Draft NISTIR 8320

Hardware-Enabled Security:

Enabling a Layered Approach to Platform Security for Cloud and Edge Computing Use Cases

Michael Bartock
Murugiah Souppaya
Ryan Savino
Tim Knoll
Uttam Shetty
Mourad Cherfaoui
Raghu Yeluri
Akash Malhotra
Karen Scarfone

This publication is available free of charge from:
https://doi.org/10.6028/NIST.IR.8320-draft

National Institute of Standards and Technology
U.S. Department of Commerce

Draft NISTIR 8320

Hardware-Enabled Security:

Enabling a Layered Approach to Platform Security for Cloud and Edge Computing Use Cases

Michael Bartock
Murugiah Souppaya
Computer Security Division
Information Technology Laboratory

Ryan Savino
Tim Knoll
Uttam Shetty
Mourad Cherfaoui
Raghu Yeluri
Intel Data Platforms Group
Santa Clara, CA

Akash Malhotra
AMD Product Security and Strategy Group
Austin, TX

Karen Scarfone
Scarfone Cybersecurity
Clifton, VA

May 2021

U.S. Department of Commerce
Gina Raimondo, Secretary

National Institute of Standards and Technology
James K. Olthoff, Performing the Non-Exclusive Functions and Duties of the Under Secretary of Commerce for Standards and Technology & Director, National Institute of Standards and Technology

National Institute of Standards and Technology Interagency or Internal Report 8320
58 pages (May 2021)

This publication is available free of charge from:
https://doi.org/10.6028/NIST.IR.8320-draft

Public comment period: *May 27, 2021* through *June 30, 2021*

National Institute of Standards and Technology
Attn: Applied Cybersecurity Division, Information Technology Laboratory
100 Bureau Drive (Mail Stop 2000) Gaithersburg, MD 20899-2000
Email: hwsec@nist.gov

Reports on Computer Systems Technology

The Information Technology Laboratory (ITL) at the National Institute of Standards and Technology (NIST) promotes the U.S. economy and public welfare by providing technical leadership for the Nation's measurement and standards infrastructure. ITL develops tests, test methods, reference data, proof of concept implementations, and technical analyses to advance the development and productive use of information technology. ITL's responsibilities include the development of management, administrative, technical, and physical standards and guidelines for the cost-effective security and privacy of other than national security-related information in federal information systems.

Abstract

In today's cloud data centers and edge computing, attack surfaces have significantly increased, hacking has become industrialized, and most security control implementations are not coherent or consistent. The foundation of any data center or edge computing security strategy should be securing the platform on which data and workloads will be executed and accessed. The physical platform represents the first layer for any layered security approach and provides the initial protections to help ensure that higher-layer security controls can be trusted. This report explains hardware-enabled security techniques and technologies that can improve platform security and data protection for cloud data centers and edge computing.

Keywords

confidential computing; container; hardware-enabled security; hardware security module (HSM); secure enclave; trusted execution environment (TEE); trusted platform module (TPM); virtualization.

Disclaimer

Any mention of commercial products or reference to commercial organizations is for information only; it does not imply recommendation or endorsement by NIST, nor does it imply that the products mentioned are necessarily the best available for the purpose.

Acknowledgments

The authors thank everyone who contributed their time and expertise to the development of this report, including:

- From Intel Corporation: Ravi Sahita, Alex Eydelberg, Sugumar Govindarajan, Kapil Sood, Jeanne Guillory, David Song, Scott Raynor, Scott Huang, Matthew Areno, Charlie Stark, Subomi Laditan, Kamal Natesan, Haidong Xia, Jerry Wheeler, Dhinesh Manoharan, and John Pennington
- From AMD: David Kaplan and Kathir Nadarajah

Audience

The primary audiences for this report are security professionals, such as security engineers and architects; system administrators and other information technology (IT) professionals for cloud service providers; and hardware, firmware, and software developers who may be able to leverage hardware-enabled security techniques and technologies to improve platform security for cloud data centers and edge computing.

Trademark Information

Call for Patent Claims

This public review includes a call for information on essential patent claims (claims whose use would be required for compliance with the guidance or requirements in this Information Technology Laboratory (ITL) draft publication). Such guidance and/or requirements may be directly stated in this ITL Publication or by reference to another publication. This call also includes disclosure, where known, of the existence of pending U.S. or foreign patent applications relating to this ITL draft publication and of any relevant unexpired U.S. or foreign patents.

ITL may require from the patent holder, or a party authorized to make assurances on its behalf, in written or electronic form, either:

a) assurance in the form of a general disclaimer to the effect that such party does not hold and does not currently intend holding any essential patent claim(s); or

b) assurance that a license to such essential patent claim(s) will be made available to applicants desiring to utilize the license for the purpose of complying with the guidance or requirements in this ITL draft publication either:

   i. under reasonable terms and conditions that are demonstrably free of any unfair discrimination; or
   ii. without compensation and under reasonable terms and conditions that are demonstrably free of any unfair discrimination.

Such assurance shall indicate that the patent holder (or third party authorized to make assurances on its behalf) will include in any documents transferring ownership of patents subject to the assurance, provisions sufficient to ensure that the commitments in the assurance are binding on the transferee, and that the transferee will similarly include appropriate provisions in the event of future transfers with the goal of binding each successor-in-interest.

The assurance shall also indicate that it is intended to be binding on successors-in-interest regardless of whether such provisions are included in the relevant transfer documents.

Such statements should be addressed to: hwsec@nist.gov

Table of Contents

List of Appendices

List of Figures

1 Introduction

In today's cloud data centers and edge computing, there are three main forces that impact security: (1) the introduction of billions of connected devices and increased adoption of the cloud have significantly increased attack surfaces; (2) hacking has become industrialized with sophisticated and evolving techniques to compromise data; and (3) solutions composed of multiple technologies from different vendors result in a lack of coherent and consistent implementations of security controls. Given these forces, the foundation for a data center or edge computing security strategy should have a consolidated approach to comprehensively secure the entire hardware platform on which workloads and data are executed and accessed.

In the scope of this document, the *hardware platform* is a server (e.g., application server, storage server, virtualization server) in a data center or edge compute facility. The server's hardware platform, also called the *server platform*, represents the first part of the layered security approach. *Hardware-enabled security*—security with its basis in the hardware platform—can provide a stronger foundation than one offered by software or firmware, which can be modified with relative ease. Hardware root of trust presents a smaller attack surface due to the small codebase. Existing security implementations can be enhanced by providing a base-layer, immutable hardware module that chains software and firmware verifications from the hardware all the way to the application space or specified security control. In that manner, existing security mechanisms can be trusted even more to accomplish their security goals without compromise, even when there is a lack of physical security or attacks originate from the software layer.

This report explains hardware-based security techniques and technologies that can improve server platform security and data protection for cloud data centers and edge computing. The rest of this report covers the following topics:

- Section 2 provides an overview of hardware platform security.
- Section 3 discusses the measurement and verification of platform integrity.
- Section 4 explores software runtime attacks and protection mechanisms.
- Section 5 considers protecting data in use, also known as confidential computing.
- Section 6 examines remote attestation services, which can collate platform integrity measurements to aid in integrity verification.
- Section 7 describes a number of cloud use case scenarios that take advantage of hardware-enabled security.
- Section 8 states the next steps for this report and how others can contribute.
- The References section lists the cited references for this report.
- Appendix A describes vendor-agnostic technology examples.
- Appendix B describes Intel technology examples.
- Appendix C describes technology examples from AMD.
- Appendix D lists the acronym and abbreviations used in the report.

- Appendix E provides a glossary of selected terms used in the report.

As technology and security capabilities evolve, NIST is continuously seeking feedback from the community on the content of the report and soliciting additional technology example contributions from other companies.

Although this document does not address other platforms like laptops, desktops, mobile devices, or Internet of Things (IoT) devices, the practices in this report can be adapted to support those platforms and their associated use cases.

Please send your feedback and comments to hwsec@nist.gov.

2 Hardware Platform Security Overview

The data center threat landscape has evolved in recent years to encompass more advanced attack surfaces with more persistent attack mechanisms. With increased attention being applied to high-level software security, attackers are pushing lower in the platform stack, forcing security administrators to address a variety of attacks that threaten the platform firmware and hardware. These threats can result in:

- Unauthorized access to and potential extraction of sensitive platform or user data, including direct physical access to dual in-line memory modules (DIMMs)
- Modification of platform firmware, such as that belonging to the Unified Extensible Firmware Interface (UEFI)/Basic Input Output System (BIOS), Board Management Controller (BMC), Manageability Engine (ME), Peripheral Component Interconnect Express (PCIE) device, and various accelerator cards
- Supply chain interception through the physical replacement of firmware or hardware with malicious versions
- Access to data or execution of code outside of regulated geopolitical or other boundaries
- Circumvention of software and/or firmware-based security mechanisms

For example, LoJax, discovered in August 2018, manifests itself in UEFI malware, allowing it to continuously persist in the firmware layer despite operating system (OS) reinstallations, and thus remain invisible to standard kernel-based virus scans [1]. These attacks can be devastating to cloud environments because they often require server-by-server rebuilds or replacements, which can take weeks. Although still rare, these attacks are increasing as attackers become more sophisticated.

Workloads subject to specific regulations or containing sensitive data present additional security challenges for multi-tenant clouds. While virtualization and containers significantly benefit efficiency, adaptability, and scalability, these technologies consolidate workloads onto fewer physical platforms and introduce the dynamic migration of workloads and data across platforms. Consequently, cloud adoption results in a loss of consumer visibility and control over the platforms that host virtualized workloads and data, and introduces the usage of third-party infrastructure administrators. Cloud providers and cloud adopters follow a shared responsibility model, where each party has responsibility for different aspects of the overall implementation. Cloud providers can expose information related to infrastructure security and platform capability in order to provide their tenants with security assurances. Furthermore, cloud providers often have data centers that span multiple geopolitical boundaries, subjecting workload owners to complicated legal and regulatory compliance requirements from multiple countries. Hybrid cloud architectures, in particular, utilize multiple infrastructure providers, each with their own infrastructure configurations and management.

Without physical control over or visibility into platform configurations, conventional security best practices and regulatory requirements become difficult or impossible to implement. With new regulatory structures like the European General Data Protection Regulation (GDPR) introducing high-stakes fines for noncompliance, having visibility and control over where data may be accessed is more important than ever before. Top concerns among security professionals

include the protection of workloads from general security risks, the loss or exposure of data in the event of a data breach, and regulatory compliance.

Existing mitigations of threats against cloud servers are often rooted in firmware or software, making them vulnerable to the same attack strategies. For example, if the firmware can be successfully exploited, then the firmware-based security controls can most likely be circumvented in the same fashion. Hardware-enabled security techniques can help mitigate these threats by establishing and maintaining *platform trust*—an assurance in the integrity of the underlying platform configuration, including hardware, firmware, and software. By providing this assurance, security administrators can gain a level of visibility and control over where access to sensitive workloads and data is permitted. Platform security technologies that establish platform trust can provide notification or even self-correction of detected integrity failures. Platform configurations can automatically be reverted back to a trusted state and give the platform resilience against attack.

All security controls must have a *root of trust (RoT)*—a starting point that is implicitly trusted. Hardware-based controls can provide an immutable foundation for establishing platform integrity. Combining these functions with a means of producing verifiable evidence that these integrity controls are in place and have been executed successfully is the basis of creating a trusted platform. Minimizing the footprint of this RoT translates to reducing the number of modules or technologies that must be implicitly trusted. This substantially reduces the attack surface.

Platforms that secure their underlying firmware and configuration provide the opportunity for trust to be extended higher in the software stack. Verified platform firmware can, in turn, verify the OS boot loader, which can then verify other software components all the way up to the OS itself and the hypervisor or container runtime layers. The transitive trust described here is consistent with the concept of the *chain of trust (CoT)*—a method where each software module in a system boot process is required to measure the next module before transitioning control.

Rooting platform integrity and trust in hardware security controls can strengthen and complement the extension of the CoT into the dynamic software category. There, the CoT can be extended even further to include data and workload protection. Hardware-based protections through CoT technology mechanisms can form a layered security strategy to protect data and workloads as they move to multi-tenant environments in a cloud data center or edge computing facility.

In addition, there are other hardware platform security technologies that can protect data at rest, in transit, and in use by providing hardware-accelerated disk encryption or encryption-based memory isolation. Many of these capabilities can help mitigate threats from speculative execution and side-channel attacks. By using hardware to perform these tasks, the attack surface is mitigated, preventing direct access or modification of the required firmware. Isolating these encryption mechanisms to dedicated hardware can allow performance to be addressed and enhanced separately from other system processes as well. An example of hardware-based isolation is discussed later in the document.

3 Platform Integrity Verification

A key concept of trusted computing is verification of the underlying platform's integrity. Platform integrity is typically comprised of two parts:

- **Cryptographic measurement of software and firmware.** In this report, the term *measurement* refers to calculating a cryptographic hash of a software or firmware executable, configuration file, or other entity. If there is any change in an entity, a new measurement will result in a different hash value than the original [2]. By measuring software and firmware prior to execution, the integrity of the measured modules and configurations can be validated before the platform launches or before data or workloads are accessed. These measurements can also act as cryptographic proof for compliance audits.
- **Firmware and configuration verification.** When firmware and configuration measurements are made, local or remote attestations can be performed to verify if the desired firmware is actually running and if the configurations are authorized [3]. Attestation can also serve as the foundation for further policy decisions that fulfill various cloud security use case implementations. For instance, encryption keys can be released to client workloads if a proof is performed that the platform server is trusted and in compliance with policies.

