White Paper

A Tour Beyond BIOS with the UEFI TPM2 Support in EDKII

Jiewen Yao
Intel Corporation

Vincent J. Zimmer
Intel Corporation

September 2014

Executive Summary

This paper presents the design and boot flow of the TPM2 support in the Security Package of the EDKII. The EDKII code acts as the “Root of Trust for Measurement” (RTM) in this scenario.

Prerequisite
This paper assumes that audience has EDKII/UEFI firmware development experience. This paper assumes the audience has basic knowledge of the Trusted Platform Module (TPM) and does not introduce detail on that. For the general concept of trusted boot, the audience can refer to [Trusted Platform] and [Win8 Secure Boot].

Table of Contents

Overview 4
Introduction to the TPM2 4
Introduction to the EDKII 4
Algorithm Flexibility 6
Why algorithm flexibility? 6
How to choose the algorithm? 6
Event log reporting 7
Simplified Management 8
Supported PPI (Physical Presence Interface) 8
Dedicate BIOS Support 10
Platform Hierarchy 10
UEFI Platform Boot Process 11
TrEE protocol 11
Event Log 11
Secure Boot 11
Hardware Interface 13
TPM1.2 v.s TPM2.0 13
dTPM2.0 v.s fTPM2.0 13
TPM2 ACPI table 14
dTPM2.0 FIFO v.s CRB 14
Platform Reset Attack Mitigation 15
Memory override 15
Conclusion 16
Glossary 17
References 18

Overview

Introduction to the TPM2

TPM2 (trusted platform module 2) is defined in TCG (Trusted Computing Group). The goal is to replace the TPM1.2 (Trusted Platform Module 1.2) because of various limitations in TPM1.2, including SHA-1 nature of TPM 1.2 [SHA-1]. See [TPM2] for detail.

A) For BIOS perspective, the main difference is below:
   1) Algorithm Flexibility: TPM1.2 only supports SHA1, TPM2.0 can support any hash algorithm. So TPM2 does not use EFI_TCG_PROTOCOL [TCG Protocol], but EFI_TrEE_PROTOCOL [TrEE Protocol]
   2) Simplify management: TPM1.2 has enable/activate/disable/deactivate state. TPM2 removes those concepts. So TPM2 replaces ACPI PPI interface [TCG PPI] with a simplified version PPI interface [TrEE ACPI].
   3) Dedicate BIOS Support: TPM1.2 can only be taken owner ship and used by OS. TPM2 adds a storage hierarchy controlled by platform firmware, so OEM can use this hierarchy regardless of the support provided to OS. This solution is OEM specific, so there is no generic solution.

B) Some BIOS features in TPM1.2 and TPM2 are the same or only have minor differences.
   1) UEFI Platform Boot Process: This PCR measurement component is nearly same in [TCG Platform]. TPM2 [TrEE Protocol] has some special requirements for PCR7, such as measuring the UEFI Secure Boot authorities [UEFI Secure Boot].
   2) TPM Hardware Interface: The first generation discrete TPM2 can still use TPM1.2 FIFO interface defined in [TCG TIS] plus Cancel [TrEE ACPI]. The later TPM2 can use new CRB interface defined in [TCG PTP]
   3) Platform Reset Attack Mitigation [TPM MOR]: This is pure software/firmware solution.

Introduction to the EDKII

EDKII is open source implementation for UEFI firmware. TPM2 support in UEFI is added to SecurityPkg (https://svn.code.sf.net/p/edk2/code/trunk/edk2/SecurityPkg).

The main [TrEE Protocol] related component is at
https://svn.code.sf.net/p/edk2/code/trunk/edk2/SecurityPkg/Tcg/TrEEPei and
https://svn.code.sf.net/p/edk2/code/trunk/edk2/SecurityPkg/Tcg/TrEEDxe.

The main [TrEE ACPI] related component is at
https://svn.code.sf.net/p/edk2/code/trunk/edk2/SecurityPkg/Tcg/TrEESmm and
https://svn.code.sf.net/p/edk2/code/trunk/edk2/SecurityPkg/Library/DxeTrEEPhysicalPresenceLib.

