

Withdrawn Draft

Warning Notice

The attached draft document has been withdrawn, and is provided solely for historical purposes. It has been superseded by the document identified below.

Withdrawal Date October 27, 2021

Original Release Date May 20, 2021

Superseding Document

Status 2nd Public Draft (2PD)

Series/Number NIST Interagency or Internal Report 8320

Title Hardware-Enabled Security: Enabling a Layered Approach to Platform Security for Cloud and Edge Computing Use Cases

Publication Date October 2021

DOI <https://doi.org/10.6028/NIST.IR.8320-draft2>

CSRC URL <https://csrc.nist.gov/publications/detail/nistir/8320/draft>

Additional Information

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>

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*

60 National Institute of Standards and Technology Interagency or Internal Report 8320
61 58 pages (May 2021)

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

64 Certain commercial entities, equipment, or materials may be identified in this document in order to describe an
65 experimental procedure or concept adequately. Such identification is not intended to imply recommendation or
66 endorsement by NIST, nor is it intended to imply that the entities, materials, or equipment are necessarily the best
67 available for the purpose.

68 There may be references in this publication to other publications currently under development by NIST in accordance
69 with its assigned statutory responsibilities. The information in this publication, including concepts and methodologies,
70 may be used by federal agencies even before the completion of such companion publications. Thus, until each
71 publication is completed, current requirements, guidelines, and procedures, where they exist, remain operative. For
72 planning and transition purposes, federal agencies may wish to closely follow the development of these new
73 publications by NIST.

74 Organizations are encouraged to review all draft publications during public comment periods and provide feedback to
75 NIST. Many NIST cybersecurity publications, other than the ones noted above, are available at
76 <https://csrc.nist.gov/publications>.

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

78 National Institute of Standards and Technology
79 Attn: Applied Cybersecurity Division, Information Technology Laboratory
80 100 Bureau Drive (Mail Stop 2000) Gaithersburg, MD 20899-2000
81 Email: hwsec@nist.gov

82 All comments are subject to release under the Freedom of Information Act (FOIA).

83

84

Reports on Computer Systems Technology

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

93

Abstract

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

102

Keywords

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

106

Disclaimer

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

110

111

Acknowledgments

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

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

119

Audience

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

125

Trademark Information

126 All registered trademarks or trademarks belong to their respective organizations.

127

128

Call for Patent Claims

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

135

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

138

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

141

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

145

146 i. under reasonable terms and conditions that are demonstrably free of any unfair
147 discrimination; or

148 ii. without compensation and under reasonable terms and conditions that are
149 demonstrably free of any unfair discrimination.

150

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

156

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

159

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

161

162 **Table of Contents**

163 **1 Introduction 1**

164 **2 Hardware Platform Security Overview 3**

165 **3 Platform Integrity Verification 5**

166 3.1 Hardware Security Module (HSM) 5

167 3.2 The Chain of Trust (CoT) 6

168 3.3 Supply Chain Protection 7

169 **4 Software Runtime Protection Mechanisms 8**

170 4.1 Return Oriented Programming (ROP) and Call/Jump Oriented Programming

171 (COP/JOP) Attacks 8

172 4.2 Address Translation Attacks 8

173 **5 Data Protection and Confidential Computing 10**

174 5.1 Memory Isolation 10

175 5.2 Application Isolation 11

176 5.3 VM Isolation 11

177 5.4 Cryptographic Acceleration 11

178 **6 Remote Attestation Services 13**

179 6.1 Platform Attestation 13

180 6.2 TEE Attestation 15

181 **7 Cloud Use Case Scenarios Leveraging Hardware-Enabled Security 17**

182 7.1 Visibility to Security Infrastructure 17

183 7.2 Workload Placement on Trusted Platforms 17

184 7.3 Asset Tagging and Trusted Location 19

185 7.4 Workload Confidentiality 20

186 7.5 Protecting Keys and Secrets 22

187 **8 Next Steps 24**

188 **References 25**

189 **List of Appendices**

190

191 **Appendix A— Vendor-Agnostic Technology Examples 29**

192 A.1 Platform Integrity Verification 29

193 A.1.1 UEFI Secure Boot (SB) 29

194 **Appendix B— Intel Technology Examples 31**

195	B.1 Platform Integrity Verification	31
196	B.1.1 The Chain of Trust (CoT).....	31
197	B.1.2 Supply Chain Protection	35
198	B.2 Software Runtime Protection Mechanisms	36
199	B.2.1 Return Oriented Programming (ROP) and Call/Jump Oriented	
200	Programming (COP/JOP) Attacks.....	36
201	B.2.2 Address Translation Attacks.....	36
202	B.3 Data Protection and Confidential Computing	38
203	B.3.1 Memory Isolation	38
204	B.3.2 Application Isolation.....	39
205	B.3.3 VM Isolation.....	40
206	B.3.4 Cryptographic Acceleration	40
207	B.3.5 Technology Example Summary.....	41
208	B.4 Remote Attestation Services.....	42
209	B.4.1 Intel Security Libraries for the Data Center (ISecL-DC).....	42
210	B.4.2 Technology Summary.....	42
211	Appendix C— AMD Technology Examples.....	43
212	C.1 Platform Integrity Verification	43
213	C.1.1 AMD Platform Secure Boot (AMD PSB).....	43
214	C.2 Data Protection and Confidential Computing	43
215	C.2.1 Memory Isolation: AMD Secure Memory Encryption	
216	(SME)/Transparent Memory Encryption (TSME).....	43
217	C.2.2 VM Isolation: AMD Secure Encrypted Virtualization (SEV)	44
218	Appendix D— Acronyms and Abbreviations	45
219	Appendix E— Glossary.....	49
220		
221		
	List of Figures	
222	Figure 1: Notional Example of Remote Attestation Service.....	14
223	Figure 2: Notional Example of TEE Attestation Flow.....	16
224	Figure 3: Notional Example of Orchestrator Platform Labeling	18
225	Figure 4: Notional Example of Orchestrator Scheduling.....	19
226	Figure 5: Notional Example of Key Brokerage	20
227	Figure 6: Notional Example of Workload Image Encryption	21

228 Figure 7: Notional Example of Workload Decryption 22

229 Figure 8: Firmware and Software Coverage of Existing Chain of Trust Technologies .. 34

230

231

232 1 Introduction

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

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

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

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

- 269
- Appendix E provides a glossary of selected terms used in the report.

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

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

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

277

278 **2 Hardware Platform Security Overview**

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

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

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

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

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

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

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

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

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

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

351 In addition, there are other hardware platform security technologies that can protect data at rest,
352 in transit, and in use by providing hardware-accelerated disk encryption or encryption-based
353 memory isolation. Many of these capabilities can help mitigate threats from speculative
354 execution and side-channel attacks. By using hardware to perform these tasks, the attack surface
355 is mitigated, preventing direct access or modification of the required firmware. Isolating these
356 encryption mechanisms to dedicated hardware can allow performance to be addressed and
357 enhanced separately from other system processes as well. An example of hardware-based
358 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

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

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

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

418 **3.2 The Chain of Trust (CoT)**

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

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

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

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

- 441 • [UEFI Secure Boot \(SB\)](#)
- 442 • [Intel Trusted Execution Technology \(TXT\)](#)
- 443 • [Intel Boot Guard](#)
- 444 • [Intel Platform Firmware Resilience \(PFR\)](#)
- 445 • [Intel Technology Example Summary](#)
- 446 • [AMD Platform Secure Boot](#)

447 **3.3 Supply Chain Protection**

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

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

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

- 462 • [Intel Transparent Supply Chain \(TSC\)](#)
- 463 • [Intel PFR with Protection in Transit \(PIT\)](#)

464

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.

