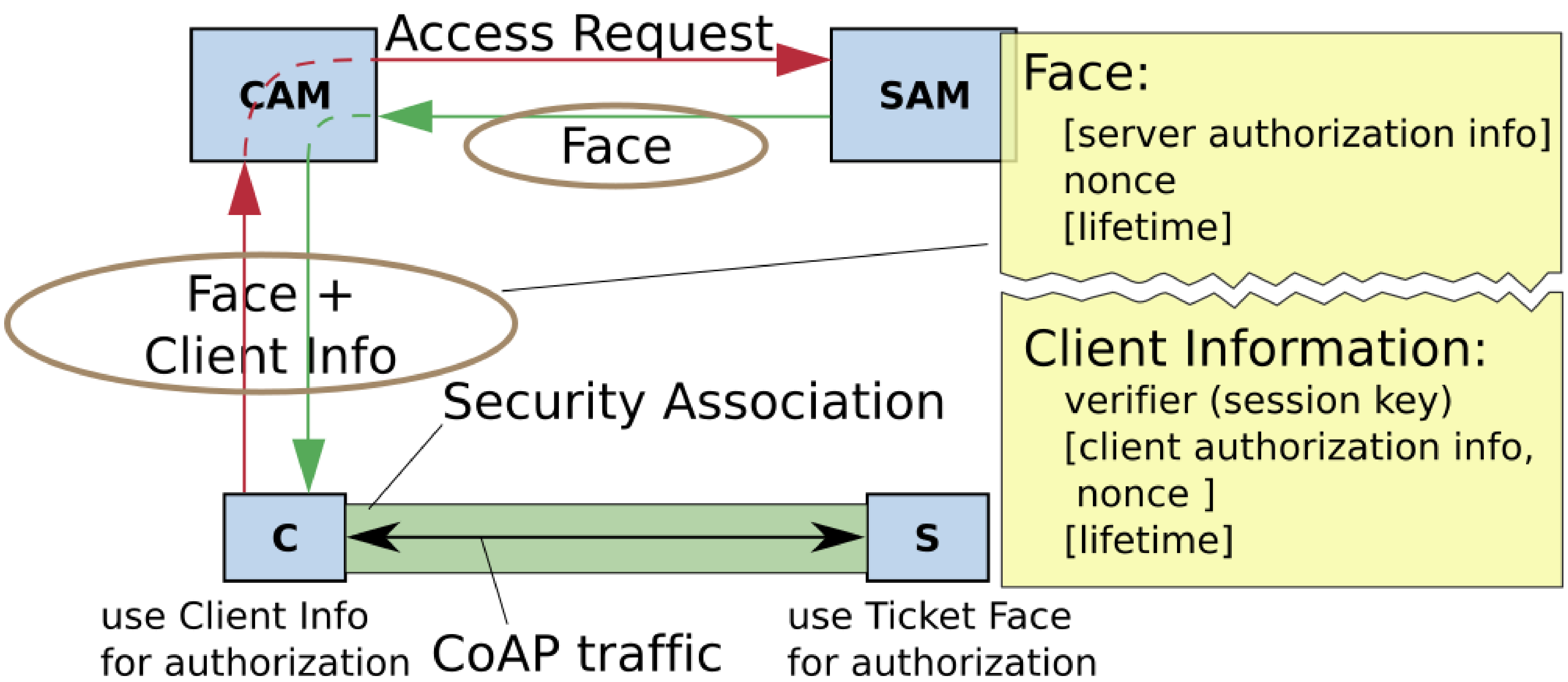
- Without (S)AM, a constrained S cannot authenticate C
- Without (S)AM, a constrained S cannot obtain ROP's authorization policies

Four legs



- Every endpoint has its own AM
- CAM is controlled by COP, SAM is controlled by ROP
- CAM helps authenticating S for C and provides authorization information about S to C
- SAM helps authenticating C for S and provides authorization information about C to S

Example Protocol Flow (DCAF)



Evaluation (DCAF-DTLS)

Reference implementation of DCAF-DTLS adds

- about 440 Bytes Code
- 54 Bytes data for ticket face
- 722 Bytes parser for CBOR payload

to existing CoAP/DTLS server (ARM Cortex M3) representing S.

Conclusion (DCAF)

- mutual authenticated authorization client-server, with symmetric keys (no need to separately obtain RPK to authenticate server)
- considers security goals of both COP and ROP.
- can make good use of DTLS-PSK
- can also use COSE with MAC, for transiting untrusted proxies

Backup

Features of DCAF

- Secure exchange of authorization information.
- Establish security association between constrained nodes (secure distribution of session keys).
- Establish security association between a constrained and a less-constrained nodes.
- Support of class-1 devices (RFC 7228).
- Requires only symmetric key cryptography on the constrained nodes.
- DCAF-DTLS supports CoAP Observe (RFC 7641) and blockwise transfer without additional overhead.
- Relieve constrained nodes from managing complex authentication and authorization tasks.

Features of DCAF (2)

- Supports multiple owners.
- Defines cross-domain constrained to constrained communication
- Relays security associations of less-constrained devices to constrained devices: Constrained devices only need the security association with their less-constrained device.
- Protects both sides of the communication (not only access to resources).
- Privacy: no device identifiers required on the constrained level.
- Provides a high level of implementation details.
- Explicit transfer of authorization information to the constrained devices possible: no additional knowledge required by the constrained nodes.
- Other formats for transmission of authorization information possible.

The DCAF universe

- Communication Security using DTLS (draft-gerdes-ace-dcaf-authorize)
- Server-Initiated Ticket Request (draft-gerdes-ace-dcaf-sitr)
- Application Level Security using COSE (draft-bergmann-ace-dcaf-cose)

related:

- Examples for using DCAF with less-constrained devices (draft-gerdes-ace-dcaf-examples)
- Authorization Transitions in the lifecycle of constrained devices (draft-gerdes-ace-a2a)