In some cases, a third part is added to platform integrity:

- **Firmware and configuration recovery.** If the verification step fails (i.e., the attestations do not match the expected measurements), the firmware and configuration can automatically be recovered to a known good state, such as rolling back firmware to a trusted version. The process by which these techniques are implemented affects the overall strength of the assertion that the measured and verified components have not been accidentally altered or maliciously tampered. Recovery technologies allow platforms to maintain resiliency against firmware attacks and accidental provisioning mistakes [4].

There are many ways to measure platform integrity. Most technologies center around the aforementioned concept of the CoT. In many cases, a hardware security module is used to store measurement data to be attested at a later point in time. The rest of this section discusses hardware security modules and various chain of trust technology implementations.

3.1 Hardware Security Module (HSM)

A *hardware security module (HSM)* is "a physical computing device that safeguards and manages cryptographic keys and provides cryptographic processing" [5]. Cryptographic operations such as encryption, decryption, and signature generation/verification are typically hosted on the HSM device, and many implementations provide hardware-accelerated mechanisms for cryptographic operations.

A *trusted platform module (TPM)* is a special type of HSM that can generate cryptographic keys and protect small amounts of sensitive information, such as passwords, cryptographic keys, and cryptographic hash measurements. [3] The TPM is a standalone device that can be integrated with server platforms, client devices, and other products. One of the main use cases of a TPM is

to store digest measurements of platform firmware and configuration during the boot process. Each firmware module is measured by generating a digest, which is then extended to a TPM platform configuration register (PCR). Multiple firmware modules can be extended to the same PCR, and the TPM specification provides guidelines for which firmware measurements are encompassed by each PCR [6].

TPMs also host functionality to generate binding and signing keys that are unique per TPM and stored within the TPM non-volatile random-access memory (NVRAM). The private portion of this key pair is decrypted inside the TPM, making it only accessible by the TPM hardware or firmware. This can create a unique relationship between the keys generated within a TPM and a platform system, restricting private key operations to the platform firmware that has ownership and access to the specified TPM. Binding keys are used for encryption/decryption of data, while signing keys are used to generate/verify cryptographic signatures. The TPM provides a random number generator (RNG) as a protected capability with no access control. This RNG is used in critical cryptographic functionality as an entropy source for nonces, key generation, and randomness in signatures [6].

There are two versions of TPMs: 1.2 and 2.0. The 2.0 version supports additional security features and algorithms [6]. TPMs also meet the National Institute of Standards and Technology (NIST) Federal Information Processing Standard (FIPS) 140 validation criteria and support NIST-approved cryptographic algorithms [7].

3.2 The Chain of Trust (CoT)

The *chain of trust (CoT)* is a method for maintaining valid trust boundaries by applying a principle of transitive trust. Each firmware module in the system boot process is required to measure the next module before transitioning control. Once a firmware module measurement is made, it is recommended to immediately extend the measurement value to an HSM register for attestation at a later point in time [6]. The CoT can be extended further into the application domain, allowing for files, directories, devices, peripherals, etc. to be measured and attested.

Every CoT starts with an RoT module. It can be composed of different hardware and firmware components. For several platform integrity technologies, the RoT core firmware module is rooted in immutable read-only memory (ROM) code. However, not all technologies define their RoTs in this manner [6]. The RoT is typically separated into components that verify and measure. The core root of trust for verification (CRTV) is responsible for verifying the first component before control is passed to it. The core root of trust for measurement (CRTM) is the first component that is executed in the CoT and extends the first measurement to the TPM. The CRTM can be divided into a static portion (SCRTM) and dynamic portion (DCRTM). The SCRTM is composed of elements that measure firmware at system boot time, creating an unchanging set of measurements that will remain consistent across reboots. The DRTM allows a CoT to be established without rebooting the system, permitting the root of trust for measurement to be reestablished dynamically.

An RoT that is built with hardware protections will be more difficult to change, while an RoT that is built solely in firmware can easily be flashed and modified.

Various platform integrity technologies build their own CoTs. Please refer to the following technology examples in the appendices for more information:

- UEFI Secure Boot (SB)
- Intel Trusted Execution Technology (TXT)
- Intel Boot Guard
- Intel Platform Firmware Resilience (PFR)
- Intel Technology Example Summary
- AMD Platform Secure Boot

3.3 Supply Chain Protection

Organizations are increasingly at risk of supply chain compromise, whether intentional or unintentional. Managing cyber supply chain risks requires, in part, ensuring the integrity, quality, and resilience of the supply chain, its products, and its services. Cyber supply chain risks may include counterfeiting, unauthorized production, tampering, theft, and insertion of malicious or otherwise unexpected software and hardware, as well as poor manufacturing and development practices in the cyber supply chain [8] [9] [10].

Special technologies have been developed to help ascertain the authenticity and integrity of platform hardware, including its firmware and configuration. These technologies help ensure that platforms are not tampered with or altered from the time that they are assembled at the manufacturer site to the time that they arrive at a consumer data center ready for installation. Verification of these platform attributes is one aspect of securing the supply chain.[1] Some technologies include an additional feature for locking the boot process or access to these platforms until a secret is provided that only the consumer and manufacturer know.

Please refer to the following technology examples in the appendices for more information:

- Intel Transparent Supply Chain (TSC)
- Intel PFR with Protection in Transit (PIT)

[1] For more information on supply chain security, see the National Cybersecurity Center of Excellence (NCCoE) Supply Chain Assurance project page at https://www.nccoe.nist.gov/projects/building-blocks/supply-chain-assurance.

4 Software Runtime Protection Mechanisms

This section describes various software runtime attacks and protection mechanisms.

4.1 Return Oriented Programming (ROP) and Call/Jump Oriented Programming (COP/JOP) Attacks

ROP attacks focus on utilizing buffer overflows and targeted memory overwrites of return addresses in the stack. Attackers redirect return flows by corrupting addresses on the data stack to be locations in already-executable code. These small selected sequences of code called *gadgets* result in malicious modifications to the system or the invocation of normally unauthorized operations. A common example is a call to the shell executable within the system interface [11].

COP/JOP attacks are similar to ROP attacks, relying on gadget building blocks. They target indirect jump instructions at the end of a gadget, many of which are intentionally emitted by the compiler. However, a jump gadget performs a one-directional control flow transfer to its target, as opposed to ROP, where gadgets return control back to the stack. This can make it difficult for attackers to regain control after executing their gadgets, but solutions to this problem, such as the one presented in [11], are beginning to appear.

Applications can utilize a parallel stack, known as the *shadow stack*, to help mitigate software attacks which attempt to modify the control flow. Utilizing special hardware, the shadow stack is used to store a copy of return addresses; the address is checked against the normal program stack on return operations. If the content differs, an exception is generated, which can help prevent malicious code from gaining control of the system with techniques such as ROP. In this way, shadow stack hardware can help mitigate some of the most common and exploitable types of software bugs.

Several defenses and preventative measures have been developed within industry to accommodate ROP and COP/JOP attacks, including:

- Intel Control-Flow Enforcement Technology (CET)

4.2 Address Translation Attacks

Commodity operating systems rely on virtual memory protection models enabled via paging enforced by the processor memory management unit. Operating systems isolate process and kernel memory using page tables managed by systems software, with access permissions such as user/supervisor and read/write/execute (RWX). Process and kernel memory accesses are via virtual addresses which are mapped to physical memory addresses via address translation structures. These structures used for address translation are critical to enforcing the isolation model.

Modern operating systems are single address space kernels (as opposed to micro-kernels), which provide good performance but have a large attack surface. A vulnerability in the kernel or driver can be leveraged to escalate privileges of a malicious process. Kernel read/write primitives can be leveraged with Write-What-Where vulnerabilities exploited from flaws discovered in kernel code and/or drivers.

Heuristic defense mechanisms such as Page Table randomization can be bypassed with information leaks achieved via malicious read/write primitives. Such information leaks are performed by chaining together a set of system calls (*syscalls*). For example, one syscall can allocate RWX pool memory, and a second can exploit an arbitrary memory write to overwrite the address translation structures. Two types of attacks can utilize this methodology for nefarious purposes. First, an attacker can redirect a virtual address in use to attacker-controlled contents (many times set up in user-space memory). Second, an attacker can create a malicious alias mapping which references desired physical memory with attacker-chosen permissions (e.g., read/write [RW] access to a page via an alias mapping that was originally read-only). It is important for address translation protection mechanisms to block both of these types of attacks.

In addition to protecting the integrity of address translation structures, processors can also detect and block any execution or data access setup by lower privilege code from a higher privilege access. These protections establish boundaries, requiring code to execute with only the necessary permissions and forcing elevated permission requests when needed.

Several defenses and preventative measures have been developed within industry to accommodate memory page-table attacks, including the following:

- Intel Hypervisor Managed Linear Address Translation (HLAT)
- Intel Supervisor Mode Execution Prevention (SMEP) and Supervisor Mode Access Prevention (SMAP)
- AMD Supervisor Mode Execution Prevention (SMEP) and Supervisor Mode Access Prevention (SMAP)

5 Data Protection and Confidential Computing

With the increase in adoption of consumer-based cloud services, virtualization has become a necessity in cloud data center infrastructure. Virtualization simulates the hardware that multiple cloud workloads run on top of. Each workload is isolated from others so that it has access to only its own resources, and each workload can be completely encapsulated for portability [12] [13]. Conventional virtual machines (VMs) have an isolated kernel space running all aspects of a workload alongside the kernel. Today, the virtualized environment has been extended to include containers and full-featured workload orchestration engines. Containers offer application portability by sharing an underlying kernel, which drastically reduces workload-consumed resources and increases performance.

While containers can provide a level of convenience, vulnerabilities in the kernel space and shared layers can be susceptible to widespread exploitation, making security for the underlying platform even more important. With the need for additional protection in the virtualized workspace, an emphasis has been placed on encrypting data both at rest and while in use. *At-rest* encryption provides protection for data on disk. This typically refers to an unmounted data store and protects against threats such as the physical removal of a disk drive. Protecting and securing cloud data while *in use*, also referred to as *confidential computing*, utilizes hardware-enabled features to isolate and process encrypted data in memory so that the data is at less risk of exposure and compromise from concurrent workloads or the underlying system and platform [14]. This section describes technologies that can be leveraged for providing confidential computing for cloud and edge.

A *trusted execution environment (TEE)* is an area or enclave protected by a system processor. Sensitive secrets like cryptographic keys, authentication strings, or data with intellectual property and privacy concerns can be preserved within a TEE, and operations involving these secrets can be performed within the TEE, thereby eliminating the need to extract the secrets outside of the TEE. A TEE also helps ensure that operations performed within it and the associated data cannot be viewed from outside, not even by privileged software or debuggers. Communication with the TEE is designed to only be possible through designated interfaces, and it is the responsibility of the TEE designer/developer to define these interfaces appropriately. A good TEE interface limits access to the bare minimum required to perform the task.

5.1 Memory Isolation

There are many technologies that provide data protection via encryption. Most of these solutions focus on protecting the respective data while at rest and do not cover the fact that the data is decrypted and vulnerable while in use. Applications running in memory share the same platform hardware and can be susceptible to attacks either from other workloads running on the same hardware or from compromised cloud administrators. There is a strong desire to secure intellectual property and ensure that private data is encrypted and not accessible at any point in time, particularly in cloud data centers and edge computing facilities. Various hardware technologies have been developed to encrypt content running in platform memory.

Please refer to the following technology examples in the appendices for more information:

- Intel TME and Multi-Key Total Memory Encryption (MKTME)
- AMD Secure Memory Encryption (SME)/Transparent Memory Encryption (TSME)

5.2 Application Isolation

Application isolation utilizes a TEE to help protect the memory reserved for an individual application. The trust boundary associated with the application is restricted to only the central processing unit (CPU). Future generations of these techniques will allow entire applications to be isolated in their own enclaves rather than only protecting specific operations or memory. By using separate application enclaves with unique per-application keys, sensitive applications can be protected against data exposure, even to malicious insiders with access to the underlying platform. Implementations of application isolation will typically involve developer integration of a toolkit within the application layer, and it is the developer's responsibility to ensure secure TEE design.

Please refer to the following technology examples in the appendices for more information:

- Intel Software Guard Extensions (SGX)

5.3 VM Isolation

As new memory and execution isolation technologies become available, it is more feasible to isolate entire VMs. VMs already enjoy a degree of isolation due to technologies like hardware-assisted virtualization, but the memory of each VM remains in the clear. Some existing memory isolation technologies require implicit trust of the virtual machine manager (VMM). Isolation technologies in future platform generations will remove the VMM from the trust boundary and allow full encryption of VM memory with per-VM unique keys, protecting the VMs from not only malicious software running on the hypervisor host but also rogue firmware.

VM isolation can be used to help protect workloads in multi-tenant environments like public and hybrid clouds. Isolating entire VMs translates to protection against malicious insiders at the cloud provider, or malware exposure and data leakage to other tenants with workloads running on the same platform. Many modern cloud deployments use VMs as container worker nodes. This provides a highly consistent and scalable way to deploy containers regardless of the underlying physical platforms. With full VM isolation, the virtual workers hosting container workloads can be effectively isolated without impacting the benefits of abstracting the container from the underlying platform.

Please refer to the following technology examples in the appendices for more information:

- Intel Trust Domain Extensions (TDX)
- AMD Secure Encrypted Virtualization (SEV)

5.4 Cryptographic Acceleration

Encryption is quickly becoming more widespread in data center applications as industry adopts more standards and guidelines regarding the sensitivity of consumer data and intellectual property. Because cryptographic operations can drain system performance and consume large

603 amounts of compute resources, the industry has adopted specialized hardware interfaces called
604 *cryptographic accelerators*, which offload cryptographic tasks from the main processing unit
605 onto a separate coprocessor chip. Cryptographic accelerators often come in the form of pluggable
606 peripheral adapter cards.

