Delegated Authenticated Authorization Framework (DCAF)

draft-gerdes-ace-dcaf-authorize

Stefanie Gerdes, Olaf Bergmann, **Carsten Bormann**
{gerdes | bergmann | cabo} @tzi.org

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

Review Comments

- Renzo: included in 04-version of DCAF:
  - Improved readability.
  - Removed inconsistencies.
  - Clarified definitions of CBOR keys.
  - Clarified handling of Ticket Request Messages.
  - Improved description of Nonces.
- Ludwig: addressed with 04-version of DCAF and DCAF-COSE
  - Also support COSE.
  - Address Server-Initiated Token Request ("Pull").
  - Adress piggy-backed protected content in SAM Information Message ("client-pull").
  - Use a resource to store tokens (DCAF-COSE).
  - Bind an authorization token to the security context between C and RS using COSE.

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 (Required for constrained environments -> t2trg Meeting Prague).
- Relay 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.
- Supports DTLS and Object Security (COSE).

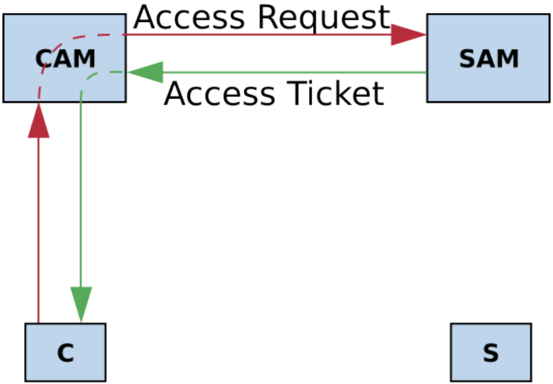
The DCAF universe

- 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)

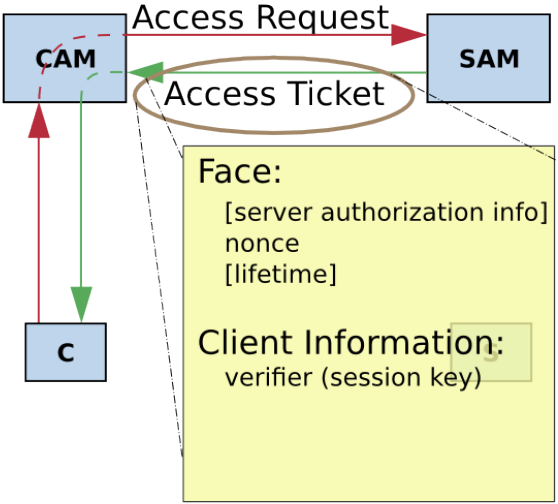
related:

- 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)

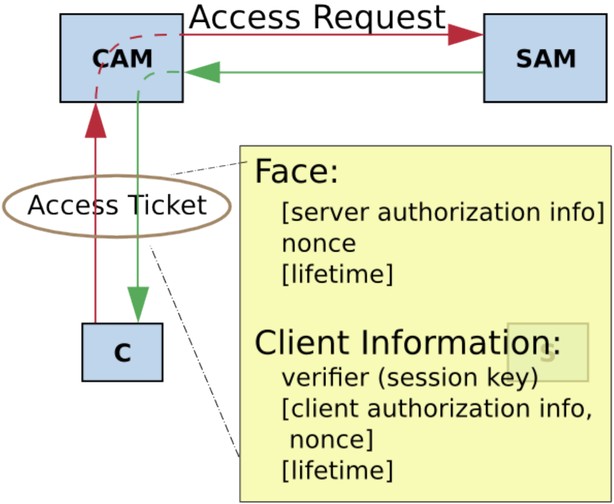
Contact S's Less Constrained Device for Authorization



