IN2120 Information Security

Lecture 7: Computer Security

Laszlo Erdödi
Ijlal Loutfi
Universitetet i Oslo

Lecture Overview

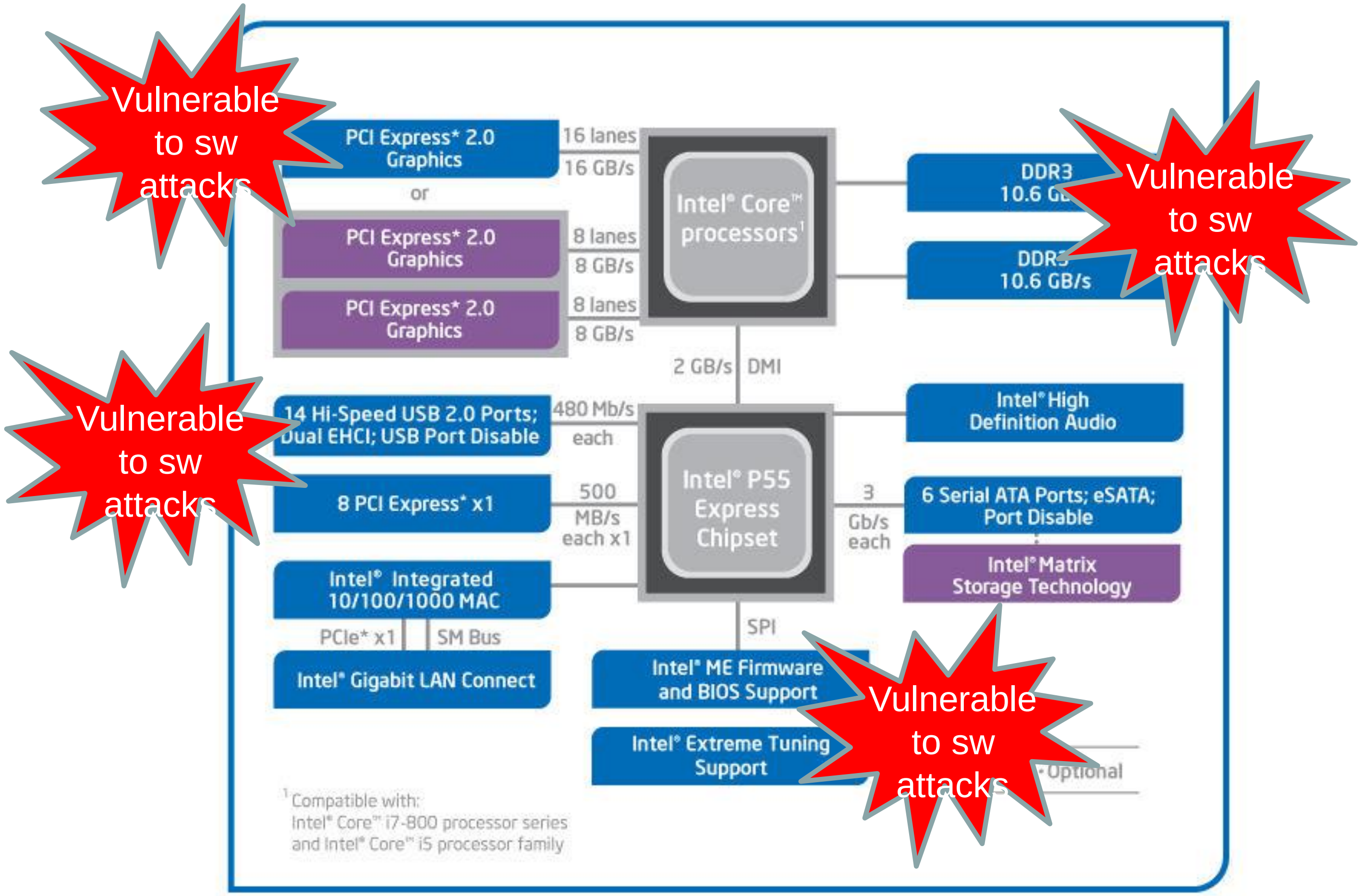
- Secure computer architectures
- Virtualization architectures
- Trusted computing

