Project Cerberus Security Architecture Overview Specification

Author:
Bryan Kelly, Principal Firmware Engineering Manager, Microsoft

Revision History

Date	Description
28-8-2017	Version 0.1 – Initial Draft
28-9-2017	Version 0.2 – Add References Section
01-22-2018	Version 0.3 – Update naming and control flow throughout
04-11-2018	Version 0.4 –Clarifications throughout

Table of Contents

List of Figures

Introduction

The cloud server continues to serve as the fundamental building block in the evolution of the cloud. The composition of the cloud server has evolved from traditional processor, memory, storage and network, to a powerful platform consisting of accelerators, offloads and smart IO devices, some more powerful than the host processors themselves. The system has evolved from Central Processing Unit (CPU) being the core instruction execution endpoint, to a fabric of sophisticated devices optimized to accelerate workloads. These accelerators and devices have independent processing units and controllers. The firmware and bootstrap code for these accelerators has long moved from single BIOS containing multiple Option ROMs, to the devices themselves carrying mutable firmware. Collectively the evaluation of the server architecture opens out new challenges to cloud hardware management and security, both during on-boarding as well as when in operation. In other words, if and when a bare-metal system is provisioned, or a cloud hardware system is repaved, one must ensure that the system is not compromised. This paper presents a vision for the inception of centralized cloud hardware management plane that extends control functions such as authentication, authorization, auditing, confidentiality, diagnostics, monitoring, system on-lining, system off-lining, debug etc.

1.1 Hardware Management

Infrastructure flexibility, scalability and reliability are core principles to operating cloud scale data centers. The flexible hardware building blocks that underpins this cloud infrastructure comes in a variety of configurations and form factors. To realize scalability in the cloud, manageability of cloud infrastructure is fundamental. Manageability must be secure, reliable and rapidly deployable. A unified approach to remote monitoring, diagnostics and hardware control is essential to rapid and repeatable deployments of modular cloud infrastructure.

1.2 Security Challenges

In a model whereby end users have Direct Access (DA) to the platform hardware or the host operating system, securing a multitude of device and operating system specific firmware interfaces becomes challenging. The following lists some existing Cloud Security Challenges:

Cloud servers have intricate boot flows that span multiple smart peripherals each containing an independent ASIC or microprocessor with disaggregated boot flows. If these peripherals do not enforce firmware digital signature authentication, any unprotected firmware update interface could become an attack vector.

Platform and peripheral firmware images are typically stored on internal flash or external NOR flash media. The media is typically writable to allow for firmware updates, during runtime or during boot. Securing the media through state transitions is not always possible.

Protection in transit across complex supply chains for Cloud infrastructure components is not easily enforced, there are multiple sources for various components comprising of cloud infrastructure.

The attack surface is expanding as coprocessors, SoC's and offload engines are becoming more popular, and their need to interconnect with the platform host CPUs over low latency bus interfaces and have Direct Memory Access (DMA) to system memory expands the attack surface for these components.

Establishing peripheral authenticity or genuine part identity at cloud scale can be a challenge. Sophisticated offloads like FPGAs and some ASICs can emulate other authentic hardware components.

Standard hardware devices can be procured by other organizations and companies. A malicious attack could occur whereby devices keyed for secure boot with another organizations asymmetric key, and made to emulate genuine part.

Establishing a consistent root of trust on very different hardware configurations while maintaining configuration and deployment flexibility is challenging. There is no uniform configuration in the cloud. Example: A systems with a host processors, has very different firmware security measures to systems without head-nodes or host processors.

2. Field Upgradeable Firmware Ingredients in Cloud

Typical Cloud hardware comprises of a host of firmware elements such as UEFI, BMC, UCODE, NIC, M.2, Hard disk, EEPROM, TPM, Accelerators, FPGAs, GPU, CPLD, etc. All system firmware is potentially an attack surface, authenticity, integrity, and confidentiality is vital. For instance, a malware author can exploit firmware updates to inject vulnerabilities that can potentially compromise the integrity of overall Cloud solution. Malware in firmware can survive system rebuilds, and firmware upgrades. They can be difficult to detect even by commercial anti-virus and security software. Having firmware attestation and a secure and reliable firmware update solution is of paramount importance. This document describes the firmware components in Gen6 hardware and the security model.

