UEFI Setvariable at Runtime Problems, status and solutions

Ilias Apalodimas, Linaro
ilias.apalodimas@linaro.org

The UEFI spec

- Getting traction within the embedded device ecosystem
- Embedded Base Boot requirements (EBBR) specification
  - Defines an interface between platform firmware and an operating system that is suitable for embedded platforms
  - Uses a small subset of EFI interfaces
  - Tries to deal with the lack of consistency between platforms which often requires per-platform customization to get an OS image to boot on multiple platforms

UEFI spec - Key storage requirements

- Defined in §32.3.6. Platform Firmware Key Storage Requirements
- PK - The public key must be stored in non-volatile storage which is tamper and delete resistant
- KEK - The public key must be stored in non-volatile storage which is tamper resistant. Careful consideration should be given to the security and configuration of any out-of-band management agent (e.g. hypervisor or service processor) such that the platform cannot exploit the management agent in order to circumvent Secure Boot
- Basically we need to store the variables in tamper protected and delete resistant medium which is always under the control of the firmware

Embedded boards status

- Having a dedicated delete resistant and tamper protected flash under the control of Secure world is rare
  - Anyone aware of any?
- Many boards have an eMMC nowadays
- What if we could store the EFI variables there ...

U-Boot status

- Supports a variety of EFI protocols. Compliant with EBBR since 2021.04
- Can boot distros on embedded boards – as long as the board drivers are included
- Has two ways of storing EFI variables
  - In a file in the ESP partition
  - In an RPMB partition of an eMMC
- Copies variables in memory and provides GetVariable and GetNextVariableName at runtime
- But doesn't support SetVariable or QueryVariableInfo at runtime

