Intel® Converged Security and Management Engine (Intel® CSME)

Security White Paper

November 2020

LEGAL NOTICE AND DISCLAIMER

Intel provides these materials as-is, with no express or implied warranties.

All products, dates, and figures specified are preliminary, based on current expectations, and are subject to change without notice.

The products described might contain design defects or errors known as errata, which might cause the product to deviate from published specifications. Current, characterized errata are available on request.

Intel technologies might require enabled hardware, software, or service activation. Some results have been estimated or simulated. Your costs and results might vary.

No product or component can be absolutely secure.

Table of Contents

Table of Figures

Introduction

Intel platforms are designed with a strong built-in security foundation. This allows the ecosystem partners to help protect the platform data and to build more trusted applications.
The Intel® Converged Security and Management Engine (Intel® CSME) was developed as a hardware-based manageability and security controller isolated from the CPU (Central Processing Unit). Intel CSME is the system's root of trust for Intel components (and optionally for an Original-Equipment Manufacturer (OEM) if it decides to use it).

The purpose of this white paper is to describe the security design and implementation of CSME 14.0 (Comet Lake) and CSME 15.0 (Tiger Lake) and its role in the platform.

What is Intel CSME?

Intel CSME is an embedded subsystem and a PCIe (Peripheral Component Interconnect Express) device that is designed to act as the security and manageability controller in the PCH (Platform Controller Hub). Intel CSME aims to implement a computing environment isolated from the main, CPU-executing, host software (SW) like BIOS (Basic Input Output System), OS (Operating System) and applications.
Intel CSME can access a limited number of interfaces, such as GPIO (General-Purpose Input/Output) and LAN/Wireless LAN (WLAN), in order to perform its intended operations. As designed, Intel CSME's firmware and configuration files are stored in NVRAM (Non-Volatile, Random-Access Memory), such as flash memory on the SPI (Serial-Peripheral-Interface) bus.

The following figure shows Intel CSME's position in the system:

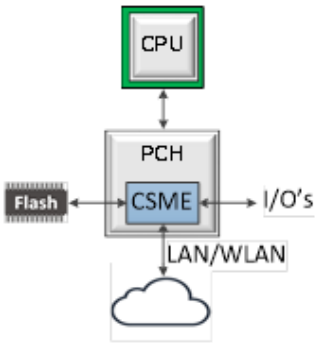

Figure 1 - Intel CSME in the system

Intel CSME is present on most Intel platforms, including client consumer and commercial systems, workstations, servers, and IoT (Internet of Things) products.
For hardware (HW)-based security, users such as content providers or IT (Information Technology) organizations can manage, for example, DRM (Digital-Right Management) and Intel® Active Management Technology (Intel® AMT), which requires hardware-level security to be available when the host is not responding or is powered down.

Intel CSME is designed to serve three main platform functions:

1. Silicon Initialization

Intel CSME is designed to serve as the platform's silicon initialization and be responsible for:

- Base initialization of PCH, including configuration of clocks and GPIO
- Authentication and loading of FW into HW engines (aka IPs) integrated within the main CPU and PCH, *e.g.*, Power-Management Controller (PMC), Audio, Camera, Type C, and Sensor FW
- Secure debug of the PCH

2. Manageability

As part of the Intel® vPro® platform, Intel AMT enhances the ability of IT organizations to manage enterprise-computing facilities by providing remote-management capability that is independent of the OS. Remote platform-management consoles are designed to access Intel AMT securely, even when the platform is powered off, as long as the platform is connected to power and to a network.

Intel-based platforms with enabled Intel AMT include the following capabilities:

- SOL (Serial Over Local-Area Network) is the capability of having a console-redirection feature
- USB-R (Universal Serial Bus Redirect) is a storage redirection feature that can be executed remotely
- KVM (Keyboard, Video and Mouse) enables the remote control over network
- Remote power control
- Alarm Clock includes scheduled wake ups for proactive maintenance
- Robust asset management enabling hardware-asset discovery and hardware information and location.
- Out-of-Band management, which includes:
    - OS-independent diagnostics
    - Non-volatile storage
    - Event management
    - Remote-control capabilities
    - System-Defense capability
    - Agent-Presence capabilities
    - Out-of-band connectivity to an Intel AMT-based platform located outside an enterprise intranet when using Intel® Endpoint Management Assistant and the CIRA (Client Initiated Remote Access) AMT feature.

3. Security

In recent years, security has increasingly become the focus of the engine. With the engine supporting both manageability and security, Intel CSME's design has added several, additional functions:

- Intel® Platform Trust Technology (Intel® PTT) is an integrated TPM (Trusted-Platform Module) compliant with the TCG TPM 2.0 standard. It enables support of UEFI (Unified

Extensible Firmware Interface) secure boot, disk encryption, secure storage, virtual smart card, remote-attestation-use cases, and all Microsoft requirements for integrated TPM if latest Win 10 OS version (19H1/H2) is installed.

- Intel® Boot Guard is a feature that aids boot-execution integrity through a chain of trust. Each module is designed to authenticate and load the next module in the boot sequence, starting from the platform root of trust. Intel CSME enables storing the Intel® Boot Guard policy in PCH Field-Programmable Fuses (FPF), so that the user can be more confident that the system is running an authentic, OEM BIOS.
- Intel CSME supports HW DRM that helps users enjoy premium services from third-party providers, with control access to copyright material
- Intel CSME enables secure loading and execution of Intel-signed DAL (Intel® Dynamic Application Loader) applets and secure firmware loading of other platform-firmware components, such as TBT (Thunderbolt), Type-C, Sensor Solution FW, etc.

Intel CSME has the capability of helping safeguard the system and protecting digital assets for users at the hardware level, which capabilities cannot be provided by programs running on the host.

Since CSME plays a critical, security role in Intel platform, Intel is committed to harden Intel CSME and implement various defense-in-depth mechanisms to help prevent abuse and attacks.

Intel CSME Hardware Overview and Capabilities

From a hardware perspective, Intel CSME is composed of a converged-manageability-hardware extension and a converged-security engine. The manageability engine contains hardware blocks connected via internal fabric that facilitate Intel AMT and debugging functionalities. The security-block design consists of the following subsystems:

- Dedicated CPU, a 32 bits processor based on Intel 486 architecture, supporting privilege execution levels, aka rings (**note:** any reference to ring in the rest of this document refers to the CSME CPU rings and not the Main CPU rings), segmentation, MMU (Memory Management Unit) for page management and also CET (Control Enforcement Technology) starting Tiger Lake platform.
- Dedicated, internal SRAM (Static Random-Access Memory) isolated from host and that cannot be probed via the chipset's external interfaces. The size of the SRAM ranges from 512KB to 1,920KB, depending on the Intel CSME SKU. The amount of SRAM required on a SKU depends on the set of applications that the SKU supports.
- ROM (Read-Only Memory) is the HW root of trust of Intel CSME firmware
- System Agent allows the CSME CPU to securely access SRAM and helps enforce access control to SRAM from internal/external devices (*e.g.*, DMA access) by using a dedicated IOMMU (Input Output Memory Management Unit).
- OCS (Offload & Cryptography Subsystem) is a Cryptography-HW accelerator with DMA (Direct-Memory-Access) engine and Secure-Key-Storage (SKS) Hardware. The SKS Hardware aims to protect keys from potential leakage from CSME SRAM, while CSME Firmware runs. It helps to ensure that CSME Firmware can use keys without knowing their actual, plaintext values. The

SKS HW contains several, SKS slots where a key can be stored. The key size could be 128 bits, 256 bits, or even 384 bits in TGL (Tiger Lake) platform with TGP (Tiger-P) PCH. Each SKS slot is designed with its own set of attributes

- **Secure Mode:** Result of AES-CBC decryption (used for unwrapping key previously wrapped by an SKS key, *i.e.*, the Wrapping Key) and HMAC (used for generating new key) can be stored in SKS only
- **Privilege Level:** Used for hardware-access control on the SKS slot. The key in this slot should be accessible if the SKS-slot-privilege level is greater or equal to the SKS-privilege level of the OCS.
- **Locked:** The key in this slot can be invalidated or replaced after Intel-CSME-hardware reset.