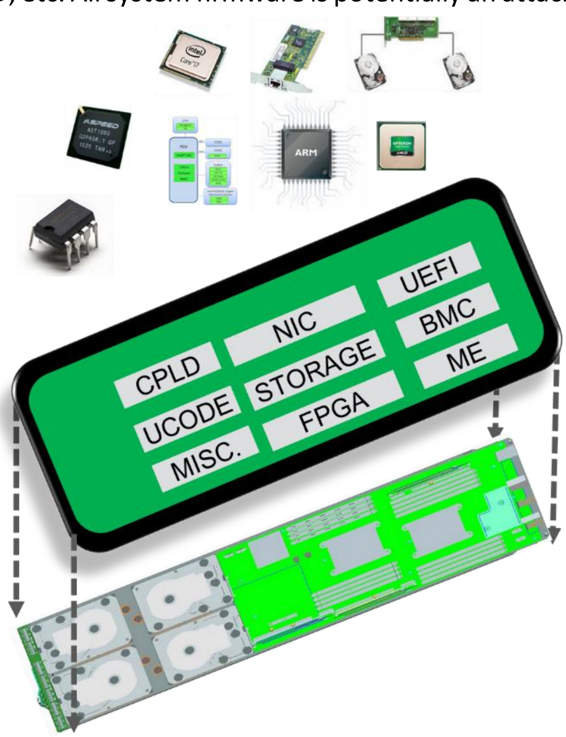


Figure 1 Firmware Components

2.3 System vs. Device Firmware

As mentioned above, there are a plethora of firmware ingredients. These can be categorized into to two broad categories. Firmware elements that are mandatory for system boot are typically referred to as System Firmware, for e.g. UEFI, BMC, etc. Firmware elements that are typically optional for system boot are referred to as Device Firmware, for e.g. NIC Option ROM (although the name contains ROM, these are often mutable), SSD Firmware etc.

2.4 Firmware update Provisions

All firmware updates on cloud platforms must be accomplished in a scale friendly manner. There are two means through which firmware can be updated; inband refers to software that is executed by host processor, and out-of-band through the Service Processor. The inband update can further be categorized as preboot and OS runtime based updates. In cloud hosting models where a customer may choose to boot to any operating system of their choice, any OS runtime based firmware update can potentially be an attack vector. Thus, maintaining supervisory control over firmware update from an OS context, prohibiting unsigned updates and detecting and remediating firmware exploits is critical.

2.5 Root of Trust

Establishing a core root of trust along with chain of trust is fundamental to overall platform security and being able to attest to its integrity. However, most industry standard trust chains only comprehend the platform BIOS and BIOS loaded peripheral Option ROMs, through to OS loader. These trust chains only cover a small portion of the total firmware present in a modern cloud server.

2.6 Secure Boot

Secure boot adds cryptographic checks to the boot process. The purpose is to verify the integrity and authenticity of firmware or firmware boot loaders that follow the primary ROM boot loader. This firmware is typically mutable and referred to as the Second Stage boot loader.

Secure boot can help establishes a firmware trust chain, as the immutable first stage ROM bootloader authenticates the mutable second stage loader, the second stage loader can authenticate subsequent firmware components. The secure boot process typically uses asymmetric ECDSA or RSA digital signature verification, whereby the ROM verifies the digital signature public key for the second stage bootloader against a hash of the public key stored in nonvolatile One Time Programmable (OTP) memory. For information on the secure boot process, refer to the document: Cerberus Security Firmware Cryptography Signatory.

2.7 Flash Encryption

Flash encryption adds confidentiality to firmware and the data stored within the flash media. Flash encryption is separate from secure boot, but enhances security by obfuscating the data contain on persistent media.

