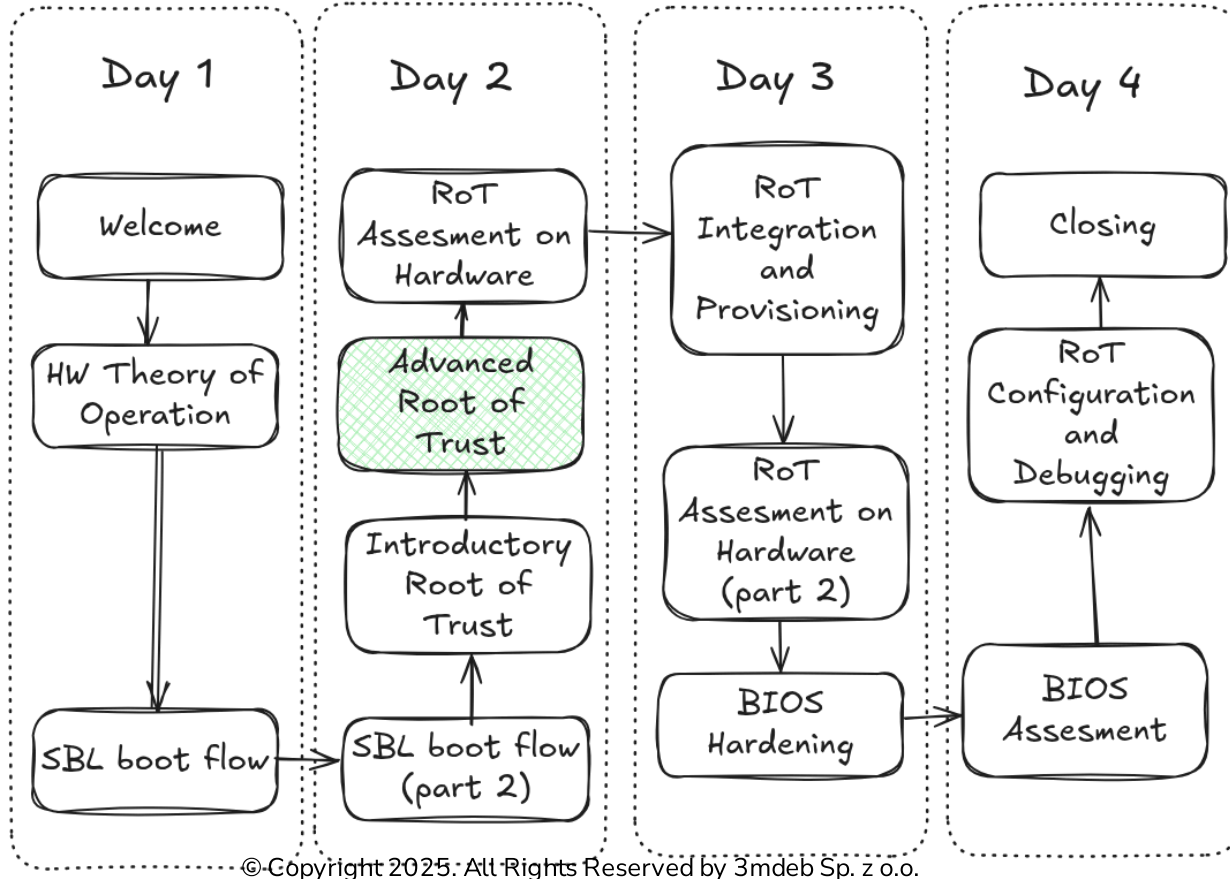


Root of Trust and Chain of Trust Technologies (part 2)

Advanced Root of Trust

Where we are in the course



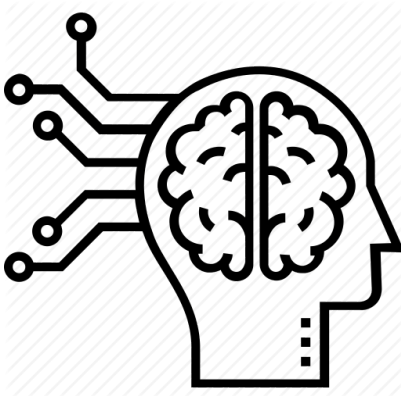
Goals of the Presentation

In this presentation, we aim to:

- Understand difference between Static and Dynamic Root of Trust
- Discuss Example implementations of S-RTM and D-RTM
- Explain role of Intel FIT and ACM for Intel Boot Guard and TXT

S-RTM vs D-RTM "holy war"

- *Static Root of Trust for Measurement (S-RTM)*
 - requires platform reset to establish a trusted state
 - hard to update (re-evaluation of PCR values)
 - vendor-specific (NXP HAB, Intel Secure Boot/Boot Guard, AMD HVB) requires proprietary tools and NDAs
 - well-established in the IBV environment
 - there are open implementations based on coreboot and vboot (Chromebook, PC Engines)
- *Dynamic Root of Trust for Measurement (D-RTM)*
 - works even in a compromised environment
 - avoid platform reboot to establish a secure state
 - avoid the problem with a secure re-evaluation of PCR values
 - small enough to pass common criteria certification
 - open implementation coming to GRUB and Linux kernel



- Intel Trusted Execution Technology (TXT)
 - uses binary blobs: ACM BIOS and ACM SINIT modules,
 - leverages `GETSEC[SENTER]` instruction and other leaf functions of it,
- AMD Secure Startup
 - it is possible to create a fully open-source implementation,
 - leverages `SKINIT` instruction introduced with the AMD-V extension,
- Qualcomm - Supposedly comply with [Arm D-RTM specification](#)
- Microsoft provides recommendations for their Secured Core PC for Intel, AMD, and Qualcomm SoC.

Quiz

Quiz

What does it mean that RoT is static?

Quiz

What does it mean that RoT is static?

- It performs its assessment only at a specific/static point in time

Quiz

What does it mean that RoT is static?

- It performs its assessment only at a specific/static point in time

What does it mean that RoT is dynamic?

Quiz

What does it mean that RoT is static?

- It performs its assessment only at a specific/static point in time

What does it mean that RoT is dynamic?

- It performs its assessment only at an arbitrary point in time

Quiz

What does it mean that RoT is static?

- It performs its assessment only at a specific/static point in time

What does it mean that RoT is dynamic?

- It performs its assessment only at an arbitrary point in time

How is D-RTM typically implemented?

Quiz

What does it mean that RoT is static?

- It performs its assessment only at a specific/static point in time

What does it mean that RoT is dynamic?

- It performs its assessment only at an arbitrary point in time

How is D-RTM typically implemented?

- By dedicated CPU instruction

Quiz

What does it mean that RoT is static?

- It performs its assessment only at a specific/static point in time

What does it mean that RoT is dynamic?

- It performs its assessment only at an arbitrary point in time

How is D-RTM typically implemented?

- By dedicated CPU instruction

What are the 2 main D-RTM threats?

Quiz

What does it mean that RoT is static?

- It performs its assessment only at a specific/static point in time

What does it mean that RoT is dynamic?