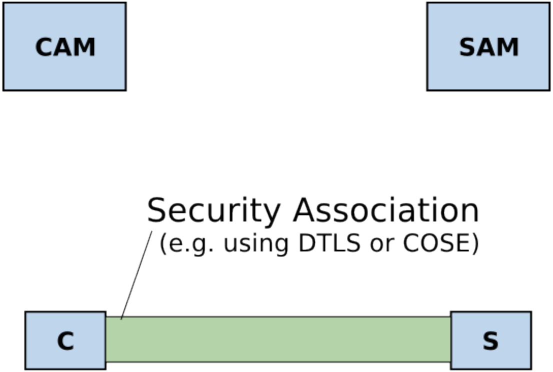
Access Ticket



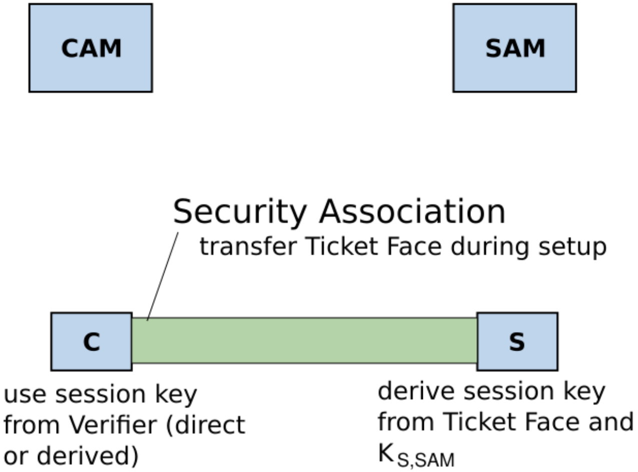
Access Ticket: Adding Client Information



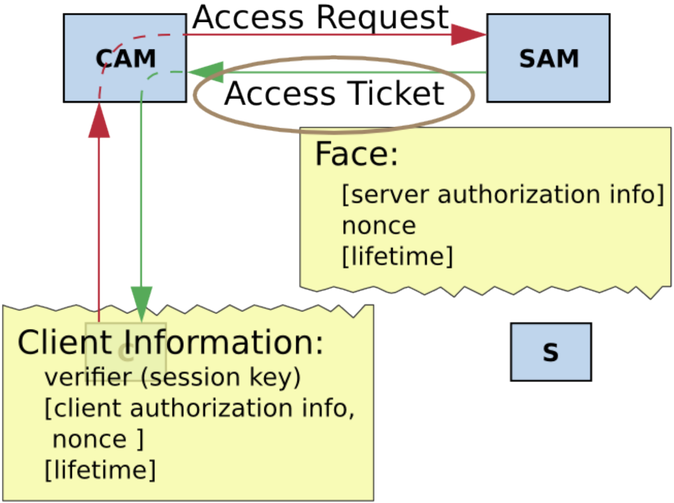
Use Access Ticket to Establish Security Context



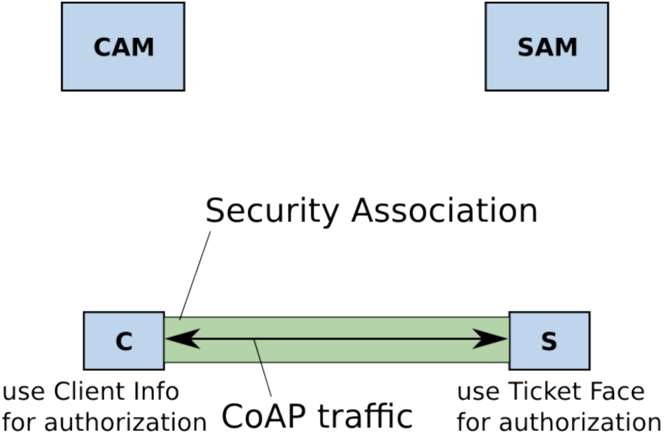
Key Derivation



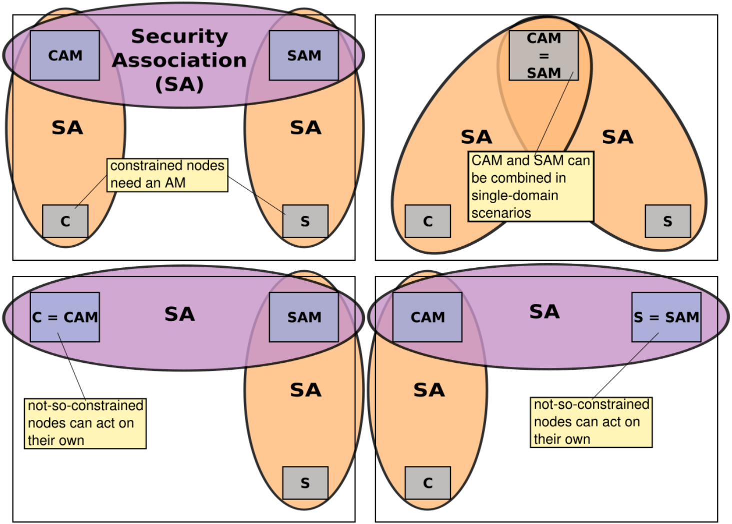
Access Ticket Parts



RS Permits Authorized Requests Over Secure Channel



Combined Actors



Flexibility

- DCAF can be used as a simple protocol for secure transmission of dynamically created session 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.