- Gasket is an interface to PCH fabric and CSME-IO (Input/Output) devices like TPM and HECI (Host-Embedded, Communication Interface), which represents the communication protocol between Intel CSME and the host.
- Manageability Devices used to help manageability and redirection (USB-R, KT, KVM, etc.)
- Protected, Real-Time Clock (PRTC) used for monotonic counters (anti-replay protection) and protected time

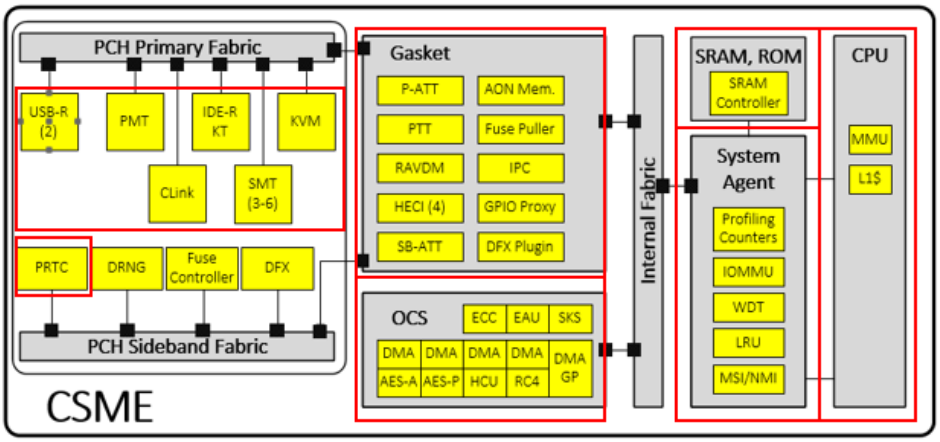

Figure 2- Hardware Architecture Conceptual Diagram

CSME's design also uses other PCH hardware blocks (aka IP) such as:

- The Intel® DRNG (Digital, Random-Number Generator), which is designed to generate non-deterministic, random numbers and complies with NIST SP800-90A, B, and C.
- The fuse controller, which helps manage two types of fuses: Intel PCH Manufacturing fuses and FPFs.
    - Intel PCH Manufacturing fuses, set by Intel manufacturing before shipment to OEM/ODM manufacturers, are used to set CSME configurations (*i.e.*, enablement of specific, Intel-CSME, signing keys, production silicon) and contain CSME-security keys that are unique per chip and encrypted using CSME-HW key (refer to The Fuse-Encryption Hardware Key, the Chipset Key and its derivatives section for details on CSME HW Key)
    - FPFs are set by OEM/ODM manufacturers before shipment to end-users and contain the manufacturers' secure settings, such as public key and Intel Boot Guard policy. They also

contain the CSME FW Anti-Rollback Security-Version Number (refer to Firmware Update and Anti-rollback (ARB) section for details on ARB)

- The DFX (Design for Testability, Debug, Security, and Validation) that helps control CSME and other, PCH micro-controllers debug interface such as JTAG. The DFX IP is also used to indicate the debug policy to other IPs within the PCH, such as CSME and fuses. Once the DFX policy is changed, the intention is that IPs are notified and will scrub any secrets that might be present inside the IP before the IPs open their debug interface

Intel CSME Firmware Throughout Platform Boot (By Component)

Intel CSME uses a typical, embedded-system architecture consisting of a boot ROM and updatable firmware. The boot ROM is permanent hardware with no patching mechanism. By design, the firmware, in its compressed form, is stored in NVM as part of the system firmware, which includes BIOS, micro-code patch, ACM (Authentication-Code Module) and others. The Intel-CSME firmware utilizes Huffman and LZMA compression to help reduce its footprint in Non-Volatile Memory (NVM).

During system boot, the boot ROM is designed to invoke DMA to load the Intel CSME binary image into internal SRAM and verify the image. The dictionary of Huffman compression is built into the SPI and decompression is conducted by the SPI controller. In contrast, as intended, the LZMA decompression is performed in firmware by the CSME process-manager module.

The CSME firmware image is digitally signed by Intel with standard, RSASSA-PSS (RSA-Signature Scheme with Appendix – Probabilistic-Signature Scheme) with a 3072-bit, RSA key and SHA-2-384 (Secure-Hash Algorithm) in CSME 15.0 and RSASSA-PKCS1-v1.5 (RSASSA Public-Key-Cryptography Standards) with a 2048-bit, RSA key and SHA-2-256 in CSME 14.0. The goal is that the public key itself and the signature are part of the firmware manifest, while the hash of the public key is hardcoded in the boot ROM. There is also a signing-key mask designed into the Intel-HVM (High-Volume-Manufacturing) Fuses to help indicate to the CSME ROM which signing key is active. The associated private signing keys are kept in an HSM (Hardware Security Module) within an Intel secured facility and are refreshed every new major generation, therefore CSME 14.0 and CSME 15.0 have different signing keys.

This section describes Intel CSME's components and the intended action of each component during platform boot.

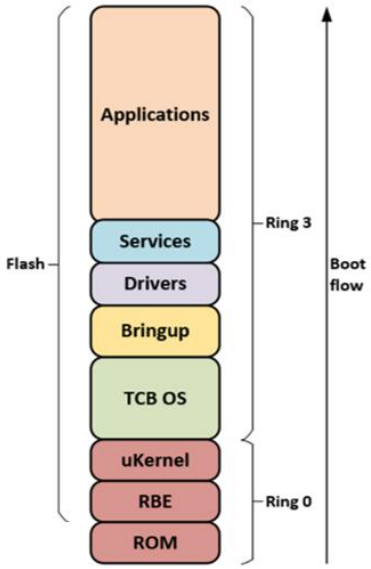

Figure 3 - Intel CSME Firmware components

1. CSME ROM

The Intel CSME ROM is integrally part of CSME HW in PCH and does not have any patching mechanism after silicon tape-in. Once an Intel HVM fuse is set during manufacturing of the chip by Intel, there is no way to subsequently bypass the ROM on production stepping.

The ROM has three, main goals:

- Initializing the CSME HW and enabling the Intel CSME CPU to work in protected mode with paging and segmentation.
- Generating the Intel-CSME-firmware keys by using the chipset key, and RBE (ROM-boot-extensions) Security-Version Number (SVN) from Firmware-manifest header.
- Loading, authenticating, and executing the RBE and Intel Debug-Launch-Module (IDLM), if present in SPI flash, by using the public-key hash embedded in ROM code and an HVM fuse indicating which key is enabled.

1.1. The Fuse-Encryption Hardware Key, the Chipset Key and its derivatives

As designed, the CSME Chipset Key is the master of all keys generated by the CSME ROM and by CSME Firmware at runtime. Main usages of those keys are to provide cryptographic protection of CSME code/ data and support attestation-based services needed by CSME applications like DRM, PTT and DAL.

All chipset keys are randomly generated by Intel's Key Generator Facility (iKGF) for Intel platforms, encrypted with a PCH family Hardware Key (the Fuse Encryption Hardware Key), and sent to Intel

Assembly facilities for final programming in the PCH manufacturing fuses before shipment to OEM, system manufacturers.

The table below lists security fuses that can be programmed during Intel Manufacturing:

Name	Length in bits	Encrypted with Fuse Encryption Hardware Key
Chipset Key (not available in debug)	256 (CML) / 512 (TGL)	Yes
EPID 2.0 Private Key	256	Yes
EPID 2.0 Group ID	32	No
PTT EK Counter	32	No
Chipset Key (available in debug)	128 (CML) / 256 (TGL)	No

The Fuse-Encryption, Hardware Key aims to prevent CSME-security keys from being exposed, should they be successfully extracted from the silicon fuses.