3. Cerberus – Hardware Enforced Platform Security

The Cerberus platform design is a hierarchical Root of Trust (RoT) architecture. All firmware is required to be secure. Processors, SoCs, microcontrollers and active components with firmware must adhere to the security and attestation requirements set forth in the Cerberus specifications. If components do not intrinsically meet the requirements, they must either place a designated security microcontroller that presents a SPI interface between their firmware load store and their processing unit, or change their controller to one that meets the security requirements set forth in the Cerberus design.

All active components are required to support both hardware and firmware combined identify through the Device Identifier Composition Engine (DICE). The DICE Compound Device Identifier (CDI) is a one-way hash of both the immutable Unique Device Secret (UDS) and the first stage mutable boot loader digest, forming a single hardware firmware composite key source known as the Compound Device Identifier. Combined with immutable secure boot, this measurement forms the foundation on which trust is established. The cryptographic device identity and attestation scheme for Cerberus is based on DICE and RIoT Core. It uses X.509 Device ID certificates and Alias Certificates for an unclonable identity. The implementation of the certificate and key chaining is described in documents referenced in the section: 4.3 DICE and RIoT Keys and Certificates.

Each independent active component in the Cerberus platform design is a root of trust for its functional domain; reporting measurements to the challenges from the platform RoT.

Active Components in the modern server boot to an operational level before the platform's host processors complete their initialization and become capable of challenging the devices. In the Cerberus design, the platform is held in power-on reset while Active Components are provided power for their RoT decompress and respond to challenges from the PA-RoT confirming the integrity of their firmware before they are released from Reset. Components that fail their challenge will be powered-off and not receive voltage, additionally the platform may be held in a reset/powered off state.

3.1 Power-on

The Cerberus designed motherboard has a T-1 power-on state whereby the power signals remain negated after rails stabilize. The motherboard RoT microcontroller powers-on, and authenticates its firmware before decompressing and loading its firmware, deriving RIoT and attestation certificates. Once loaded the Cerberus microcontroller signals for the auxiliary power rail, allowing power to the BMC while holding the BMC in reset. The Cerberus microcontroller verifies the integrity of the BMC NOR Flash, verifying the digital signature and recalculating the BMC firmware digest. If the firmware digest matches the expected signature, Cerberus releases the component from reset. When released from reset the BMC queries the RoT for its boot blocks, the RoT actively controls firmware content access to the BMC.

When the BMC has been loaded, the platform RoT signals power to the platform CPUs, while holding the system in reset. This enables power flow to PCIe devices and host CPUs. The host CPUs and any chipset components remain powered but held in a reset state. The RoT verifies the integrity of the host CPU NOR flash, verifying the digital signature and recalculating the host firmware digest. At the same time, the RoT queries PCIe devices in the Component Firmware Manifest (CFM) over I2C. PCIe components in the Cerberus designed motherboard are required to enable their RoT on standby power whereby the PCIe clock is off and reset is asserted. The PCIe Component RoTs verify the integrity and authenticate their firmware while in a powered but quarantined state. The platform RoT will challenge the component RoT's for their firmware measurements and extend the platform measurement. For detailed information on this exchange, refer to the Cerberus Challenge Protocol Specification.

After completing the component measurement challenge the Platform RoT will determine whether to continue providing 12v to all or some of the PCIe slots, or whether to keep certain components powered off or held in reset. When the Platform RoT has completed the component challenge and determined the which PCIe slots remain powered, the host CPUs are released from reset. As the platform firmware is loaded, the RoT muxes control of the I2C to the BMC for temperature monitoring of active components. During runtime operation, the platform RoT can request the BMC yield control of the I2C, repeating its measurement challenge on demand or patrol schedule. In some platform designs, it is also possible for the RoT to proxy request through the host BMC.

3.2 Secure Boot

The Platform Active RoT and Active Component RoTs use digital signature algorithms such as RSA or ECDSA. A Public Key digest stored in One Time Programmable (OTP) memory is used to authenticate the Public Key used for firmware signature verification. After first verifying their own firmware, the RoT's use a similar process of verifying the flash contents of the components they protect, calculating the digest and comparing the signature against a key manifest stored in the RoT Platform Firmware Manifest (PFM) or Component Firmware Manifest (CFM) for components. For details on the secure boot process, review the document: Cerberus Security Firmware Cryptography Signatory specification.