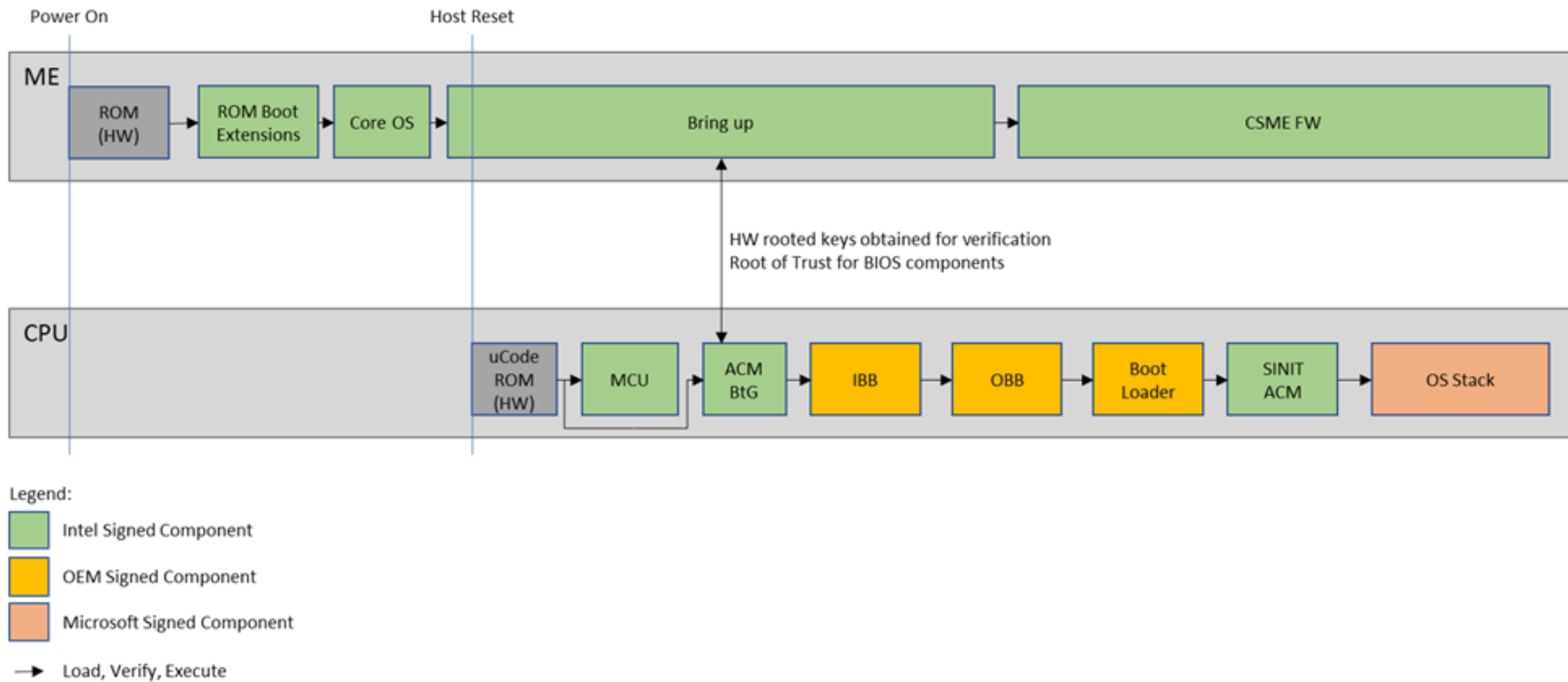
- It performs its assessment only at an arbitrary point in time

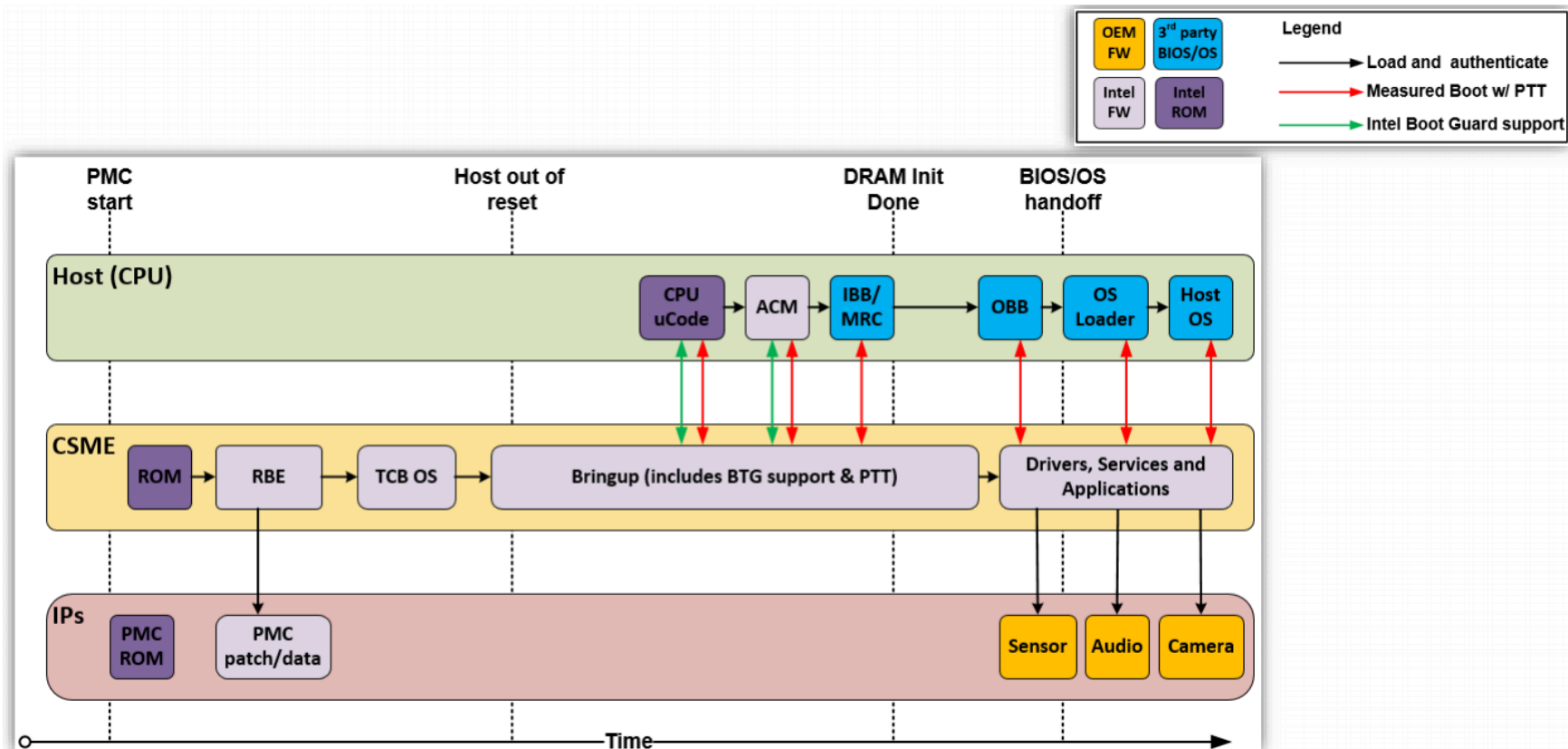
How is D-RTM typically implemented?