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

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

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

- 520 • [Intel Hypervisor Managed Linear Address Translation \(HLAT\)](#)
- 521 • [Intel Supervisor Mode Execution Prevention \(SMEP\) and Supervisor Mode Access](#)
522 [Prevention \(SMAP\)](#)
- 523 • [AMD Supervisor Mode Execution Prevention \(SMEP\) and Supervisor Mode Access](#)
524 [Prevention \(SMAP\)](#)

525

526 **5 Data Protection and Confidential Computing**

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

536 While containers can provide a level of convenience, vulnerabilities in the kernel space and
537 shared layers can be susceptible to widespread exploitation, making security for the underlying
538 platform even more important. With the need for additional protection in the virtualized
539 workspace, an emphasis has been placed on encrypting data both at rest and while in use. *At-rest*
540 encryption provides protection for data on disk. This typically refers to an unmounted data store
541 and protects against threats such as the physical removal of a disk drive. Protecting and securing
542 cloud data while *in use*, also referred to as *confidential computing*, utilizes hardware-enabled
543 features to isolate and process encrypted data in memory so that the data is at less risk of
544 exposure and compromise from concurrent workloads or the underlying system and platform
545 [14]. This section describes technologies that can be leveraged for providing confidential
546 computing for cloud and edge.

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

556 **5.1 Memory Isolation**

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

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

- 566 • [Intel TME and Multi-Key Total Memory Encryption \(MKTME\)](#)
- 567 • [AMD Secure Memory Encryption \(SME\)/Transparent Memory Encryption \(TSME\)](#)

568 **5.2 Application Isolation**

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

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

- 579 • [Intel Software Guard Extensions \(SGX\)](#)

580 **5.3 VM Isolation**

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

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

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

- 597 • [Intel Trust Domain Extensions \(TDX\)](#)
- 598 • [AMD Secure Encrypted Virtualization \(SEV\)](#)

599 **5.4 Cryptographic Acceleration**

600 Encryption is quickly becoming more widespread in data center applications as industry adopts
601 more standards and guidelines regarding the sensitivity of consumer data and intellectual
602 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

612 **6 Remote Attestation Services**

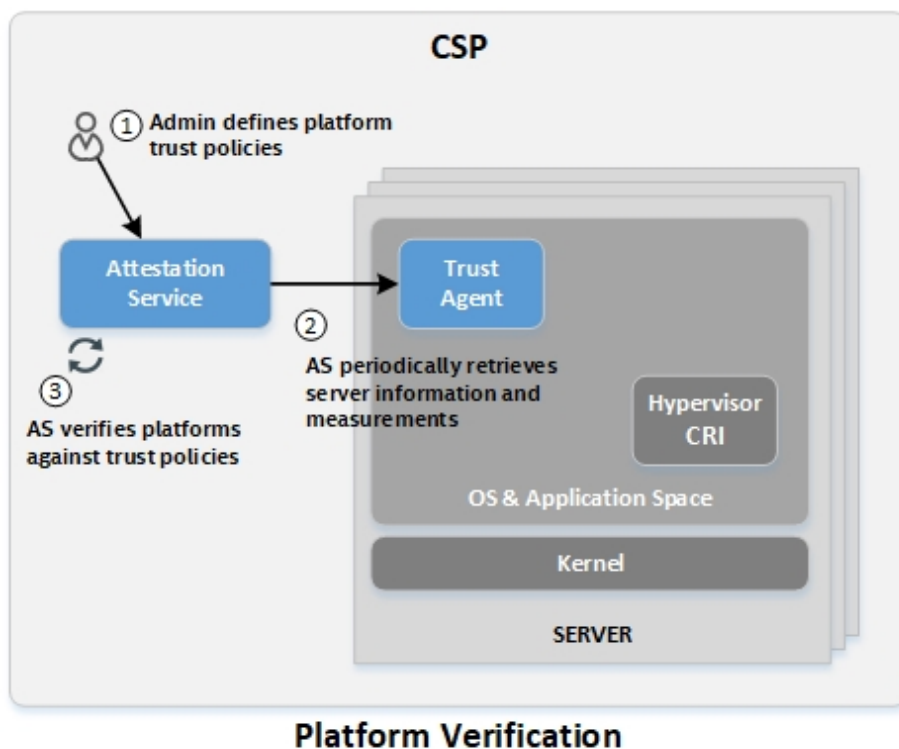
613 Measuring a server's firmware/configuration and extending these measurements to a hardware
614 interface can help keep track of which firmware is running on a platform. Some platform
615 integrity technologies can even perform local attestation and enforcement of firmware and
616 configuration on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

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

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

639 **6.1 Platform Attestation**

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



646
647

Figure 1: Notional Example of Remote Attestation Service

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

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

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

- 668
- [Intel Security Libraries for the Data Center \(ISecL-DC\)](#)

