The TCG Dynamic Root for Trusted Measurement

Author: Lee Wilson
TCG D-RTM Subgroup Chair
PureFlex Security Architect, IBM Corporation

BASIC CONCEPTS

TCB Definition

Orange Book Definition of Trusted Computing Base (TCB)

- *"the totality of protection mechanisms within it, including hardware, firmware, and software, the combination of which is responsible for enforcing a computer security policy."*
- *"[t]he ability of a trusted computing base to enforce correctly a unified security policy depends on the correctness of the mechanisms within the trusted computing base, the protection of those mechanisms to ensure their correctness, and the* correct *input of parameters related to the security policy."*

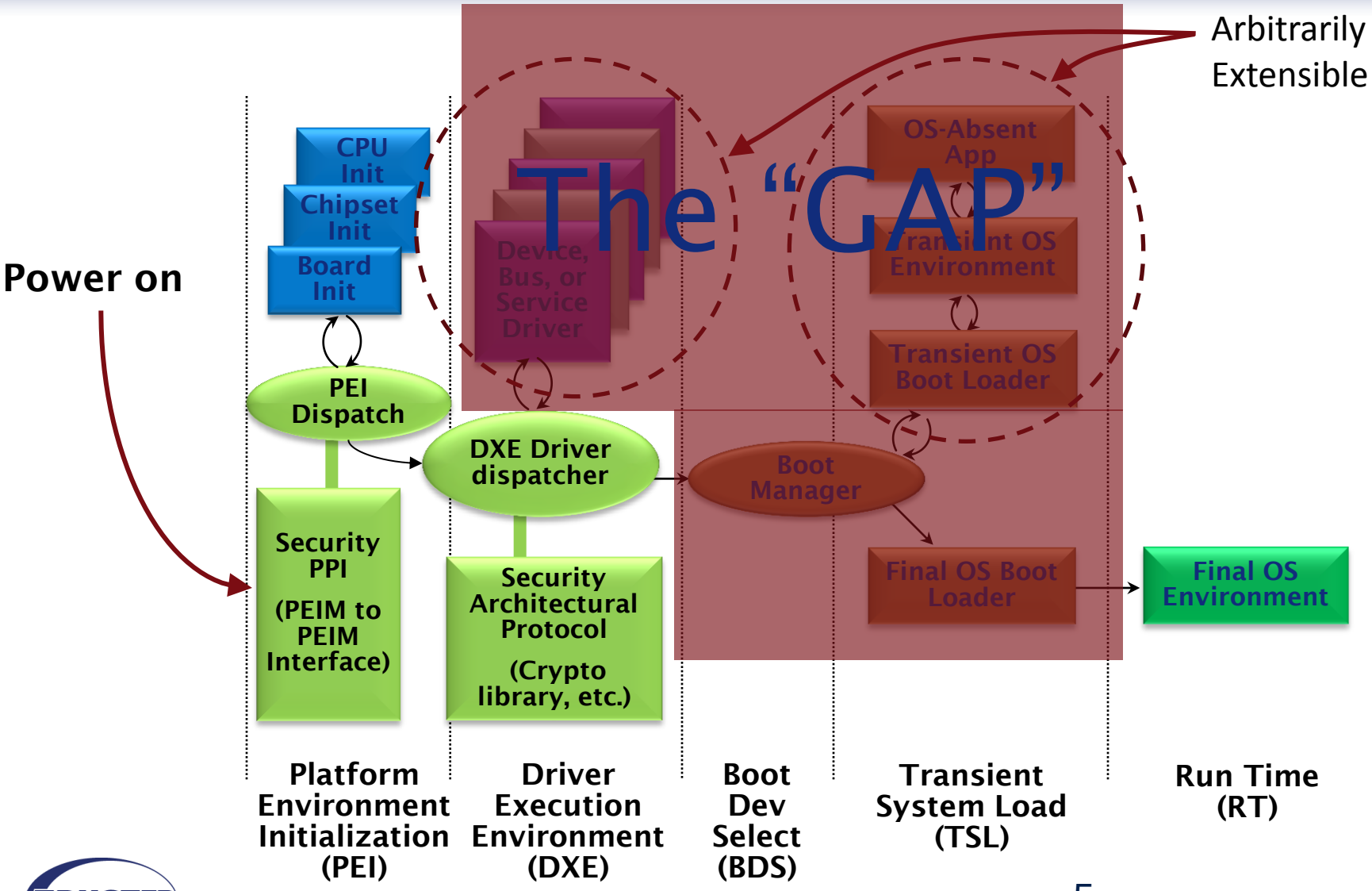
Translated….

- TCB is what protects the security of the system
- TCB must be able to protect its own security – it must "defend" itself