In next 6 sections, we will introduce all of above parts in detail.

Summary
This section provided an overview of TPM2 BIOS related components and EDKII.

Algorithm Flexibility

In the first three sections, the focus will include the differences between TPM1.2 and TPM2.0.

Why algorithm flexibility?

TPM1.2 is constrained by its data structure to using RSA and SHA1. However, SHA1 is deprecated due to its known weakness. So we need a solution. Simply to shift to another hash algorithm is not necessarily safe in the long term. Also, different geographies may want to choose different hash algorithm, like SHA256 versus SM3 [SM3].

As a solution, TPM2.0 specification just defines the framework of algorithm. TPM2.0 vendor can decide to implement which algorithm in the chip. TPM2.0 specification predefined SHA1, SHA256, SHA384, SHA512, and SM3_256. And it can be extended in the future.

How to choose the algorithm?

A TPM2.0 may maintain multiple banks of PCR. A PCR bank is a collection of PCR that are Extended with the same hash algorithm. PCR banks are identified by the hash algorithm used to Extend the PCR in that bank.

For example, a TPM2.0 may support SHA1, SHA256, and SM3_256 at same time. That means this TPM2.0 has 3 banks – SHA1 Bank, SHA256 Bank and SM3_256 Bank. TPM2.0 exposes a *TPM2_PCR_Allocate*, which can be used to allocate active Bank. For example, if only SHA1 and SHA256 are needed, BIOS can call *TPM2_PCR_Allocate*(SHA1 + SHA256). If only SM3_256 is needed, BIOS can call *TPM2_PCR_Allocate*(SM3_256). Then at next boot, only the allocated PCR Bank is active and visible.

In EDKII, we have Tpm2CommandLib/Tpm2PcrAllocate() at (https://svn.code.sf.net/p/edk2/code/trunk/edk2/SecurityPkg/Library/Tpm2CommandLib/Tpm2Integrity.c). So user can use this API to allocate desired PCR Bank.
Then user can choose HashLibTpm2 (https://svn.code.sf.net/p/edk2/code/trunk/edk2/SecurityPkg/Library/HashLibTpm2) to extend data to PCR register. This library uses *TPM2_HashSequenceStart*/ *TPM2_SequenceUpdate*/ *TPM2_EventSequenceComplete* for large data, or *TPM2_PCR_Event* for small data.

Above is a hardware solution. There might be concern on performance because the TPM2 may use the FIFO interface on the LPC bus [LPC]. As an alternative, a BIOS implementation may choose a software solution by calculating all the hash values by main CPU, and only call *TPM2_PCR_Extend* as last step. If so, the user needs to choose the other hash library instance at (https://svn.code.sf.net/p/edk2/code/trunk/edk2/SecurityPkg/Library/HashLibBaseCryptoRouter)
HashLibBaseCryptoRouter is just a router, so the user also needs to choose which hash algorithm is used. EDKII provide 2 default ones, SHA1 is at (https://svn.code.sf.net/p/edk2/code/trunk/edk2/SecurityPkg/Library/HashInstanceLibSha1) and SHA256 is at (https://svn.code.sf.net/p/edk2/code/trunk/edk2/SecurityPkg/Library/HashInstanceLibSha256)

See below figure, the left side yellow box, TPM2 instance for HashLib is hardware solution. It depends on *TPM_PCR_Allocate* to decide which PCR bank to use. The right side green box, Crypto instance for HashLib is software solution. PCR selection depends on which hash algorithm instance to be linked at build time, and policy selected at runtime. For example, an OEM can build in all SHA1/SHA256/SM3 instances into one BIOS, and expose a BIOS setup option to let the user or OS choose which one is desired. And only the hash algorithm chosen will take effect in the end.