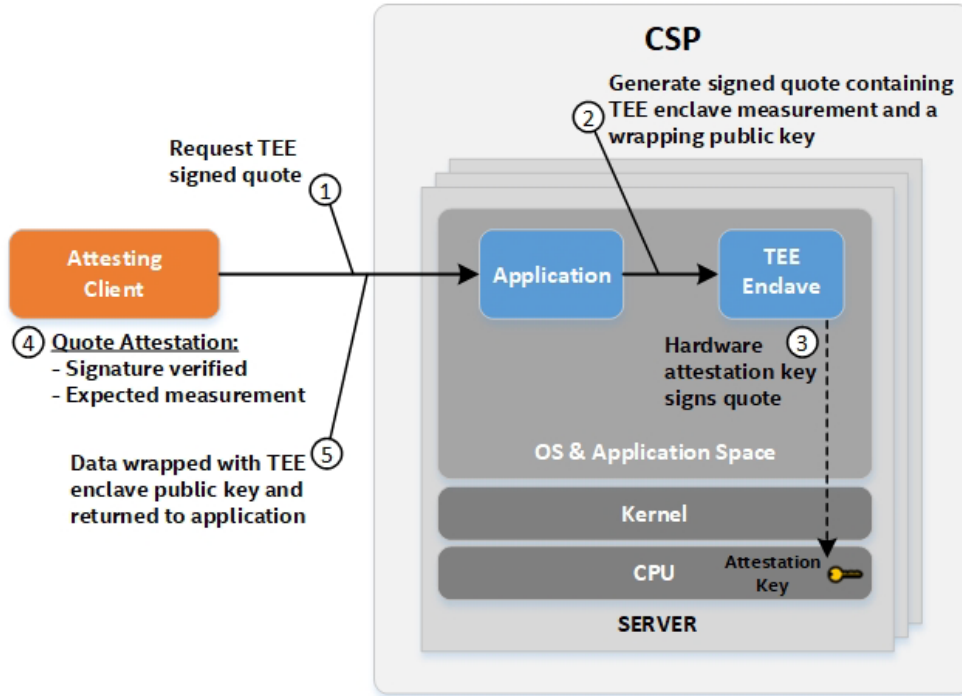
669 6.2 TEE Attestation

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

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

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

691 Figure 2 shows an example TEE attestation flow.



692

693

694

Figure 2: Notional Example of TEE Attestation Flow

695 **7 Cloud Use Case Scenarios Leveraging Hardware-Enabled Security**

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

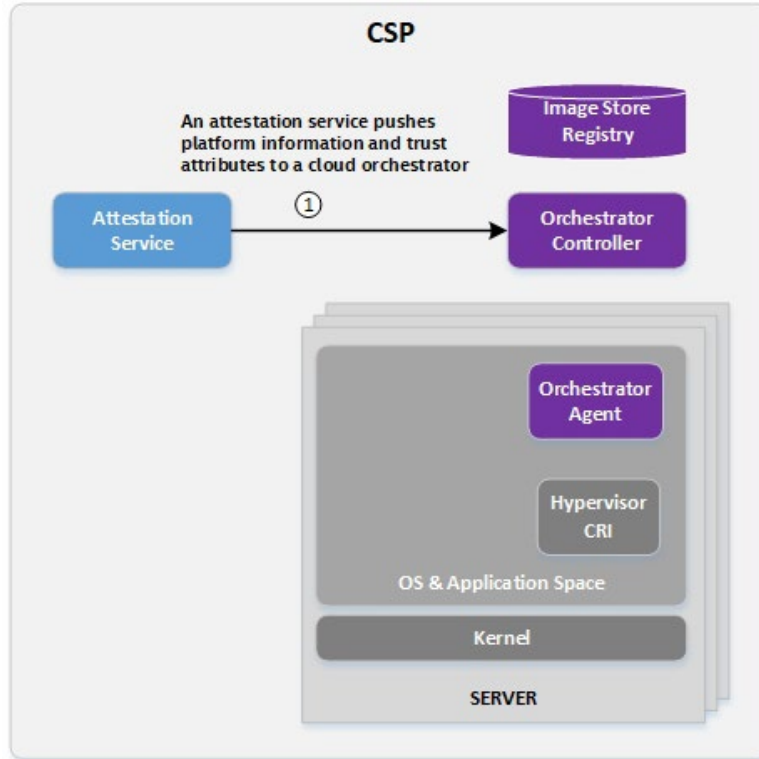
699 **7.1 Visibility to Security Infrastructure**

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

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

712 **7.2 Workload Placement on Trusted Platforms**

713 Platform information and verified firmware/configuration measurements retained within an
714 attestation service can be used for policy enforcement in countless use cases. One example is
715 orchestration scheduling. Cloud orchestrators, such as Kubernetes and OpenStack, provide the
716 ability to label server nodes in their database with key value attributes. The attestation service
717 can publish trust and informational attributes to orchestrator databases for use in workload
718 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.

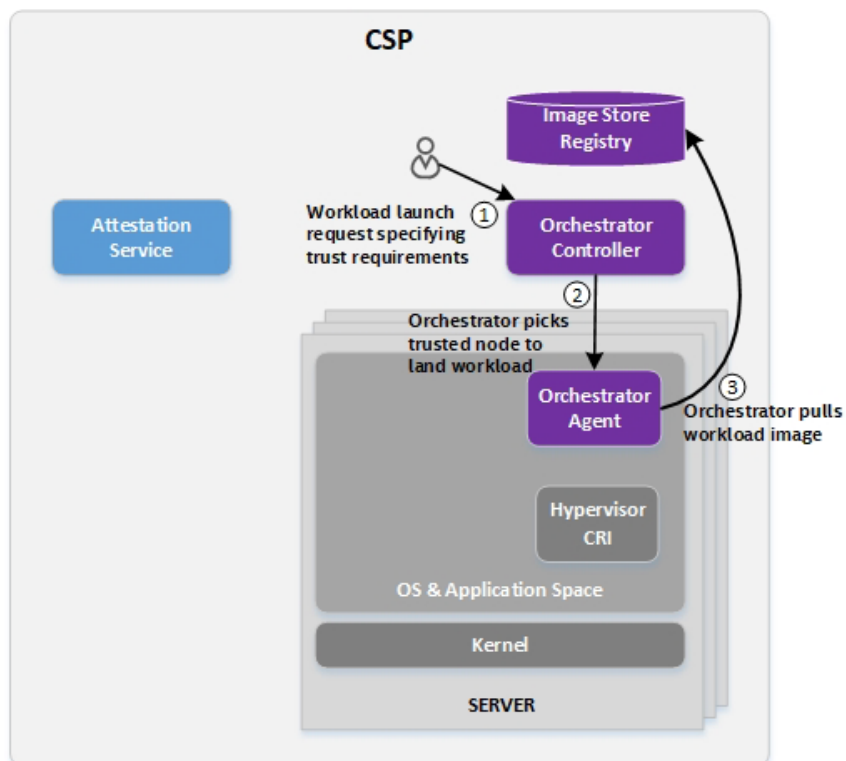


Figure 4: Notional Example of Orchestrator Scheduling

732

733

734 7.3 Asset Tagging and Trusted Location

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

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

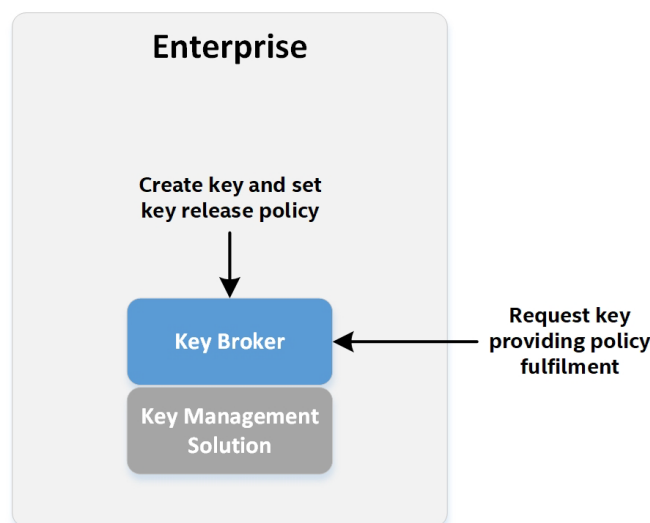
753 Outside of specific regulatory requirements, an organization may wish to segregate workloads by
 754 department. For example, human resources and finance information could be restricted to

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

763 7.4 Workload Confidentiality

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

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

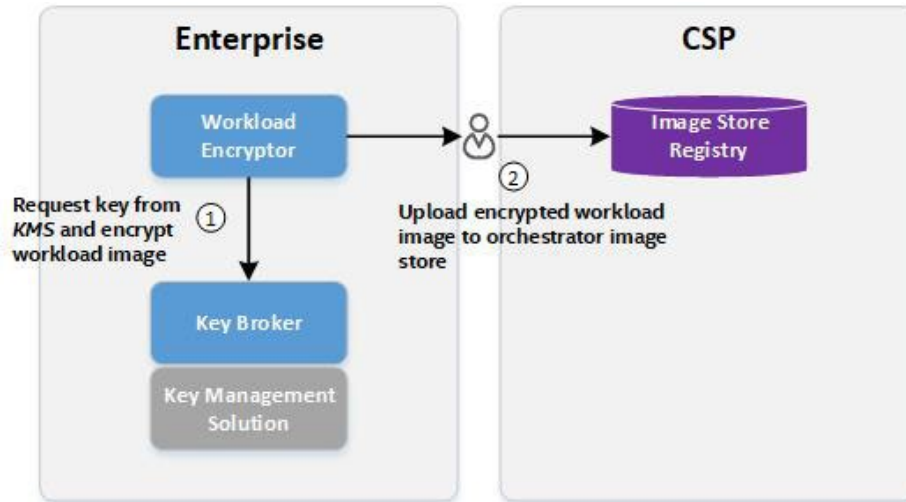


782

783

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.