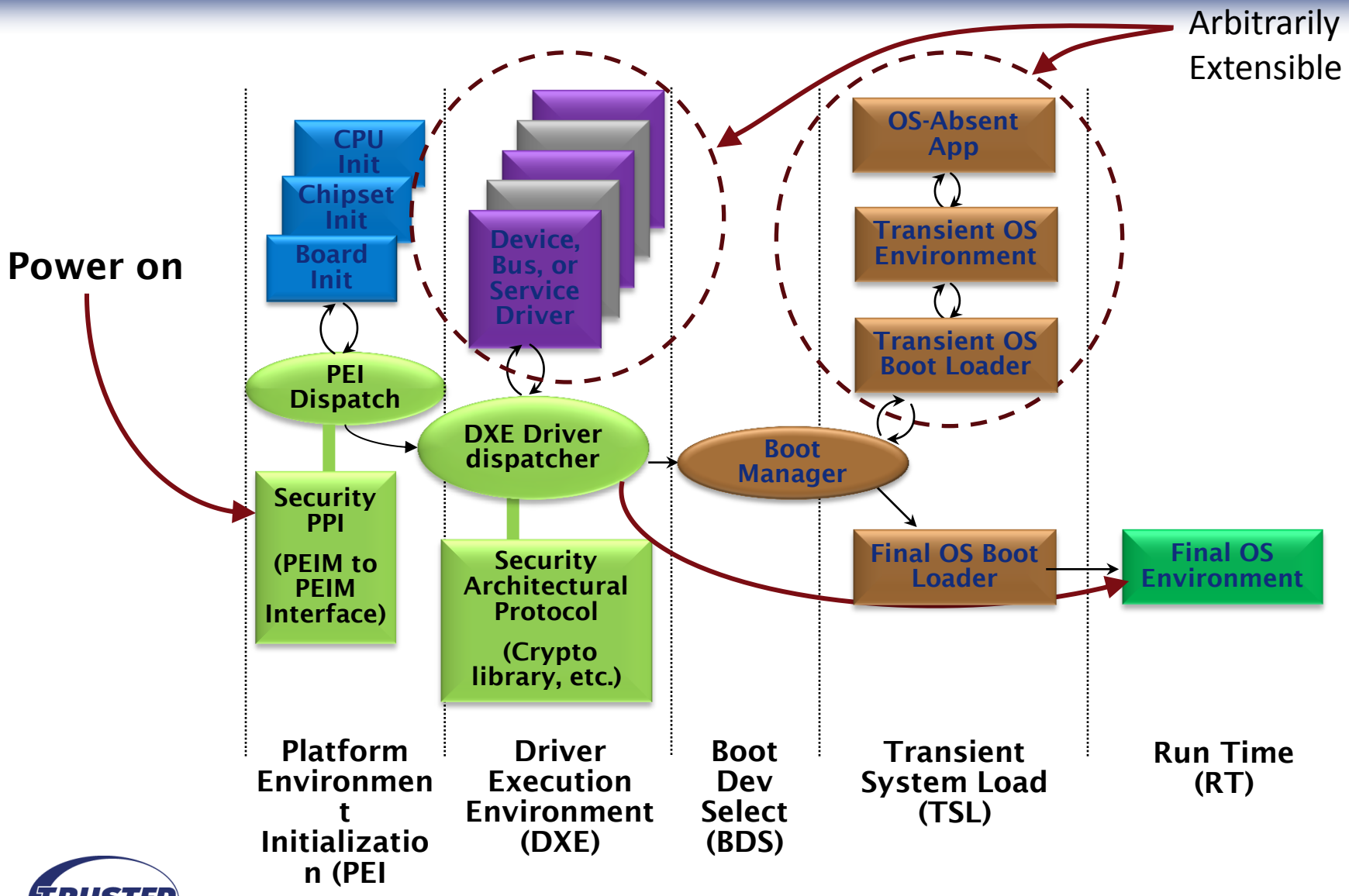
Why Move To The D-RTM?

- It is highly desirable to provide a simple trustworthy state to launch a TCB (Trusted Computing Base) in a platform without having to reset the entire platform
  - The D-RTM provides a method to launch OS'es and Hypervisors, and create a highly defended TCB with a reduced software attack surface, without resetting the entire platform.
  - The D-RTM uses a standardized software architecture with some defenses permanently enforced by hardware, which simplifies the software protection and support required to create a TCB.
- D-RTM implementations have a solution for PCR "brittleness".
  - Any change in the boot firmware causes different measurements which require new evaluation, re-sealing of secrets to PCRs, etc. This can be a cumbersome process.
- D-RTM implementations present a sufficiently constrained TOE to allow for CC Evaluation.
  - The NSA has requested that the D-RTM WG take on the development of a new protection profile (PP) for the D-RTM under the new reformed common criteria.