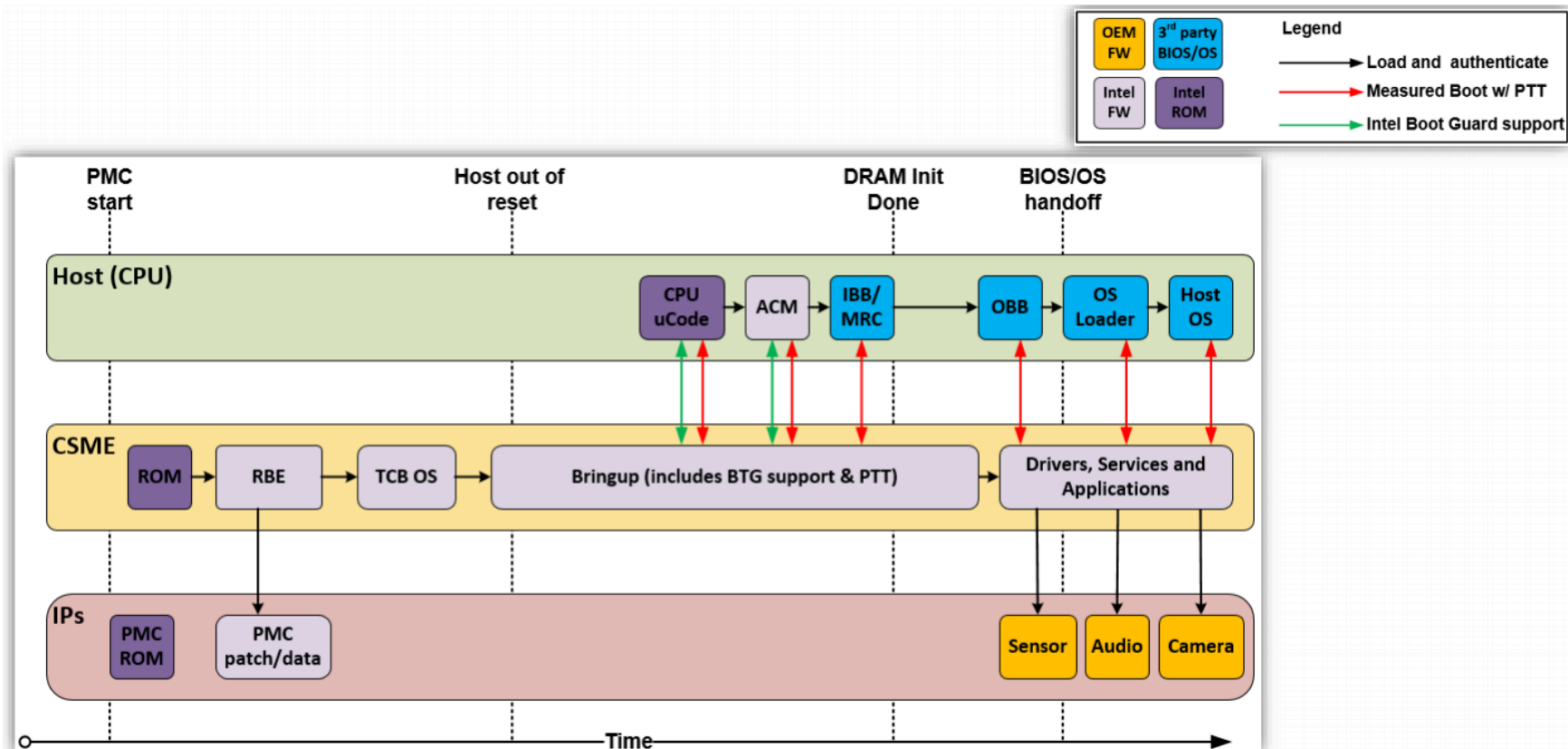
- By dedicated CPU instruction

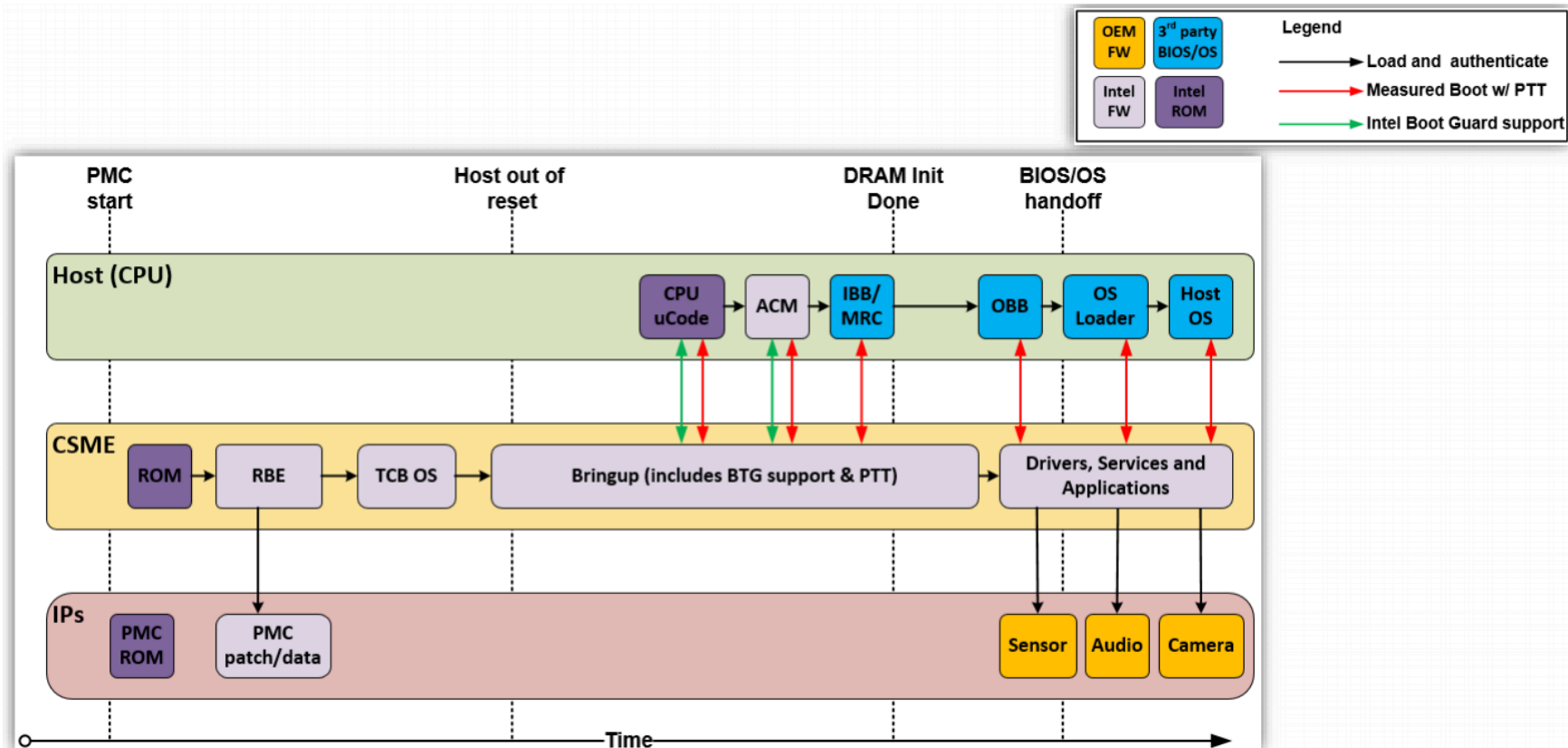
What are the 2 main D-RTM threats?