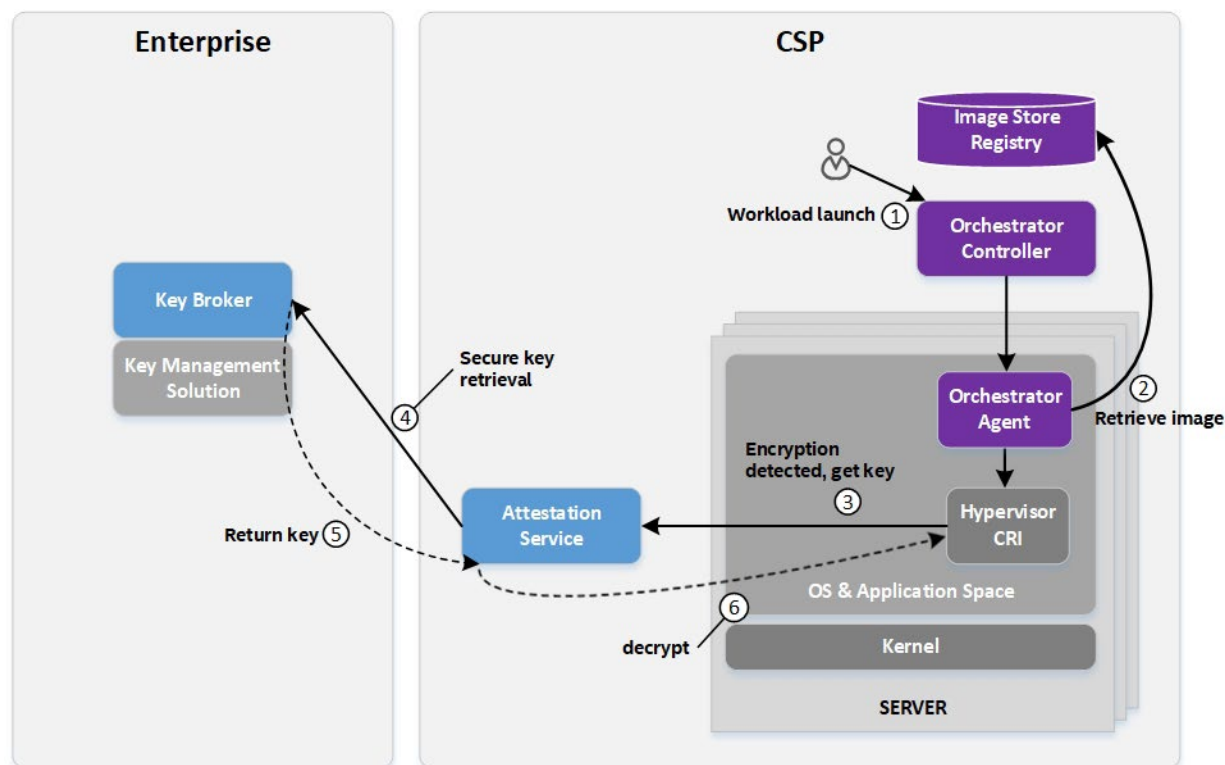


787

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.



795

796

Figure 7: Notional Example of Workload Decryption

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

805 7.5 Protecting Keys and Secrets

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

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

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

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

830

8 Next Steps

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

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

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

841

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-sec/>
- [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>

844 Appendix A—Vendor-Agnostic Technology Examples

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

847 A.1 Platform Integrity Verification

848 A.1.1 UEFI Secure Boot (SB)

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

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

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

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

878 A key part of the shim design is to allow users to control their own systems. The distro CA key is
879 built into the shim binary itself, but there is also an extra database of keys that can be managed
880 by the user—the so-called *Machine Owner Key* (MOK). Keys can be added and removed in the
881 MOK list by the user, entirely separate from the distro CA key. The *mokutil* utility can be used
882 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

890 Appendix B—Intel Technology Examples

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

893 B.1 Platform Integrity Verification

894 B.1.1 The Chain of Trust (CoT)

895 B.1.1.1 Intel Trusted Execution Technology (TXT)

896 Intel Trusted Execution Technology (TXT) in conjunction with a TPM provides a hardware RoT
897 available on Intel server and client platforms that enables “security capabilities such as measured
898 launch and protected execution” [19]. TXT utilizes *authenticated code modules (ACMs)* that
899 measure various pieces of the CoT during boot time and extend them to the platform TPM
900 [2][19]. TXT’s ACMs are chipset-specific signed binaries that are called to perform functions
901 required to enable the TXT environment. An ACM is loaded into and executed from within the
902 CPU cache in an area referred to as the *authenticated code RAM (AC RAM)*. CPU microcode,
903 which acts as the *core root of trust for measurement (CRTM)*, authenticates the ACM by
904 verifying its included digital signature against a manufacturer public key with its digest hard-
905 coded within the chipset. The ACM code, loaded into protected memory inside the processor,
906 performs various tests and verifications of chipset and processor configurations.

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

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

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

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

932 **B.1.1.2 Intel Boot Guard**

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

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

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

950 **B.1.1.3 Intel Platform Firmware Resilience (PFR)**

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

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

- 961 • **Protection:** Mechanisms for ensuring that platform firmware code and critical data
962 remain in a state of integrity and are protected from corruption, such as the process for
963 ensuring the authenticity and integrity of firmware updates
- 964 • **Detection:** Mechanisms for detecting when platform firmware code and critical data have
965 been corrupted
- 966 • **Recovery:** Mechanisms for restoring platform firmware code and critical data to a state
967 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.

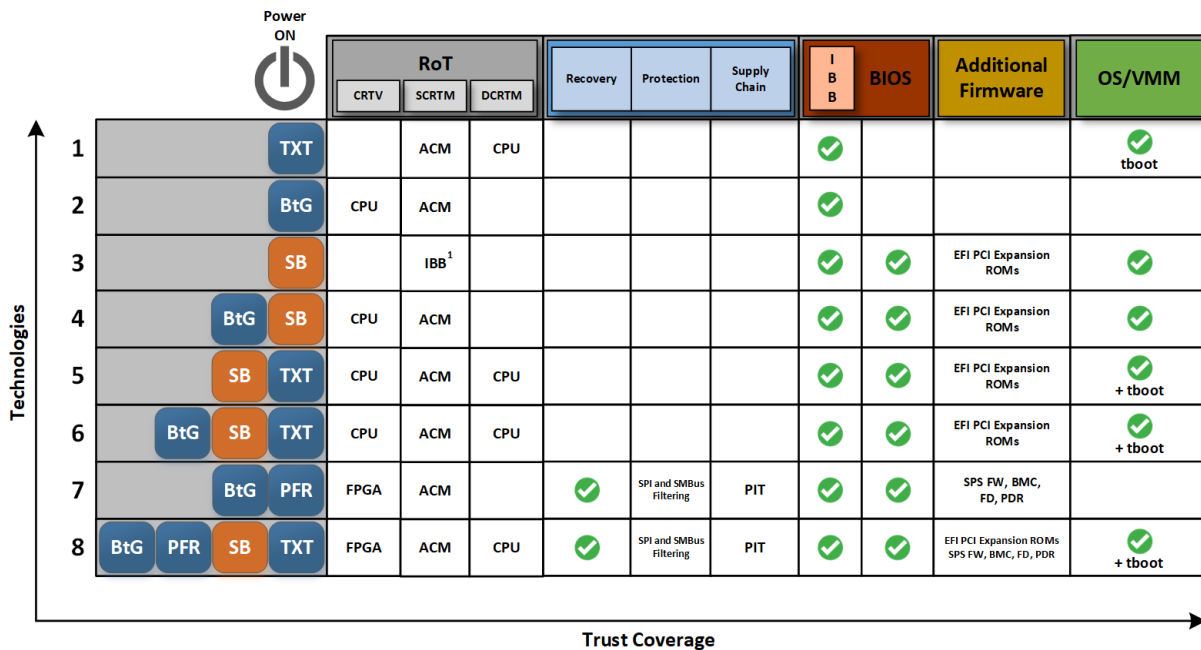
995 The PFR FPGA RoT leverages a key hierarchy to authenticate data structures residing in SPI
996 flash. The key hierarchy is based on a provisioned Root Key (RK) stored in the NVRAM of the
997 FPGA RoT and a Code Signing Key (CSK) structure, which is endorsed by the RK, stored in the
998 SPI flash, 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

1009 the other platform components for writing in order to provide an area to place digitally signed
 1010 and compressed update capsules, which are then verified by the PFR FPGA RoT before being
 1011 committed to the Active or Recovery regions. The Recovery copy can be updated in T-1 mode
 1012 once the PFR FPGA has verified the digital signature of the update capsule and confirmed that
 1013 the recovery image candidate is bootable.