System & Communication Security

- "Using encryption on the Internet is the equivalent of arranging an armored car to deliver credit card information from someone living in a cardboard box to someone living on a park bench.“

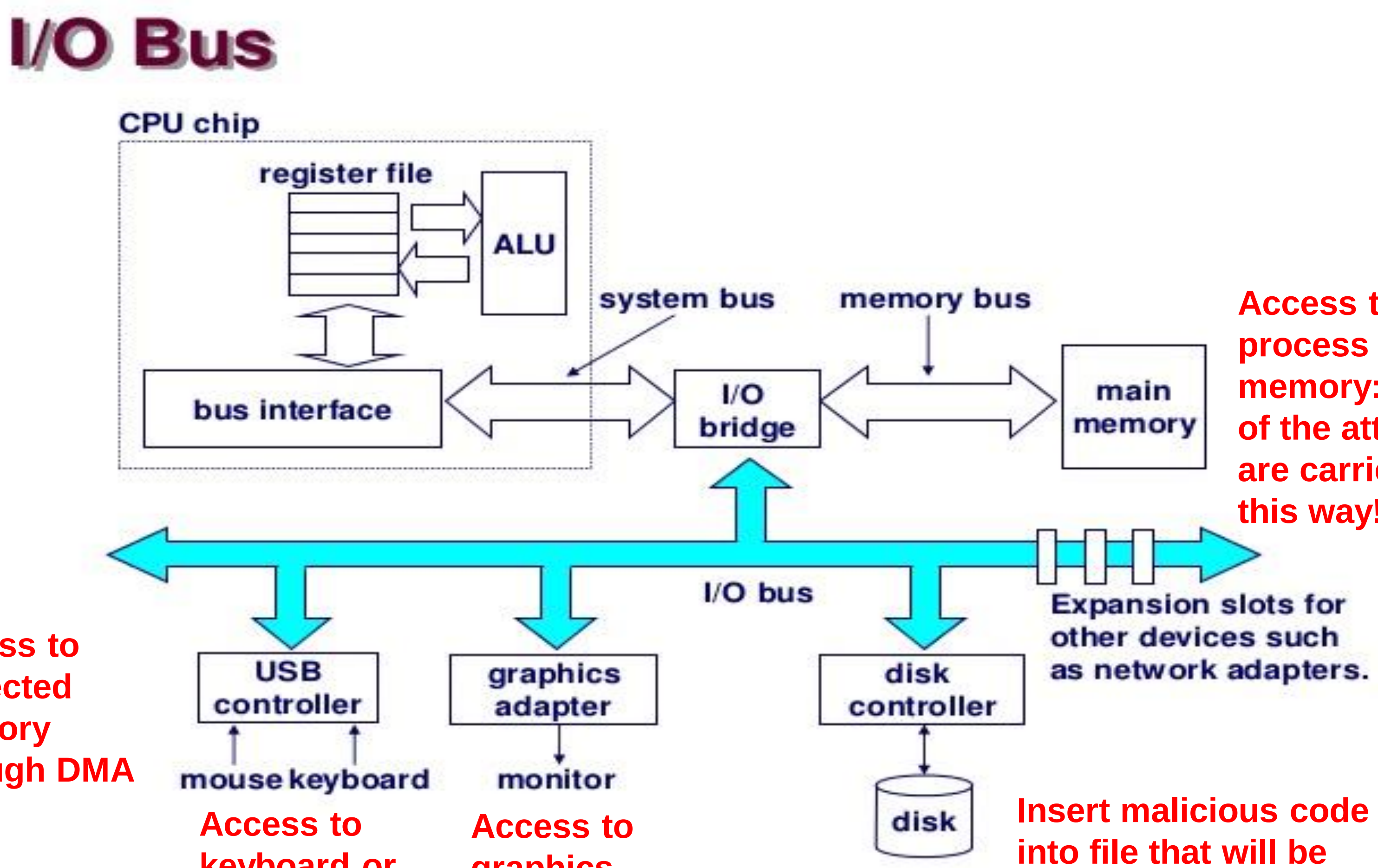
  (Gene Spafford)