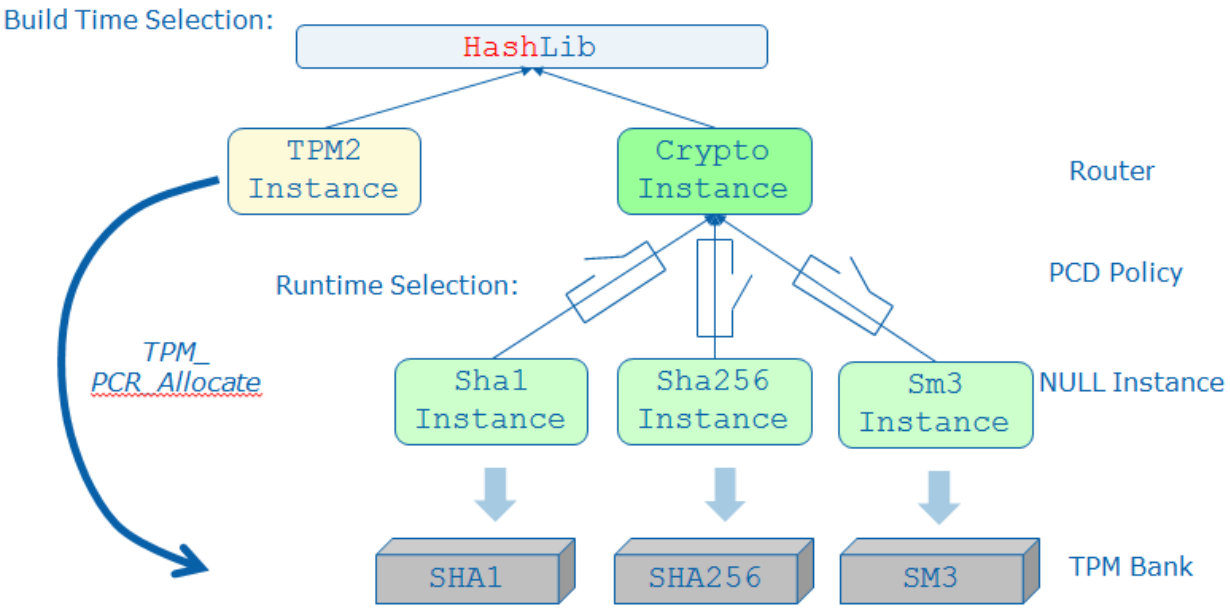

Figure 1 hash algorithm selection

Event log reporting

[TrEE Protocol] 1.0 version only supports TCG1.2 log format, so only SHA1 is in the event log area in this 1.0 version specification.

Summary

This section describes the TPM2 algorithm flexibility support in EDKII.

Simplified Management

Supported PPI (Physical Presence Interface)

TPM2 removed enable/disable/activate/deactivate concept, so the PPI interface is simplified. According to [TrEE ACPI], it only defines:

1) *TPM2_ClearControl*(NO) + *TPM2_Clear*
2) SetNoPPIClear_False
3) SetNoPPIClear_True

The TPM2 PPI design is similar as TPM1.2. There is TrEE SMM driver (https://svn.code.sf.net/p/edk2/code/trunk/edk2/SecurityPkg/Tcg/TrEESmm) to expose ACPI interface, accept request from OS and write to UEFI variable. And there is a PPI library (https://svn.code.sf.net/p/edk2/code/trunk/edk2/SecurityPkg/Library/DxeTrEEPhysicalPresenceLib) to process the request. The library should be linked with BDS and called before PI EndOfDxe event.

The PPI variable defined in (https://svn.code.sf.net/p/edk2/code/trunk/edk2/SecurityPkg/Include/Guid/TrEEPhysicalPresenceData.h) is implementation specific, not defined in any standard. EDKII uses 2 different variables. L"TrEEPhysicalPresence" variable is to record the OS request, it is read-write. L"TrEEPhysicalPresenceFlags" variable is to save TPM Management Flags and corresponding operations. It should be protected from malicious software. So EDKII use variable lock protocol to set it to be read-only after PPI process.

See the right side of below picture. The left side is for MOR which will be introduced later.