1014 **B.1.1.4 Technology Example Summary**

1015 There are several technologies that provide different levels of platform integrity and trust.
 1016 Individual technologies do not provide a complete CoT. When used in combination, they can
 1017 provide comprehensive coverage all the way up to the OS and VMM layer. Figure 8 outlines the
 1018 firmware and software coverage of each existing CoT technology example.



1019 ¹ IBB is meant to describe the portion of BIOS which performs the first measurement

1020 **Figure 8: Firmware and Software Coverage of Existing Chain of Trust Technologies**

1021 Figure 8 identifies the components of each technology that make up the RoT in their own
 1022 respective chains and also shows a rough outline of the firmware and software coverage of each
 1023 technology.

1024 Because many technologies are available, it can be difficult to decide on the correct combination
 1025 for deployment. Figure 8 illustrates the possible combinations of technologies that extend
 1026 measurements to a TPM for platform integrity attestation. Note that each combination includes at
 1027 least one hardware technology to ensure an RoT implementation. A complementary option for
 1028 extending the CoT up through the OS can also be provided. Including only the hardware
 1029 technologies would break the CoT by supplying integrity measurements for only pre-OS
 1030 firmware. Using only UEFI Secure Boot will use firmware as the RoT that does not have
 1031 hardware security protections and is much more susceptible to attack. By enabling both parts, the
 1032 CoT can be extended from a hardware RoT into the OS and beyond.

1033 These combinations will help ensure that appropriate measurements are extended to a TPM for
1034 integrity attestation and can prevent a server from booting if specific security modules are
1035 compromised. The attestation mechanisms provided by these technologies give cryptographic
1036 proof of the integrity of measured components, which can be used to provide visibility into
1037 platform security configurations and prove integrity. Note the combination of UEFI SB with
1038 TXT in Figure 8. This combination provides the UEFI SB signature verification capability on top
1039 of the tboot integrity measurement in the OS/VMM layer.

1040 In addition to attestation, PFR provides both additional verification of platform firmware and
1041 adds automatic recovery of compromised firmware to known good versions. PFR works with any
1042 combination of CoT technologies, providing a defense and resilience against firmware attack
1043 vectors. Combining a hardware-based firmware resilience technology like PFR with a hardware-
1044 based CoT configuration is part of a layered security strategy.

1045 **B.1.2 Supply Chain Protection**

1046 **B.1.2.1 Intel Transparent Supply Chain (TSC)**

1047 “Intel Transparent Supply Chain (TSC) is a set of policies and procedures implemented at ODM
1048 factories that enable end-users to validate where and when every component of a platform was
1049 manufactured” [22]. “Intel TSC tools allow platform manufacturers to bind platform information
1050 and measurements using [a TPM]. This allows customers to gain traceability and accountability
1051 for platforms with component-level reporting” [23].

1052 Intel TSC provides the following key features [22]:

- 1053 • Digitally signed statement of conformance for every platform
- 1054 • Platform certificates linked to a discrete TPM, providing system-level traceability
- 1055 • Component-level traceability via a direct platform data file that contains integrated
1056 components, including a processor, storage, memory, and add-in cards
- 1057 • Auto Verify tool that compares the snapshot of the direct platform data taken during
1058 manufacturing with a snapshot of the platform components taken at first boot
- 1059 • Firmware load verification
- 1060 • Conformity with Defense Federal Acquisition Regulation Supplement (DFARS)
1061 246.870-2 [24]

1062 **B.1.2.2 PFR with Protection in Transit (PIT)**

1063 In addition to the platform protection, detection, and recovery features, PFR also offers
1064 protection in transit (PIT) or supply chain protection. Platform lockdown requires that a
1065 password be present in the PFR FPGA as well as a radio frequency (RF) component. The
1066 password is removed before platform shipment and must be replaced before the platform will be
1067 allowed to power up. With platform firmware sealing, the PFR FPGA computes hashes of
1068 platform firmware in the PCH and BMC attached flash devices, including static and dynamic
1069 regions, and stores them in an NVRAM space before shipment. Upon delivery, the PFR FPGA

1070 will recompute the hashes and report any mismatches to ensure that the firmware has not been
1071 tampered with during system transit.

1072 **B.2 Software Runtime Protection Mechanisms**

1073 **B.2.1 Return Oriented Programming (ROP) and Call/Jump Oriented Programming** 1074 **(COP/JOP) Attacks**

1075 **B.2.1.1 Intel Control-Flow Enforcement Technology (Intel CET)**

1076 Intel Control-Flow Enforcement Technology (Intel CET) is an instruction set extension to
1077 implement control flow integrity (CFI) and defend against ROP and COP/JOP style subversion
1078 attacks. ROP and similarly COP/JOP have been the prevalent attack methodology for stealth
1079 exploit writers targeting vulnerabilities in programs. [25]

1080 Intel CET prevents this class of exploits by providing the following capabilities:

- 1081 • Shadow stack – return address protection to defend against ROP
- 1082 • Indirect branch tracking – free branch protection to defend against COP/JOP

1083 “CET introduces a shadow stack system to detect and thwart the stack manipulation required by
1084 ROP” [26]. This second stack is used exclusively for control transfer operations and is designed
1085 to be protected from application code memory accesses while keeping track of CPU stored
1086 copies of the return addresses [27]. “When CET is enabled, a CALL instruction pushes the return
1087 address into a shadow stack in addition to its normal behavior of pushing return address into the
1088 normal stack (no changes to traditional stack operation). The return instructions (e.g. RET) pops
1089 return address from both shadow and traditional stacks, and only transfers control to the popped
1090 address if return addresses from both stacks match. [...] The page table protections for shadow
1091 stack are also designed to protect the integrity of the shadow stack by preventing unintended or
1092 malicious switching of shadow stack and/or overflow and underflow of shadow stack.” [28]

1093 “CET also adds an indirect branch tracking capability to provide software the ability to restrict
1094 COP/JOP attacks.” [27] This ENDBRANCH instruction is a new addition to Intel Instruction Set
1095 Architecture (ISA). It marks legal targets for an indirect branch or jump, forcing the CPU to
1096 generate an exception for unintended or malicious operations [28].

1097 “Intel has been actively collaborating with Microsoft and other industry partners to address
1098 control-flow hijacking by using Intel’s CET technology to augment the previous software-only
1099 control-flow integrity solutions. Intel’s CET, when used properly by software, is a big step in
1100 helping to prevent exploits from hijacking the control-flow transfer instructions.” [28] A security
1101 analysis of Intel CET is published in [29].

1102 **B.2.2 Address Translation Attacks**

1103 **B.2.2.1 Intel Hypervisor Managed Linear Address Translation (HLAT)**

1104 Hypervisor managed linear address translation (HLAT) is a capability to enable Intel
1105 Virtualization Technology (Intel VT-x) based security monitors to enforce runtime protection

1106 and integrity assertions on OS-managed page tables. This helps protect kernel assets, as well as
1107 in-band security agents and agent-monitored assets from OS page-table attacks.

