PSA ADAC

Improving support and visibility

Frank Audun Kvamtrø

August 2024

ADAC – Some history

- Specification made by Arm
  - Nordic and other companies helped in the design
  - Subsequently inherited in the Trusted-Firmware organization
- Reference design can be found in TF-M architecture tests
  - But it is based on mbedtls_ prefixed crypto toolbox features
  - … and it is limited on how it handles hardware specific keys (read: platform keys)
- Added as an example on Musca-B1 boards
  - Not great awareness in TF-M side either in source form or in documentation
    - Some docs for PSA ADAC docs can be found in TF-M extras for Arms RSE-enabled devices which support a PSA friendly version of ADAC.

ADAC – Architecture and tooling

Architecture/docs:

- Figure taken from the PSA Authenticated Debug Access Control (ADAC) specification
- ADAC test procedures in PSA Architecture tests provide some documentation

Tools:

- Git: PSA ADAC target component
- Git: Secure Debug Manager

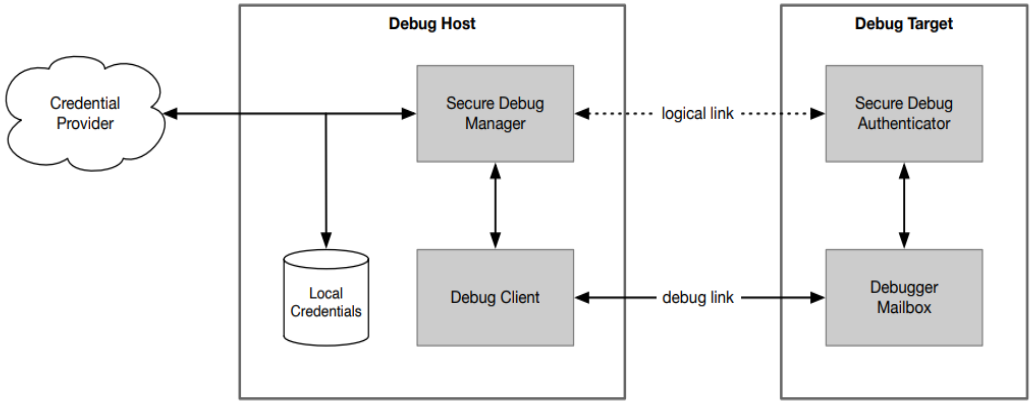

Figure 1 High level system block diagram

Not shown: The debugger mailbox is a simple request/response model implemented with word-aligned content

ADAC goals

- We want ADAC to be a **front-end item** in TF-M
  - Not hidden in vendor scope and/or hidden in a very low-level platform scope
- We want ADAC to be a **Platform Root-of-Trust** service
  - But this begs the question on placement: boot ROM vs bootloader vs Platform Root of Trust (in TF-M)
- We want to **make use of PSA crypto for all APIs**
  - Including usage of **platform keys** through the **builtin key support** or **opaque PSA driver support**
  - **Including supporting platforms like Arms RSE**
- We want to promote the **data-oriented message passing**
  - But want to investigate the possibility of **API forwarding and/or information sharing** between boot ROM/bootloaders and TF-M Image (Platform Root-of-Trust)

*We want TF-M to have a way to tie in between ADAC documentation and **standard deliverables** beyond **just the spec** and **example implementation** on specific devices*

Our suggestion: ADAC task-force:

- Arm has suggested someone take ownership of the design
  - And they are somewhat limited on capacity because of LTS releases of TF-M and Mbed TLS as well as a major reconstructing for Mbed TLS 4.0.0

Hence, we ask if it is possible to create a task-force to spread the ownership between interested parties in the Trusted Firmware organization

Open questions:

- How do we handle cooperation and ownership in Trusted Firmware?
- Should we set up an issues board to track development (by whom)?
- Should there be meetings at regular intervals to track progress?

ADAC - Special topic: “Platform keys”

TF-M introduces the concept of builtin key support in form of out-of-tree patches to Mbed TLS. This functionality is used in MCUboot to provide logic to handle keys without relying on a generic storage (ITS). This is somewhat like how TF-M has HAL APIs gives support for **Hardware Unique Keys** via target-specific resolution

We believe that the type of keys (and certificates) used for ADAC support shares a commonality with both bootloader keys and hardware unique keys, which likely should be handled as a special class altogether…

The term **platform keys** is used here to try to come up with an **umbrella term** for similar type HW keys.

Our wishes:

- We want to tie the key usage tighter towards PSA crypto and drivers
  - via **opaque driver** or **builtin key support**
  - Supported out-of-the-box e.g. in bootloader and or boot ROM (potentially forwarded)
- We will investigate using designated (and reserved) key IDs for ADAC, to simplify and standardize integration

ADAC – Special topic “Forwarded APIs”

Forwarding APIs in this scope means allowing to call into different entities outside of the TF-M secure image usable both for ***active control*** *(implemented by external entities) and for* ***info sharing*** *between them*

TF-M can standardize and show-case such forwarding through the BL2 bootloader for many device types (For LCS and ADAC and/or PSA crypto supporting platform keys)

Rationale:

- Forwarding APIs means that **ownership is centralized**, not spread
  - For keys, LCS and certificates
- Forwarding APIs means that a **PSA ADAC secure service** is relatively easy to implement
  - E.g. by implementing a pass-through ADAC library in PSA Root-of-Trust for the secure service
  - E.g. by implementing pass-through HAL APIs for crypto/keys/data residing outside of TF-M image

ADAC – Special topic “HAL abstraction”

The task-force must **investigate** and **possibly design** a **standardized HAL abstraction** for ADAC support if it is impractical to integrate towards an ADAC library API on a higher layer…

Examples:

- APIs similar to HUK for resolving, loading and using platform keys and certificates for ADAC support
- Establishing forwarding APIs to retrieve or access keys, certificates in earlier images in the boot chain, through a hypervisor and/or an external entity
- Establishing forwarding APIs to access LCS across images/entities in the system

Rationale:

- Improved and established HAL interfaces improve code reusability and ease target integration
  - And provides better visibility in TF-M project through documentation and code
- Established forwarding APIs via HALs ensure better support for different targets
  - Again, giving the opportunity to allow for better reuse.