607 Please refer to the following technology examples in the appendices for more information:

608 • Intel Advanced Encryption Standard New Instructions (AES-NI)
609 • Intel QuickAssist Technology (QAT) with Intel Key Protection Technology (KPT)
610 • AMD Advanced Encryption Standard

611

6 Remote Attestation Services

Measuring a server's firmware/configuration and extending these measurements to a hardware interface can help keep track of which firmware is running on a platform. Some platform integrity technologies can even perform local attestation and enforcement of firmware and configuration on a server. However, data centers are usually made up of thousands of servers, and keeping track of them and their respective firmware is an overwhelming task for an operator. A remote service can address this by collating server information and measurement details. Cryptographic signatures can be used to ensure the integrity of transferred measurement data. Furthermore, the remote service can be used to define allowlist policies, specifying which firmware versions and event measurements are acceptable for servers in a particular data center environment. This service would verify or attest each server's collected data against these policies, feeding the results into a policy orchestrator to report, alert, or enforce rules based on the events.

A remote attestation service can provide additional benefits besides verifying server firmware. Specifying allowlist policies for specific firmware versions can allow data center administrators to easily invalidate old versions and roll out new upgrades. In some cases, certain hardware technologies and associated capabilities on platforms can be discoverable by their specific event log measurements recorded in an HSM. The information tracked in this remote attestation service can even be exposed through the data center administration layer directly to the enterprise user. This would give endpoint consumers hardware visibility and the ability to specify firmware requirements or require platform features for the hardware on which their services are running.

The key advantage to remote attestation is the enforcement of compliance across all hardware systems in a data center. The ability to verify against a collective allowlist as opposed to a local system enforcing a supply chain policy provides operators more flexibility and control in a cryptographically secured manner. These enforcement mechanisms can even be combined to provide stronger security policies.

6.1 Platform Attestation

Figure 1 shows a remote attestation service (AS) collecting platform configurations and integrity measurements from data center servers at a cloud service provider (CSP) via a trust agent service running on the platform servers. A cloud operator is responsible for defining allowlisted trust policies. These policies should include information and expected measurements for desired platform CoT technologies. The collected host data is compared and verified against the policies, and a report is generated to record the relevant trust information in the AS database.

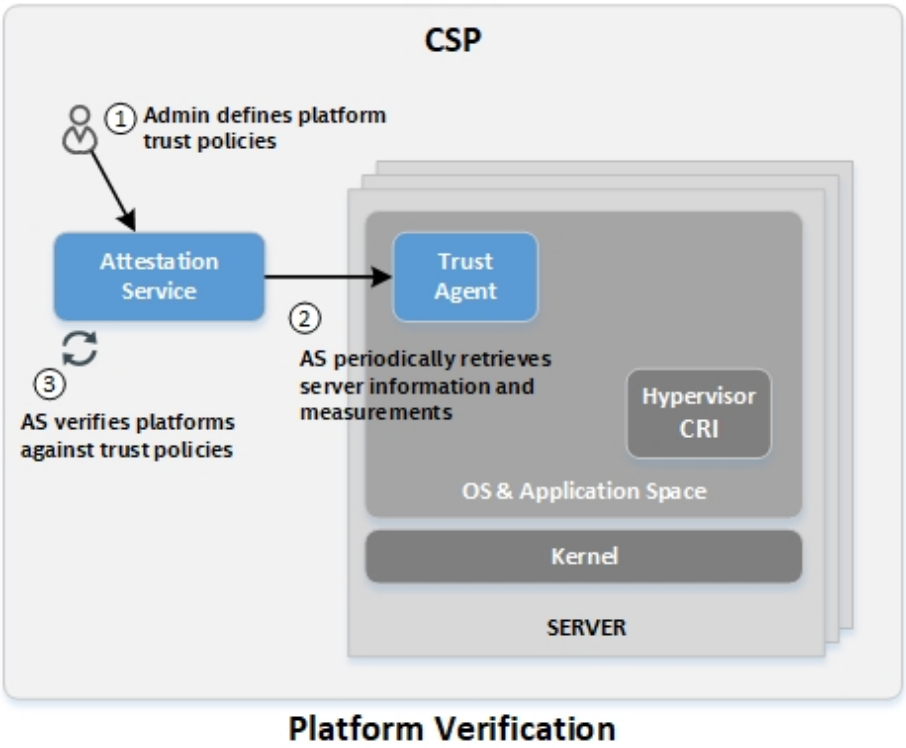


Figure 1: Notional Example of Remote Attestation Service

Platform attestation can be extended to include application integrity or the measurement and verification of the hypervisor container runtime interface (CRI) and applications installed on bare-metal servers. During boot time, an application agent on the server can measure operator-specified files and directories that pertain to particular applications. An allowlist trust policy can be defined to include these expected measurements, and this policy can be included in the overall trust assessment of the platform in the remote AS. By extending measurements to a platform TPM, applications running on the bare-metal server can be added to the CoT. The components of the trust agent and application agent can be added to the policy and measured alongside other applications to ensure that the core feature elements are not tampered with. For example, a typical Linux implementation of the application agent could run inside initrd, and measurements made on the filesystem could be extended to the platform TPM.

An additional feature commonly associated with platform trust is the concept of *asset tagging*. *Asset tags* are simple key value attributes that are associated with a platform like location, company name, division, or department. These key value attributes are tracked and recorded in a central remote service, such as the AS, and can be provisioned directly to a server through the trust agent. The trust agent can then secure these attribute associations with the host platform by writing hash measurement data for the asset tag information to a hardware security chip, such as the platform TPM NVRAM. Measurement data is then retrieved by the AS and included in the platform trust report evaluation.

Please refer to the following technology examples in the appendices for more information:

- Intel Security Libraries for the Data Center (ISecL-DC)

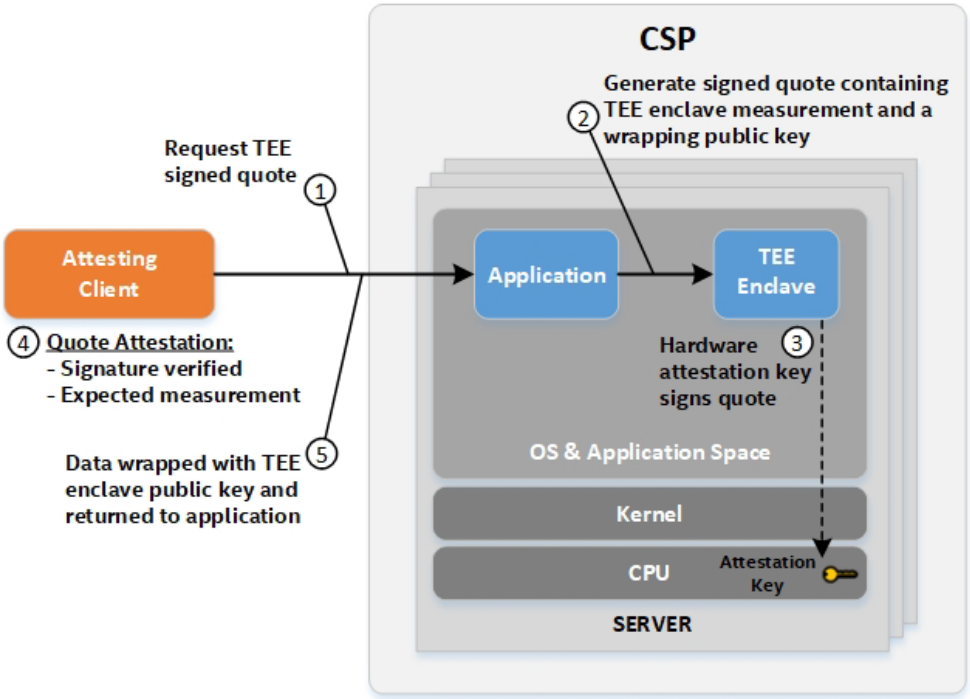
6.2 TEE Attestation

There are instances when the high assurance that the output of the processing in a TEE can be trusted should be extended to an external attesting client. This is achieved thanks to a TEE attestation flow. *TEE attestation* involves the generation of a verifiable cryptographic quote of the enclave by the TEE. The quote is then sent to the attesting client, which can validate the signature of the quote. If the signature is valid, the attesting client concludes that the remote code is running in a genuine TEE enclave.

A quote usually contains the measurement of the TEE enclave, as well as data related to the authenticity of the TEE and the compliant version of it. The measurement is a digest of the content of the enclave (e.g., code and static data) and other information. The measurement obtained at build time is typically known to the attesting client and is compared against a measurement contained in the quote that is actively taken during runtime. This allows the attesting client to determine that the remote code has not been tampered with. A quote may also contain the enclave's developer signature and platform trusted computing base (TCB) information. The authenticity and version of the TEE are verified against TEE provider certificates that are accessible to the tenant or attesting client.

The quote may also contain the public key part of an enclave key pair or a secure hash of the public/private key part if there is a limitation on the size of the quote. In the latter case, the public key part must be communicated along with the quote. The public key allows the attesting client to wrap secrets that it wants to send to the enclave. This capability allows the attesting client to provision secrets directly to the TEE enclave without needing to trust any other software running on the server.

Figure 2 shows an example TEE attestation flow.



692

693 **Figure 2: Notional Example of TEE Attestation Flow**

694

7 Cloud Use Case Scenarios Leveraging Hardware-Enabled Security

This section describes a number of cloud use case scenarios that take advantage of the hardware-enabled security capability and trust attestation capability integrated with the operator orchestration tool to support various security and compliance objectives.

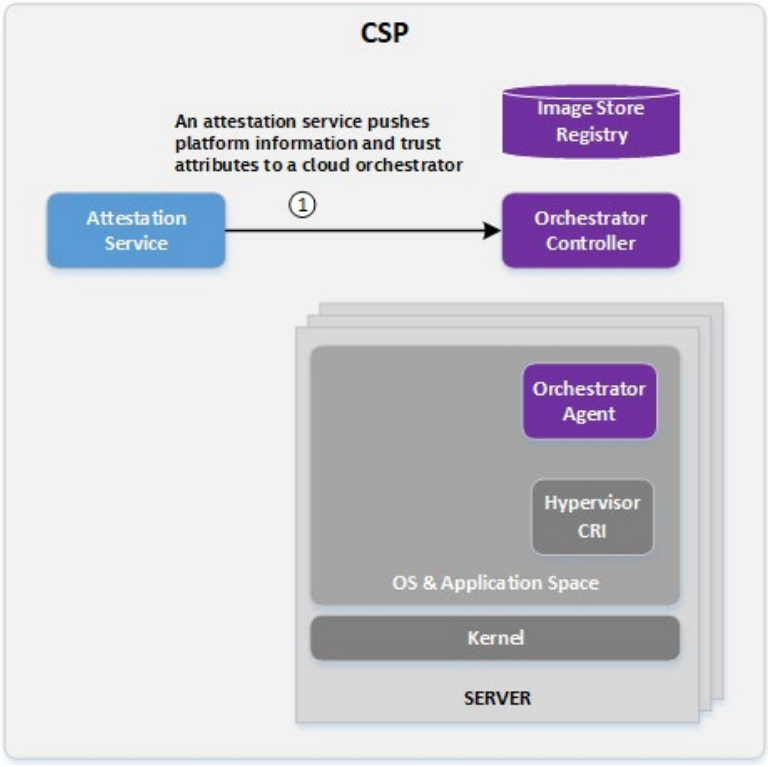
7.1 Visibility to Security Infrastructure

A typical attestation includes validation of the integrity of platform firmware measurements. These measurements are unique to a specific BIOS/UEFI version, meaning that the attestation report provides visibility into the specific firmware version currently in use, in addition to the integrity of that firmware. Attestation can also include hardware configuration and feature support information, both by attesting feature support directly and by resulting in different measurements based on which platform integrity technologies are used.

Cryptographically verifiable reports of platform integrity and security configuration details (e.g., BIOS/UEFI versions, location information, application versions) are extremely useful for compliance auditing. These attestation reports for the physical platform can be paired with workload launch or key release policies, providing traceability to confirm that data and workloads have only been accessed on compliant hardware in compliant configurations with required security technologies enabled.

7.2 Workload Placement on Trusted Platforms

Platform information and verified firmware/configuration measurements retained within an attestation service can be used for policy enforcement in countless use cases. One example is orchestration scheduling. Cloud orchestrators, such as Kubernetes and OpenStack, provide the ability to label server nodes in their database with key value attributes. The attestation service can publish trust and informational attributes to orchestrator databases for use in workload scheduling decisions. Figure 3 illustrates this.



719

720 **Figure 3: Notional Example of Orchestrator Platform Labeling**

721 In OpenStack, this can be accomplished by labeling nodes using custom traits. Workload images
722 can be uploaded to an image store containing metadata that specifies required trait values to be
723 associated with the node that is selected by the scheduling engine. In Kubernetes, nodes can be
724 labeled in etcd via node selector or node affinity. Custom resource definitions (CRDs) can be
725 written and plugged into Kubernetes to receive label values from the attestation service and
726 associate them with nodes in the etcd. When a deployment or container is launched, node
727 selector or node affinity attributes can be included in the configuration yaml to instruct
728 Kubernetes to only select nodes that have the specified labels. Other orchestrator engines and
729 flavors can be modified to accommodate a similar use case. Figure 4 illustrates how an
730 orchestrator can be configured to only launch workloads on trusted platforms or platforms with
731 specified asset tag attributes.