3.3 Attestation

The Cerberus measurement challenge is a hierarchical attestation architecture. Firmware integrity measurements are gathered locally by the Component RoTs and reported to the Platform Active RoT (PA-RoT) during measurement challenge. The Platform uses the collected component measurements combined with the platform measurements to create a platform measurement. The Datacenter Management Software collects the platform measurement.

Active Components in a traditional server boot to an operational level before the platform's host processors complete their initialization and become capable of challenging the devices. This creates an edge condition, whereby compromised component firmware could go undetected, or potentially spread during boot and compromise other components. In the Cerberus design, the platform is kept in the power-on reset mode, while the Active Components are challenged by the Platform Active RoT (PA-RoT). Upon successful verification of their firmware integrity, the PA-RoT releases platform reset.

The PA-RoT maintains a Platform Firmware Manifest (PFM) and Component Firmware Manifest (CFM). The manifests are programmable through the PA-RoT's communication interface, and updatable only through digitally signed and encrypted firmware capsules. The capsule update does not require host update or reset. Auto-detection of Active Components and computation of the PFM and CFM maybe considered in future version of the "Cerberus Challenge Protocol" Specification.

The PA-RoT uses the manifest to challenge the Active Components and record their measurements. The PA-RoT uses a digest of these measurements for the platform level measurement, creating a hierarchical platform level digest that can attest the integrity of the platform and active component firmware.

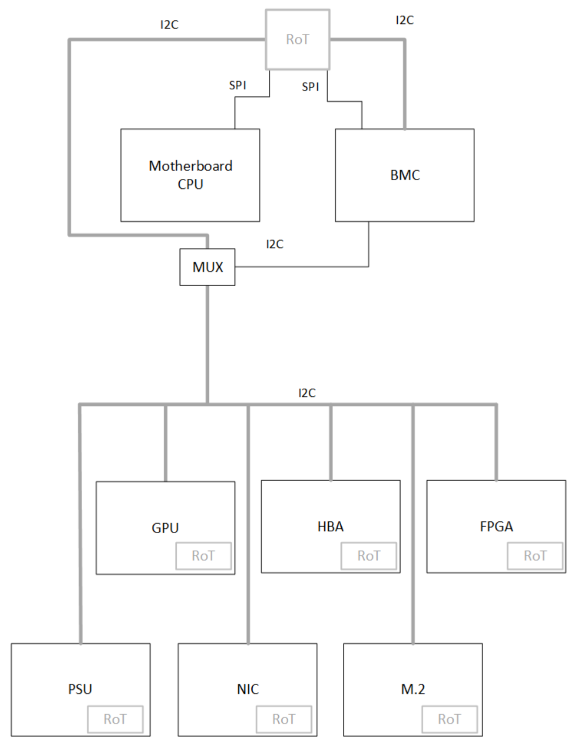
The PA-RoT and all Active Components RoT's (AC-RoT) will support Authentication, Confidentiality and Integrity of message challenges.

The PA-RoT is responsible for challenging the AC-RoT's and collecting their firmware measurements. The PA-RoT retains a manifest of active components that includes addresses, buses, firmware versions, digests and PKs. This manifest is referred to as the Platform Configuration Data (PCD).

The manifest informs the PA-RoT on all the Active Components in the system. It provides their I2C addresses, and information on how to verify their measurements against a known or expected state. Polices configured in the Platform RoT determine what action it should take should the measurements fail verification.

In the Cerberus designed motherboard, the PA-RoT orchestrates power-on. Policy settings defined in the PCD determine whether Active Components that fail verification will be released from power-on reset.