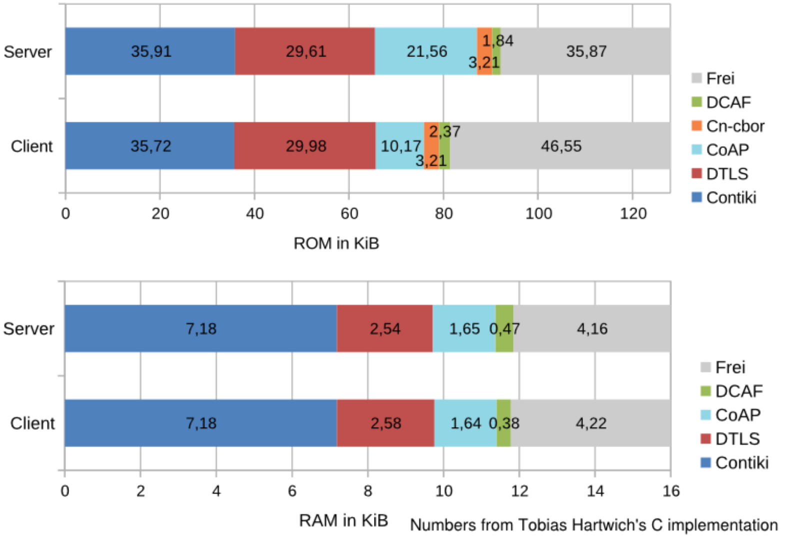
Evaluation

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).

Evaluation: DCAF Memory Usage (ROM, RAM)



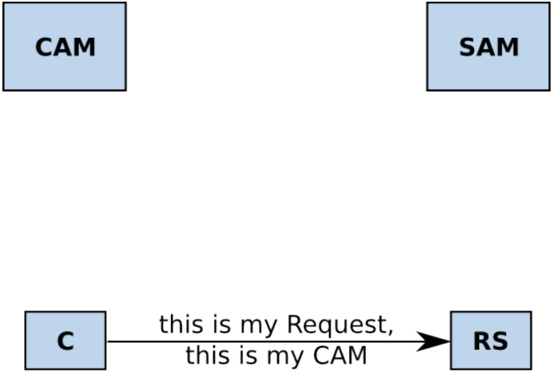
Numbers from Tobias Hartwich's C implementation for Wismote using Contiki, libcoap, tinydtls, cn-cbor

Server-Initiated Ticket Request (SITR)

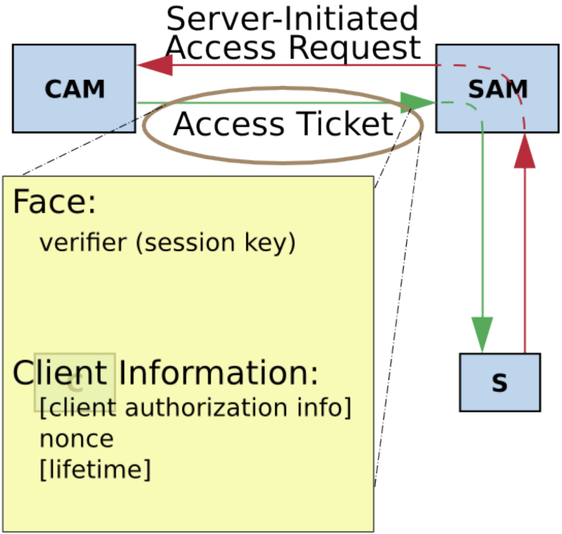
draft-gerdes-ace-dcaf-sitr

- In some scenarios, C might not be able to reach CAM or SAM
- S requests ticket for C
- C sends CAM information message to S to initiate SITR