Vulnerabilities of the PC today


Vulnerable to sw attacks
PCI Express* 2.0 Graphics
16 lanes
16 GB/s
or
PCI Express* 2.0 Graphics
8 lanes
8 GB/s
PCI Express* 2.0 Graphics
8 lanes
8 GB/s
Intel® Core™ processors¹
DDR3 10.6 GB/s
DDR3 10.6 GB/s
Vulnerable to sw attacks
2 GB/s
DMI
Vulnerable to sw attacks
14 Hi-Speed USB 2.0 Ports; Dual EHCI; USB Port Disable
480 Mb/s each
8 PCI Express* x1
500 MB/s each x1
Intel® P55 Express Chipset
Intel® High Definition Audio
3 Gb/s each
6 Serial ATA Ports; eSATA; Port Disable
Intel® Matrix Storage Technology
Intel® Integrated 10/100/1000 MAC
PCIe* x1
SM Bus
Intel® Gigabit LAN Connect
SPI
Intel® ME Firmware and BIOS Support
Intel® Extreme Tuning Support
Vulnerable to sw attacks
Optional
¹Compatible with: Intel® Core™ i7-800 processor series and Intel® Core™ i5 processor family

Intel® P55 Express Chipset Platform Block Diagram

Vulnerabilities of the PC today



Access to process memory: most of the attacks are carried out this way!!!

Access to protected memory through DMA

Access to keyboard or mouse data

Access to graphics frame buffer

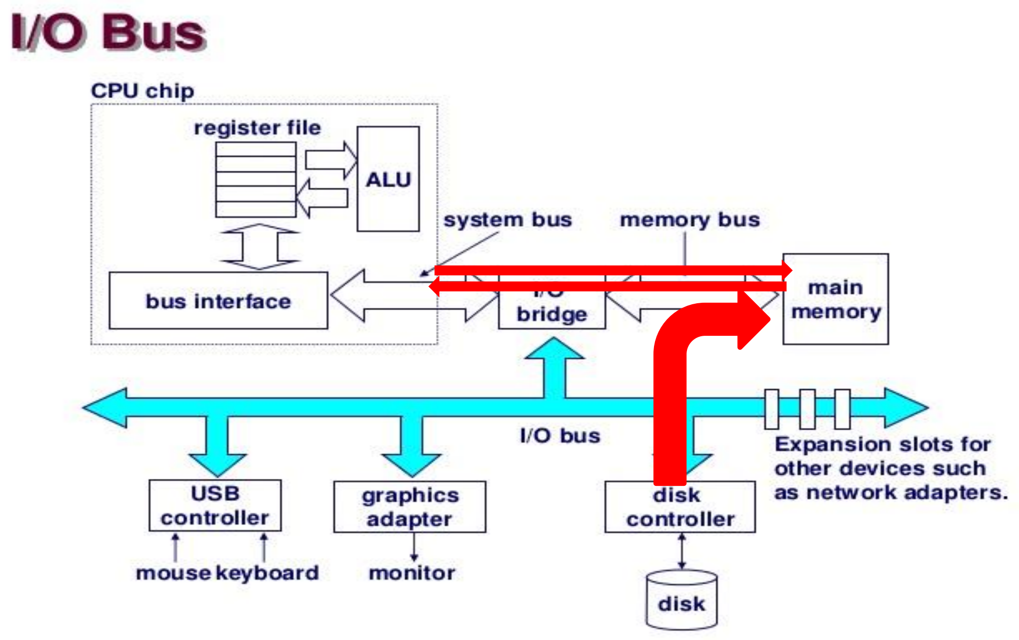
Insert malicious code into file that will be executed later