> **Note:** The Fuse Encryption Hardware Key is not meant to protect chipset key and Intel® Enhanced-Privacy ID (Intel® EPID) private key from the CSME ROM.

When the Intel-CSME ROM starts executing, it is designed to execute the following sequence:

1. The ROM requests from the Cryptography-Hardware accelerator to generate the Fuse-Encryption-Hardware Key.
2. The ROM sets the SKS-slot attributes (privilege level, secure mode, and lock) and then loads the Fuse-Encryption-Hardware Key into the SKS Hardware.
3. The ROM pulls the security keys from the fuses into CSME SRAM and then locks the security-key fuses.
   Note: Once the fuses have been locked, it is not possible to retrieve the security keys until the next Intel-CSME reset; this seeks to allow that only the Intel CSME ROM can retrieve the security keys.
4. The ROM decrypts the chipset key encrypted with the Hardware Key from Intel-CSME SRAM into SKS.
5. The ROM can use the chipset key to derive additional keys using a) HMAC-SHA256 as a Key-Derivation Function, b) hard-coded string (different per key), and c) CSME Firmware Security Version Number after the CSME-Firmware manifest has been verified by CSME ROM.

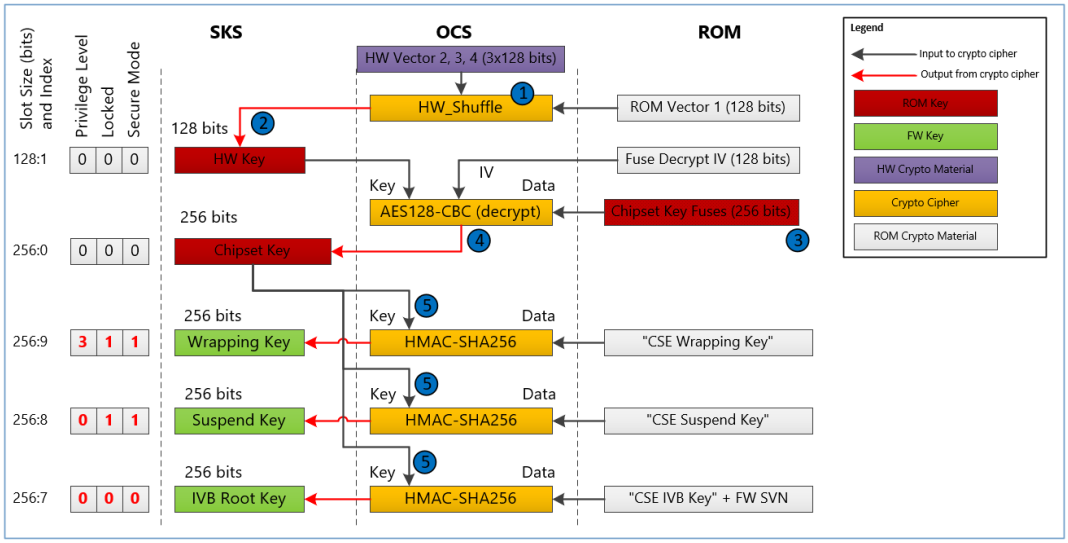

Figure 4 - Keys Derivation Conceptual Diagram

The ROM can create several keys from the Chipset Key:

1. **Intel Root Key** –the master key for the Intel CSME firmware. For example, used to generate storage keys to help protect Intel CSME data in NVM.
2. **Wrapping Key** –used to wrap a key in SRAM and help ensure that the key is unwrapped into SKS so it can be used. Once a key has been wrapped, it is designed to be bound to the Intel CSME Hardware and not be used outside Intel CSME.
3. **Memory Guard Key** –used to wrap and unwrap a key in SRAM. This key is typically used to protect attestation keys like Intel EPID or Endorsement Key (EK) that cannot be stored in SKS, which supports only Advanced-Encryption Standard (AES) or Hash-Based, Message-Authentication-Code (HMAC) key types.
4. **Suspend Key** –used to generate one-time keys to help allow saving and restoring the Intel-CSME-Firmware context when Intel CSME enters Power Gating or when the platform goes into standby (on client systems only). This key can be critical to Intel-CSME-execution integrity upon resuming.
5. **IVB (ICV Blob) Root Key** –used to generate keys that help protect the ICV (Integrity-Check Value)-Private Key (IPK) and the Intel-CSME-firmware modules' ICV in the ICV blob kept in IVBP partition. Those ICVs are calculated using IPK over Intel-CSME-firmware module's 4KB pages in NVM and the Cryptography-Hardware accelerator AES-P (Programmable, Advanced-Encryption, Standard Engine). It is designed to allow fast boot of Intel-CSME Firmware and secure page-in / page-out of Intel-CSME code and data from Intel-CSME SRAM to SPI/DRAM (Dynamic, Random-Access Memory). By design, the IPK and ICV would be invalidated by an Intel-CSME-firmware update or by an ICV-integrity failure. IVB Root Key, keys derived from it and IPK are crucial to Intel-CSME-execution integrity during boot and runtime.
6. **Provisioning Key** –used by the-Intel CSME firmware in CML (Comet Lake) to help retrieve a new, Intel-EPID key from the Intel server when TCB (Trusted-Computing Base) Recovery is required in the event the Intel-CSME-EPID-attestation key has been compromised. The Provisioning Key is

based on the Intel-CSME-firmware SVN. Using this key is relevant when the SVN is greater than 1.

7. **Intel PTT EK Root Key** –by design, exported by the Intel-CSME ROM, encrypted with the Memory-Guard key, and used by Intel PTT in CML to generate its EK by using the PTT EK counters that are also exported by CSME ROM to Intel PTT.
8. **Intel On-Die-CA (Certificate Authority) FW private ECC (Elliptic-Curve-Cryptographic) key** – used by the Intel-CSME firmware (Crypto Driver) in TGL to issue unique keys and x509v3 certificates to CSME applications (*e.g.*, PAVP, AMT, DAL) to help prove their identity to a remote server. Note, in TGL platform, the On-Die-CA keys replace EPID key.

Once the Intel-CSME ROM has finished generating all the keys, it is designed to zero the chipset key and the fuse encryption key (and lock them) before executing the Intel CSME firmware. This is done to help ensure that these keys cannot be accessed and used from outside the Intel CSME ROM.

2. Intel CSME ROM Boot Extension (RBE)

The Intel-CSME RBE extends ROM functionality in firmware. The RBE is designed to be updated in the field and address initializing and configuring the Intel CSME for the boot. The RBE can be viewed as a bootloader for the Intel-CSME operating system.

The RBE has three, main goals:

- Perform hardware-based, anti-rollback check on the Intel CSME firmware. This applies to all platforms up to and including Intel CSME 14.0. From Intel CSME 15.0, the ROM is designed to perform this check.
- Perform early silicon initialization task, like authentication of debug token as well as secure loading of PMC FW and data. Intel CSME assists the main CPU boot by applying a PMC-firmware patch before the main CPU comes out of reset as a system cannot boot without a proper, PMC patch.
- Load, authenticate, and execute the Intel-CSME, core, operating system.

3. Intel CSME Secure Boot Flow

As explained in previous section, the CSME boot process is designed to always start in ROM and continue in RBE; the intention is this chain of trust is extended by the CSME kernel and process manager, which help load and authenticate the rest of the firmware from external memory (SPI flash) as shown in Figure 5 below.