Figure 2 Physical Challenge Channels



3.4 Platform and Component Firmware Manifests

The PA-RoT contains a Platform Firmware Manifest (PFM) that describes the platform firmware permitted to run on the platform, and Component Firmware Manifest (CFM) that describes the Active Component firmware permitted in the system. Note: The PFM is different from the boot key manifest described in the Processor Secure Boot Requirements specification. The PFM describes firmware permitted to run on the platform, while the CFM describes the firmware permitted on Active Components. A record of actual platform and component measurements is generated by the PA-RoT following platform and component challenges. These records are logged in the Reported Firmware

Manifest (RFM). The PFM, CFM and RFM are stored encrypted in the PA-RoT. The symmetric encryption key for the PA-RoT is hardware generated and unique to each microcontroller. The symmetric key is not exportable or firmware readable; and only accessible to the crypto engine for encryption/decryption. For further information on the PFM, CFM, RFM and measurement challenge review the "Cerberus Challenge Protocol" specification.

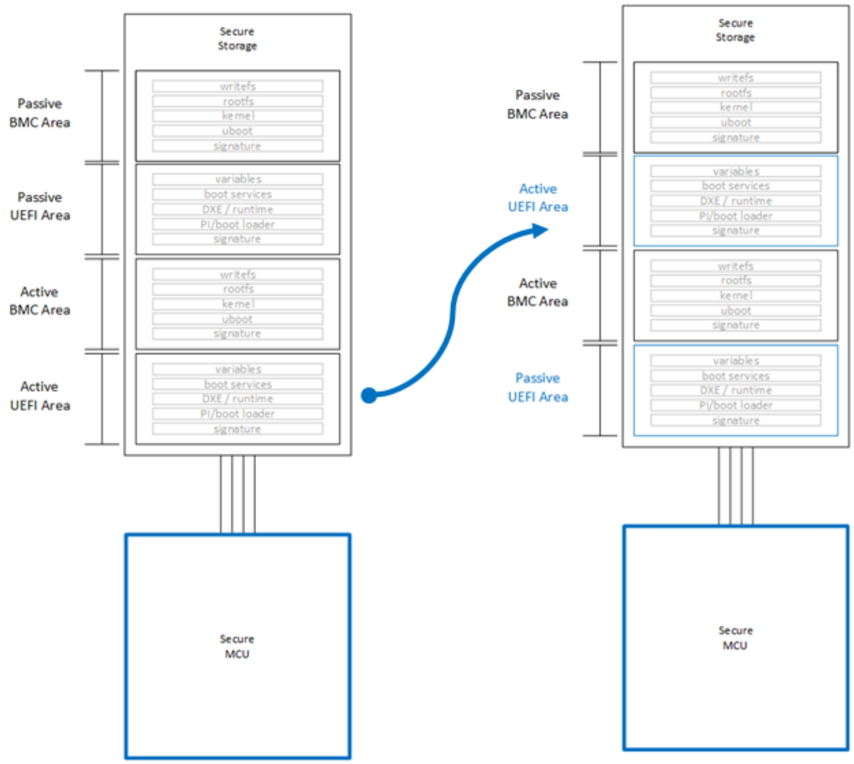
3.5 Recovery

The RoT's recovers firmware to a known good state if for any reason the firmware integrity becomes corrupted. Detection of corruption occurs through the verification of the signed digest compared with the actual or calculated digest. RoTs have a recovery region for their own firmware and the firmware of the component they protect. By default, the firmware region groups are "A" (Active) and "B" (Backup). The A and B firmware regions are updatable, but not be simultaneously updatable. The "B" region is only updatable with a verified "A" region, and vice versa.

The RoTs have multi-stage boot loaders and contain subareas within each firmware region. The subareas containing the second stage bootloader (key manifest + RIoT Core) is typically not updated, however updates are possible to this region. There is very little execution code in the second stage boot loader area. Changing the second stage boot loader key manifest would result in regeneration of the DICE CDI. The second stage bootloader area contains the boot verification public keys and third stage bootloader straps. This area is normally only updated during key revocation.

As the RoT interposes between the component and SPI flash, it governs reads and writes to various flash regions. The RoT has the ability to limit read and write access to different regions of flash for the component it protects, preventing corruption and ensuring firmware is written to a staging area, where verification is enforced before transitioning to the active region.