Approaches to strengthening platform security

- Harden the operating system
  - SE (Security Enhanced) Linux, Trusted Solaris, Windows 7/8/10
- Add security features to the CPU
  - Protection Layers, NoExecute, ASLR
- Virtualisation technology
  - Separates processes by separating virtual systems
- Trusted Computing
  - Add secure hardware to the commodity platform
  - E.g. TPM (Trusted Platform Module)
- Rely on secure hardware external to commodity platform
  - Smart cards
  - Hardware tokens

OS protections – Launching a process

- When the system runs an executable, the code and the data of the executable are loaded into the Random Access Memory
- The OS analyze the code and also copies the required dependencies and the necessary OS API files into the memory
- Each process has an own Virtual Address Space, where all necessary code and data are loaded
- The process can access only its own Virtual Address Space by CPU instructions

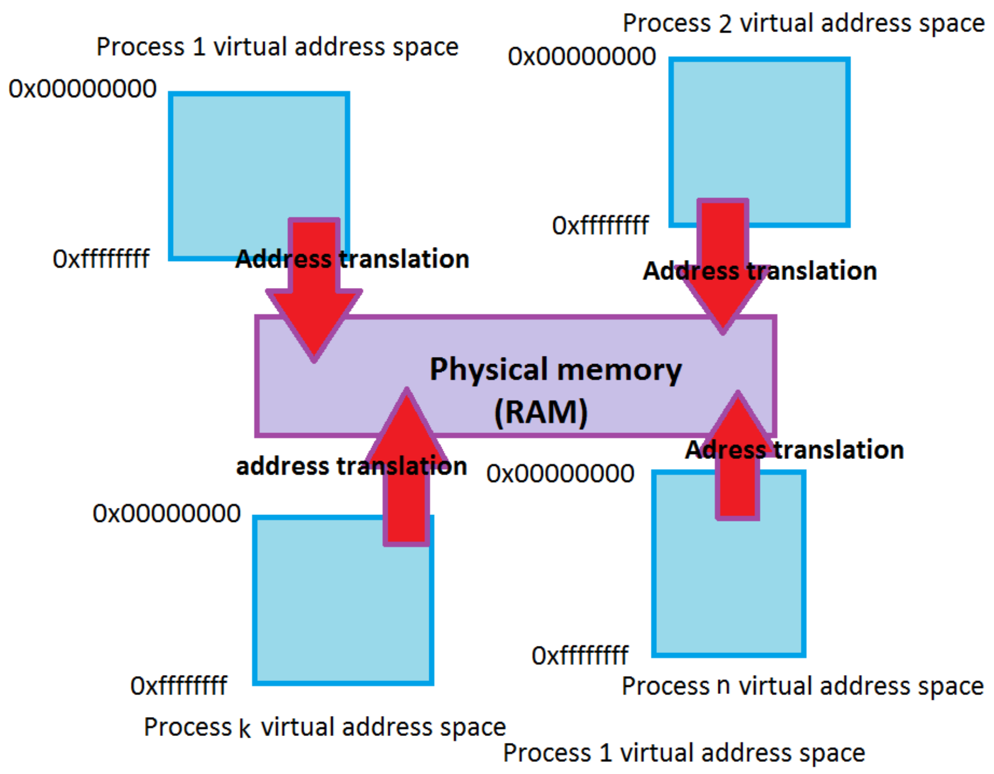


OS protections – Process Virtual Address Space

- The process code can access its own address space, but this addressing is different from the global physical addressing. Each process has the maximum addressable memory range: a virtual memory in a virtual address space (this is 2^{32} (4GB) i.e. in 32bit systems 0x00000000 – 0xffffffff)
- To ensure the correct memory operations the OS has to provide a runtime address translation between virtual memory to the physical memory
- Because of this way of operation the OS provides 2 protections by design: The program code that is executed cannot access the physical memory directly (We cannot write code to access physical memory in none of the programming languages)
- And also the OS protects the process code and data from eachother. Normally a process cannot have access to another process memory (except debugging a process by another process or using specific OS features such as CreateRemoteThread)