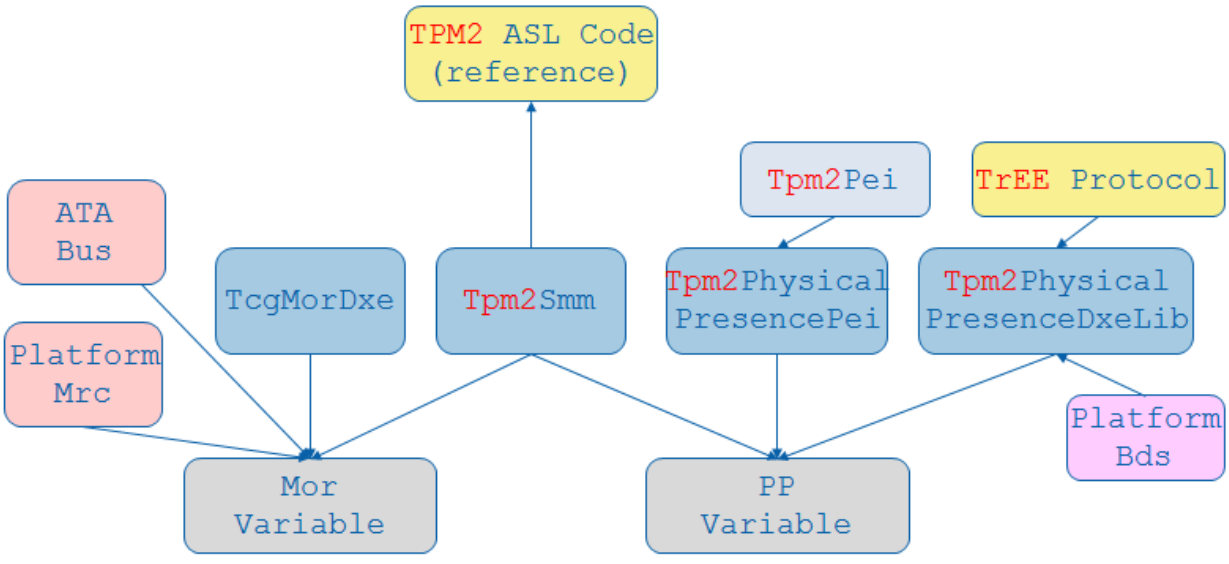

Figure 2 EDKII TPM2 PPI and MOR

Summary
This section describes the TPM2 management, PPI related implementation.

Dedicate BIOS Support

TPM 2.0 adds a Storage hierarchy controlled by platform firmware, letting the OEM benefit from the cryptographic capabilities of the TPM regardless of the support provided to the OS. This solution is OEM specific, so there is no generic solution.

Platform Hierarchy

EDKII provides library for TPM2 hierarchy, like Tpm2HierarchyChangeAuth. (https://svn.code.sf.net/p/edk2/code/trunk/edk2/SecurityPkg/Library/Tpm2CommandLib/Tpm2Hierarchy.c)

For example, *TPM_HierarchyChangeAuth* should be called during BIOS boot. The Auth value should be saved as secret, so that only platform firmware can send command required platform Auth.

Summary
This section describes the TPM2 platform hierarchy. Again it is not documented as standard on how to use platform hierarchy, which is designed for OEM use.

UEFI Platform Boot Process

We introduced the difference between TPM1.2 and TPM2.0. Now let's focus on similar feature between TPM1.2 and TPM2.0 in next 3 sections.

TrEE protocol

TPM2.0 uses [TrEE Protocol], while TPM1.2 uses [TCG Protocol]. Detail below:

1) TrEE_PROTOCOL.GetCapability() returns a capability structure, which is similar as TCG_PROTOCOL.StatusCheck().
2) TrEE_PROTOCOL.GetEventLog() returns event log location, which is also similar as TCG_PROTOCOL.StatusCheck().
3) TrEE_PROTOCOL.HashLogExtendEvent() hash data and record to event log, which is similar as TCG_PROTOCOL.HashLogExtendEvent (). There is one special feature in TrEE protocol is that, it can input a flag to hash PE/COFF image directly.
4) TrEE_PROTOCOL.SubmitCommand() runs TPM2 command, which is similar as TCG_PROTOCOL.PassThroughToTpm().

The TrEE protocol implantation is at (https://svn.code.sf.net/p/edk2/code/trunk/edk2/SecurityPkg/Tcg/TrEEDxe).