The CSME RBE and FW kernel create ICVs for every page of the CSME-FW module (drivers, services, or application) loaded and authenticated by process manager. The Intel-CSME-page size is 4KB.
The BringUp process can expose HECI commands for BIOS to tell CSME that the system memory has been initialized and can be used by CSME to configure the IMRs (Isolated Memory Region). One of these IMRs is used by CSME for its page replacement mechanism that is needed because the entire CSME FW cannot fit into its SRAM. Therefore, Intel CSME is designed to utilize this isolated region in the system's DRAM to securely evict CSME code and data from CSME SRAM. The isolated memory is also referred to as Unified-Memory Access or UMA and can be found in dedicated IMR for platform with IMR-based

CPU.
On a non-IMR-based CPU, such as CML, the BringUp process is designed to ask the BIOS during boot to allocate between 8MB and 32MB of UMA exclusively for Intel CSME to use. The exact amount of UMA depends on the Intel-CSME SKU.
On an IMR-based CPU, such as TGL, the aim is for BIOS to allocate stolen memory based on IMR total size reported by CSME during the early handshake with BIOS. CSME helps configure IMR and is designed to remove the CSME's own, write permissions to IMR settings to help prevent CSME from further changing IMR, base addresses and limits.

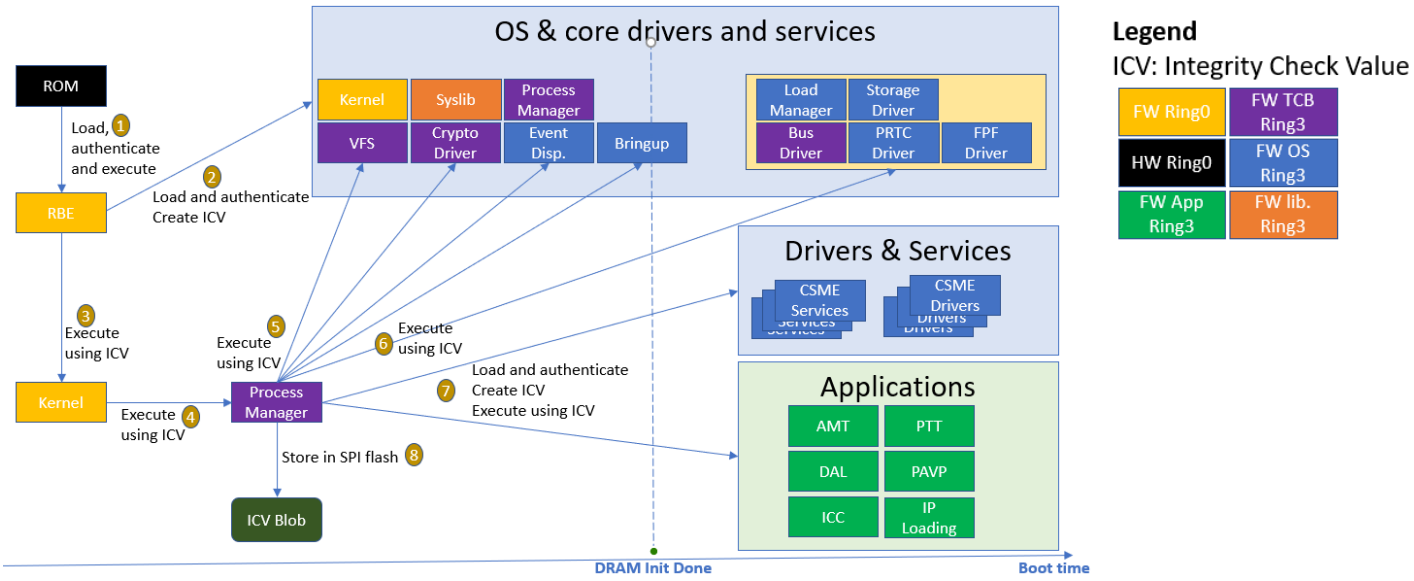

Figure 5- First Boot using new Intel CSME Firmware

Once all CSME modules have been loaded, Process Manager enables storage of all CSME modules' ICVs and IPK in ICV Blob Partition (IVBP) as encrypted and integrity- and replay-protected in SPI flash. At next boot of CSME OS, the applications can be loaded and authenticated using those ICVs as shown in Figure 6 below.

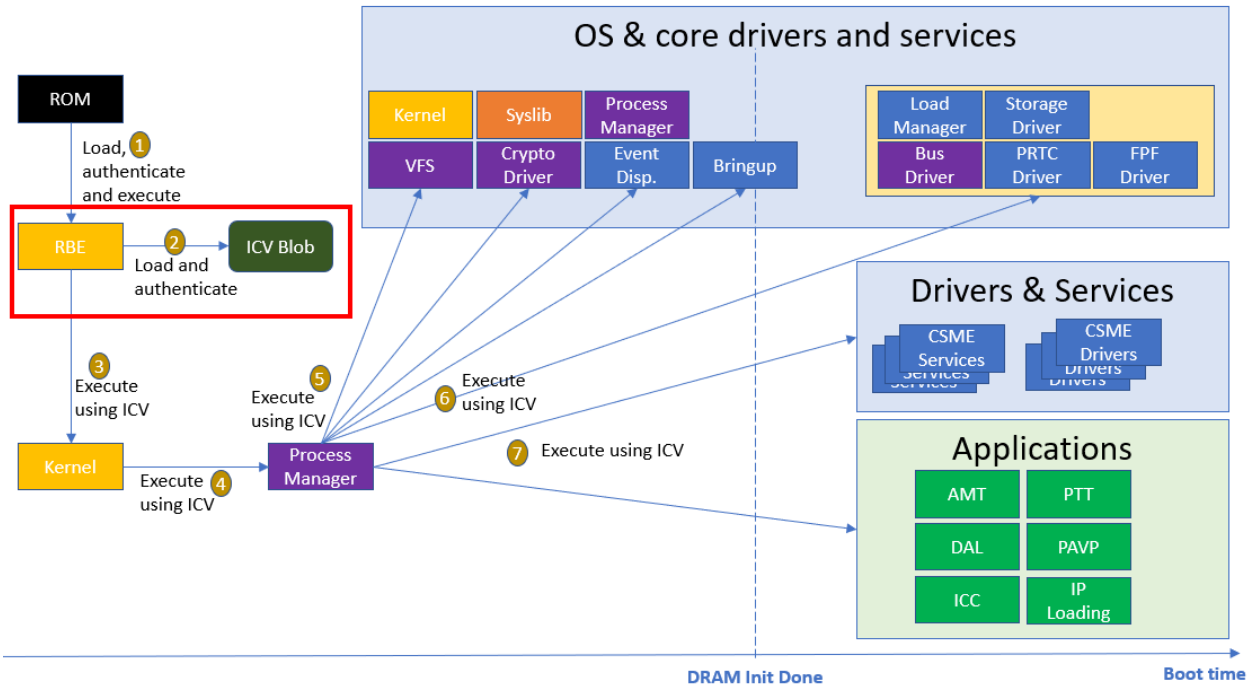

Figure 6- Subsequent boots of Intel CSME Firmware

Note that, before DRAM is initialized by BIOS, CSME-FW-code-pages replacement is designed to be done by kernel from SPI flash using ICVs. Once DRAM is initialized, CSME kernel can make use of DRAM to evict both code and data pages. However, since BIOS is outside TCB for CSME paging to DRAM, as conceived, this memory is considered untrusted from CSME standpoint. Therefore, CSME code and data pages evicted to UMA are designed to be encrypted, anti-replay-protected, integrity- and confidentiality-protected cryptographically with keys known to CSME only, *i.e.*, IPK for calculating ICVs and UMA key (a randomly generated key per boot) for encrypting evicted pages. When pages are copied back from DRAM to CSME SRAM, by design, pages are decrypted, and integrity checked. Some pages are locked in SRAM and not paged out to UMA. These can include pages that contain critical, kernel code that executes very frequently, critical data, including the AES and HMAC keys for UMA protection, and the ICVs. As intended, page faults are handled by the firmware kernel, and paging operations between SRAM and UMA are optimized by Intel CSME's DMA engine and Cryptography-HW accelerator.

4. CSME OS Main Security Principles

CSME OS (Operating System) is based on Minix OS architecture and has two main security principles: a Micro-Kernel-OS architecture and a minimal TCB.

By design, the micro-kernel is the only runtime component executing at CSME ring0, while CSME Firmware applications, drivers and services all run at CSME ring3. The micro-kernel implements the bare minimum required to implement an OS.