Figure 3 Firmware Recovery of Active Area shifting



The RoT prevents direct writes to A regions and permits a single region update at a time. Key revocation triggers sequential updates following verification of integrity of a given region. Refer the Project Cerberus Firmware Update Specification for additional details.

Devices that have firmware regions containing changeable variables and temporal logs, should partition these regions separately from the firmware code. These writable regions should have reportable digests separate to the digital signature for authenticating the firmware. Refer to the Project Cerberus Host Processor Firmware Requirements specification.

Policies in the RoT PCD determine which default recovery action should be taken upon detection of firmware corruption. By default, the RoTs will force maintain firmware in a good state (updating B maintaining A). This requires that A images are maintained current and unrevoked. Refer to the Project Cerberus Firmware Update Specification for additional details.

3.6 Firmware Update

The RoT internal flash containing the firmware for the RoT is partitioned into two sections: Active (A), Backup (B). The A partition contains the current firmware image being used by the processor. The B partition maintains a known-good firmware image that can be used to restore the A partition in the event of corruption. Whenever updates are attempted, valid B image is always verified. If B is corrupt, then B must be updated before A can be attempted.

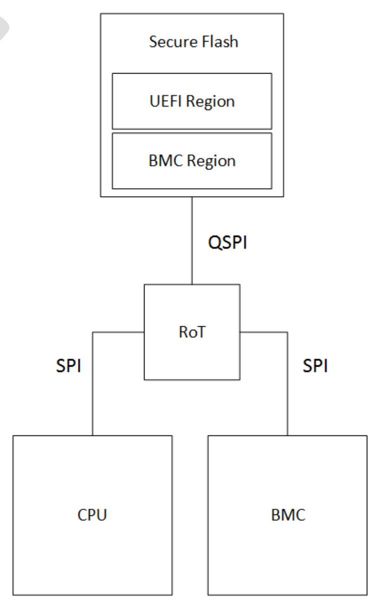
The RoT extents the A, B firmware update methodology to all platform firmware components. Similar to the RoT's update and authentication mechanism, the RoT extends the update mechanism to the flash it protects. The RoT does not permit the A or B firmware code areas to get updated while the other is deemed corrupted. Instead, only after a validate recovery firmware is available can the A area be updated. In this model the A and B areas may be interchanged. Only validated firmware is committing to flash. From a resilience perspective, there is a window whereby A area is updated, should a power loss even occur it's possible that B recovery would be needed. Therefore, verifying B before writing A is required.

The authentication of firmware updates verifies the digital signature of the firmware image against the corresponding public key in the key manifest, and ensuring the firmware exists in the Platform and Component Firmware Manifest. Cryptographic digital signatures generated with a valid private key corresponding to the public key in the Platform Key Manifest, are required for all firmware updates. Additionally, the firmware attributes should exist in the Platform Firmware Manifest or Component Firmware Manifest.

3.7 Authenticating Flash Writes

Regions in the firmware image are separated into read-only and read-write areas. For example, in an embedded Linux kernel firmware base such as the BMC; uboot, the kernel and Root File System are read-only, while the overlay directories are for configuration and log files are stored in a different region that is write permitted. Similarly, in UEFI it may be desirable to have certain variables writable while integrity on the boot block and code regions of the firmware are protected. The secure RoT enforces security on flash-based boot devices by maintaining writable and read-only areas, with integrity digests with either RSA or ECDSA signature authentication.

Figure 4 Flash Offset



The RoT is required to have knowledge of the flash layout for each of the protected components, for the PA-RoT this would include the CPU chipset or BMC. The RoT maintains the secure encrypted flash with RSA or ECDSA key based authentication for images for both the CPU and BMC. The RoT maintains two separate SPI interfaces to the CPU and BMC. The RoT will offset the read/write requests to different locations on the physical flash based on the SPI interface the commands is received.