Event Log

Most UEFI platform boot process [TCG Platform] should be unchanged between TPM1.2 and TPM2.0. If a component is measured in TPM1.2 boot, most likely it is measured in TPM2.0 boot. In PEI phase, the event log (measurement for CRTM Version, main BIOS, FvImage) is recorded at (https://svn.code.sf.net/p/edk2/code/trunk/edk2/SecurityPkg/Tcg/TrEEPei). In DXE phase, the event log (measurement for boot variable, SMBIOS table, multi-processor information, etc) is recorded at (https://svn.code.sf.net/p/edk2/code/trunk/edk2/SecurityPkg/Tcg/TrEEDxe).

In order to support TCG trusted boot, BIOS will also measure PE/COFF image and GPT partition. The TPM2 version is at (https://svn.code.sf.net/p/edk2/code/trunk/edk2/SecurityPkg/Library/DxeTpm2MeasureBootLib).

Platform is also required to use TrEE Protocol or TCG protocol to measure platform specific component like static ACPI table, or CPU Microcode. EDKII provide a TPM measurement library (https://svn.code.sf.net/p/edk2/code/trunk/edk2/SecurityPkg/Library/DxeTpmMeasurementLib) to abstract the TPM1.2 or TPM2.0. So platform just need call TpmMeasureAndLogData() API no matter there is TPM1.2 on system or TPM2.0 on system.

Secure Boot

According to [TrEE Protocol], for Windows, PCR[7] is used to reflect the UEFI 2.3.1 Secure Boot policy. This policy relies on the firmware authenticating all boot components launched

prior to the UEFI environment and the UEFI platform initialization code (or earlier firmware code) invariantly recording the Secure Boot policy information into PCR[7].

Platform firmware adhering to the policy must therefore measure the following values into PCR[7]: PK, KEK, db/dbx, L"SecureBoot", and the entries in the EFI_IMAGE_SECURITY_DATABASE that are used to validate EFI Drivers or EFI Boot Applications in the boot path.

The first 4 items are recorded at UEFI Auth Variable driver.
(https://svn.code.sf.net/p/edk2/code/trunk/edk2/SecurityPkg/VariableAuthenticated/RuntimeDxe/Measurement.c).
The last item is recorded at UEFI Secure Boot Image Verification library.
(https://svn.code.sf.net/p/edk2/code/trunk/edk2/SecurityPkg/Library/DxeImageVerificationLib/Measurement.c).