The TCB is composed of minimum amount of processes and acts as a root of trust for all security-hardening measures and security services. Its main roles are:

- Helping protect the access to keys and hardware (the Intel-CSME assets)
- Enabling Intel-CSME-firmware-code integrity at boot time and at runtime
- Helping protect Intel-CSME modules from each other and their data stored in SPI flash memory
- Enabling Intel-CSME modules to execute with minimum privileges.

5. Micro-Kernel (uKernel)

The Micro-Kernel enables the infrastructure for Intel CSME, including threading, memory management, and process isolation.

The Micro-Kernel has four, main goals:

- CPU's driver: Micro-Kernel helps a) enforce that code execution is from SRAM only, b) apply process isolation using CPU rings and x86 segments, c) set up page attribute (*i.e.*, Read, Write, and User bit) enforced by MMU, and d) control HW access to CSME ring3 via MMIO (Memory-Mapped, Input Output).
- IOMMU's driver: it helps configure the IOMMU policies to allow/deny DMA access to SRAM.
- Support standard-kernel service, such as Inter-Process Communication (IPC), processes and threads management, and handling of interrupts and exceptions.
- Aid page replacement between SRAM and DRAM/SPI flash to optimize SRAM utilization. By design, the evicted pages to DRAM are encrypted and integrity-protected.

6. Intel-CSME, TCB, Operating System

Intel-CSME TCB Operating System is composed of various components whose main security roles as designed are described below:

- Process Manager enables i) loading and authenticating of CSME drivers, services, and applications, ii) process creation and termination, and iii), using the Kernel Service, creation of x86 process running at ring3 CPU privileges for driver, services, and applications.
- Crypto Driver implements FW-key management as well as Cryptographic and DMA service.
- VFS (Virtual-File System) enables secure storage with data-migration support and helps enforce a permission check on data files and special files exposed by drivers and services inside CSME OS. VFS can offer a mechanism for applications and common services to store non-volatile data in the data region of the same NVRAM in which the Intel CSME image is stored. The mechanism is designed to provide encryption, integrity, and anti-replay protection. The aim is for the storage key to be managed by a crypto driver and protected using the SKS HW at runtime. This can eliminate potential, key-leak vulnerabilities. AES-GCM with 256 bits key is used to help protect files in NVM. Replay protection can be supported using either RPMC or PRTC. The Intel-CSME processes enable deciding if encryption or/and anti-replay protection are required for the blobs they own. Blob protection is designed to include integrity. To help avoid flash wear-out, a general rule for Intel-CSME-firmware design is to try to minimize the number of writes to flash. The storage manager supports implementing an algorithm to detect access patterns that would accelerate flash wear-out; the algorithm is designed to then apply the relevant remediation.

- Bus Driver is intended to allow CSME drivers to configure their own device-configuration space and help enforce access control as to which device can be configured by which driver.

The following table shows the designated, access control and permissions for CSME ring3 TCB components:

Ring3 TCB Component	Create CSME Process	Control access to CSME keys	Control access to CSME OS services	Control access to Hardware
Process Manager	**Yes**	No	No	No
Crypto Driver	No	**Yes**	No	No
VFS	No	No	**Yes**	No
Bus Driver	No	No	No	**Minimal**

Figure 7 - CSME ring3 Component Access Control and Permissions

7. BringUp (BUP)

The CSME BringUp (BUP) is a FW module that implements CSME functionality required for supporting platform boot and configuration. BUP's design includes Boot Guard, a sub-set of Intel-PTT commands, and a minimal implementation of some CSME-FW drivers and services. The goal is for it to launch quickly, while the platform waits for all the drivers, services, and applications to start executing.

Starting with Intel CSME 12, the BringUp is intended to run with reduced privileges (refer to section on BUP Deprivilege and Enhanced RBE Security Architecture). Due to the deprivilege, the aim is that BringUp cannot create any Intel CSME processes or have access to FW-root keys, attestation key, OCS, or DFX hardware.

8. CSME-Operating-System Drivers and Services

The Intel-CSME Operating-System drivers and services are designed to be executed at CSME ring3, with the two main components as Maestro and Load Manager. These components enable orchestrating the loading of all CSME ring3 processes, which are classified into three categories: Drivers, Services, and Applications. As intended, access to drivers and services is overseen jointly by the VFS and the Micro-Kernel that help enforce pre-defined, static permission for each CSME-FW process.

Intel-CSME drivers are processes designed to abstract a hardware device, such as the Cryptography-HW accelerator and DMA engine, and expose APIs (Application-Programming Interfaces) for other processes to consume. The drivers are also designed to only have access to the specific hardware that they need to manage, with such access via MMIO. Some drivers do not communicate directly with the external world, while others, such as LAN, WLAN, and NVRAM driver, help transport data from the external world to associated applications and services.

An example of a driver is HECI, which can provide a transport mechanism by which the host and Intel CSME can pass messages to one another through two, circular buffers: one for messages sent from Intel CSME to the host and the other for messages in the opposite direction. The host and the Intel CSME are both designed to implement a HECI driver meant to act as a message dispatcher. On the host OS, by

design, the HECI driver is installed as a kernel-mode driver, also known as Intel® MEI (Intel® Management Interface). On the Intel CSME, the aim is for the HECI driver to be a CSME ring3 driver in CSME OS consumed by other processes. The HECI interface does not provide any native, security measures. By design, clients of HECI are responsible for implementing the applicable protocols to protect the transmitted messages. Intel-CSME-firmware processes help expose HECI commands of various functions for the host to invoke. On both the host and the Intel CSME, various clients can connect to the driver to send and receive messages to and from their counterparts, respectively. For example, Intel CSME's PAVP (Protected-Audio-Video Path)-application client helps implement HECI commands for the media-player, software client on the host to provision device key-box, process content licenses, retrieve playback statistics, and so on.

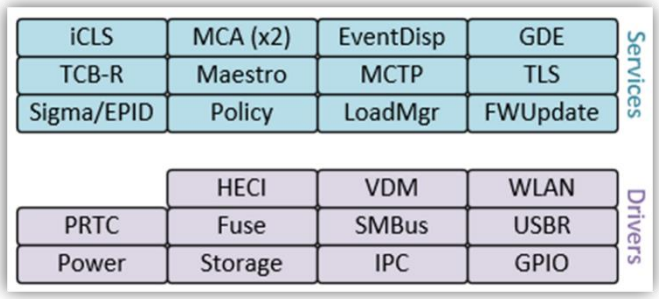

Figure 8 - Example: Services and Drivers

9. Intel-CSME Applications

Following BringUp and the loading of drivers and services, the Intel-CSME applications are designed to run in CSME ring3. The CSME Micro-Kernel helps enforce isolation of each Intel-CSME-application execution and prevent one CSME application from accessing the memory space of another application. Application secrets are also better protected in flash memory by VFS, which helps provide such protection using a storage key designed to be protected in SKS and managed by crypto.

Not all features are supported by all SKUs, and some features require specific configuration and enablement. For example, Intel AMT is not supported on consumer SKUs, Intel® Atom platforms or servers. Intel® Boot Guard requires OEMs to provision and enable the feature for usage. Some features can also be disabled by OEMs.