3.8 Firmware Storage

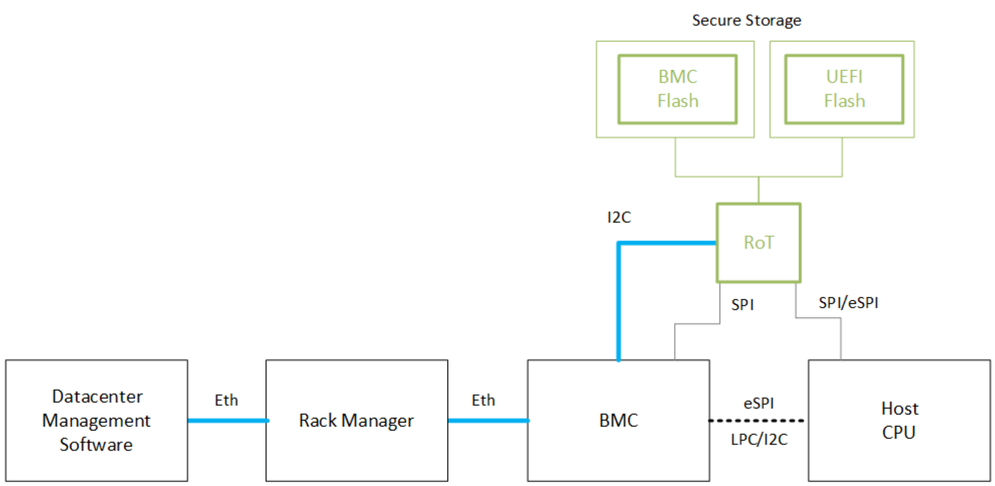
The RoT internal flash supports encryption with a device unique secret Physical Unclonable Function (PUF) key, or secret derived key. This special key provides AES encryption of the flash unique to the device creating a device specific vault for storying device measurements. The encrypted flash area is device specific, and decrypted only by the physical RoT with the unique PUF.

3.9 OOB Reporting

The initial RoT firmware authentication and platform challenge occurs pre-platform power-on. Should verification and recovery fail for any reason, RoT platform policies may determine the platform needs to stay powered off. Datacenter Management Software responsible for Out-Of-Band (OOB) management of the server fleet obtains the failure log and encrypted challenge measurements. The failure log triggers remediation actions in fabric, with results in the node been kept offline until the issue has been resolved.

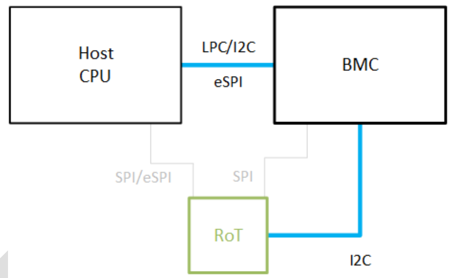
The platform RoT is accessible to the Data Center Management Software through an encrypted channel tunneled through the Rack Manager and BMC. The Rack Manager interfaces with the Datacenter Management Software over the manageability LAN. The Rack Manager interfaces with both the Management LAN and the Baseboard Management Controller (BMC). The BMC interfaces with the RoT over I2C, offering an Ethernet to I2C tunnel. The channel enables the Datacenter Management Software control hardware and apply firmware polices that persist on systems where the host operating system is unmanaged or inaccessible.

Figure 5 RoT Interfaces



Servers with host CPUs and Microsoft hosted operating systems have the option for communicating with the RoT via a secondary BMC tunnel through the in-band KCS interface to the BMC. This interface is physically I2C, LPC or eSPI to the BMC. The SSIF or KCS protocol from the host to the BMC can create a secure virtual tunnel and enable the host CPU to authenticate with the RoT and retrieve firmware measurements.

Figure 6 IB Interface



4. References

4.1 DICE Architecture

https://trustedcomputinggroup.org/work-groups/dice-architectures

4.2 RIoT

https://www.microsoft.com/en-us/research/publication/riot-a-foundation-for-trust-in-the-internet-of-things

4.3 DICE and RIoT Keys and Certificates

https://www.microsoft.com/en-us/research/publication/device-identity-dice-riot-keys-certificates