SRTM TCB – UEFI Version

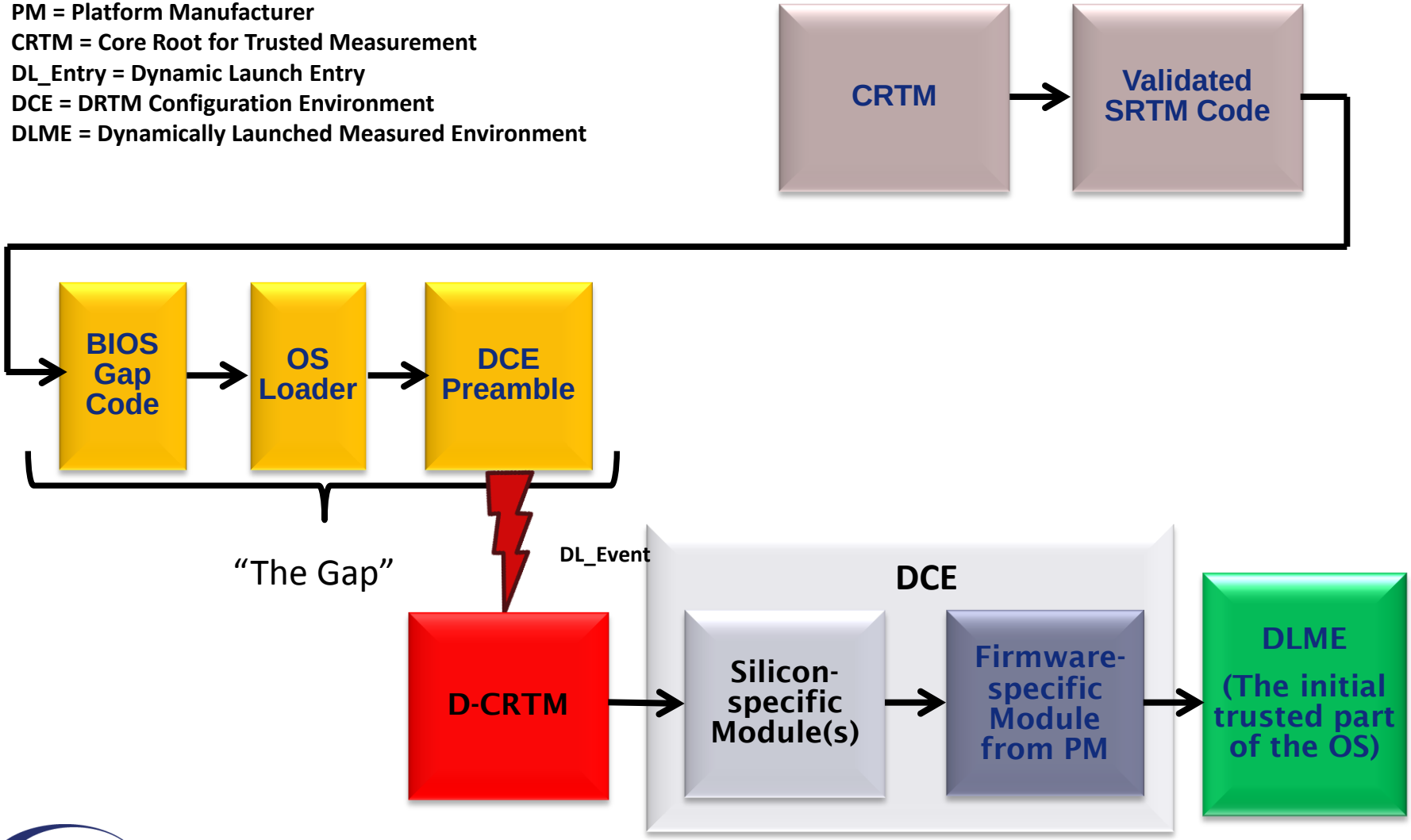


Smaller, Controlled, Attestable TCB

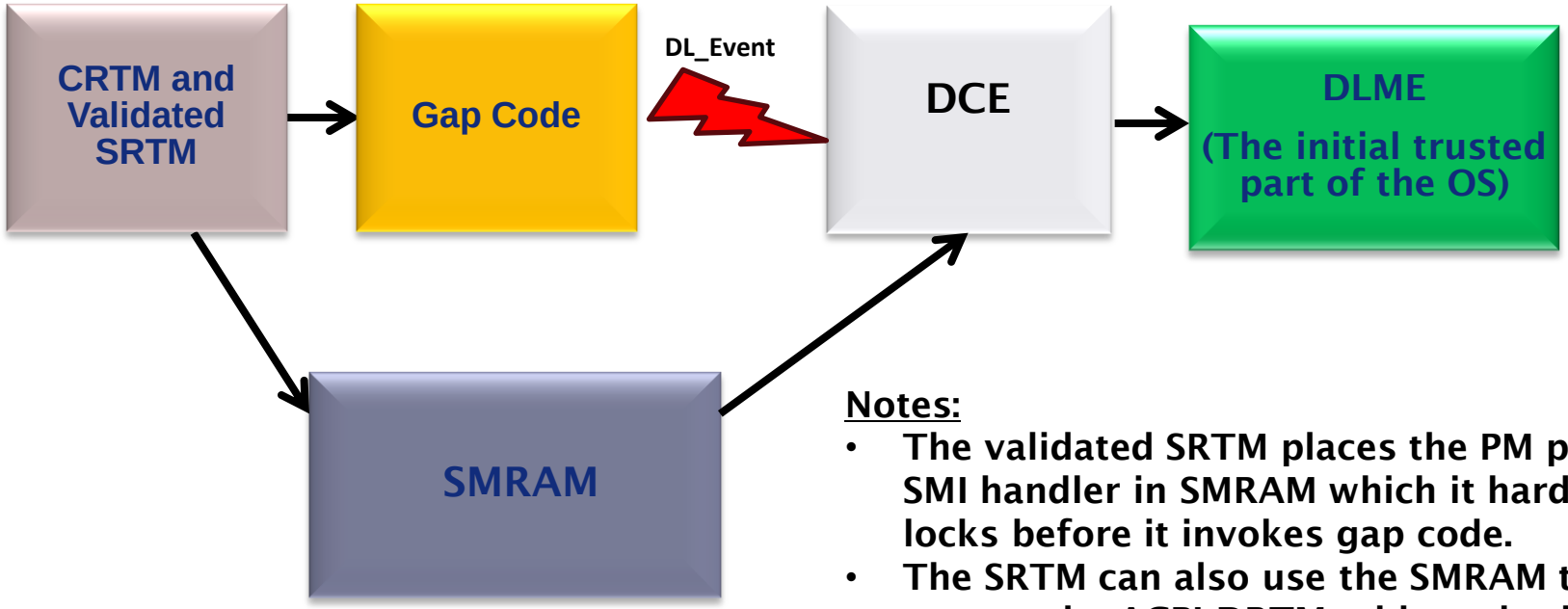


DRTM Flow

PM = Platform Manufacturer
CRTM = Core Root for Trusted Measurement
DL_Entry = Dynamic Launch Entry
DCE = DRTM Configuration Environment
DLME = Dynamically Launched Measured Environment



Locked SMRAM



Notes:

- The validated SRTM places the PM provided SMI handler in SMRAM which it hardware locks before it invokes gap code.
- The SRTM can also use the SMRAM to protect the ACPI DRTM table and other tables with security properties. By depositing them in SMRAM, the SRTM insures they are passed unaltered to the DCE.
- The DCE must be validated before it is executed. It is trusted code.