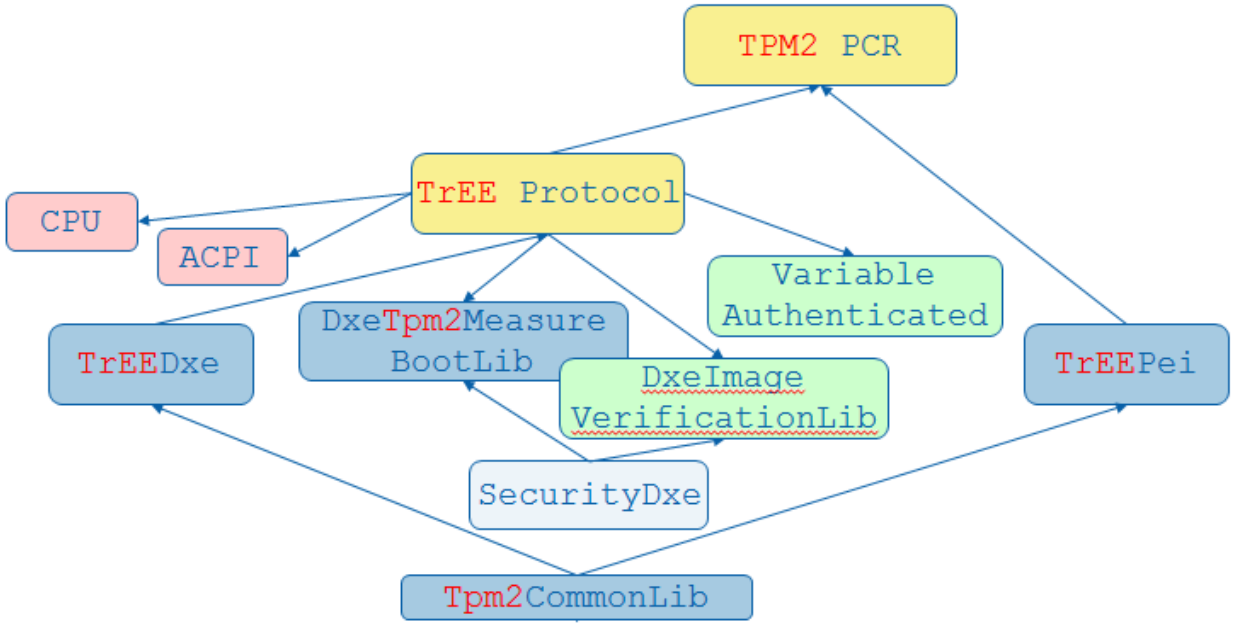

Figure 3 EDKII TPM2 measured boot

Summary
This section describes the UEFI Platform Boot Process for TPM2.

Hardware Interface

TPM1.2 v.s TPM2.0

We know that TPM2.0 hardware may use same [TPM TIS] interface as TPM1.2. So how BIOS know if there is TPM1.2 on platform or TPM2.0 on platform?
EDKII TrEEConfig driver (https://svn.code.sf.net/p/edk2/code/trunk/edk2/SecurityPkg/Tcg/TrEEConfig) will do the detection at BIOS boot runtime. During normal boot, TpmDetection.c will check if dTPM present by reading TPM Base address (0xFED40000 for PC). If dTPM is present, this module will send TPM1.2 startup command. If startup success, it means TPM1.2 on system. Or it means TPM2.0 on system. This TPM module selection information will be saved into a PCD - PcdTpmInstanceGuid. Later, TcgPei/Dxe/Smm and TrEEPei/Dxe/Smm will check this PCD to determine if it need running. So all TPM1.2 and TPM2.0 code are integrated, but at most one of them will run. Worst case is that no TPM module detected, so that none of them will run.

Later in DxePhase, this PCD will be saved into a UEFI variable, with read-only attribute. So in S3 resume phase, there is no need to do the detection again, but to use the UEFI variable data.

dTPM2.0 v.s fTPM2.0

TPM2.0 may have different implementation. dTPM2.0 uses TCG defined standard interface. A platform may have fTPM2.0 (firmware TPM2.0) implementation, which does not use [TPM TIS] interface. Instead fTPM2.0 can have a Control Area interface defined in [TrEE ACPI]. If a platform has fTPM2.0, it might need have its own TrEEConfig driver and have its own logic on fTPM2.0 detection.

In EDKII, we design a set of library to hide the TPM2.0 device difference. In previous figure 3, all TrEE drivers call Tpm2CommandLib (https://svn.code.sf.net/p/edk2/code/trunk/edk2/SecurityPkg/Library/Tpm2CommandLib), which hide how to send command to TPM2 device.

See figure 4 below, the Tpm2CommandLib calls Tpm2DeviceLib to sub TPM2 command. Tpm2DeviceLib can have different instances. E.g. Tpm2DeviceLibTrEE (https://svn.code.sf.net/p/edk2/code/trunk/edk2/SecurityPkg/Library/Tpm2DeviceLibTrEE) is the instance to consume TrEE protocol, which is typically used by PlatformBds and all TPM2 application.

Tpm2DeviceLibRouter (https://svn.code.sf.net/p/edk2/code/trunk/edk2/SecurityPkg/Library/Tpm2DeviceLibRouter) is another instance, which is typically used by TrEEPei and TrEEDxe driver. This library instance can do runtime TPM2.0 interface selection. Tpm2DeviceLibRouter need link different TPM2 device NULL instance. EDKII provides dTPM2.0 instance (https://svn.code.sf.net/p/edk2/code/trunk/edk2/SecurityPkg/Library/Tpm2DeviceLibDTpm). A

platform may provide fTPM2.0 instance. During runtime, Tpm2DeviceLibRouter will do selection based upon the PCD – PcdTpmInstanceGuid value set by TrEEConfig driver, and send command to the expected TPM device.