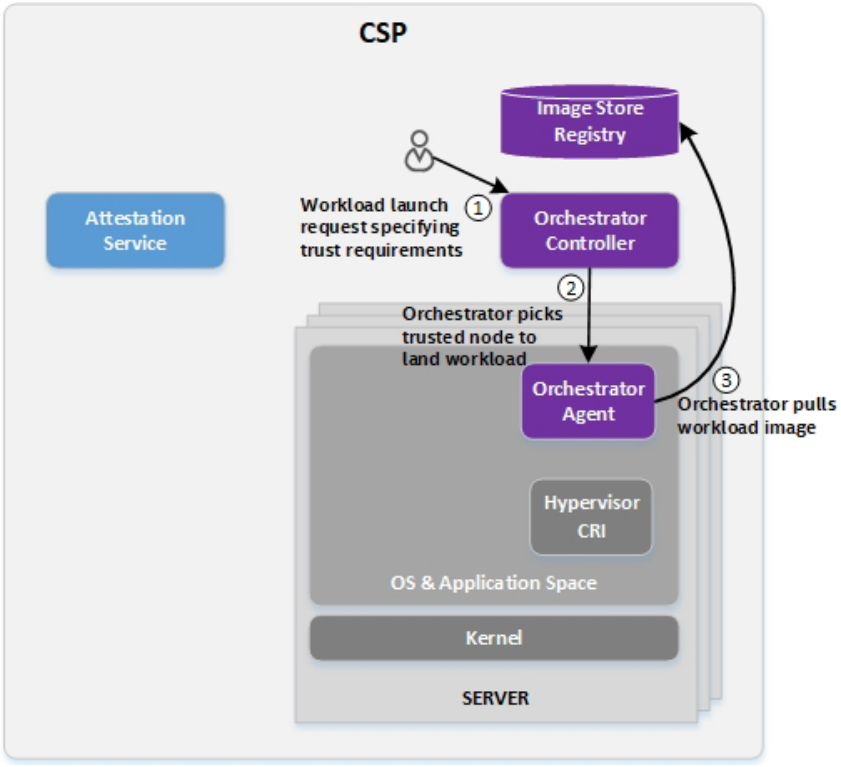


Figure 4: Notional Example of Orchestrator Scheduling

7.3 Asset Tagging and Trusted Location

Trusted geolocation is a specific implementation of the aforementioned trusted asset tag feature used with platform attestation. Key attribute values specifying location information are used as asset tags and provisioned to server hardware, such as the TPM. In this way, location information can be included in platform attestation reports and therefore consumed by cloud orchestrators, infrastructure management applications, policy engines, and other entities [15]. Orchestration using asset tags can be used to segregate workloads and data access in a wide variety of scenarios. Geolocation can be an important attribute to consider with hybrid cloud environments subject to regulatory controls like GDPR, for example. Violating these constraints by allowing access to data outside of specific geopolitical boundaries can trigger substantial penalties.

In addition to location, the same principle can apply to other sorts of tag information. For example, some servers might be tagged as appropriate for storing health information subject to Health Insurance Portability and Accountability Act (HIPAA) compliance. Data and workloads requiring this level of compliance should only be accessed on platforms configured to meet those compliance requirements. Other servers may be used to store or process information and workloads not subject to HIPAA requirements. Asset tags can be used to flag which servers are appropriate for which workloads beyond a simple statement of the integrity of those platforms. The attestation mechanisms help ensure that the asset tag information is genuine, preventing easy subversion.

Outside of specific regulatory requirements, an organization may wish to segregate workloads by department. For example, human resources and finance information could be restricted to

platforms with different security profiles, and big data workloads could be required to run on platforms tagged for performance capabilities. For cloud orchestration platforms that do not natively support discovery or scheduling of workloads based on specific platform features, asset tags can provide a mechanism for seamlessly adding such a capability. For example, workloads that require Intel SGX can be orchestrated to only run on platforms that support the SGX platform feature, even if the cloud platform does not natively discover support for SGX. The open-ended user-configurable asset tag functionality allows virtually any level of subdivision of resources for business, security, or regulatory needs.

7.4 Workload Confidentiality

Consumers who place their workloads in the cloud or the edge are typically forced to accept that their workloads are secured by their service providers without insight or knowledge as to what security mechanisms are in place. The ability for end users to encrypt their workload images can provide at-rest cryptographic isolation to protect consumer data and intellectual property. Key control is integral to the workload encryption process. While it is preferable to transition key storage, management, and ownership to the endpoint consumer, an appropriate key release policy must be defined that includes a guarantee from the service provider that the utilized hardware platform and firmware are secure and uncompromised.

There are several key management solutions (KMSs) in production that provide services to create and store keys. Many of these are compliant with the industry-standardized Key Management Interoperability Protocol (KMIP) and can be deployed within consumer enterprises. The concept is to provide a thin layer on top of the KMS called a *key broker*, as illustrated in Figure 5, that applies and evaluates policies to requests that come into the KMS. Supported requests to the key broker include key creation, key release policy association, and key request by evaluating associated policies. The key release policy can be any arbitrary set of rules that must be fulfilled before a key is released. The policy for key release is open-ended and meant to be easily extendible, but for the purpose of this discussion, a policy associated with platform trust is assumed.

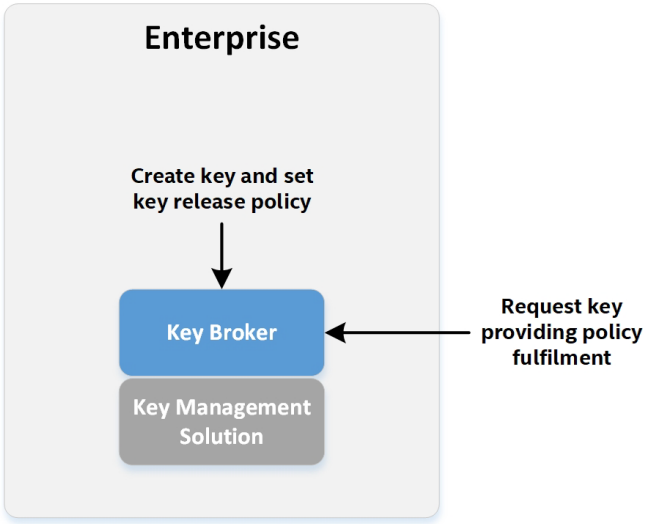
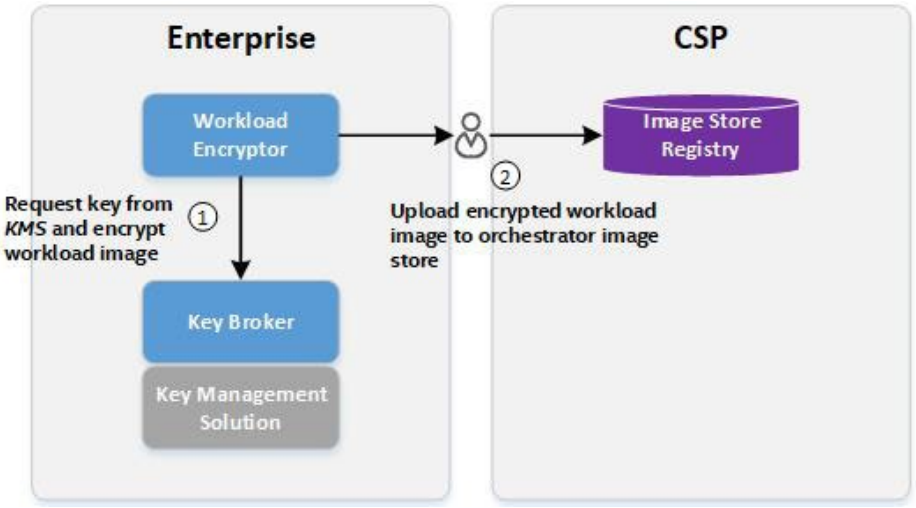


Figure 5: Notional Example of Key Brokerage

784 Once the key policy has been determined, a KMS-created and managed key can be used to
785 encrypt a workload image, as shown in Figure 6. The enterprise user may then upload the
786 encrypted image to a CSP orchestrator image store or registry.



787

788 **Figure 6: Notional Example of Workload Image Encryption**

789 The key retrieval and decryption process is the most complex piece of the workload
790 confidentiality story, as Figure 7 shows. It relies on a secure key transfer between the enterprise
791 and CSP with an appropriate key release policy managed by the key broker. The policy for key
792 release discussed here is based on platform trust and the valid proof thereof. The policy can also
793 dictate a requirement to wrap the key using a public wrapping key, with the private portion of the
794 wrapping key only known to the hardware platform within the CSP.

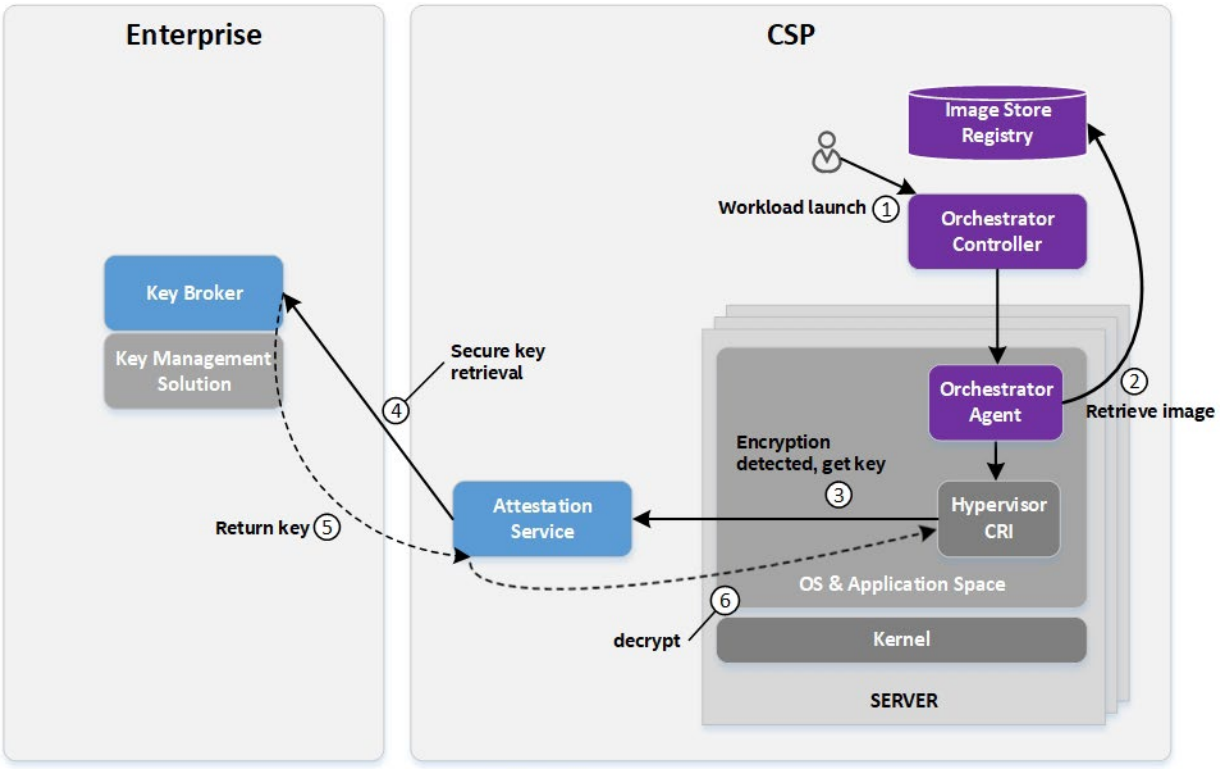


Figure 7: Notional Example of Workload Decryption

When the runtime node service receives the launch request, it can detect that the image is encrypted and make a request to retrieve the decryption key. This request can be passed through an attestation service so that an internal trust evaluation for the platform can be performed. The key request is forwarded to the key broker with proof that the platform has been attested. The key broker can then verify the attested platform report and release the key back to the CSP and node runtime services. At that time the node runtime can decrypt the image and proceed with the normal workload orchestration. The disk encryption kernel subsystem can provide at-rest encryption for the workload on the platform.

7.5 Protecting Keys and Secrets

Cryptographic keys are high-value assets in workloads, especially in environments where the owner of the keys is not in complete control of the infrastructure, such as public clouds, edge computing, and network functions virtualization (NFV) deployments. In these environments, keys are typically provisioned on disk as flat files or entries in configuration files. At runtime, workloads read the keys into random access memory (RAM) and use them to perform cryptographic operations like data signing, encryption/decryption, or Transport Layer Security (TLS) termination.

Keys on disk and in RAM are exposed to conventional attacks like privilege escalation, remote code execution (RCE), and input buffer mismanagement. Keys can also be stolen by malicious administrators or be disclosed because of operational errors. For example, an improperly protected VM snapshot can be used by a malicious agent to extract keys.

An HSM can be attached to a server and used by workloads to store keys and perform cryptographic operations. This results in keys being protected at rest and in use. In this model, keys are never stored on disk or loaded into RAM. If attaching an HSM to a server is not an option, or if keys are needed in many servers at the same time, an alternative option is to use a network HSM. Workloads send the payload that needs cryptographic processing over a network connection to the network HSM, which then performs the cryptographic operations locally, typically in an attached HSM.

An HSM option is not feasible in some environments. Workload owners may not have access to a cloud or edge environment in order to attach their HSM to a hardware server. Network HSMs can suffer from network latency, and some workloads require an optimized response time. Additionally, network HSMs are often provided as a service by the cloud, edge, or NFV providers and are billed by the number of transactions. Cost is often a deciding factor for using a provider network HSM.

8 Next Steps

NIST is seeking feedback from the community on the content of the report and soliciting additional technology example contributions from other companies. The report is intended to be a living document that will be frequently updated to reflect advances in technology and the availability of commercial implementations and solutions. This can help raise the bar on platform security and evolve the use cases.

Please send your feedback and comments on this report to hwsec@nist.gov.