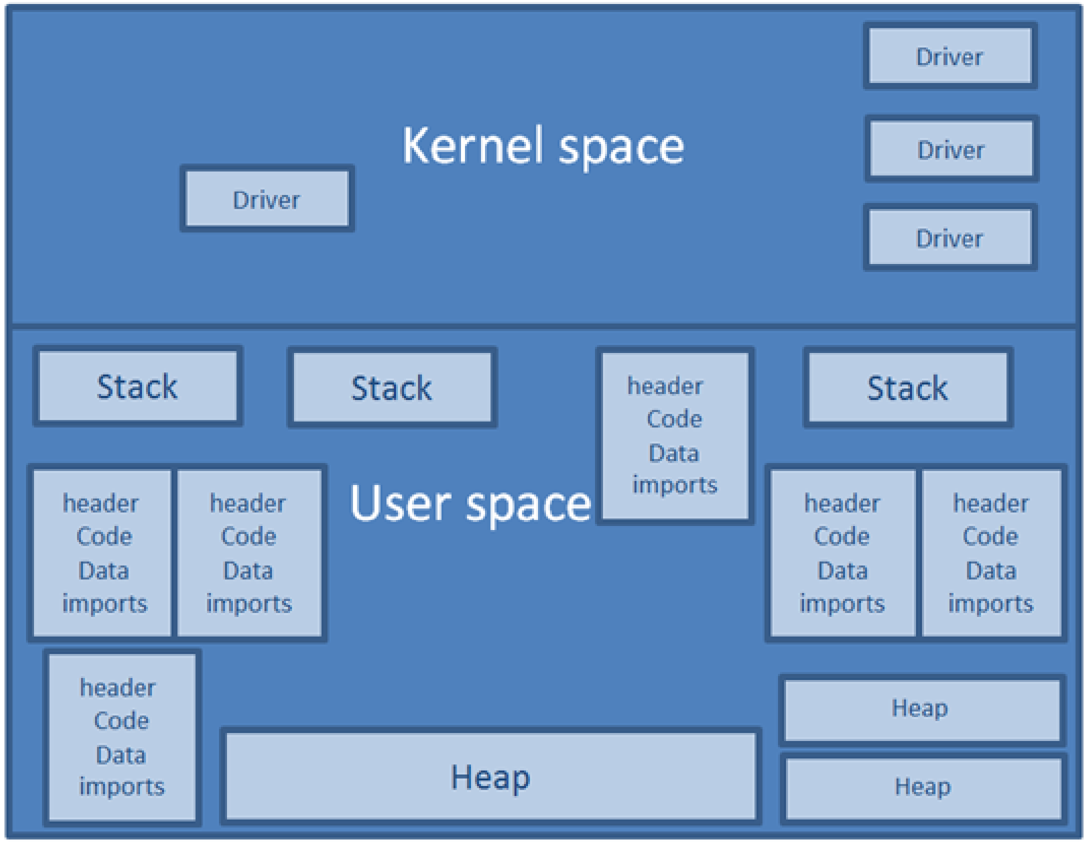
OS protections – Runtime Address Translation

- The runtime address translation is to optimize memory usage of processes and also to protect the process data