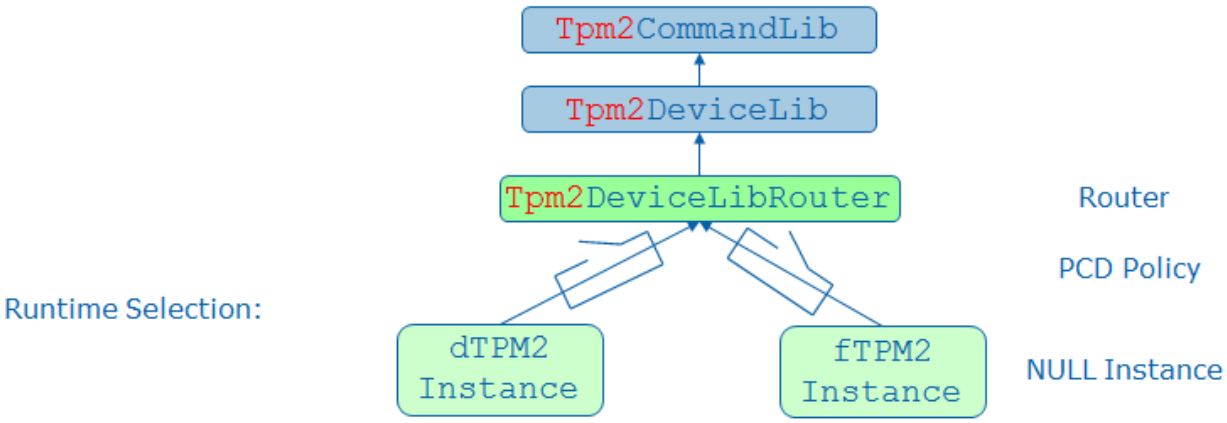

Figure 4 EDKII TPM2 device selection

TPM2 ACPI table

Now we resolve the BIOS issue on TPM2 device selection. Then how OS know the information? [TrEE ACPI] defined a TPM2 ACPI table, which has a field - Start Method. 6 is reserved for the Memory mapped I/O Interface (TIS 1.2+Cancel), which is for early dTPM2.0. 7 or 8 means using the Command Response Buffer Interface, which is for fTPM.

dTPM2.0 FIFO v.s CRB

Later [TCG PTP] specification brings Command Response Buffer interface into standard, so a dTPM2.0 implementation may also choose CRB interface.

Summary

This section describes the TPM2 hardware interface.

Platform Reset Attack Mitigation

Memory override

The memory override feature [TCG MOR] is to mitigate platform reset attack. TPM2.0 has exactly same interface with TPM1.2.

The ACPI OS interface is produced by TrEESmm driver. (https://svn.code.sf.net/p/edk2/code/trunk/edk2/SecurityPkg/Tcg/TrEESmm). TrEESmm driver will write MOR variable, which is defined in [TCG MOR] specification.

In next boot, the MOR variable will be checked by a silicon specific MRC (Memory reference code) driver. If MOR bit is set, the MRC code will clear memory right after memory initialization to remove secret from DRAM.

An ATA bus driver (https://svn.code.sf.net/p/edk2/code/trunk/edk2/MdeModulePkg/Bus/Ata/AtaBusDxe) can also refer to MOR variable. If MOR bit is set, ATA bus driver will send TPer Reset command to reset eDrive to lock all protected bands. Again, this is to protect the secret in eDriver.

Finally, the MOR driver (https://svn.code.sf.net/p/edk2/code/trunk/edk2/SecurityPkg/Tcg/MemoryOverwriteControl) will clear MOR bit.

See figure 2 left side.

Summary
This section describes the TPM MOR feature.

Conclusion

TPM is important for TCG trusted boot and need be supported in UEFI BIOS. TPM2 is latest TPM standard. This paper describes detail boot flow of TPM2 based trusted boot in the EDKII SecurityPkg.

Glossary

PCR – Platform Configuration Register (in TPM)

PI – Platform Initialization. Volume 1-5 of the UEFI PI specifications.

TCG – Trusted Computing Group.