NIST is also working on other publications on hardware-enabled security as part of the NCCoE Trusted Cloud project. More information on the project and links to the other publications are available at https://www.nccoe.nist.gov/projects/building-blocks/trusted-cloud.

842 # References

[1] Barrett B (2018) *Russia's Elite Hackers Have a Clever New Trick That's Very Hard to Fix*, Wired. Available at https://www.wired.com/story/fancy-bear-hackers-uefi-rootkit

[2] Intel Corporation (2021) *Intel® Trusted Execution Technology (Intel® TXT) – Software Development Guide – Measured Launched Environment Developer's Guide, Revision 017*. Available at https://www.intel.com/content/www/us/en/software-developers/intel-txt-software-development-guide.html

[3] Regenscheid AR (2014) BIOS Protection Guidelines for Servers. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-147B. https://doi.org/10.6028/NIST.SP.800-147B

[4] Regenscheid AR (2018) Platform Firmware Resiliency Guidelines. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-193. https://doi.org/10.6028/NIST.SP.800-193

[5] Barker EB, Barker WC (2019) Recommendation for Key Management: Part 2 – Best Practices for Key Management Organizations. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-57 Part 2, Rev. 1. https://doi.org/10.6028/NIST.SP.800-57pt2r1

[6] Trusted Computing Group (2019) *Trusted Platform Module Library Specification, Family "2.0"*. Available at https://trustedcomputinggroup.org/work-groups/trusted-platform-module/

[7] National Institute of Standards and Technology (2001) Security Requirements for Cryptographic Modules. (U.S. Department of Commerce, Washington, DC), Federal Information Processing Standards Publication (FIPS) 140-2, Change Notice 2 December 03, 2002. https://doi.org/10.6028/NIST.FIPS.140-2

[8] Diamond T, Grayson N, Paulsen C, Polk T, Regenscheid A, Souppaya M, Brown C (2020) *Validating the Integrity of Computing Devices: Supply Chain Assurance*. (National Institute of Standards and Technology, Gaithersburg, MD). Available at https://www.nccoe.nist.gov/projects/building-blocks/supply-chain-assurance

[9] Boyens JM, Paulsen C, Moorthy R, Bartol N (2015) Supply Chain Risk Management Practices for Federal Information Systems and Organizations. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-161. https://doi.org/10.6028/NIST.SP.800-161

[10] National Institute of Standards and Technology (2020) *Cyber Supply Chain Risk Management*. Available at https://csrc.nist.gov/Projects/cyber-supply-chain-risk-management

[11] Bletsch T, Jiang X, Freeh V, Liang Z (2011) *Jump-Oriented Programming: A New Class of Code-Reuse Attack*. Available at https://www.comp.nus.edu.sg/~liangzk/papers/asiaccs11.pdf

[12] Scarfone KA, Souppaya MP, Hoffman P (2011) Guide to Security for Full Virtualization Technologies. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-125. https://doi.org/10.6028/NIST.SP.800-125

[13] Intel Corporation (2020) *Intel® Virtualization Technology (Intel® VT).* Available at https://www.intel.com/content/www/us/en/virtualization/virtualization-technology/intel-virtualization-technology.html

[14] Linux Foundation (2020) *Confidential Computing Consortium*. Available at https://confidentialcomputing.io

[15] Bartock MJ, Souppaya MP, Yeluri R, Shetty U, Greene J, Orrin S, Prafullchandra H, McLeese J, Scarfone KA (2015) Trusted Geolocation in the Cloud: Proof of Concept Implementation. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Interagency or Internal Report (IR) 7904. https://doi.org/10.6028/NIST.IR.7904

[16] Debian Wiki (2019) *Secure Boot*. Available at https://wiki.debian.org/SecureBoot

[17] Wilkins R, Richardson B (2013) *UEFI Secure Boot in Modern Computer Security Solutions* (Unified Extensible Firmware Interface Forum). Available at https://uefi.org/sites/default/files/resources/UEFI_Secure_Boot_in_Modern_Computer_Security_Solutions_2013.pdf

[18] RedHat (2018) *README* (for shim). Available at https://github.com/rhboot/shim

[19] Intel Corporation (2018) *Intel® Trusted Execution Technology (Intel® TXT) Overview*. Available at https://www.intel.com/content/www/us/en/support/articles/000025873/technologies.html

[20] Futral W, Greene J (2013) *Intel® Trusted Execution Technology for Server Platforms* (Apress, Berkeley, CA). Available at https://www.apress.com/gp/book/9781430261483

[21] Linux Kernel Organization (2020) *Intel® TXT Overview*. Available at https://www.kernel.org/doc/Documentation/intel_txt.txt

[22] Intel Corporation (2020) *Transparent Supply Chain*. Available at https://www.intel.com/content/www/us/en/products/docs/servers/transparent-supply-chain.html

[23] Intel Corporation (2020) *Intel Highlights Latest Security Investments at RSA 2020*. Available at https://newsroom.intel.com/news-releases/intel-highlights-latest-security-investments-rsa-2020/

[24] Department of Defense (2018) Subpart 246.870, Contractors' Counterfeit Electronic Part Detection and Avoidance Systems, *Defense Federal Acquisition Regulation Supplement*. Available at https://www.acq.osd.mil/dpap/dars/dfars/html/current/246_8.htm#246.870-2

[25] Intel Corporation (2021) *Intel 64 and IA-32 Architectures Software Developer's Manual Volume 1: Basic Architecture*. Available at https://software.intel.com/content/www/us/en/develop/download/intel-64-and-ia-32-architectures-software-developers-manual-volume-1-basic-architecture.html

[26] Nichols S (2020) *RIP ROP, COP, JOP? Intel to bring anti-exploit tech to market in this year's Tiger Lake chip family*. (The Register, San Francisco, CA). Available at https://www.theregister.com/2020/06/15/intel_cet_tiger_lake

[27] Patel BV (2020) *A Technical Look at Intel's Control-flow Enforcement Technology*. (Intel Fellow Client Computing Group, Intel Corporation). Available at https://software.intel.com/content/www/us/en/develop/articles/technical-look-control-flow-enforcement-technology.html

[28] Patel BV (2016) *Intel Releases New Technology Specifications to Protect Against ROP attacks*. (Intel Corporation). Available at https://software.intel.com/content/www/us/en/develop/blogs/intel-release-new-technology-specifications-protect-rop-attacks.html

[29] Shanbhogue V, Gupta D, Sahita R (2019) Security Analysis of Processor Instruction Set Architecture for Enforcing Control-Flow Integrity. (HASP '19: Proceedings of the 8th International Workshop on Hardware and Architectural Support for Security and Privacy). https://doi.org/10.1145/3337167.3337175

[30] Intel Corporation (2021) Intel Architecture Instruction Set Extensions and Future Features Programming Reference. Available at https://software.intel.com/content/www/us/en/develop/download/intel-architecture-instruction-set-extensions-programming-reference.html

[31] Intel Corporation (2018) *Intel Security Features and Technologies Related to Transient Execution Attacks*. Available at https://software.intel.com/content/www/us/en/develop/articles/software-security-guidance/best-practices/related-intel-security-features-technologies.html

[32] Anvin HP (2012) *Description x86: Supervisor Mode Access Prevention*. Available at https://lwn.net/Articles/517251/

[33] Corbet J (2012) *Supervisor Mode Access Prevention*. Available at https://lwn.net/Articles/517475/

[34] Intel Corporation (2020) *Strengthen Enclave Trust with Attestation*. Available at https://software.intel.com/en-us/sgx/attestation-services

[35] European Telecommunications Standards Institute (2020) *Network Functions Virtualisation (NFV)*. Available at http://www.etsi.org/technologies-clusters/technologies/nfv

[36] Intel Corporation (2020) *Intel® Trust Domain Extensions.* Available at https://software.intel.com/content/dam/develop/external/us/en/documents/tdx-whitepaper-v4.pdf

[37] Intel Corporation (2010) *Intel Data Protection Technology with AES-NI and Secure Key*. Available at https://www.intel.com/content/www/us/en/architecture-and-technology/advanced-encryption-standard-aes/data-protection-aes-general-technology.html

[38] Intel Corporation (2020) *Flexible Workload Acceleration on Intel Architecture Lowers Equipment Cost*. Available at https://www.intel.fr/content/dam/www/public/us/en/documents/white-papers/communications-quick-assist-paper.pdf

[39] Tadepalli, H (2017) *Intel QuickAssist Technology with Intel Key Protection Technology in Intel Server Platforms Based on Intel Xeon processor Scalable Family*. (Intel Corporation). Available at https://www.aspsys.com/images/solutions/hpc-processors/intel-xeon/Intel-Key-Protection-Technology.pdf

[40] Intel Corporation (2020) *Intel® Security Libraries for Data Center (Intel® SecL-DC).* Available at https://01.org/intel-secl

[41] Kaplan D, Powell J, Woller T (2016) *AMD Memory Encryption*. (Advanced Micro Devices, Inc.). Available at https://developer.amd.com/wordpress/media/2013/12/AMD_Memory_Encryption_Whitepaper_v7-Public.pdf

[42] Kaplan D (2017) *Protecting VM Register State with SEV-ES*. (Advanced Micro Devices, Inc.). Available at https://www.amd.com/system/files/TechDocs/Protecting%20VM%20Register%20State%20with%20SEV-ES.pdf

[43] Advanced Micro Devices, Inc. (2020) *AMD SEV-SNP: Strengthening VM Isolation with Integrity Protection and More*. Available at https://www.amd.com/system/files/TechDocs/SEV-SNP-strengthening-vm-isolation-with-integrity-protection-and-more.pdf

843

Appendix A—Vendor-Agnostic Technology Examples

This section describes vendor-agnostic technology examples that map back to the key concepts described in the various sections of the document.

A.1 Platform Integrity Verification

A.1.1 UEFI Secure Boot (SB)

"UEFI Secure Boot (SB) is a verification mechanism for ensuring that code launched by a computer's UEFI firmware is trusted" [16]. SB prevents malware from taking "advantage of several pre-boot attack points, including the system-embedded firmware itself, as well as the interval between the firmware initiation and the loading of the operating system" [17].

The basic idea behind SB is to sign executables using a public-key cryptography scheme. The public part of a *platform key (PK)* can be stored in the firmware for use as a root key. Additional key exchange keys (KEKs) can also have their public portion stored in the firmware in what is called the *signature database*. This database contains public keys that can be used to verify different components that might be used by UEFI (e.g., drivers), as well as bootloaders and OSs that are loaded from external sources (e.g., disks, USB devices, network). The signature database can also contain *forbidden signatures*, which correspond to a revocation list of previously valid keys. The signature database is meant to contain the current list of authorized and forbidden keys as determined by the UEFI organization. The signature on an executable is verified against the signature database before the executable can be launched, and any attempt to execute an untrusted program will be prevented [16][17].

Before a PK is loaded into the firmware, UEFI is considered to be in *setup mode*, which allows anyone to write a PK or KEK to the firmware. Writing the PK switches the firmware into *user mode*. Once in user mode, PKs and KEKs can only be written if they are signed using the private portion of the PK. Essentially, the PK is meant to authenticate the platform owner, while the KEKs are used to authenticate other components of the distribution (distro), like OSs [17].

Shim is a simple software package that is designed to work as a first-stage bootloader on UEFI systems. It is a common piece of code that is considered safe, well-understood, and audited so that it can be trusted and signed using PKs. This means that firmware certificate authority (CA) providers only have to worry about signing shim and not all of the other programs that vendors might want to support [16]. Shim then becomes the RoT for all the other distro-provided UEFI programs. It embeds a distro-specific CA key that is itself used to sign additional programs (e.g., Linux, GRUB, fwupdate). This allows for a clean delegation of trust; the distros are then responsible for signing the rest of their packages. Ideally, shim will not need to be updated often, which should reduce the workload on the central auditing and CA teams [16].

A key part of the shim design is to allow users to control their own systems. The distro CA key is built into the shim binary itself, but there is also an extra database of keys that can be managed by the user—the so-called *Machine Owner Key (MOK)*. Keys can be added and removed in the MOK list by the user, entirely separate from the distro CA key. The mokutil utility can be used to help manage the keys from Linux OS, but changes to the MOK keys may only be confirmed

883 directly from the console at boot time. This helps remove the risk of OS malware potentially
884 enrolling new keys and therefore bypassing SB [16].

885 On systems with a TPM chip enabled and supported by the system firmware, shim will extend
886 various PCRs with the digests of the targets it is loading [18]. Certificate hashes are also
887 extended to the TPM, including system, vendor, MOK, and shim denylisted and allowlisted
888 certificate digests.

889

Appendix B—Intel Technology Examples

This section describes a number of Intel technology examples that map back to the key concepts described in the various sections of the document.

B.1 Platform Integrity Verification

B.1.1 The Chain of Trust (CoT)

B.1.1.1 Intel Trusted Execution Technology (TXT)

Intel Trusted Execution Technology (TXT) in conjunction with a TPM provides a hardware RoT available on Intel server and client platforms that enables "security capabilities such as measured launch and protected execution" [19]. TXT utilizes *authenticated code modules (ACMs)* that measure various pieces of the CoT during boot time and extend them to the platform TPM [2][19]. TXT's ACMs are chipset-specific signed binaries that are called to perform functions required to enable the TXT environment. An ACM is loaded into and executed from within the CPU cache in an area referred to as the *authenticated code RAM (AC RAM)*. CPU microcode, which acts as the *core root of trust for measurement (CRTM)*, authenticates the ACM by verifying its included digital signature against a manufacturer public key with its digest hard-coded within the chipset. The ACM code, loaded into protected memory inside the processor, performs various tests and verifications of chipset and processor configurations.