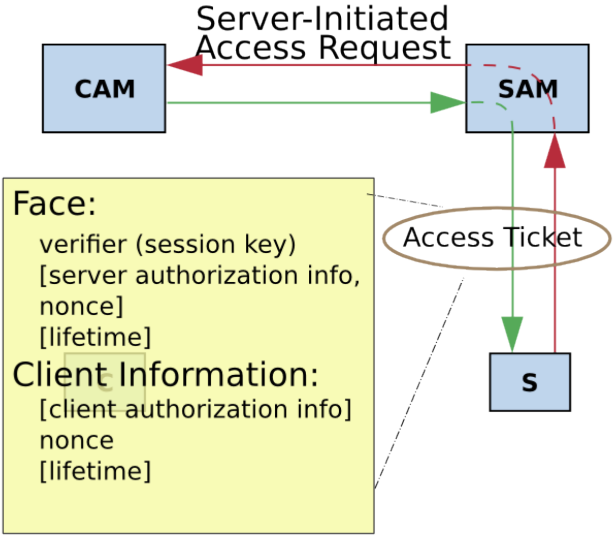
CAM Information Message



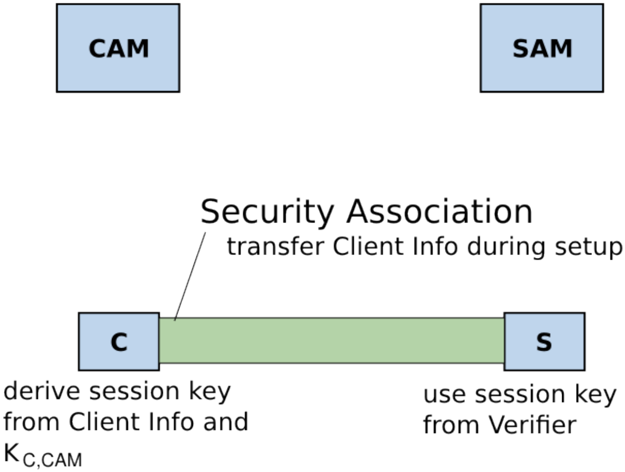
SI Access Ticket



SI Access Ticket: Adding Server Information



SIT Key Derivation



Problem with Server-Initiated Solutions

- All solutions where the server requests a ticket for the client ("Pull Model") are prone to DOS attacks.
- Use solutions where the Client request the ticket whenever possible

Summary

- mutual authentication client-server, with symmetric keys (no need to separately obtain RPK to authenticate server)
- can make good use of DTLS-PSK
- can also use COSE with MAC, for transition of untrusted proxies

DCAF-COSE vs. OSCOAP

	DCAF-COSE	OSCOAP
Changes to COSE	use COSE as is (-06) no changes required	invent "Secure Message format" (COSE-profile in Appendix A) invent "COSE Optimizations" that are not COSE-compatible (new message types, remove unprot-ected header, alg ...)
Security Context	use parameter kid (identifies auth info and session key)	invent new parameter cid (identifies cipher suite, keys, alg-specific parameters, different for client and server: "typically identifies the sending party")
Replay protection	use parameter nonce (-> local time)	invent new parameter seq (-> sequence number, no freshness information)
Re-key	Server sends SAM Information Message	"out of scope" (Section 7.1)
Signaling	use existing payload types two new options (not critical due to usual content-format handling)	implicit, new payload type new critical option
Handling of unknown options	COSE extension parameter to signal required options	not supported
RFC 7252, 7641 options block-wise	needs more work in CoRE WG	

DCAF-COSE vs. OAuth Profiling

	DCAF	OAuth Profiling
C may be class 1	yes	only in single domain
cross-domain	yes	not for constrained-to-constrained communication
multi-owner	yes	?
PoP tokens	yes	yes
Authn support	for C and RS	for RS; for C only in single domain
Authz support	for C and RS	for RS; for C only in single domain
/token	no	only in single domain
csp signaling	by RS or resource description	by AS
token introspection	optional	optional
dynamic session keys	(D)TLS-PSK COSE	(D)TLS OSCOAP
CWT	possible	possible
Privacy	no endpoint identifiers required	?

Discussion

Transport of Ticket Face for DTLS-PSK:

- `psk_identity`
  - Opaque for the client, no semantic restrictions
  - mandatory -> good interoperability
  - All known DTLS libraries pass it to the application to determine the PSK
- supplemental data (RFC 4680)
  - Client and server must support this extension.
  - Needs to define a new `SupplementalDataType` or a new `AuthzDataFormat` for `client_authz` (cf. RFC 5878)
  - Derivation of master-secret from supplemental data is not allowed (*"Information provided in a supplemental data object [...] MUST NOT need to be processed by the TLS protocol."*, RFC 4680)

How to proceed

- Accept DCAF as one of the building blocks that ACE is working on