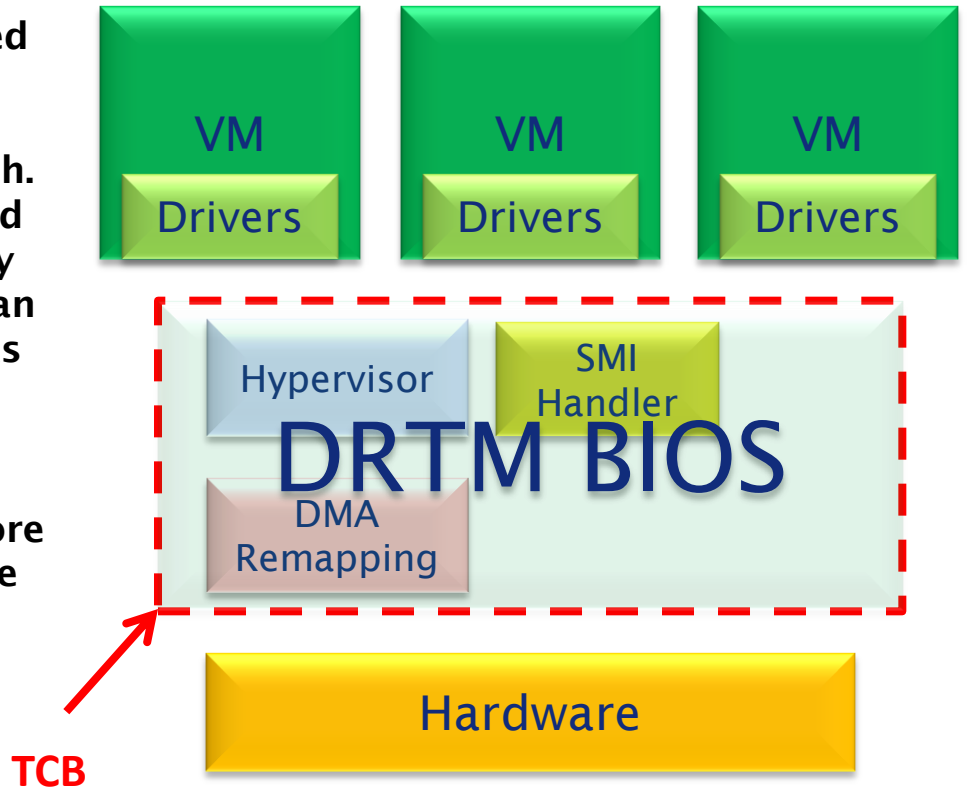
Restarting the Chain of Trust

- The objective of DL_Event (eg. SENTER, SKINIT) is to “pop” the hardware back to a lower entropy state and clear the dynamic TPM PCRs making it easier to check the security properties of the system before continuing to boot.
- In essence, a new chain of trust is established.
  - The CPU contains a CRTM which is invoked by an instruction called DL_Event.
  - DL_Event (the CPU CRTM) measures some code and starts executing that code
  - Code is protected from DMA before it is measured.
  - The DCE is run after DL_Event
- The DCE:
  - Checks the hardware necessary for a minimal TCB.
  - Verifies that that the security sensitive portions of the ACPI are correct
  - Passes control to OS code (DLME).

D-RTM Based Hypervisor TCB

Notes:

- The use of the D-RTM is not limited to hypervisors, but hypervisors offer additional TCB protection to accompany a trusted D-RTM launch.
- With DMA remapping and a trusted hypervisor, the TCB established by the DRTM and subsequent code can be better protected against attacks by VM guests.
- The SMI handler, hypervisor, DMA remapping logic and DRTM BIOS which launched them (and therefore has a lingering effect) are the code components that make up the resultant TCB.

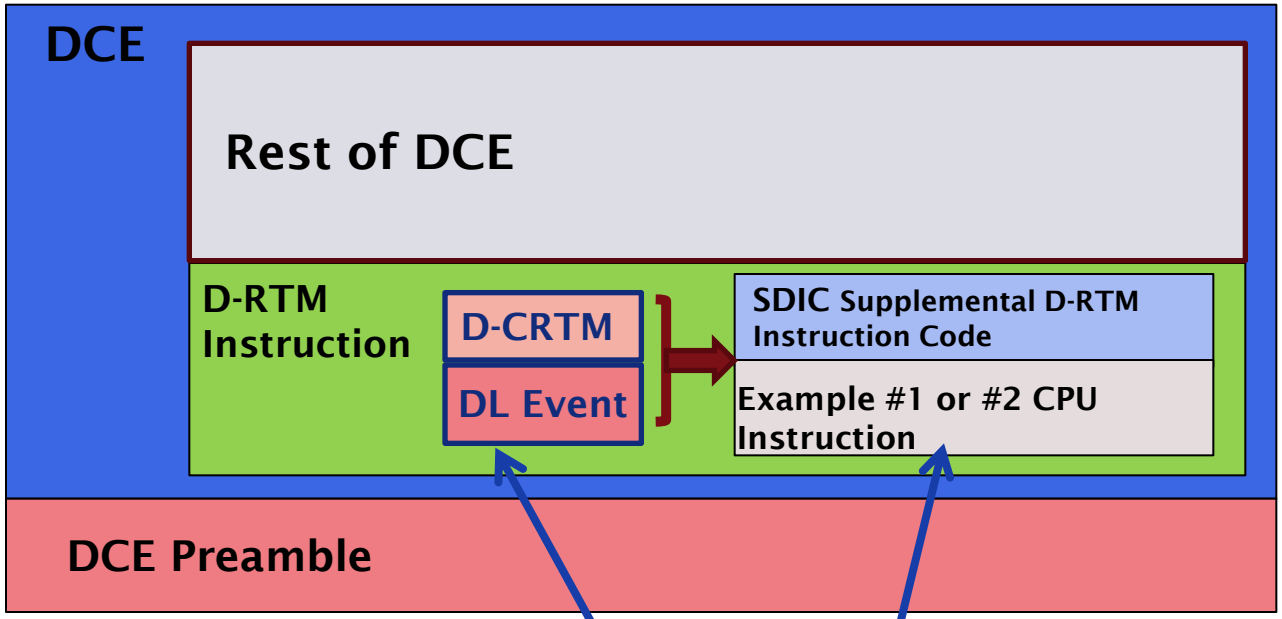


ACPI Tables – Sensitive Resources

- The ACPI DRTM table is the single most important ACPI table in the ACPI tree from a security perspective.
- The ACPI DRTM table points to other ACPI tables defined by the PM which the PM has determined are necessary to the TCB of the system.
- A sensitive resources list is also provided to the DLME which contains pointers to all security sensitive ACPI tables and any other address ranges which the PM has determined are necessary to defending the TCB of the system.
- The DCE must insure that these security sensitive ACPI tables and the sensitive resources list is correct before passing them to the DLME.
- It is anticipated that most vendors will do this be sequestering these in SMRAM to protect them from the gap. The DCE can then extract known good copies of them from SMRAM to provide them to the DLME.