The ACMs needed to initialize the TXT environment are the BIOS and the Secure Initialization (SINIT) ACMs. Both are typically provided within the platform BIOS image. The SINIT ACM can be provisioned on disk as well [2][20]. The BIOS ACM is responsible for measuring the BIOS firmware to the TPM and performs additional BIOS-based security operations. The latest version of TXT converged with Intel Boot Guard Technology labels this ACM as the Startup ACM to differentiate it from the legacy BIOS ACM. The SINIT ACM is used to measure the system software or operating system to the TPM, and it "initializes the platform so the OS can enter the secure mode of operation" [20].

When the BIOS startup procedures have completed, control is transitioned to the OS loader. In a TXT-enabled system, the OS loader is instructed to load a special module called Trusted Boot before loading the first kernel module [20]. Trusted Boot (tboot) is an open-source, pre-kernel/virtual machine manager (VMM) module that integrates with TXT to perform a measured launch of an OS kernel/VMM. The tboot design typically has two parts: a preamble and the trusted core. The tboot preamble is most commonly executed by the OS loader but can be loaded at OS runtime. The tboot preamble is responsible for preparing SINIT input parameters and is untrusted by default. It executes the processor instruction that passes control to the CPU microcode. The microcode loads the SINIT into AC RAM, authenticates it, measures SINIT to the TPM, and passes control to it. SINIT verifies the platform configuration and enforces any present Launch Control Policies, measuring them and tboot trusted core to the TPM. The tboot trusted core takes control and continues the CoT, measuring the OS kernel and additional modules (like initrd) before passing control to the OS [21].

Intel TXT includes a policy engine feature that provides the capability to specify known good platform configurations. These *Launch Control Policies* (LCPs) dictate which system software is

permitted to perform a secure launch. LCPs can enforce specific platform configurations and tboot trusted core versions required to launch a system environment [20].

B.1.1.2 Intel Boot Guard

Intel Boot Guard provides a hardware RoT for authenticating the BIOS. An original equipment manufacturer (OEM) enables Boot Guard authentication on the server manufacturing line by permanently fusing a policy and OEM-owned public key into the silicon. When an Intel processor identifies that Boot Guard has been enabled on the platform, it authenticates and launches an ACM. The ACM loads the initial BIOS or Initial Boot Block (IBB) into the processor cache, authenticates it using the fused OEM public key, and measures it into the TPM.

If the IBB authenticates properly, it verifies the remaining BIOS firmware, loads it into memory, and transfers execution control. The IBB is restricted to this limited functionality, which allows it to have a small enough size to fit in the on-die cache memory of Intel silicon. If the Boot Guard authentication fails, the system is forced to shut down. When the Boot Guard execution completes, the CoT can continue for other components by means of UEFI Secure Boot. TXT can be used in conjunction with these technologies to provide a dynamic trusted launch of the OS kernel and software.

Because Boot Guard is rooted in permanent silicon fuses and authenticates the initial BIOS from the processor cache, it provides resistance from certain classes of physical attacks. Boot Guard also uses fuses to provide permanent revocation of compromised ACMs, BIOS images, and input polices.

B.1.1.3 Intel Platform Firmware Resilience (PFR)

Intel Platform Firmware Resilience (PFR) technology is a platform-level solution that creates an open platform RoT based on a programmable logic device. It is designed to provide firmware resiliency (in accordance with NIST SP 800-193 [4]) and comprehensive protection for various platform firmware components, including BIOS, Server Platform Services Firmware (SPS FW), and BMCs. PFR provides the platform owner with a minimal trusted compute base (TCB) under full platform-owner control. This TCB provides cryptographic authentication and automatic recovery of platform firmware to help guarantee correct platform operation and to return to a known good state in case of a malicious attack or an operator error such as a failed update.

NIST SP 800-193 [4] outlines three guiding principles to support the resiliency of platforms against potentially destructive attacks:

- **Protection:** Mechanisms for ensuring that platform firmware code and critical data remain in a state of integrity and are protected from corruption, such as the process for ensuring the authenticity and integrity of firmware updates
- **Detection:** Mechanisms for detecting when platform firmware code and critical data have been corrupted
- **Recovery:** Mechanisms for restoring platform firmware code and critical data to a state of integrity in the event that any such firmware code or critical data are detected to have

968 been corrupted or when forced to recover through an authorized mechanism. Recovery is
969 limited to the ability to recover firmware code and critical data.

970 In addition, NIST SP 800-193 [4] provides guidance on meeting those requirements via three
971 main functions of a Platform Root of Trust:

972 • **Root of Trust for Update (RTU),** which is responsible for authenticating firmware
973 updates and critical data changes to support platform protection capabilities; this includes
974 signature verification of firmware updates as well as rollback protections during update.

975 • **Root of Trust for Detection (RTD),** which is responsible for firmware and critical data
976 corruption detection capabilities.

977 • **Root of Trust for Recovery (RTRec),** which is responsible for recovery of firmware and
978 critical data when corruption is detected or when instructed by an administrator.

979 PFR is designed to support NIST guidelines and create a resilient platform that is able to self-
980 recover upon detection of attack or firmware corruption. This includes verification of all
981 platform firmware and configuration at platform power-on time, active protection of platform
982 non-volatile memory at runtime, and active protection of the Serial Peripheral Interface (SPI
983 flash) and System Management Bus (SMBus). PFR functionality also incorporates monitoring
984 the platform component's boot progress and providing automatic firmware recovery to a known
985 good state upon detection of firmware or configuration corruption. PFR achieves this goal by
986 utilizing a Field-Programmable Gate Array (FPGA) to establish an RoT.

987 PFR technology defines a special pre-boot mode (T-1) where only the PFR FPGA is active. Intel
988 Xeon processors and other devices that could potentially interfere with the boot process, such as
989 the Platform Controller Hub (PCH)/Manageability Engine (ME) and BMC, are not powered.
990 Boot critical firmware, like the BIOS, ME, and BMC, are cryptographically verified during T-1
991 mode. In case of corruption, a recovery event is triggered, and the corrupted firmware in the
992 active regions of the SPI flash is erased and restored with a known-good recovery copy. Once
993 successful, the system proceeds to boot in a normal mode, leveraging Boot Guard for static RoT
994 coverage.

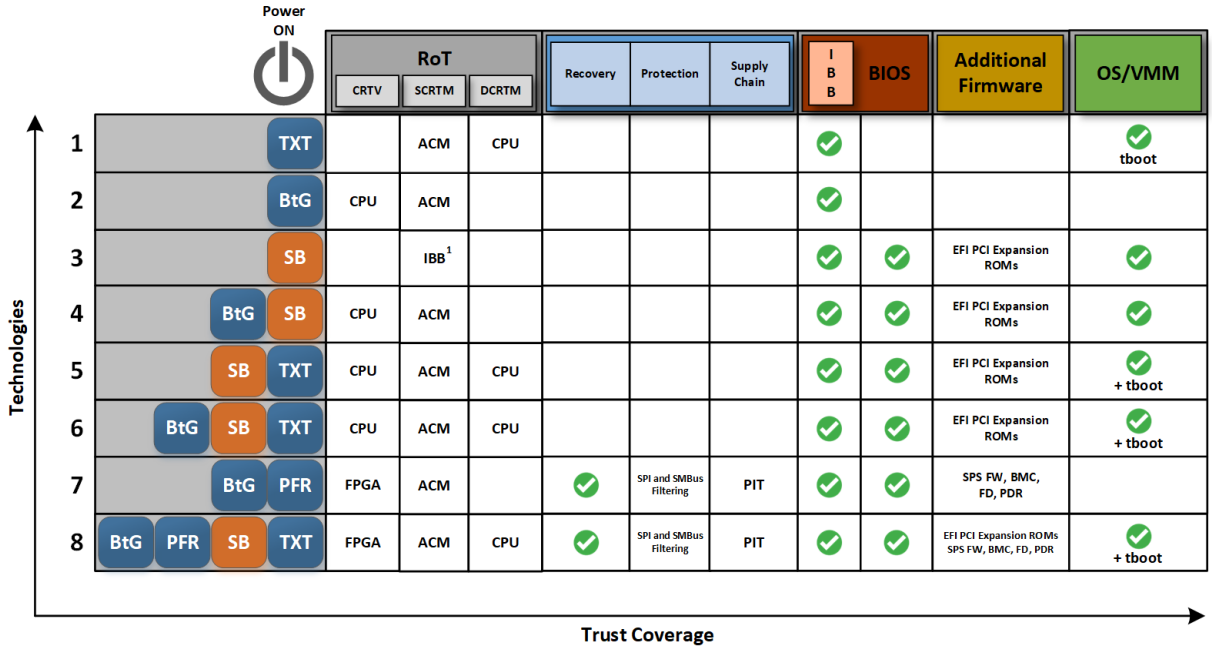
995 The PFR FPGA RoT leverages a key hierarchy to authenticate data structures residing in SPI
996 flash. The key hierarchy is based on a provisioned Root Key (RK) stored in the NVRAM of the
997 FPGA RoT and a Code Signing Key (CSK) structure, which is endorsed by the RK, stored in the
998 SPI flash, and used for the signing of lower-level data structures. The PFR FPGA uses this CSK
999 to verify the digital signature of the Platform Firmware Manifest (PFM), which describes the
1000 expected measurements of the platform firmware. The PFR FPGA RoT verifies those
1001 measurements before allowing the system to boot. When a recovery is needed, either because
1002 measurements do not match the expected value or because a hang is detected during system
1003 bootup, the PFR FPGA RoT uses a recovery image to recover the firmware. The recovery image
1004 and any update images are stored in a compressed capsule format and verified using a digital
1005 signature.

1006 Each platform firmware storage is divided into three major sections: Active, Recovery, and
1007 Staging. The Recovery regions, as well as the static parts of the Active regions, are write-
1008 protected from other platform components by the PFR FPGA RoT. The Staging region is open to

the other platform components for writing in order to provide an area to place digitally signed and compressed update capsules, which are then verified by the PFR FPGA RoT before being committed to the Active or Recovery regions. The Recovery copy can be updated in T-1 mode once the PFR FPGA has verified the digital signature of the update capsule and confirmed that the recovery image candidate is bootable.

B.1.1.4 Technology Example Summary

There are several technologies that provide different levels of platform integrity and trust. Individual technologies do not provide a complete CoT. When used in combination, they can provide comprehensive coverage all the way up to the OS and VMM layer. Figure 8 outlines the firmware and software coverage of each existing CoT technology example.



[1] IBB is meant to describe the portion of BIOS which performs the first measurement

Figure 8: Firmware and Software Coverage of Existing Chain of Trust Technologies

Figure 8 identifies the components of each technology that make up the RoT in their own respective chains and also shows a rough outline of the firmware and software coverage of each technology.

Because many technologies are available, it can be difficult to decide on the correct combination for deployment. Figure 8 illustrates the possible combinations of technologies that extend measurements to a TPM for platform integrity attestation. Note that each combination includes at least one hardware technology to ensure an RoT implementation. A complementary option for extending the CoT up through the OS can also be provided. Including only the hardware technologies would break the CoT by supplying integrity measurements for only pre-OS firmware. Using only UEFI Secure Boot will use firmware as the RoT that does not have hardware security protections and is much more susceptible to attack. By enabling both parts, the CoT can be extended from a hardware RoT into the OS and beyond.

These combinations will help ensure that appropriate measurements are extended to a TPM for integrity attestation and can prevent a server from booting if specific security modules are compromised. The attestation mechanisms provided by these technologies give cryptographic proof of the integrity of measured components, which can be used to provide visibility into platform security configurations and prove integrity. Note the combination of UEFI SB with TXT in Figure 8. This combination provides the UEFI SB signature verification capability on top of the tboot integrity measurement in the OS/VMM layer.

In addition to attestation, PFR provides both additional verification of platform firmware and adds automatic recovery of compromised firmware to known good versions. PFR works with any combination of CoT technologies, providing a defense and resilience against firmware attack vectors. Combining a hardware-based firmware resilience technology like PFR with a hardware-based CoT configuration is part of a layered security strategy.

B.1.2 Supply Chain Protection

B.1.2.1 Intel Transparent Supply Chain (TSC)

"Intel Transparent Supply Chain (TSC) is a set of policies and procedures implemented at ODM factories that enable end-users to validate where and when every component of a platform was manufactured" [22]. "Intel TSC tools allow platform manufacturers to bind platform information and measurements using [a TPM]. This allows customers to gain traceability and accountability for platforms with component-level reporting" [23].

Intel TSC provides the following key features [22]:

- Digitally signed statement of conformance for every platform
- Platform certificates linked to a discrete TPM, providing system-level traceability
- Component-level traceability via a direct platform data file that contains integrated components, including a processor, storage, memory, and add-in cards
- Auto Verify tool that compares the snapshot of the direct platform data taken during manufacturing with a snapshot of the platform components taken at first boot
- Firmware load verification
- Conformity with Defense Federal Acquisition Regulation Supplement (DFARS) 246.870-2 [24]

B.1.2.2 PFR with Protection in Transit (PIT)

In addition to the platform protection, detection, and recovery features, PFR also offers protection in transit (PIT) or supply chain protection. Platform lockdown requires that a password be present in the PFR FPGA as well as a radio frequency (RF) component. The password is removed before platform shipment and must be replaced before the platform will be allowed to power up. With platform firmware sealing, the PFR FPGA computes hashes of platform firmware in the PCH and BMC attached flash devices, including static and dynamic regions, and stores them in an NVRAM space before shipment. Upon delivery, the PFR FPGA

will recompute the hashes and report any mismatches to ensure that the firmware has not been tampered with during system transit.