1108 “[HLAT] is intended to be used by a Hypervisor/Virtual Machine Monitor (VMM) to enforce
1109 guest linear translation (to guest physical mappings). When combined with the existing Extended
1110 Page Table (EPT) capability, HLAT enables the VMM to ensure the integrity of combined guest
1111 linear translation (mappings and permissions) cached by the processor TLB, via a reduced
1112 software TCB managed by the VMM.” [30] In this fashion, the VMM-enforced guest
1113 translations are more protected from alterations by untrusted system software adversaries. [30]

1114 “This feature is intended to augment the security functionality for a type of Virtual Machine
1115 Monitor (VMM) that may use legacy EPT read/write/execute (XWR) permission bits (bits 2:0 of
1116 the EPTE) as well as the new user-execute (XU) access bit (bit 10 of the EPTE) to ensure the
1117 integrity of code/data resident in guest physical memory assigned to the guest operating system.
1118 EPT permissions are also used in these VMMs to isolate memory; for example, to host a Secure
1119 Kernel (SK) that can manage security properties for the General Purpose Kernel (GPK). For such
1120 usages, it is important that the VMM ensure that the guest linear address mappings which are
1121 used by the General Purpose Kernel to refer to the EPT monitored guest physical pages are
1122 access-controlled as well.” [30]

1123 “VMMs could enforce the integrity of these specific guest linear to guest physical mappings
1124 (paging structures) by using legacy EPT permissions to mark the guest physical memory
1125 containing the relevant guest paging structures as read-only. The intent of marking these guest
1126 paging structures as read-only is to ensure an invalid mapping is not created by guest software.
1127 However, such page-table edit control techniques are known to cause very high overheads due to
1128 the requirement that the VMM must monitor all paging contexts created by the (Guest) operating
1129 system. HLAT enables a VMM to enforce the integrity of guest linear mappings without this
1130 high overhead.” [30]

1131 HLAT utilizes a processor mechanism that implements an alternate Intel Itanium architecture
1132 (IA) paging structure managed in guest physical memory by a Secure Kernel. This paging
1133 structure contains guest linear to guest physical translations that the VMM/Secure Kernel wants
1134 to enforce.

1135 Additionally, to accommodate legacy page-table monitoring approaches, HLAT defines two new
1136 EPT control bits in EPT leaf entries. A “Paging-Write” control bit specifies which guest physical
1137 pages hold HLAT or legacy IA paging structures. This allows the processor to use the Paging-
1138 Write as permission to perform A/D bit writes, instead of the software W permission in the
1139 EPTE. A “Verify Paging-Write” control bit specifies which guest physical pages should only be
1140 referenced via translation (guest) paging structures marked as Paging-writable under EPT [30].

1141 **B.2.2.2 Intel Supervisor Mode Execution Prevention (SMEP) and Supervisor Mode** 1142 **Access Prevention (SMAP)**

1143 Supervisor Mode Execution Prevention (SMEP) and Supervisor Mode Access Prevention
1144 (SMAP) are opt-in capabilities that can be used by systems software (such as the kernel) to
1145 harden the privilege separation between user-mode and kernel-mode. These capabilities further

1146 enforce the user/supervisor properties specified via address translation mechanisms by mitigating
1147 malicious code execution or malicious use of data setup by processes executing in user-mode.

1148 Intel OS Guard, also known as SMEP, helps prevent execution out of untrusted application
1149 memory while operating at a more privileged (supervisor) level. “[When] enabled, the operating
1150 system will not be allowed to directly execute application code, even speculatively. This makes
1151 branch target injection attacks on the OS substantially more difficult by forcing the attacker to
1152 find gadgets within the OS code. It is also more difficult for an application to train OS code to
1153 jump to an OS gadget. All major operating systems enable SMEP support by default.” [31]

1154 SMAP is a security feature that helps prevent unauthorized kernel consumption of data
1155 accessible to user space [32]. An enabling SMAP bit in the CR4 control register will cause a
1156 page fault to be triggered when there is any attempt to access user-space memory while running
1157 in a privileged mode. When access to user space memory is needed by the kernel, a separate AC
1158 flag is toggled to allow the required access [33]. “Two new instructions (STAC and CLAC) are
1159 provided to manipulate that flag relatively quickly.” When the AC flag is set in protection mode
1160 under normal operating circumstances, SMAP blocks a whole class of exploits where the kernel
1161 is fooled into reading from (or writing to) user-space memory by mistake. SMAP also allows for
1162 the early discovery of kernel bugs where developers dereference user space pointers directly
1163 from the kernel [33].

1164 **B.3 Data Protection and Confidential Computing**

1165 **B.3.1 Memory Isolation**

1166 **B.3.1.1 Intel TME and Intel Multi-Key TME (Intel MKTME)**

1167 Intel Total Memory Encryption (Intel TME) provides the capability to encrypt the entire physical
1168 memory of a system. This capability is typically enabled in the very early stages of the boot
1169 process with a small change to the BIOS. Once this change is configured and locked, all data on
1170 the external memory buses of a CPU and any additional DIMMs will be encrypted using 128-bit
1171 keys utilizing the NIST standard AES-XTS algorithm. The encryption key used for Intel TME
1172 uses a hardware random number generator implemented in the Intel CPU, and the keys are not
1173 accessible by software or by using external interfaces to the CPU. The architecture is flexible and
1174 will support additional memory protection schemes in the future. Intel TME is intended to
1175 support unmodified existing system and application software. The overall performance impact of
1176 TME is likely to be relatively small and highly dependent on workload.

1177 Intel Multi-Key Total Memory Encryption (Intel MKTME) builds on Intel TME and adds
1178 support for multiple encryption keys. The CPU implementation supports a fixed number of
1179 encryption keys, and software can configure a CPU to use a subset of available keys. Software
1180 manages the use of keys and can use each of the available keys for encrypting any page of the
1181 memory. Thus, Intel MKTME allows page granular encryption of memory. By default, Intel
1182 MKTME uses the Intel TME encryption key unless explicitly specified by software.

1183 In addition to supporting a CPU-generated ephemeral key (not accessible by software or by using
1184 external interfaces to a CPU), Intel MKTME also supports software-provided keys. Software-
1185 provided keys are particularly useful when used with nonvolatile memory, when combined with

1186 attestation mechanisms or used with key provisioning services. An OS may be enabled to take
1187 additional advantage of the Intel MKTME capability, both in native and virtualized
1188 environments. When properly enabled, Intel MKTME is available to each guest OS in a
1189 virtualized environment, and the guest OS can take advantage of Intel MKTME in the same ways
1190 as a native OS.

1191 **B.3.2 Application Isolation**

1192 **B.3.2.1 Intel Software Guard Extensions (SGX)**

1193 Intel Software Guard Extensions (SGX) is a set of instructions that increases the security of
1194 application code and data. Developers can partition security-sensitive code and data into an *SGX*
1195 *enclave*, which is executed in a CPU protected region. The developer creates and runs SGX
1196 enclaves on server platforms where only the CPU is trusted to provide attestations and protected
1197 execution environments for enclave code and data. SGX also provides an enclave remote
1198 attestation mechanism. This mechanism allows a remote provider to verify the following [34]:

- 1199 1. The enclave is running on a real Intel processor inside an SGX enclave.
- 1200 2. The platform is running at the latest security level (also referred to as the *TCB version*).
- 1201 3. The enclave's identity is as claimed.
- 1202 4. The enclave has not been tampered with.

1203 Once all of this is verified, the remote attester can then provision secrets into the enclave. SGX
1204 enclave usage is reserved for Ring-3 applications and cannot be used by an OS or BIOS
1205 driver/module.