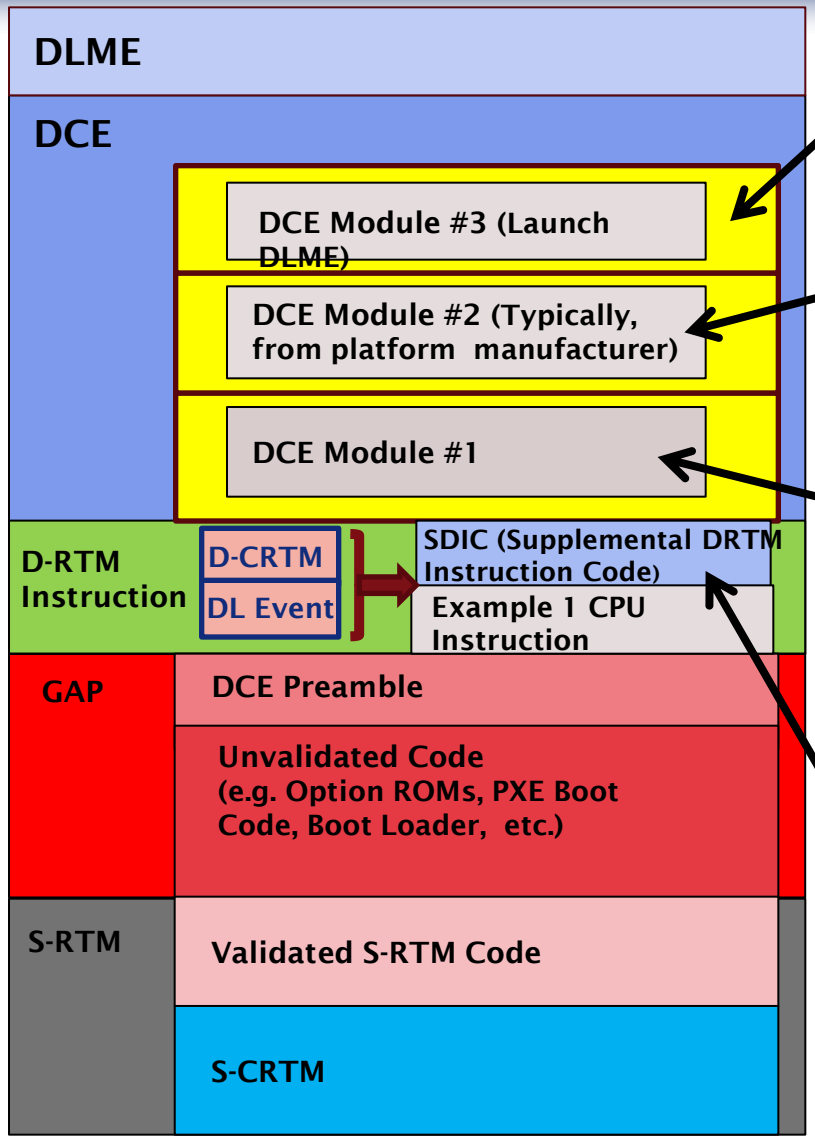
TAXONOMY OF A D-RTM

Components of the DCE



In current implementations, the aggregate function achieved by the DL Event and the D-CRTM are achieved with the actual CPU D-RTM instruction and additional code which we will call SDIC for simplicity (Supplemental D-RTM Instruction Code). The SDIC is some additional CPU instructions that complete the requirements for the architected D-RTM instruction and it is constructed differently by different CPU manufacturers.

D-RTM Example #1



Yellow wrapper indicates the module is signed. Separate yellow boxes are separately signed modules.

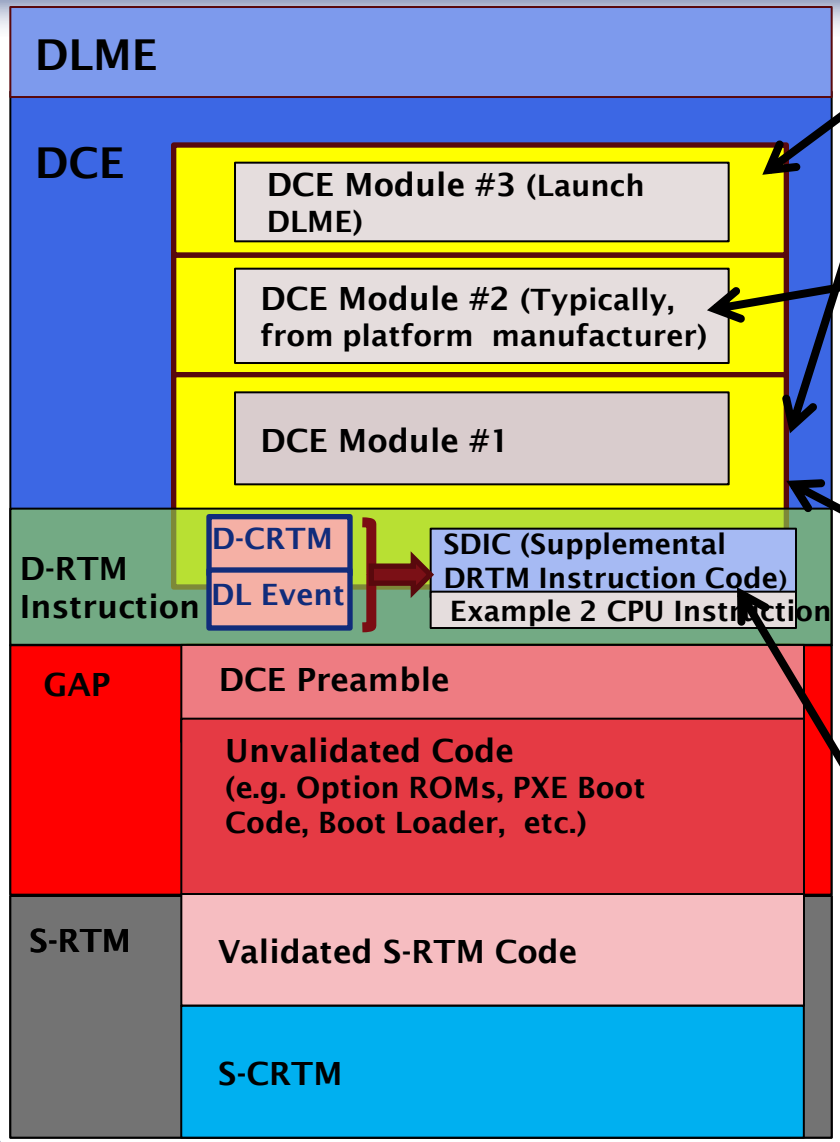
DCE Module #3 must be signature checked and launched by DCE Module #2, etc. It will register PCR.DLME.Authorities and launch the DLME.

