Communication in Constrained Environments

- Constrained Application Protocol (CoAP, RFC 7252)
  - designed for special requirements of constrained environments
  - Similar to HTTP (RESTful architecture style)
    - server has items of interest
    - client requests representation of current state
- Datagram Transport Layer Security (DTLS) binding
Features of DCAF

- Secure exchange of authorization information.
- Establish DTLS channel between constrained nodes.
- Establish DTLS channel between a constrained and a less-constrained nodes.
- Support of class-1 devices (RFC 7228).
- Use only symmetric key cryptography on the constrained nodes.
- Support of CoAP Observe and blockwise transfer without additional overhead.
- Relieve constrained nodes from managing complex authentication and authorization tasks.
Initial Trust Relationships

DTLS/TLS

CAM

SAM

DTLS (PSK)

C

we have these

DTLS (PSK)

S
Trust: The Complete Picture

DTLS/TLS

CAM

SAM

we have these

DTLS

(PSK)

we want this

DTLS

(PSK)

C

S
Unauthorized Access Request

Request

4.01 Unauthorized you should ask SAM
Contact S’s Less Constrained Device for Authorization

Access Request

Access Ticket
Access Ticket

Face:
[server authorization info]
timestamp
[lifetime]

Client Information:
verifier (session key)
Access Ticket: Adding Client Information

Access Request

CAM

SAM

Access Ticket

Face:
[server authorization info]
timestamp
[lifetime]

Client Information:
verifier (session key)
[client authorization info]
timestamp
[lifetime]
Use Access Ticket to Establish DTLS Channel
PSK Derivation

DTLS channel

psk_identity = Ticket Face

PSK = Verifier

derive PSK from Ticket Face and $K_{S,SAM}$
Access Ticket Parts

Access Request

CAM

Access Ticket

SAM

Face:
[server authorization info]
timestamp
[lifetime]

Client Information:
verifier (session key)
[client authorization info]
timestamp
[lifetime]
RS Permits Authorized Requests Over DTLS

DTLS channel

C

CoAP traffic

S

use Client Info for authorization

use Ticket Face for authorization
Lessons learned from the Use Cases:

▶ In some cases, binary authorization (all authenticated entities have the same authorization) is sufficient.
▶ Use Cases often require more sophisticated authorization on the client and/or on the server side.

Consequence:

▶ A solution that always transmits authorization information generates unnecessary overhead.
▶ Authorization information must be securely transmitted when needed.
Flexibility

- DCAF can be used as a simple protocol for secure transmission of DTLS pre-shared keys (implicit authorization).
- DCAF can additionally securely transmit authorization information to the server and/or the client.
- DCAF defines how combinations of actors work together.
- DCAF can be used as needed.
**Combined Actors**

- **Constrained nodes need an AM**
  - `C` (Client) and `S` (Server) are part of DTLS/TLS.
  - `CAM` (Client Authentication Module) and `SAM` (Server Authentication Module) are also part of DTLS/TLS.

- **Not-so-constrained nodes can act on their own**
  - When `C = CAM` and `S = SAM`, DTLS can be used independently.

- **CAM = SAM**
  - DTLS/TLS can be used when both the client and server authenticate themselves.

- **DTLS**
  - DTLS provides a secure communication channel between two parties.

- **DTLS/TLS**
  - Combines the security features of DTLS and TLS for enhanced security.
Evaluation

Reference implementation 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).
How to proceed

- Define interaction with protocols on the less-constrained level (how to use DCAF with existing solutions such as OAuth)
- Accept DCAF as one of the building blocks that ACE is working on.