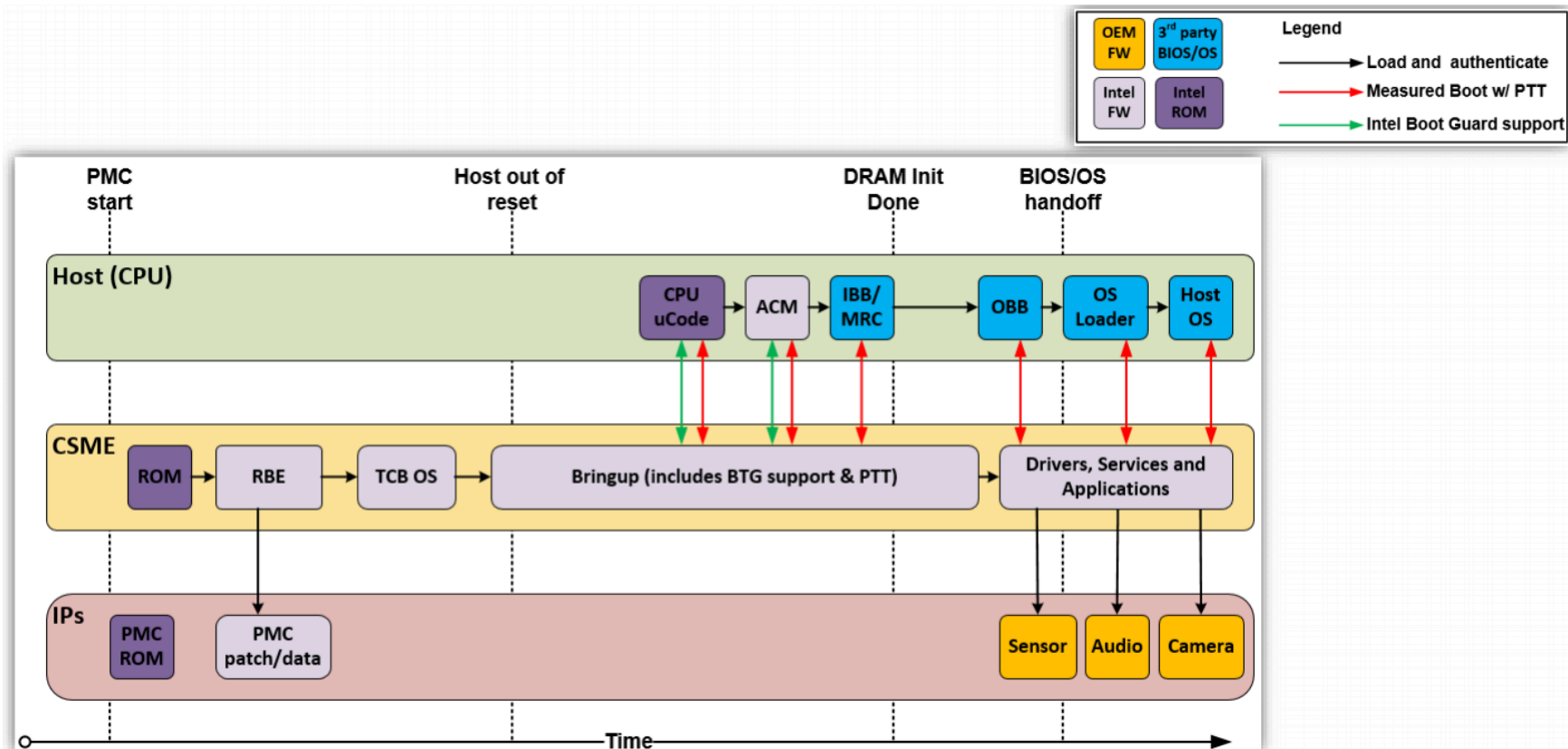
- SMI attacks
- DMA attacks

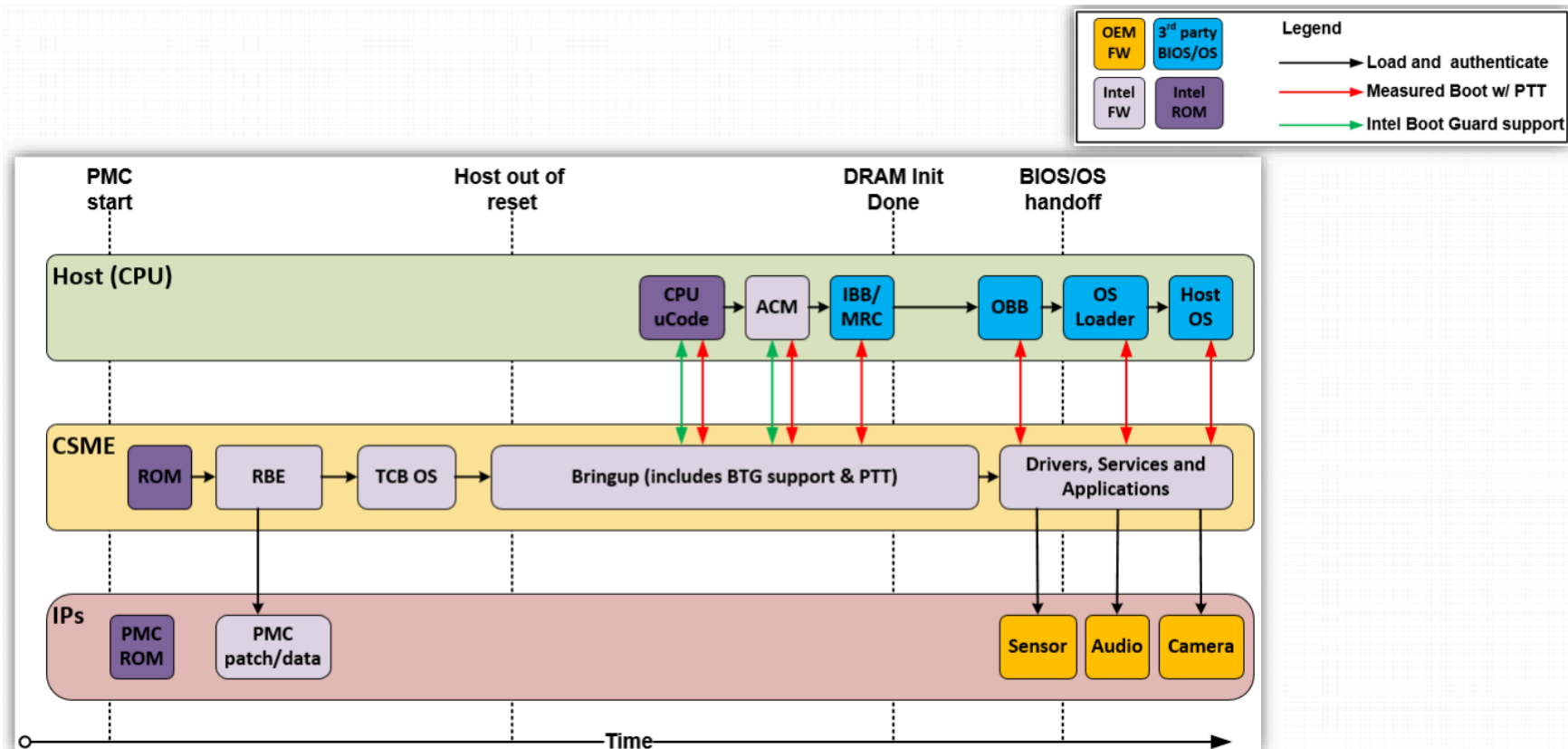












U0000: 00626803f200	tmp15:= MOVEFROMCREG_DSZ64(CORE_CR_CUR_UIP)
U0001: 000801030008	tmp0:= ZEROEXT_DSZ32(0x00000001)
018e5e40	SEQW GOTO U0e5e
<hr/>	
U0002: 004800013000	tmp7:= ZEROEXT_DSZ64(0x00000000)
U0004: 05b900013000	mm7:= unk_5b9(0x00000000)
U0005: 000a01000200	TESTUSTATE(UCODE, UST_MSLOOPCTR_NONZERO)
0b000240	? SEQW GOTO U0002
U0006: 014800000000	SYNCWAIT→ URET(0x00)
<hr/>	
U0008: 000c6c97e208	tmp14:= SAVEUIP(0x01, U056c)
01890900	SEQW GOTO U0909

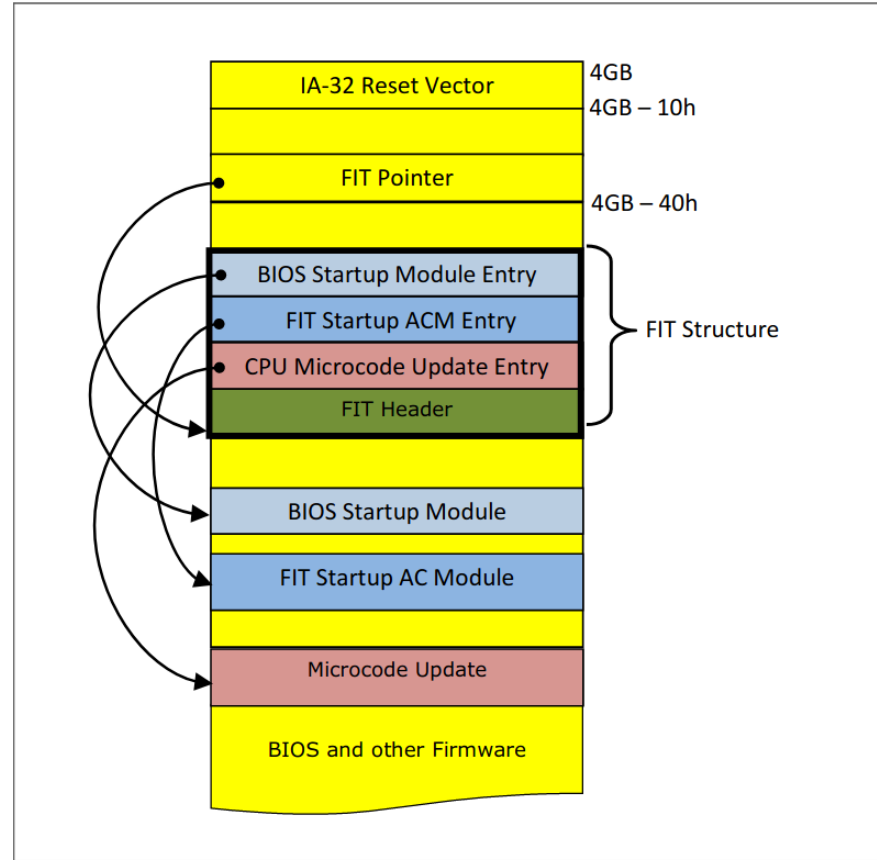
- Microcode is probably the lowest level machine code.
- Implements the programmer-visible instruction set architecture.
- Intel microcode is available on [GitHub](#).
- Signed and encrypted by Intel to prevent unauthorized microcode updates.
- The key is burned into the CPUs.
- Used to authenticate ACMs.