1206 SGX removes the privileged software (e.g., OS, VMM, System Management Mode [SMM],
1207 devices) and unprivileged software (e.g., Ring-3 applications, VMs, containers) from the trust
1208 boundary of the code running inside the enclave, enhancing security of sensitive application code
1209 and data. An SGX enclave trusts the CPU for execution and memory protections. SGX encrypts
1210 memory to protect against memory bus snooping and cold boot attacks for enclave code and data
1211 in host DRAM. SGX includes ISA instructions that can be used to handle Enclave Page Cache
1212 (EPC) page management for creating and initializing enclaves.

1213 SGX relies on the system UEFI BIOS and OS for initial provisioning, resource allocation, and
1214 management. However, once an SGX enclave starts execution, it is running in a
1215 cryptographically isolated environment separate from the OS and BIOS.

1216 SGX can allow any application (whole or part of) to run inside an enclave and puts application
1217 developers in control of their own application security. However, it is recommended that
1218 developers keep the SGX code base small, validate the entire system (including software side
1219 channel resistance), and follow other secure software development guidelines.

1220 SGX enclaves can be used for applications ranging from protecting private keys and managing
1221 security credentials to providing security services. In addition, industry security standards, like
1222 European Telecommunications Standards Institute (ETSI) Network Functions Virtualization
1223 (NFV) Security (ETSI NFV SEC) [35], have defined and published requirements for Hardware

1224 Mediated Execution Enclaves (HMEEs) for the purposes of NFV, 5G, and edge security. SGX is
1225 an HMEE.

1226 **B.3.3 VM Isolation**

1227 **B.3.3.1 Intel Trust Domain Extensions (Intel TDX)**

1228 Intel Trust Domain Extensions (Intel TDX) introduces new architectural elements to deploy
1229 hardware-isolated VMs called trust domains (TDs). Intel TDX is designed to isolate VMs from
1230 the VMM/hypervisor and any non-TD software on the platform to protect TDs from a broad
1231 range of software. TDX is built using a combination of Virtual Machine Extensions (VMX) ISA
1232 extensions, MKTME technology, and a CPU-attested software module called the TDX-SEAM
1233 module. TDX isolates VMs from many hardware threats and most software-based threats,
1234 including from the VMM and other CSP software. TDX helps give the cloud tenant control of
1235 their own data security and IP protection. TDX does this while maintaining the CSP role of
1236 managing resources and cloud platform integrity.

1237 The TDX solution provides the following capabilities to TDs to address the security challenges:

- 1238 • Memory and CPU state confidentiality and integrity to help keep the sensitive IP and
1239 workload data secure from most software-based attacks and many hardware-based
1240 attacks. The workload now has a tool that supports excluding the firmware, software,
1241 devices, and operators of the cloud platform from the TCB. The workloads can use this
1242 tool to foster more secure access to CPU instructions and other CPU features. The
1243 workload can have this ability irrespective of the cloud infrastructure used to deploy the
1244 workload.
- 1245 • Remote attestation enables a relying party (either the owner of the workload or a user of
1246 the services provided by the workload) to establish that the workload is running on a
1247 TDX-enabled platform located within a TD prior to providing that workload data.
1248 Remote attestation aims to allow the owners and consumers of the service to digitally
1249 determine the version of the TCB they are relying on to help secure their data. The VMM
1250 remains the platform resource manager, and TDs should not cause denial of service to the
1251 VMM. Defending TDs against denial of service by the VMM is not a goal.

1252 TDX also augments defense of the TD against limited forms of attacks that use physical access
1253 to the platform memory, such as offline, DRAM analysis (example: cold-boot attacks), and
1254 active attacks of DRAM interfaces, including capturing, modifying, relocating, splicing, and
1255 aliasing memory contents [36]. The VMM continues to be the resource manager, and TDs do not
1256 have privileges to deny service to the VMM.

1257 **B.3.4 Cryptographic Acceleration**

1258 **B.3.4.1 Intel Advanced Encryption Standard New Instructions (Intel AES-NI)**

1259 Intel AES New Instructions (Intel AES-NI) is an encryption instruction set that improves
1260 hardware performance of the Advanced Encryption Standard (AES) algorithm and accelerates
1261 data encryption. Intel AES-NI consists of seven new instructions that accelerate encryption and
1262 decryption and improve key generation and matrix manipulation, all while aiding in carry-less

1263 multiplication. This minimizes application performance concerns inherent in conventional
1264 cryptographic processing and helps provide enhanced security by addressing side channel attacks
1265 on AES associated with conventional software methods of table lookups [37].

1266 AES is the most widely used standard for protecting network traffic, personal data, and corporate
1267 IT infrastructures. By implementing certain intensive sub-steps of the AES algorithm into the
1268 hardware, Intel AES-NI strengthens and accelerates execution of the AES application [37].

1269 **B.3.4.2 Intel QuickAssist Technology (QAT) with Intel Key Protection Technology** 1270 **(KPT)**

1271 Intel QuickAssist Technology (QAT) is a high-performance hardware accelerator for performing
1272 cryptographic, security, and compression operations. Applications like VMs, containers, and
1273 Function as a Service (FaaS) call Intel QAT using industry-standard OpenSSL, TLS, and Internet
1274 Protocol Security (IPsec) interfaces to offload symmetric and asymmetric cryptographic
1275 operations. Cloud, multi-tenancy, NFV, edge, and 5G infrastructures and applications are best
1276 suited for QAT for all types of workloads, including software-defined networks (SDNs), content
1277 delivery networks (CDNs), media, and storage [38].

1278 Intel Key Protection Technology (KPT) helps enable customers to secure their keys to be used
1279 with QAT through a bring-your-own-key (BYOK) paradigm. KPT allows customers to deliver
1280 their own cryptographic keys to the QAT device in the target platform where their workload is
1281 running. KPT-protected keys are never in the clear in host DRAM or in transit. The customers
1282 encrypt their workload key (e.g., RSA private key for Nginx) using KPT inside their HSMs. This
1283 encrypted workload key is delivered to the target QAT platform, where it is decrypted
1284 immediately prior to use. KPT provides key protection at rest, in transit, and while in use [39].

1285 **B.3.5 Technology Example Summary**

1286 Cloud infrastructure creates improvements in the efficiency, agility, and scalability of data center
1287 workloads by abstracting hardware from the application layer. This introduces new security
1288 concerns as workloads become multi-tenant, attack surfaces become shared, and infrastructure
1289 administrators from the cloud operator gain access to underlying platforms. Isolation techniques
1290 provide answers to these concerns by adding protection to VMs, applications, and data during
1291 execution, and they represent a crucial layer of a layered security approach for data center
1292 security architecture.

1293 Various isolation techniques exist and can be leveraged for different security needs. Full memory
1294 isolation defends a platform against physical memory extraction techniques, while the same
1295 technology extended with multiple keys allows individual VMs or platform tenants to have
1296 uniquely encrypted memory. Future generations of these technologies will allow full memory
1297 isolation of VMs, protecting them against malicious infrastructure insiders, multi-tenant
1298 malware, and more. Application isolation techniques allow individual applications to create
1299 isolated enclaves that require implicit trust in the platform CPU and nothing else and that have
1300 the ability to provide proof of the enclave to other applications before data is sent.

1301 **B.4 Remote Attestation Services**

1302 **B.4.1 Intel Security Libraries for the Data Center (ISecL-DC)**

1303 Intel Security Libraries for the Data Center (ISecL-DC) is an open-source remote attestation
1304 implementation of a set of building blocks that utilize Intel Security features to discover, attest,
1305 and enable critical foundation security and confidential computing use-cases. This middleware
1306 technology provides a consistent set of application programming interfaces (APIs) for easy
1307 integration with cloud management software and security monitoring and enforcement tools.
1308 ISecL-DC applies the remote attestation fundamentals described in this section and standard
1309 specifications to maintain a platform data collection service and an efficient verification engine
1310 to perform comprehensive trust evaluations. These trust evaluations can be used to govern
1311 different trust and security policies applied to any given workload, as referenced in the workload
1312 scheduling use case in Section 7.2. In future generations, the product will be extended to include
1313 TEE attestation to provide assurance and validity of the TEE to enable confidential computing
1314 [40].