In this example, the SDIC and DCE Module #1 are separate. DCE Module #1 must signature check DCE module #2. To do this, it will need to have the public key for checking DCE module #2.

The SDIC (Supplement D-RTM Instruction Code) for example #1 is responsible for the following (may do additional tasks);

- Setting PCR.Authorities and PCR.Details values.
- Determining if the DCE is intended for the platform.
- Measuring the CPU microcode level.
- Measuring and launching DCE module #1.

D-RTM Example #2



Yellow wrapper indicates the module is signed. Separate yellow boxes are separately signed modules.

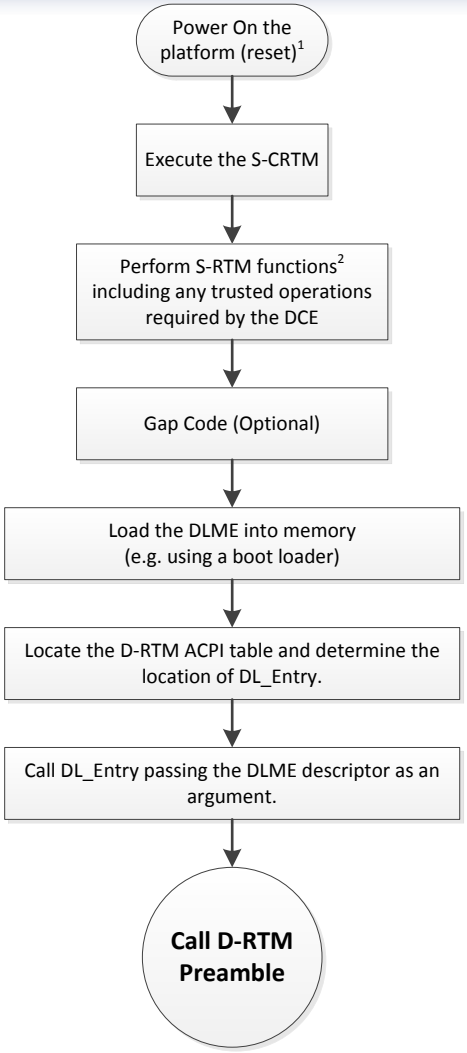
DCE Module #3 must be signature checked and launched by DCE Module #2, etc. It will register PCR.DLME.Authorities and launch the DLME.

In this example, the SDIC and DCE Module #1 are signed together. DCE Module #1 must signature check DCE module #2. To do this, it will need to have the public key for checking DCE module #2.

The SDIC (Supplement D-RTM Instruction Code) for example #2 is responsible for the following (may do additional tasks);

- Setting PCR.Authorities and PCR.Details values.
- Determining if the DCE is intended for the platform.