source: [Disassembled microcode source](#) Chip Red Pill Team

FIT table

- The Firmware Interface Table (FIT) is a data structure inside the BIOS binary that describes certain BIOS attributes.
- The absolute address of the beginning of the table is located at a physical address of 4GB - 40h (64 bytes from the top of the flash).
- It is processed by CPU **before** the first BIOS instruction at the reset vector is executed.
- Contains pointers to microcode updates and ACMs (Authenticated Code Modules) to be executed, etc.
- FIT is one of the structures used to establish a Root of Trust in Intel-based systems.
- It is documented in `599500_FW_Interface_Table_BIOS_Spec_Rev1p5.pdf` :
 - "The BIOS flash must include a Firmware Interface Table (FIT) with Type 0 (FIT Header) and Type 1 (Microcode Update) entries."
 - "A microcode update must exist for every processor stepping supported by the platform."

Figure 2-1. FIT Layout in Flash / ROM



Practice #301 Exercise #0

Find ucode in the AMI BIOS and Slim Bootloader image using UEFI Tool or any other tool of your choice.

The UEFI Tool should be started graphically, for example via RDP.

- AMI BIOS binary is in `$HOME/training_materials/hardware/odroid/bios`

UEFITool NE alpha 72 (Jun 16 2025) - ADLN-H4_B1.08.bin

File
Action
View
Help

Structure

Name	A	Type	Subtype	Text
Intel image		Image	Intel	
Descriptor region		Region	Descriptor	
ME region		Region	ME	
Padding		Padding	Empty (FFh)	
BIOS region		Region	BIOS	
AmiNvramMainRomAreaGuid		Volume	FFSv2	
Padding		Padding	Empty (FFh)	
F649FC2D-C0E6-4262-AD51-0CE64F76429F		Volume	FFSv2	
1A803C55-F034-4E60-AD9E-9D3F32CE273C		Volume	FFSv2	
DC616298-3CF8-4AA1-8FEF-1DD5E51684E		Volume	FFSv2	
D45EB923-8B43-411F-BB1C-EE506FBA4BAE		Volume	FFSv2	
47B18F92-5ADC-4739-8D70-9BFFCD9A2EC2		Volume	FFSv2	
4F1C52D3-D824-4D2A-A2F0-EC40C23C5916		Volume	FFSv2	
3DA4F21C-1890-46A3-98C0-AF814EBF01B0		Volume	FFSv2	
EF1FirmwareFileSystem2Guid		Volume	FFSv2	
52F1AFB6-78A6-448F-8274-F370549AC5D0		Volume	FFSv2	
Padding		Padding	Empty (FFh)	
7BEBD21A-A1E5-4C4C-9CA1-A0C168BCBD9D		Volume	FFSv2	
61C0F511-A691-4F54-974F-B9A42172CE53		Volume	FFSv2	
4409A1B5-8D3E-436C-B424-80C9F7AF1C9F		Volume	FFSv2	
AFDD39F1-19D7-4501-A730-CE5A27E1154B		Volume	FFSv2	
8487985B-65AD-44D1-BA36-FD65190CEACD		Volume	FFSv2	
289EFB98-9F4C-47F5-9AC4-35A4D847B0A9		Volume	FFSv2	

Information

Fixed: Yes
Base: 0h
Address: FF000000h
Offset: 0h
Full size: 1000000h (16777216)
Flash chips: 1
Regions: 1
Masters: 3
PCH straps: 70
PROC straps: 1

Parser
FIT
Security
Search
Builder

	Address	Size	Version	Checksum	Type	Information
1	FIT_	00000060h	0100h	00h	FIT Header	
2	00000000FFEA9000h	00020C00h	0100h	00h	Microcode	CpuSignature: 000B06E0h, Revision: 00000010h, Date: 19.12.2022
3	00000000FFF40000h	00009800h	0100h	00h	Startup ACM	LocalOffset: 00000018h, EntryPoint: 0000EBD6h, ACM SVN: 0001h, Date: 28.06.2022
4	007D040100710070h	00000000h	0000h	00h	TXT Policy	Index: 007Dh, BitPosition: 04h, AccessWidth: 01h, DataRegAddr: 0071h, ...
5	00000000FFEE3B00h	00000000h	0100h	00h	BootGuard Key Manifest	
6	00000000FFEE3080h	00000000h	0100h	00h	BootGuard Boot Policy	

UEFITool NE alpha 72 (Jun 16 2025) - ifwi-release.bin

File
Action
View
Help

Structure

Name	A	Type	Subtype	Text
Intel image		Image	Intel	
Descriptor region		Region	Descriptor	
ME region		Region	ME	
Padding		Padding	Empty (00h)	
BIOS region		Region	BIOS	
Padding		Padding	Non-empty	
Intel microcode		Microcode	Intel	
Padding		Padding	Empty (FFh)	
Intel microcode		Microcode	Intel	
Padding		Padding	Empty (FFh)	
Intel microcode		Microcode	Intel	
Padding		Padding	Non-empty	
EfiFirmwareFileSystem2Guid		Volume	FFSv2	
52F1AFB6-78A6-448F-8274-F370549AC5D0		Volume	FFSv2	
Intel microcode		Microcode	Intel	
Padding		Padding	Empty (FFh)	
Intel microcode		Microcode	Intel	
Padding		Padding	Empty (FFh)	
Intel microcode		Microcode	Intel	
Padding		Padding	Non-empty	
EfiFirmwareFileSystem2Guid		Volume	FFSv2	
52F1AFB6-78A6-448F-8274-F370549AC5D0		Volume	FFSv2	
Padding		Padding	Empty (FFh)	
7BEBD21A-A1E5-4C4C-9CA1-A0C168BCBD9D		Volume	FFSv2	
EfiFirmwareFileSystem2Guid		Volume	FFSv2	
Padding		Padding	Empty (FFh)	
7BEBD21A-A1E5-4C4C-9CA1-A0C168BCBD9D		Volume	FFSv2	
EfiFirmwareFileSystem2Guid		Volume	FFSv2	

Information

Fixed: Yes
Base: 0h
Address: FF000000h
Offset: 0h
Full size: 1000000h (16777216)
Flash chips: 1
Regions: 1
Masters: 3
PCH straps: 70
PROC straps: 1

Parser
FIT
Security
Search
Builder

	Address	Size	Version	Checksum	Type	Information
1	_FIT_	000000B0h	0100h	65h	FIT Header	
2	00000000FFC2D000h	00037400h	0100h	00h	Microcode	CpuSignature: 00090672h, Revision: 0000003Ah, Date: 12.12.2024
3	00000000FFC68000h	00022000h	0100h	00h	Microcode	CpuSignature: 000B06E0h, Revision: 0000001Dh, Date: 06.12.2024
4	00000000FFCA3000h	00036400h	0100h	00h	Microcode	CpuSignature: 000906A3h, Revision: 00000434h, Date: 22.02.2024
5	00000000FFCDE000h	00000000h	0100h	00h	Microcode	
6	00000000FFF80000h	00000000h	0100h	00h	Startup ACM	
7	00000000FFFF5340h	00000ACCh	0100h	00h	BIOS Startup Module	
8	00000000FFFE5000h	00001028h	0100h	00h	BIOS Startup Module	
9	00000000FFFE0000h	00010000h	0100h	00h	BIOS Startup Module	
10	00000000FFFE4600h	00000400h	0100h	00h	Secure Guard Key Manifest	
11	00000000FFFE4A00h	00000600h	0100h	00h	Secure Guard Boot Policy	

Practice #301 Exercise #1

Find a FIT table in the image and decode all entry types it contains.