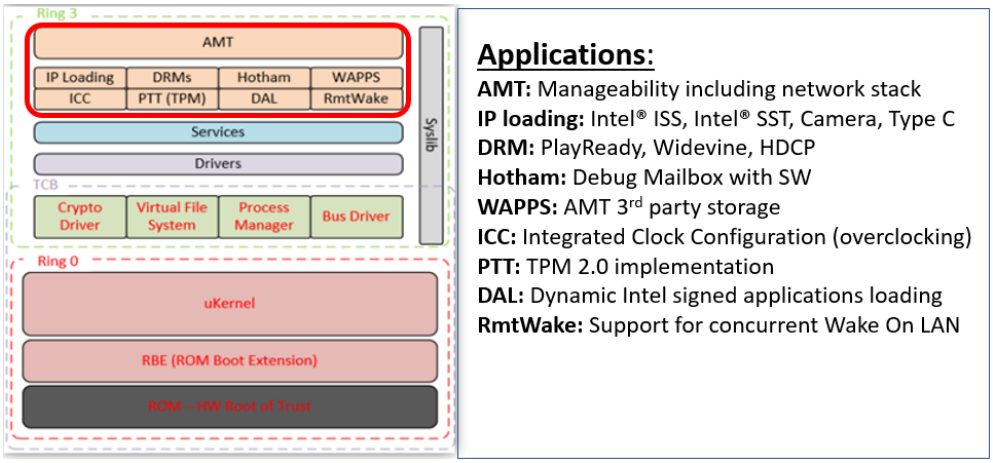

Figure 9 - Application examples (outlined in red)

CSME-Recoverability Aspects

The following section describes various designed recoverability aspects of CSME: from simple firmware update to re-key-generation process in specific cases of security breaches.

Firmware Update and Anti-rollback (ARB)

Intel CSME's firmware can be updated in the field. Intel might release new firmware for several reasons:

- Functional- or security-bug mitigations
- Performance improvements
- New, infrastructural features
- New, security hardening
- New applications

The new firmware is delivered to end users by individual OEMs – not by Intel. Users are encouraged to download and install the latest version available for the device. An Intel-CSME-firmware update is designed to be installed using a software agent via the HECI interface.
The Firmware-Update process supports a fault-tolerant mechanism to help protect the user from system bricking in the case of an interrupted update.

If the new firmware implements mitigations for security bugs, it might not be possible to downgrade the firmware or roll back to old and vulnerable firmware due to firmware or hardware mechanisms. The firmware-enforced, anti-rollback mechanism helps prevent downgrading via the firmware-update interface. This can protect users from accidentally performing downgrades or prevent an attacker with OS-admin privileges from performing a downgrade to an old FW with known vulnerabilities.

The hardware-enforced, anti-rollback mechanism is designed to provide an OEM-managed capability. When enabled, it properly mitigates physical rollback of CSME Firmware in SPI flash on the platform.

It helps the system remain protected against known vulnerabilities that have been mitigated in the most recent, CSME-Firmware release. To achieve this, the new firmware is designed to burn specific, FPFs to record the FW ARB SVN into hardware. At a very early stage of the boot, the Intel CSME's boot ROM or RBE (depending of the platform generation) helps examine these FPFs. As a remediation mechanism, if the SVN of the firmware from flash is lower than the SVN from the FPFs, then CSME can detect it is under rollback attack and prevent the system from booting a vulnerable version. The Intel-CSME, hardware-enforced, anti-rollback mechanism was first implemented in the Cannon Lake PCH (CSME 12.0).

Intel® Enhanced Privacy ID (Intel® EPID) and On-Die, Certificate Authority (ODCA)

The Intel EPID scheme is an anonymous authentication algorithm invented by Intel and standardized by the International Organization for Standardization (ISO). Intel EPID 1.0 was first introduced in Intel CSME 5 in 2008. Intel EPID 2.0, with performance and security enhancements, intercepted Intel CSME 12 in 2017 and is supported up to Intel CSME 14.0.

The Intel EPID helps a member prove its membership to a verifier without disclosing its identity. The ecosystem of Intel EPID is designed with three actors:

1. **Authority**: generator and issuer of all member private keys and group public keys.
2. **Member:** belongs to a group of many members. The member is provisioned with its unique member private key and a group certificate that is shared by all members of the same group. A member uses its private key to help generate an Intel-EPID signature to show its membership to the verifier.
3. **Verifier:** designed to authenticate a member by checking the member's signature against the group certificate.

The fundamental advantage of Intel EPID over traditional authentication methods built on asymmetric cryptography is Intel EPID's enabling of anonymity. The Authority can decide the number of members in a group. Every member owns its unique, private key, but, by design, all members of the same group share the same, group, public key and certificate. During authentication, the intention is that a member signs the challenge from the verifier, and the verifier verifies the member's signature using the group certificate. The verifier helps confirm that the signature is generated by one of the group members, but it is designed not to be able to identify exactly which of the members generated the signature.

A member or a group of members may be revoked by the Authority in three ways:

- Group revocation: Revoke the entire group.
- Private-key revocation: The Authority maintains a list of values of compromised, private keys. A verifier helps detect whether a received signature was generated by a private key in the private-key-revocation list and reject the member if it was.
- Signature revocation: Sometimes the private key of a compromised member is not disclosed, but a signature from the member is known. The Authority helps maintain a list of signatures from compromised members. A verifier is designed to detect whether a received signature was generated by the same member who generated a signature on the signature-revocation list; if it was, the goal is for the Authority to reject the member.

The Intel EPID is especially useful for applications that require privacy. On the Intel CSME, it is consumed by applications including PAVP, DRM, Intel DAL, and Intel PTT.

CSME 15.0 and forward replace Intel EPID with an alternate architecture: the ODCA (On-Die, Certificate Authority). Using its ODCA-private key, Intel CSME ROM can act as embedded, Certificate Authority and issue a certificate to the CSME FW and its application. Examples of certificate usages are Boot Guard, DRM, PTT, and DAL. The aim is for each application/usage to use its certificate to prove to the verifier it is a genuine CSME application running inside the CSME subsystem. For example, Content Provider can be used for DRM, ACM for Boot Guard, and AMT authentication by the management console. ODCA is also used to help generate the Endorsement Key for PTT. A CSME-FW update may revoke the ODCA chain of certificates for its application.

TCB Recovery and Intel Capability-Licensing Service

As conceived, TCB Recovery consists of:

1. Blob migration – enabled migration of all CSME data kept in NVM from the previous, storage key to a new, storage key generated by ROM from a newer, FW SVN.
2. Help acquisition of a new, Intel EPID and a new, EK Certificate on the CML platform where ODCA is not supported. On the TGL platform, ROM can issue a new, ODCA-FW, private key and certificate based on incremented, FW SVN without a need to connect to the Intel server over the internet.

Once TCB Recovery has successfully completed, by design, the platform will obtain a new, Intel® EPID key, a new EK with its corresponding certificate, and data blobs protected using a new, storage key. Starting with TGL and moving onward, the intention is that a new, ODCA-FW, private key, and certificate will be generated.

In order to support TCB recovery on CML platform, impacted systems can be updated to a new, Intel-CSME firmware that contains the relevant mitigation and has an incremented SVN.

The impacted system also needs an iCLS (Intel Capability-Licensing Service) installed. The iCLS is a software service delivered with the Intel®-ME-software package and that is designed to run on client machines. The iCLS supports establishing a trusted connection, that uses standard, Transport-Layer Security (TLS) over port 443, to communicate with Intel-backend-licensing servers to obtain various licenses and keys. If the end-user system is inside an intranet (*e.g.*, inside an IT organization), he or she might need to provide a proxy to allow iCLS to connect properly to the Intel-iCLS-server backend over the internet.
The iCLS-local service requires an internet connection to connect to the iCLS server and executes the Intel EPID Re-Key in case of TCB recovery up to Intel CSME 14.0.

The following is a description of the re-key process designed for use on versions up to and including Intel CSME 14.0:

a. If the system is not using the latest version of the iCLS software service, the iCLS software should be updated.
b. Manufacturers should update the Intel-CSME firmware with firmware that has a higher SVN. Following the firmware update, on the next boot, the Intel-CSME firmware is designed to

migrate the Intel-CSME data from the previous, CSME-storage key to a new, storage key derived from the Intel-Root key created by ROM by using the incremented SVN in the FW-manifest header.

c. The iCLS-software service uses the Intel-designed, sigma protocol to enable a secure connection over the internet to Intel-backend servers to complete TCB recovery and retrieves both the new, Intel-EPID key and Intel certificate for the new, Intel-PTT EK (the TPM EK).
d. At some point, Intel revokes the Intel-EPID keys and the Intel-PTT EK by publishing a Certificate-Revocation List (CRL). Once revocation is performed, for example, the intention is that content providers can stop streaming content to systems that have not been updated.