TPM – Trusted Platform Module

UEFI – Unified Extensible Firmware Interface. Firmware interface between the platform and the operating system. Predominate interfaces are in the boot services (BS) or pre-OS. Few runtime (RT) services.

References

[EDK2] UEFI Developer Kit www.tianocore.org

[LPC] Low pin-count bus http://www.intel.com/design/chipsets/industry/lpc.htm

[SHA-1] Bruce Schneier, "SHA-1 broken"
https://www.schneier.com/blog/archives/2005/02/sha1_broken.html

[SM3] SM3 Hash function, http://tools.ietf.org/id/draft-shen-sm3-hash-00.txt

[TPM2] Trusted Platform Module Library Specification, Family "2.0",
http://www.trustedcomputinggroup.org/resources/tpm_library_specification

[TCG MOR] PC Client Work Group Platform Reset Attack Mitigation Specification
http://www.trustedcomputinggroup.org/resources/pc_client_work_group_platform_reset_attack_mitigation_specification_version_10

[TCG PTP] PC Client Platform TPM Profile Specification
http://www.trustedcomputinggroup.org/resources/pc_client_platform_tpm_profile_ptp_specification

[TCG Platform] TCG EFI Platform Specification
http://www.trustedcomputinggroup.org/resources/tcg_efi_platform_specification

[TCG PPI] TCG Physical Presence Interface Specification
http://www.trustedcomputinggroup.org/resources/tcg_physical_presence_interface_specification

[TCG Protocol] TCG EFI Protocol Specification
http://www.trustedcomputinggroup.org/resources/tcg_efi_protocol_specification

[TCG TIS] PC Client Work Group PC Client Specific TPM Interface Specification
http://www.trustedcomputinggroup.org/resources/pc_client_work_group_pc_client_specific_tpm_interface_specification_tis

[TrEE Protocol] Trusted Execution Environment EFI protocol http://msdn.microsoft.com/en-us/library/windows/hardware/jj923068.aspx

[TrEE ACPI] Trusted Execution Environment ACPI Profile http://msdn.microsoft.com/en-us/library/windows/hardware/jj923067.aspx

[Trusted Platform] Zimmer, Dasari, Brogan, "Trusted Platforms - UEFI, PI and TCG-based firmware" September 2009, http://www.cs.berkeley.edu/~kubitron/courses/cs194-24-S14/hand-outs/SF09_EFIS001_UEFI_PI_TCG_White_Paper.pdf

[UEFI] Unified Extensible Firmware Interface (UEFI) Specification, Version 2.4.b www.uefi.org

[UEFI Book] Zimmer, et al, "Beyond BIOS: Developing with the Unified Extensible Firmware Interface," 2nd edition, Intel Press, January 2011

[UEFI Overview] Zimmer, Rothman, Hale, "UEFI: From Reset Vector to Operating System," Chapter 3 of *Hardware-Dependent Software*, Springer, February 2009

[UEFI Secure Boot] Magnus Nystrom, Martin Nicholes, Vincent Zimmer, "UEFI Networking and Pre-OS Security," in *Intel Technology Journal - UEFI Today: Boostrapping the Continuum*, Volume 15, Issue 1, pp. 80-101, October 2011, ISBN 978-1-934053-43-0, ISSN 1535-864X https://www.researchgate.net/publication/235258577_UEFI_Networking_and_Pre-OS_Security/file/9fcfd510b3ff7138f4.pdf

[UEFI PI Specification] UEFI Platform Initialization (PI) Specifications, volumes 1-5, Version 1.3 www.uefi.org

[Win8 Secure Boot] Windows 8 Boot Security FAQ http://technet.microsoft.com/en-us/windows/dn168169.aspx

Authors

Jiewen Yao (jiewen.yao@intel.com) is EDKII BIOS architect, EDKII FSP package maintainer with Software and Services Group at Intel Corporation.

Vincent J. Zimmer (vincent.zimmer@intel.com) is a Senior Principal Engineer with the Software and Services Group at Intel Corporation. Vincent chairs the security and network subteams in the UEFI Forum and was the co-author of the original EFI TCG Protocol and Platform Specifications for TPM1.2.