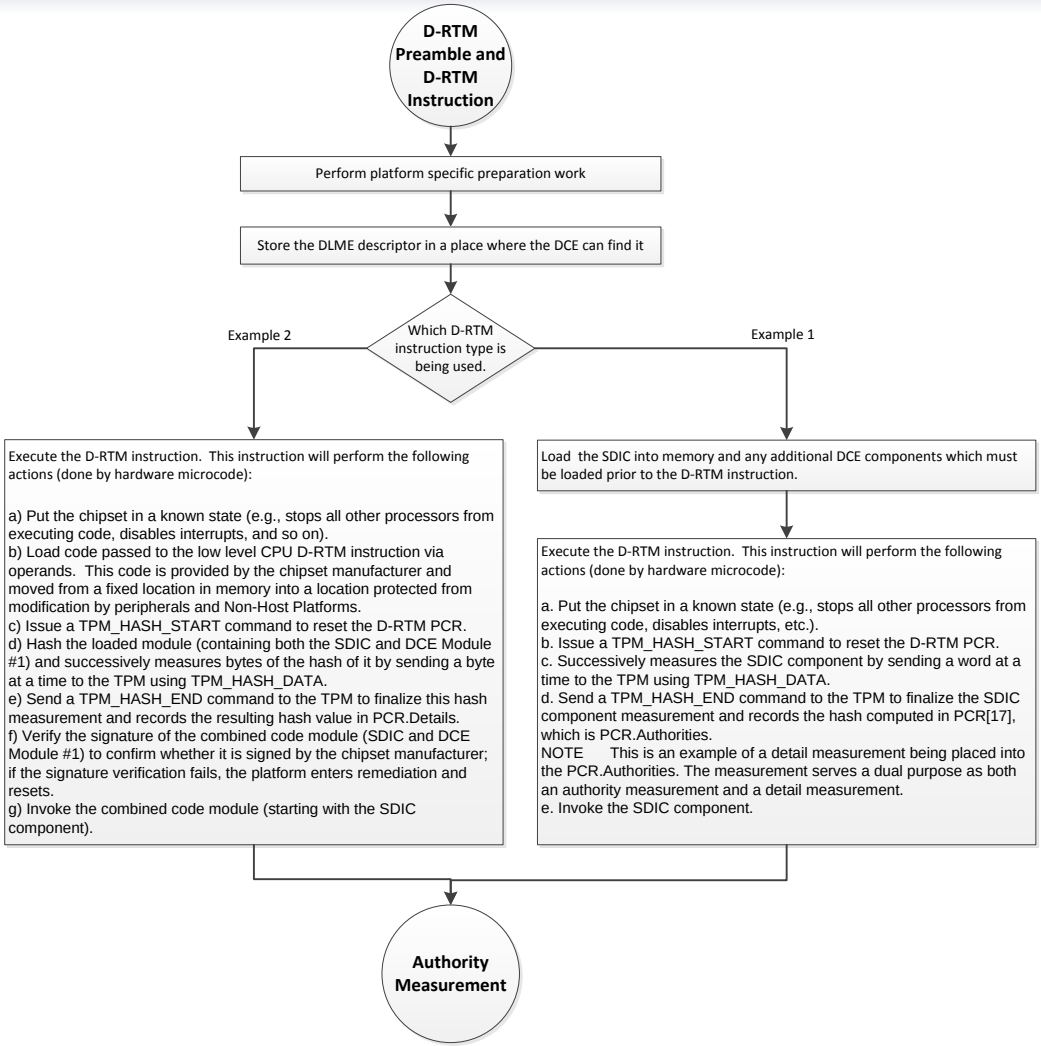
Power On



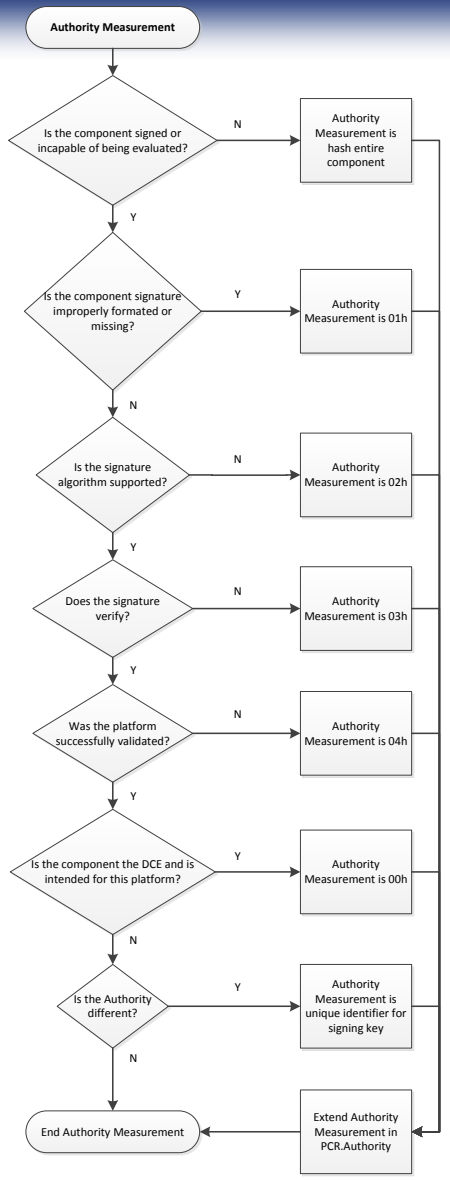
(1) PCR(0-7) are zero after reset.

(2) S-RTM operations are defined in the PC Client specification.

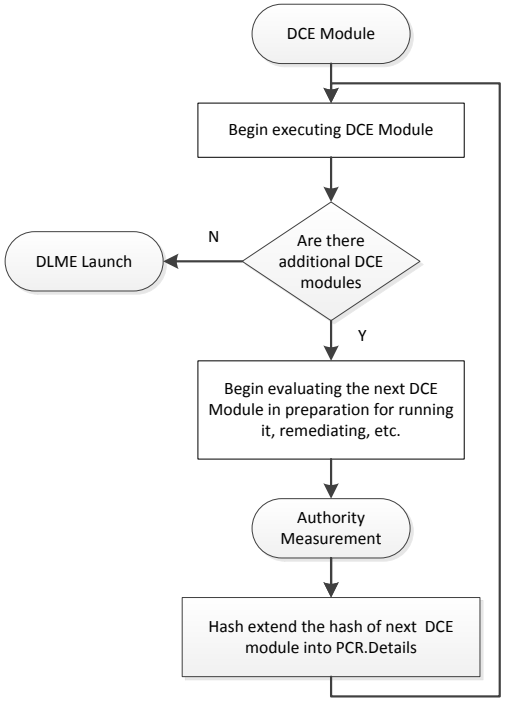
DRTM Preamble and CPU Instruction



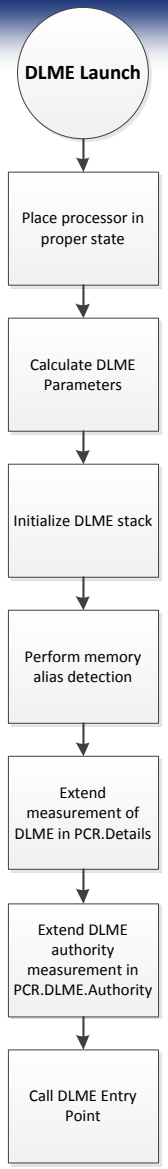
Initial PCR.Authorities Measurement



Executing the DCE Modules



Launch the DLME



ADDITIONAL DETAILS

Non-Host Platforms (NHPs)

- A non-host platform (NHP) is field-programmable processor against which the primary/host software TCB has no defense.
- Things that are not non-host platforms:
  - A computer on a card that is 'outside' of the perimeter afforded by DMA remapping hardware is a peripheral.
  - Something that is programmed at the factory and cannot be modified in the field is hardware because it has the same security properties as a bunch of gates.
- NHPs can have a substantial impact on the security of a system if they have access to the TCB.
- The DRTM specification categorizes NHPs and specifies how they must be measured, secured and reported so that their effect on the system TCB is accounted for and reported correctly.
- Server NHPs are especially powerful – consider trusted computing architectures in conjunction with very good security design for server class NHPs (service processors)!