- UEFI Tool is in your `$HOME/training_materials/uefi` directory.
- AMI BIOS binary is in `$HOME/training_materials/hardware/odroid-h4/bios`
- Dasharo (Slim Bootloader+UEFI) binary should be in `$HOME/training_materials/src/slimbootloader/Output`
- FIT Entry Types can be found in documentation in Table 4-2
(<https://edc.intel.com/content/www/br/pt/design/products-and-solutions/software-and-services/firmware-and-bios/firmware-interface-table/1.6/firmware-interface-table/>).

Parser	FIT	Security	Search	Builder	
Address	Size	Version	Checksum	Type	Information
1 _FIT_	00000070h	0100h	00h	FIT Header	
2 00000000FFDD0400h	00017C00h	0100h	00h	Microcode	CpuSignature: 000506E3h, Revision: 000000B2h, Date: 01.02.2017
3 00000000FFDE8000h	00017800h	0100h	00h	Microcode	CpuSignature: 000906E9h, Revision: 00000048h, Date: 15.11.2016
4 00000000FFDF8000h	00017800h	0100h	00h	Microcode	CpuSignature: 000506E8h, Revision: 00000034h, Date: 10.07.2016
5 00000000FFFD0000h	00008000h	0100h	00h	BIOS ACM	LocalOffset: 00000018h, EntryPoint: 00003BB1h, ACM SVN: 0000h, Date: 28.08.2016
6 00000000FFFD9100h	00000241h	0100h	00h	BootGuard Key Manifest	
7 00000000FFFD8080h	000002DFh	0100h	00h	BootGuard Boot Policy	

- Signed by Intel binary blobs to initialize
 - Intel Boot Guard - technology used as HRTM, RTD, RTV and RTM.
 - BIOS Guard - technology used as RTU.
 - TXT - technology used for DRTM.
 - And probably other future technologies.



Intel Authenticated Code Module

- ACMs are authenticated by microcode and a key burned in CPU.

- ACMs are authenticated by microcode and a key burned in CPU.
- TXT ACMs
 - BIOS ACMs
 - SINIT ACMs

- ACMs are authenticated by microcode and a key burned in CPU.
- TXT ACMs
 - BIOS ACMs
 - SINIT ACMs
- BIOS Guard ACMs

- ACMs are authenticated by microcode and a key burned in CPU.
- TXT ACMs
 - BIOS ACMs
 - SINIT ACMs
- BIOS Guard ACMs
- Boot Guard ACMs

- ACMs are authenticated by microcode and a key burned in CPU.
- TXT ACMs
 - BIOS ACMs
 - SINIT ACMs
- BIOS Guard ACMs
- Boot Guard ACMs
- Intel TDX ACMs

- Startup ACMs (S-ACM) (FIT entry type 0x2)

- Startup ACMs (S-ACM) (FIT entry type 0x2)
- Diagnostics ACMs (FIT entry type 0x3)

- Startup ACMs (S-ACM) (FIT entry type 0x2)
- Diagnostics ACMs (FIT entry type 0x3)
- Runtime ACMs (non-official nomenclature)

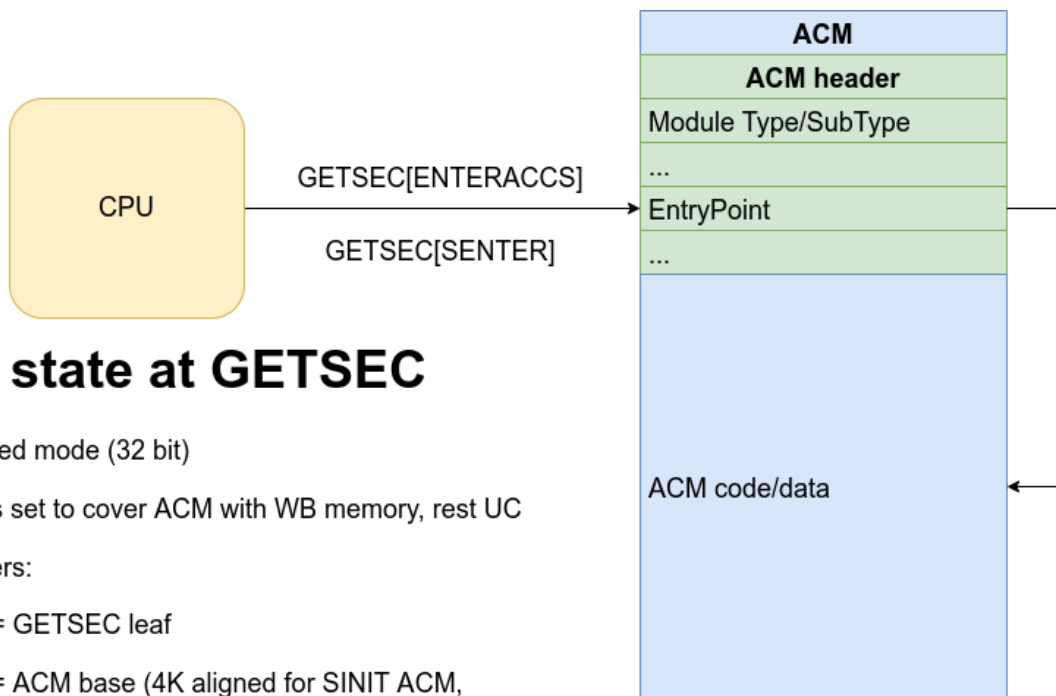
- [SlimBootloader: BuildLoader](#)
- [ACM Type 3](#)
- [Get Started with Intel® Time Coordinated Computing Tools \(Intel® TCC Tools\) 2022.1 for Slim Bootloader](#)
- [Intel TDX Documanation](#)
- [Intel TDX Interface Specification](#)

Parser	FIT	Security	Search	Builder
BootGuard ACM found at base 7D0000h				
ModuleType: 0002h		ModuleSubtype: 0003h	HeaderLength: 00008000h	
HeaderVersion: 00000000h		ChipsetId: 0000h	Flags: 0000h	
ModuleVendor: 8086h		Date: 28.08.2016	ModuleSize: 00008000h	
EntryPoint: 00003BB1h		AcmSvn: 0000h	Unknown1: 00000000h	
Unknown2: 00000000h		GdtBase: 00000598h	GdtMax: 00000020h	
SegSel: 00000008h		KeySize: 00000100h	Unknown3: 0000023Ch	
ACM RSA Public Key (Exponent: 11h):				
C71AC1E2A457E7FCAA585572AFE2BAABFCFC17BAFBC5EED971E12883A268F7EA 6E2C9738F493D7F597144B1AF3F187156839783C503392C92088F89C75BDBC43 0E9BA63DE6890CAC5F22177909D7C2CFCDA313F0C7E299932558B7403BD1D2DF B4874F4FC7DCE34524D896404B64FA1E88AF6349439827F139243F4BD63AE297 E2353A5837F05D701F057E39BAF3BD8007F1A1AD52BAE40964465E1D04304B 63221DC2FB5FD2A62D2AE7DB2FD47F62C993F490F5C7F43EABC6B45DB20ECC69 863550D48BB390FD5EBF45AFC3A7AF051396121732CAF1328F79CBB0C96FCEE 819CDEB2E0E5B1E89A3B3BB79D7167DDEF31F663959564905EDD6A7FA7F75AEE				
ACM RSA Signature:				
1DB30EBED8DE48495C7E4A7A4ED0EB204120D63510108F8534F11F91312DCF7D 71FD9362AEF7474408D0C4B57256E822F064E0880B29B1B4A388F8D888050994 46B47F3A3E788484C67D03BEF6ECFE070A741BD5A1D272AB53A7FE1EFE4D7209 A2CED9AE2BD997450D7EDD388F8D9928CD68C222C6ED858573E3F4C8A647FD63 C56A74D10873BFB7D7A27FCAC42F87311884A6520DF0DD55F9774C01A7AFEF3 CC7EFF53EA6ED1BBDB5A8FE5E0BCFBD2D4662C95F0F43E6E03B1F6482FB3B3DF 6076B2DFAE311D92A3A3C7701A80BFB71BB6A94A2C11E430DFA1D7BB1C014C54 5D778E4CEA5EAE8CB9329091F0EA1F7BAFEFF658561A86999A60445ED5B28633				