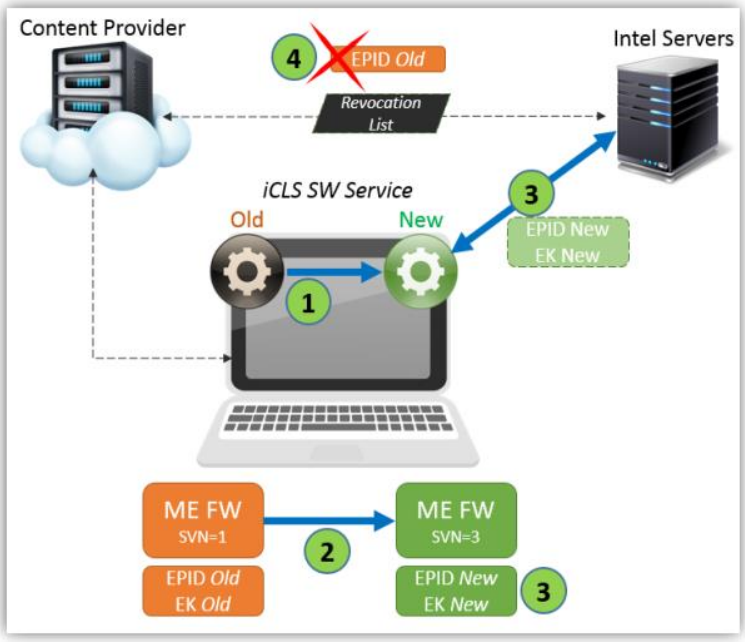

Figure 10 -TCB Recovery with iCLS connection

If a critical vulnerability that causes one or more components of the TCB to be compromised is found in the Intel CSME, after such vulnerability is brought to Intel's attention, Intel will evaluate the situation and might trigger TCB recovery, which can revoke the compromised credentials and replace them with new ones.

Starting with CSME 15.0, TCB recovery is designed to be done with a FW update that triggers the ODCA-re-key process by the ROM and FW only. The CSME-crypto driver helps generate and renew application certificates using ODCA without the need for server connection.

Security Assets

Intel CSME utilizes a list of CSPs (Critical-Security Parameters) to help protect itself from attacks and to support security services and applications. These CSPs are Intel CSME's security assets. Examples of native CSPs (provisioned during the manufacturing by Intel or the OEM or derived internally in Intel CSME) include:

- Keys such as Chipset key, Intel EPID key, Intel-PTT-endorsement credentials as described in the section about The Chipset Key and the Fuse-Encryption Hardware Key
- OEM, public-key hash: to enable secure boot, the OEM burns the hash of its public key onto Intel CSME's FPF during manufacturing. The corresponding, OEM-private key can be used to sign OEM-relevant components (*e.g.*, the BIOS-initial-boot block for Boot Guard, ISH code etc.).
- DRM-device keys: some DRM-device keys are provisioned by OEMs during manufacturing, while other DRMs have their device keys provisioned by license servers in the field.
- The crypto driver can derive additional keys from the FW-root keys derived by ROM and pass them to the FW applications and services. The Intel-EPID key, Intel-PTT EKs, and derivatives of the chipset key are keys used by TCB.
- In CSME 15.0 with the ODCA architecture, ROM can act as an embedded, Certificate Authority and issue a certificate to the CSME FW. The CSME FW using its certificate and private key generated by ROM can issue a certificate for different usages, such as Boot Guard, AMT, PTT, DAL, and PAVP.

The CSME-FW integrity is designed to be reviewed at boot by the Intel-CSME signing key and at runtime by keys indirectly derived from the chipset key. As intended, such reviews also address the integrity of CSME OS and applications. The CSME Micro-Kernel and TCB-ring3 processes help implement a trusted execution environment to better ensure the Intel EPID key, ODCA Keys, PTT keys, and other CSME-application or service keys are protected even within the CSME subsystem.

There is another category of CSPs that, by design, are installed or generated by Intel-CSME applications after Intel manufacturing. These include:

- Intel-AMT-administrator passwords
- Verified, Intel-DAL applets' hash digests (for faster loading by skipping asymmetric, signature verification)
- Secrets owned by Intel-DAL applets
- Intel PTT's AIK (Attestation-Identity Key) and other keys
- DRM keys

At runtime, the goal is that CSPs are either saved in SRAM or paged out to UMA in protected form. When Intel CSME is powered off, the application CSPs can be stored in NVRAM and better protected by the CSME-storage key.

End-of-Manufacturing (EoM) Mandatory Step

The EoM step is mandatory for all OEMs to help make their assets more secure. As designed, it consists of a command that OEMs need to run before shipping platforms to end users. The command has the following goals:

- Write and lock manufacturers' settings into FPFs along with RPMC (Replay-Protection, Monotonic Counter) provisioning if supported.
- Close some CSME APIs used for debugging and manufacturing.
- Lock the SPI flash descriptor (the SPI controller implements access control on BIOS, Intel CSME, and other SPI regions).

For Intel CSME 15.0, a more flexible EoM was developed in response to OEM requests to minimize OEMs missing EoM steps before shipment. To help OEMs customize their manufacturing procedures, the EoM is designed to split the three steps above into three stages.

Security-Process Improvements, Design Enhancement and Hardening

Intel is pursuing a course of continuous improvement of Intel CSME's security capabilities.
Intel aims to follow the industry's best security practices in all cycles of Intel-CSME development. For example:

- Reducing attack surfaces by minimizing external and internal interfaces.
- Enhancing defense-in-depth instead of relying on a single measure to protect critical assets.
- Enhancing the Intel-CSME-security-mitigation mechanism ability to address the rapidly changing threat model and make attacks more difficult and costlier for an attacker.
- Significant efforts like external audits, independent third-party code reviews, high-assurance-development practices, and expansion of red-team activities.
- Designed integration of automated-security checks in the CI/CD (Continuous Integration / Continuous Development) pipeline used to better catch security issues automatically during build, such as static-code analyzer
- Security-Advanced research intended to align with industry-best-in-class, security practices with the goal of development and deployment of Advanced Fuzzing and advanced-code-analyzer capabilities (refer to Intel CSME Security-Validation Technologies section).
- Establishing an "Intel platform update (IPU)" release to help manage security mitigations reaching the end points.
- Using the latest, known, cryptography guidelines published by authorities and standard bodies. For example, starting in 2020, the Intel CSME 15 is aims to comply with the U.S. government's guidelines for the transition period to quantum cryptography. Such compliance would include increasing sizes to 256 bits for AES keys, 3072 bits for RSA keys, 384 bits for ECC (Elliptic-Curve Cryptography), and 384 bits for SHA-2 digests.
- Policy of applying SDL (Security-Development-Lifecycle) methodology to all stages of product development. Intel's SDL is a set of activities and milestones that drive high-quality, security outcomes in product and service development at Intel. The SDL process makes security and privacy an integral part of Intel-CSME-production definition, design, development, and validation.

The FW re-design includes revising security-hardening features to better match or exceed their levels in previous generations. For instance:

- HW-based SKS designed to hold the kernel's applications' AES and HMAC keys in hardware. When the keys are needed, the intention is for AES and HMAC cryptographic accelerators to fetch the keys from the SKS directly and thereby not expose the key values in firmware memory.
- The module of cryptographic-accelerator hardware (OCS) and the CSME-firmware, crypto driver passed stringent, third-party review and testing and achieved the FIPS 140-2 level 1 certificate from the NIST (National Institute of Standards and Technology).