1315 **B.4.2 Technology Summary**

1316 Platform attestation provides auditable foundational reports for server firmware and software
1317 integrity and can be extended to include the location of other asset tag information stored in a
1318 TPM, as well as integrity verification for applications installed on the server. These reports
1319 provide visibility into platform security configurations and can be used to control access to data
1320 and workloads. Platform attestation is performed on a per-server basis and typically consumed
1321 by cloud orchestration or a wide variety of infrastructure management platforms.

1322 TEE attestation provides a mechanism by which a user or application can validate that a genuine
1323 TEE enclave with an acceptable TCB is actually being used before releasing secrets or code to
1324 the TEE. Formation of a TEE enclave is performed at the application level, and TEE attestations
1325 are typically consumed by a user or application requiring evidence of enclave security before
1326 passing secrets.

1327 These different attestation techniques serve complementary purposes in a cloud deployment in
1328 the data center or at the edge computing facility.

1329

1330 Appendix C—AMD Technology Examples

1331 This section describes a number of AMD technology examples that map back to the key
1332 concepts described in the various sections of the document.

1333 C.1 Platform Integrity Verification**1334 C.1.1 AMD Platform Secure Boot (AMD PSB)**

1335 AMD Platform Secure Boot (AMD PSB) provides a hardware RoT to authenticate the initial
1336 Platform BIOS code during the boot process of the server. Manufacturers of server systems, like
1337 OEMs or Original Device Manufacturers (ODMs), enable the functionality of AMD PSB in their
1338 manufacturing flow by permanently fusing policy into the silicon.

1339 The OEM or ODM's final BIOS image contains the AMD public key and the OEM BIOS-
1340 signing public key (signed with the AMD private key). When a system powers on, the AMD
1341 Security Processor (ASP) starts executing the immutable on-chip Boot ROM. It authenticates
1342 and loads multi-stage ASP Boot Loaders from SPI/Low Pin Count (LPC) Flash into its internal
1343 memory, which initializes the silicon and the system memory.

1344 Once the system memory is initialized, the ASP Boot Loaders load and authenticate the OEM
1345 BIOS-signing public key, followed by authenticating the initial BIOS code. Once the verification
1346 is successful, ASP releases the x86 core to execute authenticated initial BIOS code. The BIOS
1347 can continue CoT for other components by means of UEFI Secure Boot. If PSB authentication
1348 fails, the system is forced to shut down.

1349 AMD PSB supports revocation and rollback protection of BIOS images through the OEM BIOS-
1350 signing key revision ID and rollback protection.

1351 C.2 Data Protection and Confidential Computing**1352 C.2.1 Memory Isolation: AMD Secure Memory Encryption (SME)/Transparent Memory
1353 Encryption (TSME)**

1354 AMD Secure Memory Encryption (SME) is a memory encryption technology from AMD which
1355 helps protect data in DRAM by encrypting system memory content [41]. When enabled, memory
1356 content is encrypted via dedicated hardware in the on-die memory controllers. Each controller
1357 includes a high-performance AES engine that encrypts data when it is written to DRAM and
1358 decrypts it when read. The encryption of data is done with an encryption key in a mode that
1359 utilizes an additional physical address-based tweak to protect against ciphertext block move
1360 attacks.

1361 The encryption key used by the AES engine with SME is randomly generated on each system
1362 reset and is not visible to any software running on the CPU cores. This key is managed entirely
1363 by the AMD Secure Processor (AMD-SP) that functions as a dedicated security subsystem
1364 integrated within the AMD System-on-Chip (SOC). The key is generated using the onboard
1365 NIST SP 800-90 compliant hardware random number generator and is stored in dedicated
1366 hardware registers where it is never exposed outside the SOC in the clear.

1367 Two modes of memory encryption are supported for various use cases. The simplest mode is
1368 Transparent Secure Memory Encryption (TSME), which is a BIOS option and enables memory
1369 encryption automatically on all memory accesses. TSME works in the background and requires
1370 no software interaction. Another supported mode is the OS-managed Secure Memory Encryption
1371 (SME) mode in which individual pages of memory may be marked for encryption via CPU page
1372 tables. SME provides additional flexibility if only a subset of memory needs to be encrypted but
1373 does require appropriate software support.

1374 Encrypted memory provides strong protection against cold boot, DRAM interface snooping, and
1375 similar types of attacks.

1376 **C.2.2 VM Isolation: AMD Secure Encrypted Virtualization (SEV)**

1377 The AMD Secure Encrypted Virtualization (SEV) feature is designed to isolate VMs from the
1378 hypervisor. When SEV is enabled, individual VMs are encrypted with an AES encryption key.
1379 When a component such as the hypervisor attempts to read memory inside a guest, it is only able
1380 to see the data in its encrypted form. This provides strong cryptographic isolation between the
1381 VMs, as well as between the VMs and the hypervisor.

1382 To protect SEV-enabled guests, the SEV firmware assists in the enforcement of three main
1383 security properties: authenticity of the platform, attestation of a launched guest, and
1384 confidentiality of the guest's data.

1385 Authenticating the platform prevents malicious software or a rogue device from masquerading as
1386 a legitimate platform. The authenticity of the platform is proven with its identity key. This key is
1387 signed by AMD to demonstrate that the platform is an authentic AMD platform with SEV
1388 capabilities.

1389 Attestation of the guest launch proves to guest owners that their guests securely launched with
1390 SEV enabled. A signature of various components of the SEV-related guest state, including initial
1391 contents of memory, is provided by the firmware to the guest owner to verify that the guest is in
1392 the expected state. With this attestation, a guest owner can ensure that the hypervisor did not
1393 interfere with the initialization of SEV before transmitting confidential information to the guest.

1394 Confidentiality of the guest is accomplished by encrypting memory with a memory encryption
1395 key that only the SEV firmware knows. The SEV management interface does not allow the
1396 memory encryption key or any other secret SEV state to be exported outside of the firmware
1397 without properly authenticating.

1398 AMD SEV has two additional modes:

- 1399 • SEV With Encrypted State (SEV-ES): This mode encrypts and protects VM registers
1400 from being read or modified by a malicious hypervisor or VM [42].
- 1401 • SEV with Secure Nested Paging (SEV-SNP): This mode adds strong memory integrity
1402 protection to help prevent malicious hypervisor-based attacks like data replay and
1403 memory remapping. [43]

1404

1405 **Appendix D—Acronyms and Abbreviations**

1406 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	European Telecommunications Standards Institute Network Functions
SEC	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
SCR TM	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

1408 **Appendix E—Glossary**

1409	Asset Tag	Simple key value attributes that are associated with a platform (e.g., location, company name, division, or department).
1410		
1411		
1412	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.
1413		
1414		
1415		
1416		
1417	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.
1418		
1419		
1420		
1421	Cryptographic Accelerator	A specialized separate coprocessor chip from the main processing unit where cryptographic tasks are offloaded to for performance benefits.
1422		
1423		
1424	Hardware-Enabled Security	Security with its basis in the hardware platform.
1425	Platform Trust	An assurance in the integrity of the underlying platform configuration, including hardware, firmware, and software.
1426		
1427		
1428	Root of Trust (RoT)	A starting point that is implicitly trusted.
1429	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.
1430		
1431		
1432		
1433	Trusted Execution Environment (TEE)	An area or enclave protected by a system processor.