ACM structure definition may be found in Appendix A of [Intel Trusted Execution Measured Launch Environment Developer's Guide](#)

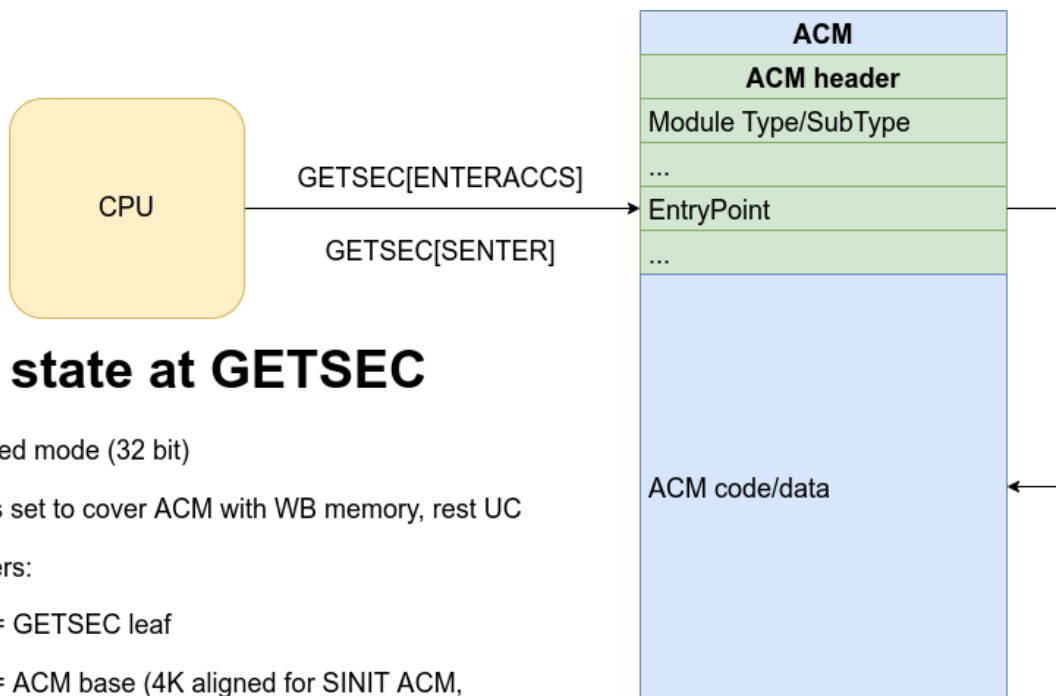
Intel SMX

- Safer Mode Extension provide a programming interface to establish measured environment (Intel TXT)
 - described in Intel SDM chapter 6
 - key technology to implement DRTM for Intel platforms
- Includes:
 - Measured Launched Environment (MLE), which may be based on measured VMM (MVMM)
 - secure storage and location for measurement
 - VMM protection against modification
- SMX functionality is provided by `GETSEC` instruction
- BIOS is responsible for loading ACM and calling `GETSEC[ENTERACCS]`
 - In contrast to Boot Guard, where the ACM is loaded and `GETSEC[ENTERACCS]` is executed automatically by the Boot ROM
- This extension is not available in all processors and requires compatible chipset be fully functional.



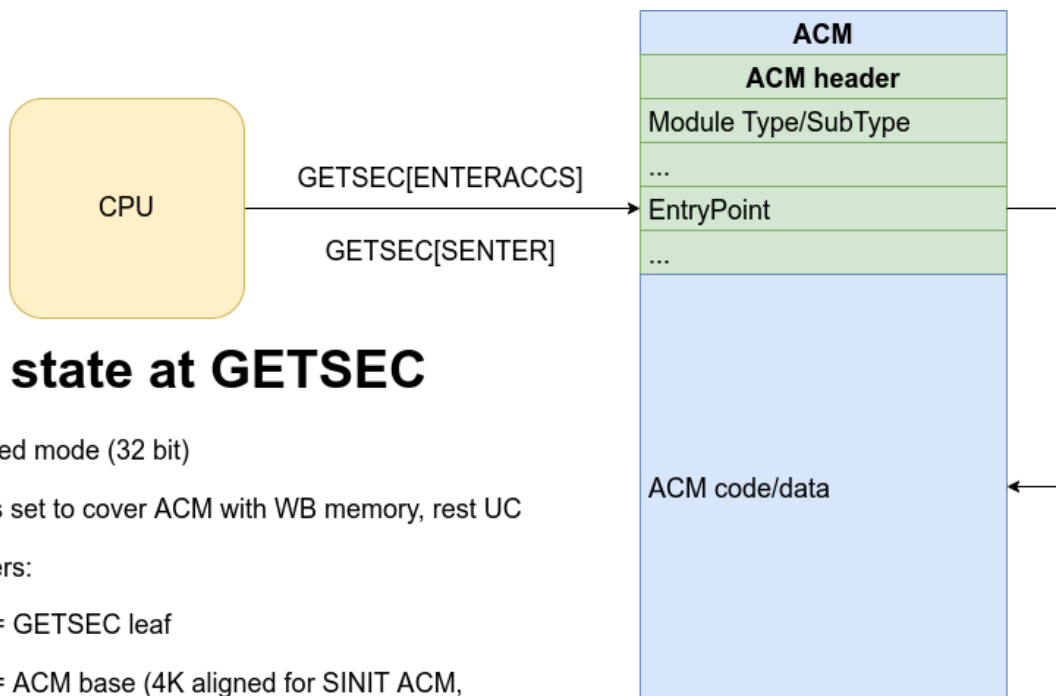
CPU state at GETSEC

1. Protected mode (32 bit)
2. MTRRs set to cover ACM with WB memory, rest UC
3. Registers:
 - EAX = GETSEC leaf
 - EBX = ACM base (4K aligned for SINIT ACM, 64K/128K/256K for BIOS ACM platform-specific)
 - ECX = ACM size
 - EDX, EDI = 0
 - ESI = BIOS ACM function to call, else 0



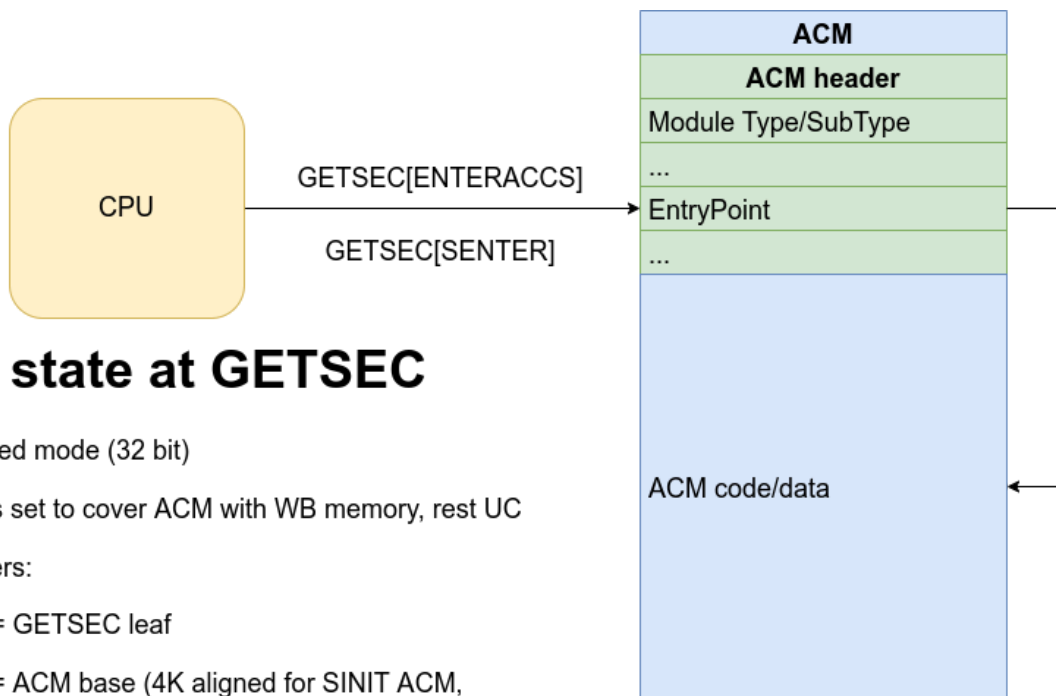
CPU state at GETSEC

1. Protected mode (32 bit)
2. MTRRs set to cover ACM with WB memory, rest UC
3. Registers:
 - EAX = GETSEC leaf
 - EBX = ACM base (4K aligned for SINIT ACM, 64K/128K/256K for BIOS ACM platform-specific)
 - ECX = ACM size
 - EDX, EDI = 0
 - ESI = BIOS ACM function to call, else 0