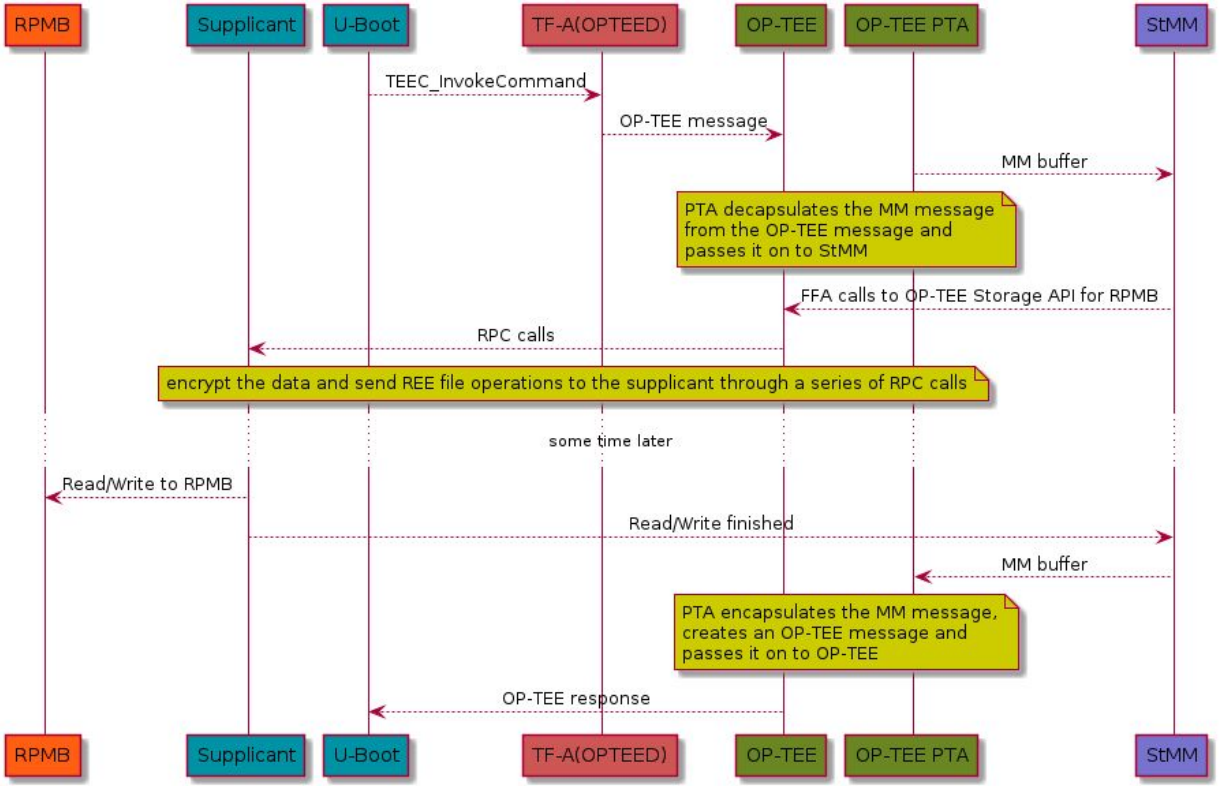
Need for RT variable services

- Runtime services are rarely needed in an OS
- EBBR only defines SetVirtualAddressMap and ConvertPointer as mandatory runtime services
- SetVariableRT is required when
  - Distro installers need to set SHIM as the first boot option or any other boot option (usually Boot0000)
  - Control BootOrder and/or BootNext and boot a different OS
  - Set OsIndications to trigger a capsule update on-disk
- Anything else? PSTORE almost impossible due to supplicant complications (see below)
- Distro installers used to fail with a scary error if SetVariable failed
  - It's just a warning now on most installers
  - The boards can still boot even without setting the Boot0000 variable

Variables in an RPMB & U-Boot

- Adheres to the EFI spec delete resistant and tamper protected medium requirements
- Needs OP-TEE, which launches StandAloneMM from EDK2 as-is
- StandAloneMM performs all the variable validation & authentication and uses OP-TEE APIs to read/write from/to the RPMB
- Authenticated variables are supported
- The cryptographic checks happen in the Secure World
- **Since the eMMC is in control of the non-secure world (firmware or OS) we need a userspace supplicant to mediate the RPMB reads & writes**

Variable accesses in U-Boot



Linaro

RPMB problems

- Doing a SetVariableRT requires the kernel to call into firmware runtime services
- Since we use OP-TEE and StandAloneMM we need to retain runtime sections of that
- Remember the supplicant requirement?
- When the request gets processed and exits OP-TEE it will search for the supplicant in that (firmware)context
- So we have to retain the supplicant as well
- And the eMMC drivers ...
- But the kernel is up and running at that point
- The eMMC is now controlled by the kernel and his userspace supplicant ...

A ‘solution’

- Patches under review
- OP-TEE provides a discoverable ‘bus’
- Devices and services reside on that bus and can be scanned e.g. a fTPM
- We can add StandAloneMM to that ‘bus’ and make it a scannable service
- Once the OP-TEE module comes up it looks for a ‘variable support’ service – in our case StandAloneMM
- When the userspace supplicant is launched a _probe() function will run, which will
  - Replace the runtime calls with OP-TEE provided ones which call directly into OP-TEE and eventually StandAloneMM
  - The efivarfs is mounted as RO if SetvariableRT isn’t supported. A notifier chain automatically remounts it as RW

The good

- eMMC and RPMB partitions are described in a spec
- StandAloneMM is self relocatable and as a result hardware independent
- Single binary that controls variables across U-Boot and EDK2
- It is also described in the PI specification
- Solves distro problems as far as SetVariable is concerned
- Allows embedded platforms to use CapsuleUpdates

The bad

- Breaks the EFI spec
- The kernel relies on a userspace application
- But this has always been the case for OP-TEE Trusted Applications which require RPMB accesses (e.g. fTPM)
  - Working towards an in-kernel supplicant which would wire up the requests directly to the eMMC subsystem
- Depending on the eMMC speed/driver the write dance might be a bit slow
  - OP-TEE also encrypts the data
- The entire solution is complex and requires 3 projects to play along – U-Boot, EDK2 and OP-TEE

Variables in a file

- UEFI authenticated variables are not supported
- The EFI spec doesn't standardize the variable format
  - But EBBR does – explicitly leaving out Authenticated variables
- U-Boot can store non-authenticated variables in a file
- Those are copied in memory during boot and provide GetVariable etc
- SetVariable is obviously not supported

Possible solutions

- Teach efitools to edit the file directly?
- Easy to implement if the format is agreed upon
  - Easy to notify/handle failures to write variables
  - Once a variable are written the kernel and file view are out of sync
  - Will need to reboot, but based on the use-cases above we usually reboot after setting a variable anyway
  - Too hacky