Solving PCR Brittleness

- Assume that the Platform Manufacturer (PM) knows what constitutes a trustworthy state.
- Record the fact that the PM checked the hardware in a PCR (the “authorities” PCR) and record details of the measurements in another PCR.
- Many uses are satisfied if the PM says things are OK so they would just need to check the authorities PCR.
- The underlying checks can change – but the authorities PCR remains the same but the details PCR reflects the change.
- The DLME can seal any secrets it needs to the authorities PCR which should not change as the platform manufacturer remains the same when boot firmware is updated.

S3 Resume

- For version 1.0 of the specification it is believed that implementers will pursue a “Gapful” resume is being addressed.
  - Note: It is also possible to run a gapless resume.
- BIOS (including the D-RTM process) is reinvoked on an S3 resume.
  - PXE, the boot loader, etc. are not re-invoked on S3 resume as there is a fully functional memory image already resident.
  - The BIOS knows it is a resume by flags set in the chipset by the DLME when the machine is put to sleep.
  - PCRs are re-established, etc.
- The DLME will once again be launched by the DCE (which in most instances means it will be signature checked).
- The DLME is responsible for establishing that the overall system memory image has not been tampered with since the system went to sleep.

Policies for DCE Launch Decisions

- The DRTM Specification does not specify how vendors will do policies for their DCE Launches of the DLME
  - But, you're going to need them.
  - Note: They are not architecturally visible to the DLME.
  - Different vendors may do them in different ways and have different features.
- Means will have to be given for customers to securely modify these policies (eg. Public keys for locally signed hypervisors).

Validating BIOS - Secure BIOS Update

- Things in BIOS that have to be validated (not just measured – insured to be correct):
  - D-CRTM
  - SRTM code between the D-CRTM before the “Gap”
  - SMI handler, security sensitive ACPI tables, sensitive resources list (stuff you put in locked SMRAM)
  - The chip and PM portions of the DCE
- How can you validate?
  - By using a secure update process for locked flash areas.
  - By signing areas outside of locked flash and checking them before running (DMA is off so it cannot tamper with these images while they are running).
  - By imbedding correct measurements for additional “validated” code in locked areas of the flash controlled by a secure update process.
  - By sealing a secret to a PCR that contains the measurements of a validated component and is required for the system to proceed booting.

Hypervisor Responsibilities

Out of scope for the DRTM Specification, however…

- Protect identified sensitive resources provided to the DLME by the DCE.
- Protect resources identified by the ACPI DRTM table and the other tables that it identifies as security sensitive.
- AML considerations:
  - AML for a security sensitive table provided by the PM should be contained within the table – not pointed to outside of the table. This makes sequestering this AML against the gap much easier.
  - AML identified as “sensitive” by the PM will be allowed to interact with sensitive resources – other AML should not have access to these resources.

DRTM – Certification Achieveable

Today

- As previously noted, today's BIOS's:
  - Are complicated….
  - Are arbitrarily extensible…
  - Contain code in peripheral option ROMs that the PM cannot foresee…
- Translation – You can forget common criteria certification of today's general purpose BIOSes/UEFIs.
  - Same issue with or without SRTM.
  - UEFI 2.3.1 has this issue too.

Tomorrow – with DRTM

- The DRTM hardware protections remove the impact of the following from the security state of the system by placing them in the "GAP":
  - Option ROMs and other code provided by 3rd parties
  - Boot loaders, PXE code, etc.
  - DMA, Etc.
- The DRTM CC target of evaluation can be tightly constrained and well defined.
- DRTM implementations should be certifiable!

Next Steps

- **The DRTM Specification v1.0 was focused on getting X86 implementations to the marketplace ASAP – It is complete and now being balloted.**
- The DRTM Specification is written to allow maximum flexibility for implementers…
  - Providing such flexibility can cause vendors difficulties who just want to know how to execute a DRTM implementation in a minimum amount of time…
  - Solution - Collateral material – presentations, white papers, etc. on X86 implementation basics
- Additional requestors of DRTM architecture:
  - Embedded controllers
  - Non-X86 computer architectures
- What are the implications of supporting other platforms?
  - Descriptors are not necessarily ACPI
  - SMI and SMRAM may be handled differently
  - Multiple processors may be active during the boot process
- DRTM provides a very strong minimal TCB. The hypervisor/OS receiving the TCB must protect it.
  - This is out of scope for the DRTM specification. However…
  - It's important so we (TCG) will need to give guidance and/or specifications here.
- A CC Protection Profile is needed for the DRTM

D-RTM Benefit Summary

- Replacing some software protections with hardware protections
  - Hardware protects sensitive resources against gap code (eg. SMRAM, etc.)
  - DL_Event (a CPU instruction) returns the machine to a known security state following the gap code where platform security inspection may occur.
- Reducing the software attack surface
  - Much of boot firmware is complex and it is built from software integrated from many parties. By running this code in a gap with hardware protections then restoring a secure state via a CPU instruction, the need to analyze the security properties of all this code is obviated.
- Countering root kits, viruses and malware
  - Following DL_Entry, the DCE runs and analyzes the machine state and the DLME image in memory confirming that the machine is free of threats and can proceed with a trusted "stand-alone" OS launch.
- D-RTM enables machine and user identity to be trusted through the use of a TPM and the building of a transitive trust chain.
- Certification possible!