B.2 Software Runtime Protection Mechanisms

B.2.1 Return Oriented Programming (ROP) and Call/Jump Oriented Programming (COP/JOP) Attacks

B.2.1.1 Intel Control-Flow Enforcement Technology (Intel CET)

Intel Control-Flow Enforcement Technology (Intel CET) is an instruction set extension to implement control flow integrity (CFI) and defend against ROP and COP/JOP style subversion attacks. ROP and similarly COP/JOP have been the prevalent attack methodology for stealth exploit writers targeting vulnerabilities in programs. [25]

Intel CET prevents this class of exploits by providing the following capabilities:

- Shadow stack – return address protection to defend against ROP
- Indirect branch tracking – free branch protection to defend against COP/JOP

"CET introduces a shadow stack system to detect and thwart the stack manipulation required by ROP" [26]. This second stack is used exclusively for control transfer operations and is designed to be protected from application code memory accesses while keeping track of CPU stored copies of the return addresses [27]. "When CET is enabled, a CALL instruction pushes the return address into a shadow stack in addition to its normal behavior of pushing return address into the normal stack (no changes to traditional stack operation). The return instructions (e.g. RET) pops return address from both shadow and traditional stacks, and only transfers control to the popped address if return addresses from both stacks match. […] The page table protections for shadow stack are also designed to protect the integrity of the shadow stack by preventing unintended or malicious switching of shadow stack and/or overflow and underflow of shadow stack." [28]

"CET also adds an indirect branch tracking capability to provide software the ability to restrict COP/JOP attacks." [27] This ENDBRANCH instruction is a new addition to Intel Instruction Set Architecture (ISA). It marks legal targets for an indirect branch or jump, forcing the CPU to generate an exception for unintended or malicious operations [28].

"Intel has been actively collaborating with Microsoft and other industry partners to address control-flow hijacking by using Intel's CET technology to augment the previous software-only control-flow integrity solutions. Intel's CET, when used properly by software, is a big step in helping to prevent exploits from hijacking the control-flow transfer instructions." [28] A security analysis of Intel CET is published in [29].

B.2.2 Address Translation Attacks

B.2.2.1 Intel Hypervisor Managed Linear Address Translation (HLAT)

Hypervisor managed linear address translation (HLAT) is a capability to enable Intel Virtualization Technology (Intel VT-x) based security monitors to enforce runtime protection

and integrity assertions on OS-managed page tables. This helps protect kernel assets, as well as in-band security agents and agent-monitored assets from OS page-table attacks.

"[HLAT] is intended to be used by a Hypervisor/Virtual Machine Monitor (VMM) to enforce guest linear translation (to guest physical mappings). When combined with the existing Extended Page Table (EPT) capability, HLAT enables the VMM to ensure the integrity of combined guest linear translation (mappings and permissions) cached by the processor TLB, via a reduced software TCB managed by the VMM." [30] In this fashion, the VMM-enforced guest translations are more protected from alterations by untrusted system software adversaries. [30]

"This feature is intended to augment the security functionality for a type of Virtual Machine Monitor (VMM) that may use legacy EPT read/write/execute (XWR) permission bits (bits 2:0 of the EPTE) as well as the new user-execute (XU) access bit (bit 10 of the EPTE) to ensure the integrity of code/data resident in guest physical memory assigned to the guest operating system. EPT permissions are also used in these VMMs to isolate memory; for example, to host a Secure Kernel (SK) that can manage security properties for the General Purpose Kernel (GPK). For such usages, it is important that the VMM ensure that the guest linear address mappings which are used by the General Purpose Kernel to refer to the EPT monitored guest physical pages are access-controlled as well." [30]

"VMMs could enforce the integrity of these specific guest linear to guest physical mappings (paging structures) by using legacy EPT permissions to mark the guest physical memory containing the relevant guest paging structures as read-only. The intent of marking these guest paging structures as read-only is to ensure an invalid mapping is not created by guest software. However, such page-table edit control techniques are known to cause very high overheads due to the requirement that the VMM must monitor all paging contexts created by the (Guest) operating system. HLAT enables a VMM to enforce the integrity of guest linear mappings without this high overhead." [30]

HLAT utilizes a processor mechanism that implements an alternate Intel Itanium architecture (IA) paging structure managed in guest physical memory by a Secure Kernel. This paging structure contains guest linear to guest physical translations that the VMM/Secure Kernel wants to enforce.

Additionally, to accommodate legacy page-table monitoring approaches, HLAT defines two new EPT control bits in EPT leaf entries. A "Paging-Write" control bit specifies which guest physical pages hold HLAT or legacy IA paging structures. This allows the processor to use the Paging-Write as permission to perform A/D bit writes, instead of the software W permission in the EPTE. A "Verify Paging-Write" control bit specifies which guest physical pages should only be referenced via translation (guest) paging structures marked as Paging-writable under EPT [30].

B.2.2.2 Intel Supervisor Mode Execution Prevention (SMEP) and Supervisor Mode Access Prevention (SMAP)

Supervisor Mode Execution Prevention (SMEP) and Supervisor Mode Access Prevention (SMAP) are opt-in capabilities that can be used by systems software (such as the kernel) to harden the privilege separation between user-mode and kernel-mode. These capabilities further

enforce the user/supervisor properties specified via address translation mechanisms by mitigating malicious code execution or malicious use of data setup by processes executing in user-mode.

Intel OS Guard, also known as SMEP, helps prevent execution out of untrusted application memory while operating at a more privileged (supervisor) level. "[When] enabled, the operating system will not be allowed to directly execute application code, even speculatively. This makes branch target injection attacks on the OS substantially more difficult by forcing the attacker to find gadgets within the OS code. It is also more difficult for an application to train OS code to jump to an OS gadget. All major operating systems enable SMEP support by default." [31]

SMAP is a security feature that helps prevent unauthorized kernel consumption of data accessible to user space [32]. An enabling SMAP bit in the CR4 control register will cause a page fault to be triggered when there is any attempt to access user-space memory while running in a privileged mode. When access to user space memory is needed by the kernel, a separate AC flag is toggled to allow the required access [33]. "Two new instructions (STAC and CLAC) are provided to manipulate that flag relatively quickly." When the AC flag is set in protection mode under normal operating circumstances, SMAP blocks a whole class of exploits where the kernel is fooled into reading from (or writing to) user-space memory by mistake. SMAP also allows for the early discovery of kernel bugs where developers dereference user space pointers directly from the kernel [33].

B.3 Data Protection and Confidential Computing

B.3.1 Memory Isolation

B.3.1.1 Intel TME and Intel Multi-Key TME (Intel MKTME)

Intel Total Memory Encryption (Intel TME) provides the capability to encrypt the entire physical memory of a system. This capability is typically enabled in the very early stages of the boot process with a small change to the BIOS. Once this change is configured and locked, all data on the external memory buses of a CPU and any additional DIMMs will be encrypted using 128-bit keys utilizing the NIST standard AES-XTS algorithm. The encryption key used for Intel TME uses a hardware random number generator implemented in the Intel CPU, and the keys are not accessible by software or by using external interfaces to the CPU. The architecture is flexible and will support additional memory protection schemes in the future. Intel TME is intended to support unmodified existing system and application software. The overall performance impact of TME is likely to be relatively small and highly dependent on workload.

Intel Multi-Key Total Memory Encryption (Intel MKTME) builds on Intel TME and adds support for multiple encryption keys. The CPU implementation supports a fixed number of encryption keys, and software can configure a CPU to use a subset of available keys. Software manages the use of keys and can use each of the available keys for encrypting any page of the memory. Thus, Intel MKTME allows page granular encryption of memory. By default, Intel MKTME uses the Intel TME encryption key unless explicitly specified by software.

In addition to supporting a CPU-generated ephemeral key (not accessible by software or by using external interfaces to a CPU), Intel MKTME also supports software-provided keys. Software-provided keys are particularly useful when used with nonvolatile memory, when combined with

attestation mechanisms or used with key provisioning services. An OS may be enabled to take additional advantage of the Intel MKTME capability, both in native and virtualized environments. When properly enabled, Intel MKTME is available to each guest OS in a virtualized environment, and the guest OS can take advantage of Intel MKTME in the same ways as a native OS.

B.3.2 Application Isolation

B.3.2.1 Intel Software Guard Extensions (SGX)

Intel Software Guard Extensions (SGX) is a set of instructions that increases the security of application code and data. Developers can partition security-sensitive code and data into an *SGX enclave*, which is executed in a CPU protected region. The developer creates and runs SGX enclaves on server platforms where only the CPU is trusted to provide attestations and protected execution environments for enclave code and data. SGX also provides an enclave remote attestation mechanism. This mechanism allows a remote provider to verify the following [34]:

1. The enclave is running on a real Intel processor inside an SGX enclave.
2. The platform is running at the latest security level (also referred to as the *TCB version*).
3. The enclave's identity is as claimed.
4. The enclave has not been tampered with.

Once all of this is verified, the remote attester can then provision secrets into the enclave. SGX enclave usage is reserved for Ring-3 applications and cannot be used by an OS or BIOS driver/module.

SGX removes the privileged software (e.g., OS, VMM, System Management Mode [SMM], devices) and unprivileged software (e.g., Ring-3 applications, VMs, containers) from the trust boundary of the code running inside the enclave, enhancing security of sensitive application code and data. An SGX enclave trusts the CPU for execution and memory protections. SGX encrypts memory to protect against memory bus snooping and cold boot attacks for enclave code and data in host DRAM. SGX includes ISA instructions that can be used to handle Enclave Page Cache (EPC) page management for creating and initializing enclaves.

SGX relies on the system UEFI BIOS and OS for initial provisioning, resource allocation, and management. However, once an SGX enclave starts execution, it is running in a cryptographically isolated environment separate from the OS and BIOS.

SGX can allow any application (whole or part of) to run inside an enclave and puts application developers in control of their own application security. However, it is recommended that developers keep the SGX code base small, validate the entire system (including software side channel resistance), and follow other secure software development guidelines.

SGX enclaves can be used for applications ranging from protecting private keys and managing security credentials to providing security services. In addition, industry security standards, like European Telecommunications Standards Institute (ETSI) Network Functions Virtualization (NFV) Security (ETSI NFV SEC) [35], have defined and published requirements for Hardware

Mediated Execution Enclaves (HMEEs) for the purposes of NFV, 5G, and edge security. SGX is an HMEE.

B.3.3 VM Isolation

B.3.3.1 Intel Trust Domain Extensions (Intel TDX)

Intel Trust Domain Extensions (Intel TDX) introduces new architectural elements to deploy hardware-isolated VMs called trust domains (TDs). Intel TDX is designed to isolate VMs from the VMM/hypervisor and any non-TD software on the platform to protect TDs from a broad range of software. TDX is built using a combination of Virtual Machine Extensions (VMX) ISA extensions, MKTME technology, and a CPU-attested software module called the TDX-SEAM module. TDX isolates VMs from many hardware threats and most software-based threats, including from the VMM and other CSP software. TDX helps give the cloud tenant control of their own data security and IP protection. TDX does this while maintaining the CSP role of managing resources and cloud platform integrity.

The TDX solution provides the following capabilities to TDs to address the security challenges:

- Memory and CPU state confidentiality and integrity to help keep the sensitive IP and workload data secure from most software-based attacks and many hardware-based attacks. The workload now has a tool that supports excluding the firmware, software, devices, and operators of the cloud platform from the TCB. The workloads can use this tool to foster more secure access to CPU instructions and other CPU features. The workload can have this ability irrespective of the cloud infrastructure used to deploy the workload.
- Remote attestation enables a relying party (either the owner of the workload or a user of the services provided by the workload) to establish that the workload is running on a TDX-enabled platform located within a TD prior to providing that workload data. Remote attestation aims to allow the owners and consumers of the service to digitally determine the version of the TCB they are relying on to help secure their data. The VMM remains the platform resource manager, and TDs should not cause denial of service to the VMM. Defending TDs against denial of service by the VMM is not a goal.

TDX also augments defense of the TD against limited forms of attacks that use physical access to the platform memory, such as offline, DRAM analysis (example: cold-boot attacks), and active attacks of DRAM interfaces, including capturing, modifying, relocating, splicing, and aliasing memory contents [36]. The VMM continues to be the resource manager, and TDs do not have privileges to deny service to the VMM.

B.3.4 Cryptographic Acceleration

B.3.4.1 Intel Advanced Encryption Standard New Instructions (Intel AES-NI)

Intel AES New Instructions (Intel AES-NI) is an encryption instruction set that improves hardware performance of the Advanced Encryption Standard (AES) algorithm and accelerates data encryption. Intel AES-NI consists of seven new instructions that accelerate encryption and decryption and improve key generation and matrix manipulation, all while aiding in carry-less

multiplication. This minimizes application performance concerns inherent in conventional cryptographic processing and helps provide enhanced security by addressing side channel attacks on AES associated with conventional software methods of table lookups [37].

AES is the most widely used standard for protecting network traffic, personal data, and corporate IT infrastructures. By implementing certain intensive sub-steps of the AES algorithm into the hardware, Intel AES-NI strengthens and accelerates execution of the AES application [37].

B.3.4.2 Intel QuickAssist Technology (QAT) with Intel Key Protection Technology (KPT)

Intel QuickAssist Technology (QAT) is a high-performance hardware accelerator for performing cryptographic, security, and compression operations. Applications like VMs, containers, and Function as a Service (FaaS) call Intel QAT using industry-standard OpenSSL, TLS, and Internet Protocol Security (IPsec) interfaces to offload symmetric and asymmetric cryptographic operations. Cloud, multi-tenancy, NFV, edge, and 5G infrastructures and applications are best suited for QAT for all types of workloads, including software-defined networks (SDNs), content delivery networks (CDNs), media, and storage [38].