It is quite difficult, if not impossible, to avoid bugs in code. Updates and patches are a regular part of modern, technology products. Protecting customers and their data continues to be a critical priority for Intel. Intel actively works on connecting deep, offensive-security research with deep, product knowledge to find and address potential vulnerabilities on an ongoing basis and to work with the research community to do the same through a bug-bounty program and by engaging academia. Given the nature of its products, Intel commonly works with its customers and other third parties, including hardware, software, services vendors, as well as end users, to develop and deploy mitigations. Effective mitigation requires all of these parties to work together in coordinated cooperation.

Anti-exploitation techniques

CSME adopts a multi-layer, anti-exploitation mechanism to make exploitability of memory-safety issues (*i.e.*, memory corruption) more difficult. The section below references some of those techniques:

Stack-Protector XORed with Return address

Intel CSME introduced an enhancement to the standard, stack-canary, stack-overflow protection: the XOR (Exclusive OR) canary.

The canaries are designed to provide a unique stack value generated randomly for every thread (some CSME processes have more than one thread). The canaries also are intended to be created by the Micro-Kernel when it creates threads of the relevant, CSME-FW module by using the Intel PCH DRNG. This enhanced implementation, included in Intel-CSME versions including and up to 14.0, can add "xoring" the random number with the return address from the present stack.

The introduction of CET-HW-based protection in Intel CSME 15.0 aims to eliminate the need for the XOR-canary technique on CSME FW. By design, the standard implementation of the GCC (GNU-Compiler-Collection)-stack canary is applied instead on functions that the GCC compiler detects as needing a canary. (Refer to section on Control Enforcement Technology (CET).)

SW-Forward, Edge-Control-Flow Integrity (F-CFI)

F-CFI is a technique that reduces the risk associated with attacks aiming to redirect the flow of a program's execution. CSME is designed to implement F-CFI by adding an end-branch tag before each indirect internal CSME-callable function, which tag validates that the call is set to a valid location with a function pointer. This helps prevent such attacks from arbitrarily controlling program behavior.

XORed-Function pointers' Control-Flow Integrity (XF-CFI)

XF-CFI is an enhancement of F-CFI technique where all function pointers are designed "xored" with a random value generated by Micro-Kernel using Intel PCH DRNG, which random value differs from the stack-canary value, is re-generated each boot, and is unique per process. As conceived, the value is exposed to the process during loading and initialization as well as during the process runtime.

By “xoring” all function pointers with a secret, random value, the intent is that an attacker cannot craft a valid function pointer in the CSME process, when he/she finds a vulnerability that allows an arbitrary write in a process’s memory, without finding an additional information-leakage vulnerability that would disclose the canary.

Data-Address-Space-Layout Randomization (DASLR)

The goal of DASLR is to randomize the process’s memory-writable sections, such as base address of stack and heap per CSME boot. The goal is to make an arbitrary-write vulnerability to CSME stack and heap less predictable and help ensure that the attacker cannot take advantage of the static data CSME holds in those writable-memory areas.

Write Protect

CSME kernel supports setting writable-memory area as read-only inside the heap. This capability is called Write Protect, and once set, helps a CSME application prevent its critical data kept in the heap from being modified. It can be used, for example, by DAL when it keeps DAL applet code in heap and can help ensure DAL applet code cannot be modified when stored in a writable-memory area like the heap.

Intel® Control-Flow Enforcement Technology (CET)

Starting with CSME 15.0, Intel® CET is supported not only by the TGL core CPU but also by the CSME-dedicated CPU. CET is designed to provide HW-based protection against control-flow-hijacking techniques used by an attacker to exploit vulnerabilities by providing two key capabilities:

- Indirect-branch tracking: indirect-branch protection to help defend against jump/call-oriented-programming (JOP/COP) attack methods.
- Shadow stack: return-address protection to help defend against return-oriented-programming (ROP) attack methods.

CET is enabled in both CSME ring 0 and ring 3 firmware.

Additional Security improvements

CSME contains additional security improvements as described in the sections below.

BUP Deprivilege and Enhanced-RBE-Security Architecture

Before the BUP-Deprivilege enhancement, an exploit leading to arbitrary-code execution in Intel-CSME BringUp or any other CSME ring 3-FW-component part of FW TCB could allow the creation of new CSME ring 3-FW components with any privilege and, as a result, access to secrets kept in the file system (*e.g.*, Intel EPID and EK).

As a security enhancement, Intel removed privileges from Intel-CSME BringUp and some other, CSME ring 3-FW components and adopted a static-process-creation design where all CSME-FW processes’ permissions are embedded in kernel and VFS at FW build-time. In addition, all FW module code hashes

are designed either embedded in RBE or Process Manager and protected by a single FW manifest (the RBE manifest), which is authenticated by ROM. This enhancement helps prevent all FW modules outside of CSME TCB from creating arbitrary processes with arbitrary permission or access secrets in memory or flash and get to critical HW resources like the Crypto-HW accelerator and DFX HW. Even the CSME ring3 TCB is designed to have a specific role and permission within the CSME OS, as explained in a previous section. Removing Intel-CSME BringUp from the TCB also reduces the risk that TCB recovery will be required in the future.

Starting with CSME 15.0, RBE security architecture was also improved. For earlier CSME version, where a module ran at CSME ring0 with highest privileges in earlier CSME versions, RBE has been broken down into multiple, security domains with minimum privileges where the aim is for only a small part of the component to still run at CSME ring 0. This privileged part is designed to act like the CSME kernel: it helps ensure memory protection between the RBE-security domain that runs at CSME ring 3 and control which HW can be accessed.

Intel-CSME-Firmware Measurements – Enhanced, Measured Boot

This feature is intended to determine the integrity of the Intel-CSME firmware on the platform. In addition, Enhanced, Measured Boot helps enforce the OEM's system to be compliant with the TCG (Trusted-Computing-Group) PC-Client Specification. Basically, this feature enables measurement of the Intel-CSME FW and all IP FW stored in flash and loaded by CSME, (*e.g.*, PMC, Type C etc.), calculating their hash value, and exposing the hash values to BIOS, which can make final measurement into the platform TPM-PCRs (Trusted-Platform Module – Platform-Configuration Registers). From the BIOS standpoint, when BIOS detects that Intel-CSME measurement is complete (via Extended-Range (ER), HECI Status Registers), BIOS is used to help read the value in the ER and extend it to TPM PCR [0]. By design, the BIOS then invokes the Intel-CSME-HECI interface to view the Intel-CSME-Measured-Boot-Event Log and update the TPM Log.
CSME ROM is designed to measure first the CSME-RBE module and extend its manifest hash and security version in the HECI ER, so that, from this point, the HECI ER cannot be modified except to extend it. Starting with TGL, the HW helps enforce this feature.

AMT-related enhancements

Starting with CSME 14.0, Intel-AMT, Host-Based Provisioning adds Mutual TLS to enable in band provisioning and deprecating unsecure, CCM (Client-Control Mode) to ACM (Admin-Control Mode) upgrade.

Intel-CSME, Security-Validation Technologies

Intel is also enhancing its own validation technologies. A few steps were added to our regular validation using the latest industry techniques on silicon:

- Address Sanitization (*i.e.*, ASAN)
- Fuzzing silicon-based capability
- Advanced, Fuzzing capable
- Fuzzing with Coverage guided (*i.e.*, AFL)
- Run on real hardware and firmware environment (*i.e.*, not simulation)
- Dedicated, security-code review
- Automated, static-code analysis
- Manual, penetration testing

Conclusion

By its architecture based on hardware root of trust and lowest-privilege security principles, Intel CSME represents a major building block for a safe computing environment. Security is continually improved and enhanced through the CSME generations via new defense in depth and anti-exploitations techniques as well as options to help recover on the field in case of vulnerabilities. Intel is committed to continue working on hardening Intel CSME and delivering state-of-the-art solutions to enable and enhance users' secure-computing experience.

References

The white paper content is based on:

- *"BlackHat 2019 - Behind the Scenes of Intel Security and Manageability Engine", https://www.blackhat.com/us-19/briefings/schedule/index.html#behind-the-scenes-of-intel-security-and-manageability-engine-15789*