CPU state at GETSEC

1. Protected mode (32 bit)
2. MTRRs set to cover ACM with WB memory, rest UC
3. Registers:
 - EAX = GETSEC leaf
 - EBX = ACM base (4K aligned for SINIT ACM, 64K/128K/256K for BIOS ACM platform-specific)
 - ECX = ACM size
 - EDX, EDI = 0
 - ESI = BIOS ACM function to call, else 0



CPU state at GETSEC

1. Protected mode (32 bit)
2. MTRRs set to cover ACM with WB memory, rest UC
3. Registers:
 - EAX = GETSEC leaf
 - EBX = ACM base (4K aligned for SINIT ACM, 64K/128K/256K for BIOS ACM platform-specific)
 - ECX = ACM size
 - EDX, EDI = 0
 - ESI = BIOS ACM function to call, else 0

Quiz

Quiz

What is Authenticated Code Module?

Quiz

What is Authenticated Code Module?

- Signed by Intel binary blob to initialize Boot Guard, TXT, BIOS Guard, TDX and FuSa.

Quiz

What is Authenticated Code Module?

- Signed by Intel binary blob to initialize Boot Guard, TXT, BIOS Guard, TDX and FuSa.

What types of ACMs form Root of Trust?

Quiz

What is Authenticated Code Module?

- Signed by Intel binary blob to initialize Boot Guard, TXT, BIOS Guard, TDX and FuSa.

What types of ACMs form Root of Trust?

- BIOS ACM - Static Root of Trust
- SINIT ACM - Dynamic Root of Trust
- TDX ACM - Root of Trust for Confidentiality

Quiz

What is Authenticated Code Module?

- Signed by Intel binary blob to initialize Boot Guard, TXT, BIOS Guard, TDX and FuSa.

What types of ACMs form Root of Trust?

- BIOS ACM - Static Root of Trust
- SINIT ACM - Dynamic Root of Trust
- TDX ACM - Root of Trust for Confidentiality

What instruction is used to execute BIOS ACM?

Quiz

What is Authenticated Code Module?

- Signed by Intel binary blob to initialize Boot Guard, TXT, BIOS Guard, TDX and FuSa.

What types of ACMs form Root of Trust?

- BIOS ACM - Static Root of Trust
- SINIT ACM - Dynamic Root of Trust
- TDX ACM - Root of Trust for Confidentiality

What instruction is used to execute BIOS ACM?

- `GETSEC[ENTERACCS]`

Leaks Effect

- Recent leaks of cryptographic materials seem to do a lot of good job for quality and transparency of documentation.

” Quote

This technical article is intended to address an issue discovered in some Original Equipment Manufacturers (OEM) or Original Design Manufacturers (ODM) production systems where system firmware (FW) or an Integrated Firmware Image (IFWI) **was found to contain one or more preproduction or example test keys**. The private keys for these preproduction or test keys were often included in system Basic Input Output System (BIOS) development information provided to OEMs or ODMs for their use in platform development and testing and were not intended to be used in production systems or environments.

source: [Introduction to Key Usage in Integrated Firmware Images](#)

- Limitation

” Quote

This document's information is generally limited to products based on 13th Generation Intel® Core™ Processor platforms and 3rd Generation Intel® Xeon® Scalable Processor platforms, or earlier. Guidance for manifesting and signing components for future products is subject to change.

source: [Introduction to Key Usage in Integrated Firmware Images](#)

” Quote

Platform manufacturers use field programmable fuses (FPFs) to provision the boot policy to be used on the system. They provide the foundational mechanism for verifying the integrity of the KM and BPM.

source: [Introduction to Key Usage in Integrated Firmware Images](#)

- FPF are persistently stored in Platform Controller Hub (PCH) in case of dual chipset design (CPU+PCH) and in SoC for mobile/embedded systems - this is our key store backed by some fusing technology.
- Once FPFs are provisioned a hardware lock is closed and those cannot be changed for the life of the platform.
- Establishing final lock is referred to as End of Manufacturing (EOM).
- FPFs are not accessible directly and can be controlled only through ME/CSE/CSME, which is inside PCH or SoC.

Quiz

Quiz

Where Field Programmable Fuses are located in modern Intel platform?

Quiz

Where Field Programmable Fuses are located in modern Intel platform?

- In case of single chip platform fuses are in SoC.
- In case of dual-chip platform (CPU+PCH) fuses are in PCH.

Quiz

Where Field Programmable Fuses are located in modern Intel platform?

- In case of single chip platform fuses are in SoC.
- In case of dual-chip platform (CPU+PCH) fuses are in PCH.

What effect on Silicon Vendor may have security vulnerabilities?

Quiz

Where Field Programmable Fuses are located in modern Intel platform?

- In case of single chip platform fuses are in SoC.
- In case of dual-chip platform (CPU+PCH) fuses are in PCH.

What effect on Silicon Vendor may have security vulnerabilities?

- They have to explain impact of vulnerability, what typically means revealing NDA design of the internals of the platform design.
- This is needed to explain to stakeholders that required mitigation was applied.
- Ecosystem gain transparency and improve auditability on the cost of temporary instability of the global computer ecosystem.

Quiz

Where Field Programmable Fuses are located in modern Intel platform?

- In case of single chip platform fuses are in SoC.
- In case of dual-chip platform (CPU+PCH) fuses are in PCH.

What effect on Silicon Vendor may have security vulnerabilities?

- They have to explain impact of vulnerability, what typically means revealing NDA design of the internals of the platform design.
- This is needed to explain to stakeholders that required mitigation was applied.
- Ecosystem gain transparency and improve auditability on the cost of temporary instability of the global computer ecosystem.

What effect on Silicon Vendor may have key and source code leaks?

Quiz

Where Field Programmable Fuses are located in modern Intel platform?

- In case of single chip platform fuses are in SoC.
- In case of dual-chip platform (CPU+PCH) fuses are in PCH.

What effect on Silicon Vendor may have security vulnerabilities?

- They have to explain impact of vulnerability, what typically means revealing NDA design of the internals of the platform design.
- This is needed to explain to stakeholders that required mitigation was applied.
- Ecosystem gain transparency and improve auditability on the cost of temporary instability of the global computer ecosystem.

What effect on Silicon Vendor may have key and source code leaks?

Conclusion

In this lecture, we learned about the following:

- Even the best SRTM requires high quality initial protection and SRTM boot chains as well as protecting it RoT are complex.
- Root of Trust does not have to be static. Dynamic RoT may bring complexity as well as opportunity to improve security posture of the platform and system.
- Reset vector is not beginning of modern x86 boot process.
- Where is Intel Boot Guard, what role ACM and uCode have in establishing platform security.
- What is assumed responsibility of various parties in the process according to Intel.

This information will provide us with a solid foundation for:

- Assessment of Intel Boot Guard state.
- Guiding future decision about system security.

The background is a dark gray with decorative circuit-like lines in the corners. These lines are light gray and form various geometric shapes, including straight lines, right angles, and small circles, resembling a printed circuit board (PCB) layout. The lines are more dense in the top-left and top-right corners and more sparse in the bottom-left and bottom-right corners.

Q&A