Intel Key Protection Technology (KPT) helps enable customers to secure their keys to be used with QAT through a bring-your-own-key (BYOK) paradigm. KPT allows customers to deliver their own cryptographic keys to the QAT device in the target platform where their workload is running. KPT-protected keys are never in the clear in host DRAM or in transit. The customers encrypt their workload key (e.g., RSA private key for Nginx) using KPT inside their HSMs. This encrypted workload key is delivered to the target QAT platform, where it is decrypted immediately prior to use. KPT provides key protection at rest, in transit, and while in use [39].

B.3.5 Technology Example Summary

Cloud infrastructure creates improvements in the efficiency, agility, and scalability of data center workloads by abstracting hardware from the application layer. This introduces new security concerns as workloads become multi-tenant, attack surfaces become shared, and infrastructure administrators from the cloud operator gain access to underlying platforms. Isolation techniques provide answers to these concerns by adding protection to VMs, applications, and data during execution, and they represent a crucial layer of a layered security approach for data center security architecture.

Various isolation techniques exist and can be leveraged for different security needs. Full memory isolation defends a platform against physical memory extraction techniques, while the same technology extended with multiple keys allows individual VMs or platform tenants to have uniquely encrypted memory. Future generations of these technologies will allow full memory isolation of VMs, protecting them against malicious infrastructure insiders, multi-tenant malware, and more. Application isolation techniques allow individual applications to create isolated enclaves that require implicit trust in the platform CPU and nothing else and that have the ability to provide proof of the enclave to other applications before data is sent.

B.4 Remote Attestation Services

B.4.1 Intel Security Libraries for the Data Center (ISecL-DC)

Intel Security Libraries for the Data Center (ISecL-DC) is an open-source remote attestation implementation of a set of building blocks that utilize Intel Security features to discover, attest, and enable critical foundation security and confidential computing use-cases. This middleware technology provides a consistent set of application programming interfaces (APIs) for easy integration with cloud management software and security monitoring and enforcement tools. ISecL-DC applies the remote attestation fundamentals described in this section and standard specifications to maintain a platform data collection service and an efficient verification engine to perform comprehensive trust evaluations. These trust evaluations can be used to govern different trust and security policies applied to any given workload, as referenced in the workload scheduling use case in Section 7.2. In future generations, the product will be extended to include TEE attestation to provide assurance and validity of the TEE to enable confidential computing [40].

B.4.2 Technology Summary

Platform attestation provides auditable foundational reports for server firmware and software integrity and can be extended to include the location of other asset tag information stored in a TPM, as well as integrity verification for applications installed on the server. These reports provide visibility into platform security configurations and can be used to control access to data and workloads. Platform attestation is performed on a per-server basis and typically consumed by cloud orchestration or a wide variety of infrastructure management platforms.

TEE attestation provides a mechanism by which a user or application can validate that a genuine TEE enclave with an acceptable TCB is actually being used before releasing secrets or code to the TEE. Formation of a TEE enclave is performed at the application level, and TEE attestations are typically consumed by a user or application requiring evidence of enclave security before passing secrets.

These different attestation techniques serve complementary purposes in a cloud deployment in the data center or at the edge computing facility.

Appendix C—AMD Technology Examples

This section describes a number of AMD technology examples that map back to the key concepts described in the various sections of the document.

C.1 Platform Integrity Verification

C.1.1 AMD Platform Secure Boot (AMD PSB)

AMD Platform Secure Boot (AMD PSB) provides a hardware RoT to authenticate the initial Platform BIOS code during the boot process of the server. Manufacturers of server systems, like OEMs or Original Device Manufacturers (ODMs), enable the functionality of AMD PSB in their manufacturing flow by permanently fusing policy into the silicon.

The OEM or ODM's final BIOS image contains the AMD public key and the OEM BIOS-signing public key (signed with the AMD private key). When a system powers on, the AMD Security Processor (ASP) starts executing the immutable on-chip Boot ROM. It authenticates and loads multi-stage ASP Boot Loaders from SPI/Low Pin Count (LPC) Flash into its internal memory, which initializes the silicon and the system memory.

Once the system memory is initialized, the ASP Boot Loaders load and authenticate the OEM BIOS-signing public key, followed by authenticating the initial BIOS code. Once the verification is successful, ASP releases the x86 core to execute authenticated initial BIOS code. The BIOS can continue CoT for other components by means of UEFI Secure Boot. If PSB authentication fails, the system is forced to shut down.

AMD PSB supports revocation and rollback protection of BIOS images through the OEM BIOS-signing key revision ID and rollback protection.

C.2 Data Protection and Confidential Computing

C.2.1 Memory Isolation: AMD Secure Memory Encryption (SME)/Transparent Memory Encryption (TSME)

AMD Secure Memory Encryption (SME) is a memory encryption technology from AMD which helps protect data in DRAM by encrypting system memory content [41]. When enabled, memory content is encrypted via dedicated hardware in the on-die memory controllers. Each controller includes a high-performance AES engine that encrypts data when it is written to DRAM and decrypts it when read. The encryption of data is done with an encryption key in a mode that utilizes an additional physical address-based tweak to protect against ciphertext block move attacks.

The encryption key used by the AES engine with SME is randomly generated on each system reset and is not visible to any software running on the CPU cores. This key is managed entirely by the AMD Secure Processor (AMD-SP) that functions as a dedicated security subsystem integrated within the AMD System-on-Chip (SOC). The key is generated using the onboard NIST SP 800-90 compliant hardware random number generator and is stored in dedicated hardware registers where it is never exposed outside the SOC in the clear.

Two modes of memory encryption are supported for various use cases. The simplest mode is Transparent Secure Memory Encryption (TSME), which is a BIOS option and enables memory encryption automatically on all memory accesses. TSME works in the background and requires no software interaction. Another supported mode is the OS-managed Secure Memory Encryption (SME) mode in which individual pages of memory may be marked for encryption via CPU page tables. SME provides additional flexibility if only a subset of memory needs to be encrypted but does require appropriate software support.

Encrypted memory provides strong protection against cold boot, DRAM interface snooping, and similar types of attacks.

C.2.2 VM Isolation: AMD Secure Encrypted Virtualization (SEV)

The AMD Secure Encrypted Virtualization (SEV) feature is designed to isolate VMs from the hypervisor. When SEV is enabled, individual VMs are encrypted with an AES encryption key. When a component such as the hypervisor attempts to read memory inside a guest, it is only able to see the data in its encrypted form. This provides strong cryptographic isolation between the VMs, as well as between the VMs and the hypervisor.

To protect SEV-enabled guests, the SEV firmware assists in the enforcement of three main security properties: authenticity of the platform, attestation of a launched guest, and confidentiality of the guest's data.

Authenticating the platform prevents malicious software or a rogue device from masquerading as a legitimate platform. The authenticity of the platform is proven with its identity key. This key is signed by AMD to demonstrate that the platform is an authentic AMD platform with SEV capabilities.

Attestation of the guest launch proves to guest owners that their guests securely launched with SEV enabled. A signature of various components of the SEV-related guest state, including initial contents of memory, is provided by the firmware to the guest owner to verify that the guest is in the expected state. With this attestation, a guest owner can ensure that the hypervisor did not interfere with the initialization of SEV before transmitting confidential information to the guest.

Confidentiality of the guest is accomplished by encrypting memory with a memory encryption key that only the SEV firmware knows. The SEV management interface does not allow the memory encryption key or any other secret SEV state to be exported outside of the firmware without properly authenticating.

AMD SEV has two additional modes:

- SEV With Encrypted State (SEV-ES): This mode encrypts and protects VM registers from being read or modified by a malicious hypervisor or VM [42].
- SEV with Secure Nested Paging (SEV-SNP): This mode adds strong memory integrity protection to help prevent malicious hypervisor-based attacks like data replay and memory remapping. [43]

Appendix D—Acronyms and Abbreviations

Selected acronyms and abbreviations used in this paper are defined below.

AC RAM	Authenticated Code Random Access Memory
ACM	Authenticated Code Module
AES	Advanced Encryption Standard
AMD PSB	AMD Platform Secure Boot
AMD-SP	AMD Secure Processor
API	Application Programming Interface
AS	Attestation Service
ASP	AMD Security Processor
BIOS	Basic Input/Output System
BMC	Board Management Controller
BYOK	Bring Your Own Key
CA	Certificate Authority
CDN	Content Delivery Network
CFI	Control Flow Integrity
COP	Call Oriented Programming
CoT	Chain of Trust
CPU	Central Processing Unit
CRD	Custom Resource Definition
CRI	Container Runtime Interface
CRTM	Core Root of Trust for Measurement
CRTV	Core Root of Trust for Verification
CSK	Code Signing Key
CSP	Cloud Service Provider
DCRTM	Dynamic Core Root of Trust for Measurement
DFARS	Defense Federal Acquisition Regulation Supplement
DIMM	Dual In-Line Memory Module
DRAM	Dynamic Random-Access Memory
EPC	Enclave Page Cache
EPT	Extended Page Table
ETSI	European Telecommunications Standards Institute
ETSI NFV SEC	European Telecommunications Standards Institute Network Functions Virtualization Security
FaaS	Function as a Service

FIPS	Federal Information Processing Standard
FOIA	Freedom of Information Act
FPGA	Field Programmable Gate Array
GDPR	General Data Protection Regulation
GPK	General Purpose Kernel
HASP	Hardware and Architectural Support for Security and Privacy
HIPAA	Health Insurance Portability and Accountability Act
HLAT	Hypervisor Managed Linear Address Translation
HMEE	Hardware Mediated Execution Enclave
HSM	Hardware Security Module
IA	Intel Itanium Architecture
IBB	Initial Boot Block
Intel AES-NI	Intel Advanced Encryption Standard New Instructions
Intel CET	Intel Control-Flow Enforcement Technology
Intel MKTME	Intel Multi-Key Total Memory Encryption
Intel TDX	Intel Trust Domain Extensions
Intel TME	Intel Total Memory Encryption
Intel TSC	Intel Transparent Supply Chain
Intel VT-x	Intel Virtualization Technology
IoT	Internet of Things
IPsec	Internet Protocol Security
IR	NIST Interagency or Internal Report
ISA	Instruction Set Architecture
ISecL-DC	Intel Security Libraries for the Data Center
IT	Information Technology
ITL	Information Technology Laboratory
JOP	Jump Oriented Programming
KEK	Key Exchange Key
KMIP	Key Management Interoperability Protocol
KMS	Key Management Service
KPT	Key Protection Technology
LCP	Launch Control Policy
LPC	Low Pin Count
ME	Manageability Engine
MOK	Machine Owner Key

NCCoE	National Cybersecurity Center of Excellence
NFV	Network Functions Virtualization
NIST	National Institute of Standards and Technology
NVRAM	Non-Volatile Random-Access Memory
ODM	Original Design Manufacturer
OEM	Original Equipment Manufacturer
OS	Operating System
PCH	Platform Controller Hub
PCIE	Peripheral Component Interconnect Express
PCR	Platform Configuration Register
PFM	Platform Firmware Manifest
PFR	Platform Firmware Resilience
PIT	Protection in Transit
PK	Platform Key
QAT	QuickAssist Technology
RAM	Random Access Memory
RCE	Remote Code Execution
RF	Radio Frequency
RK	Root Key
RNG	Random Number Generator
ROM	Read-Only Memory
ROP	Return Oriented Programming
RoT	Root of Trust
RTD	Root of Trust for Detection
RTRec	Root of Trust for Recovery
RTU	Root of Trust for Update
RW	Read/Write
RWX	Read/Write/Execute
SB	UEFI Secure Boot
SCRTM	Static Core Root of Trust for Measurement
SDN	Software Defined Network
SEV	Secured Encrypted Virtualization
SEV-ES	Secured Encrypted Virtualization with Encrypted State
SEV-SNP	Secured Encrypted Virtualization with Secured Nested Paging
SGX	Software Guard Extensions
SINIT ACM	Secure Initialization Authenticated Code Module

SK	Secure Kernel
SMAP	Supervisor Mode Access Prevention
SMBus	System Management Bus
SME	Secure Memory Encryption
SMEP	Supervisor Mode Execution Prevention
SMM	System Management Mode
SOC	System-on-Chip
SP	Special Publication
SPI	Serial Peripheral Interface
SPS FW	Server Platform Services Firmware
TCB	Trusted Compute Base, Trusted Computing Base
TD	Trust Domain
TEE	Trusted Execution Environment
TLB	Translation Lookaside Buffer
TLS	Transport Layer Security
TPM	Trusted Platform Module
TSME	Transparent Memory Encryption
TXT	Trusted Execution Technology
UEFI	Unified Extensible Firmware Interface
USB	Universal Serial Bus
VM	Virtual Machine
VMM	Virtual Machine Manager, Virtual Machine Monitor
VMX	Virtual Machine Extensions
XTS	xor-encrypt-xor (XEX) Based Tweaked-Codebook Mode with Ciphertext Stealing
XU	User-Execute
XWR	Read/Write/Execute

1407

Appendix E—Glossary

Asset Tag	Simple key value attributes that are associated with a platform (e.g., location, company name, division, or department).
Chain of Trust (CoT)	A method for maintaining valid trust boundaries by applying a principle of transitive trust, where each software module in a system boot process is required to measure the next module before transitioning control.
Confidential Computing	Hardware-enabled features to isolate and process encrypted data in memory so that the data is at less risk of exposure and compromise from concurrent workloads or the underlying system and platform.
Cryptographic Accelerator	A specialized separate coprocessor chip from the main processing unit where cryptographic tasks are offloaded to for performance benefits.
Hardware-Enabled Security	Security with its basis in the hardware platform.
Platform Trust	An assurance in the integrity of the underlying platform configuration, including hardware, firmware, and software.
Root of Trust (RoT)	A starting point that is implicitly trusted.
Shadow Stack	A parallel hardware stack that applications can utilize to store a copy of return addresses that are checked against the normal program stack on return operations.
Trusted Execution Environment (TEE)	An area or enclave protected by a system processor.