Or better

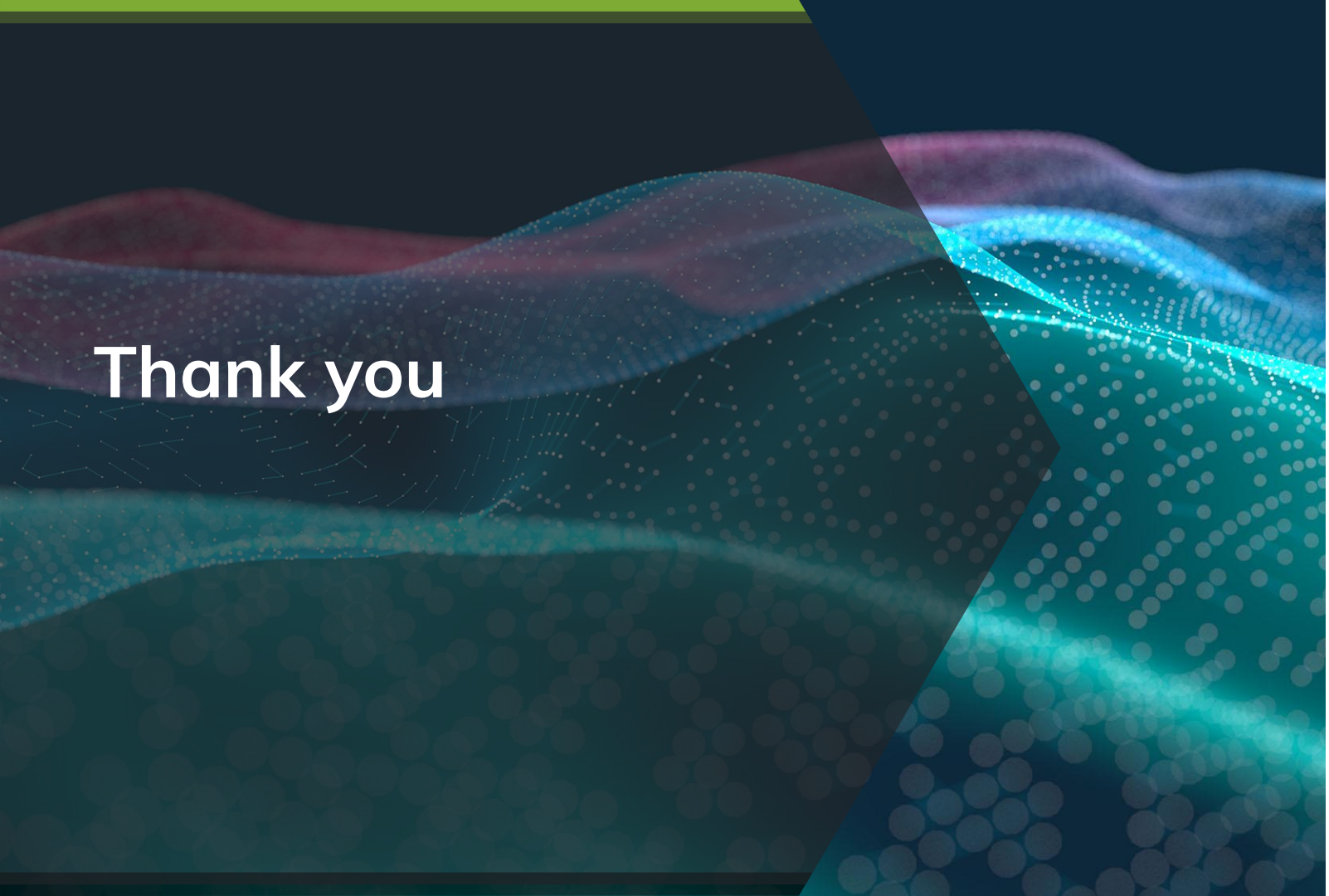
- Teach the firmware to support SetvariableRT on the memory backend
- Have a watcher application in efivarfs that syncs memory <-> file storage
  - Breaks the EFI spec since variable writes have to complete before returning success
  - Hard to notify about failures e.g. writing to file fails

Or

- Replace RT calls and write to a file?
- Too complicated for no apparent reason

Reads

- https://www.linaro.org/blog/protected-uefi-variables-with-u-boot/
- https://www.linaro.org/blog/uefi-secureboot-in-u-boot/
- https://www.linaro.org/blog/journey-to-systemready-compliance-in-u-boot/



Linaro