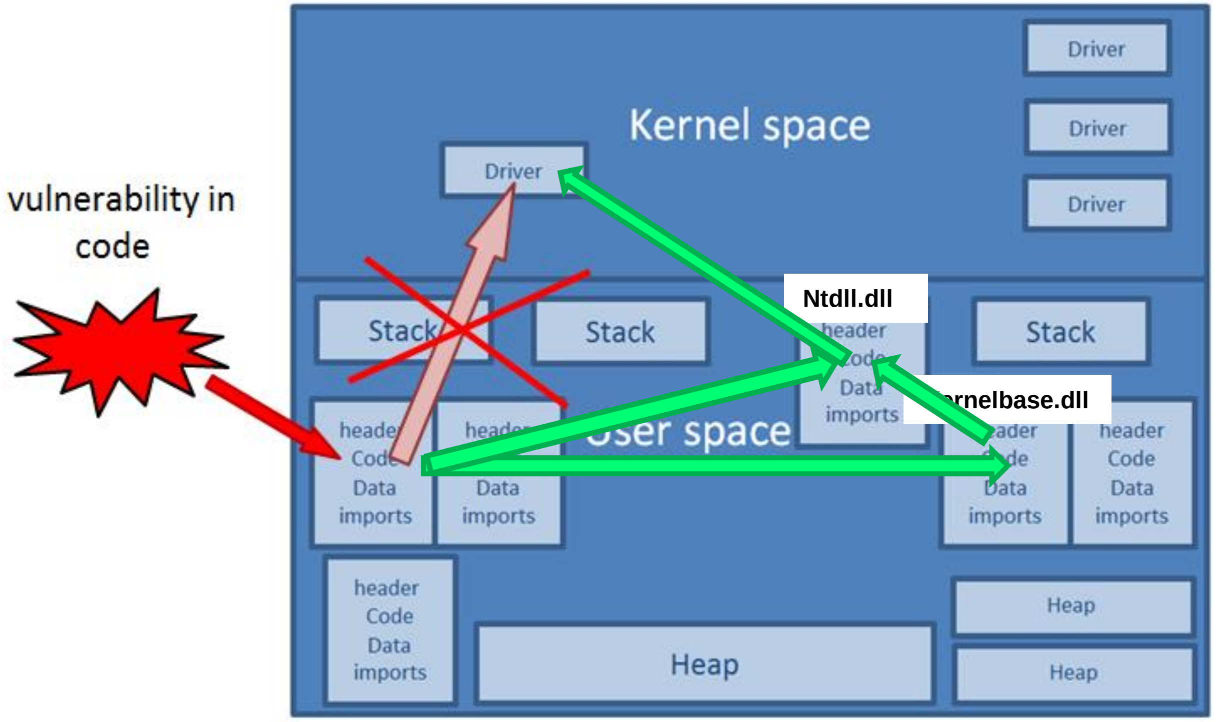
OS protections – Virtual Address Space

- The Virtual Address Space is separated into kernel and user space



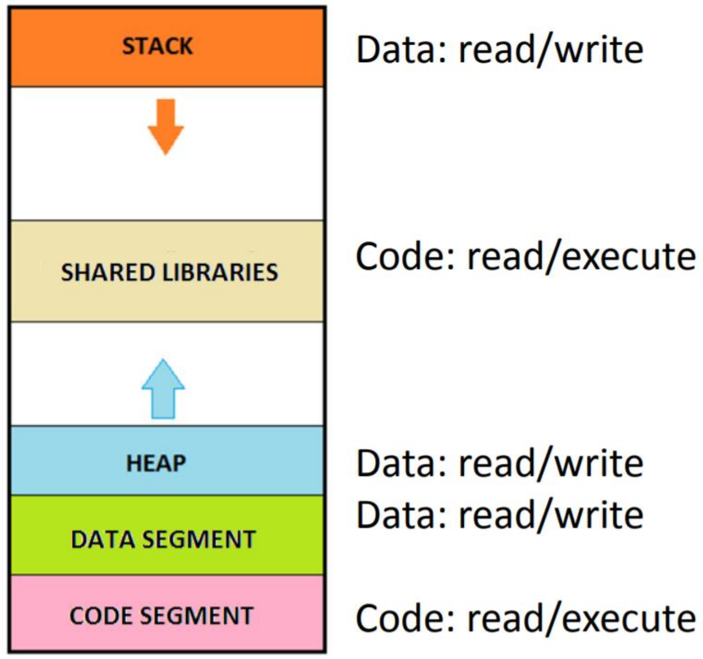
OS protections – Code vulnerabilities

- The kernel space can be accessed through only specific OS APIs (e.g. in Windows using the native API, Hardware protection CPU rings - see later)



OS protections – Software supported Data Execution Prevention (DEP)

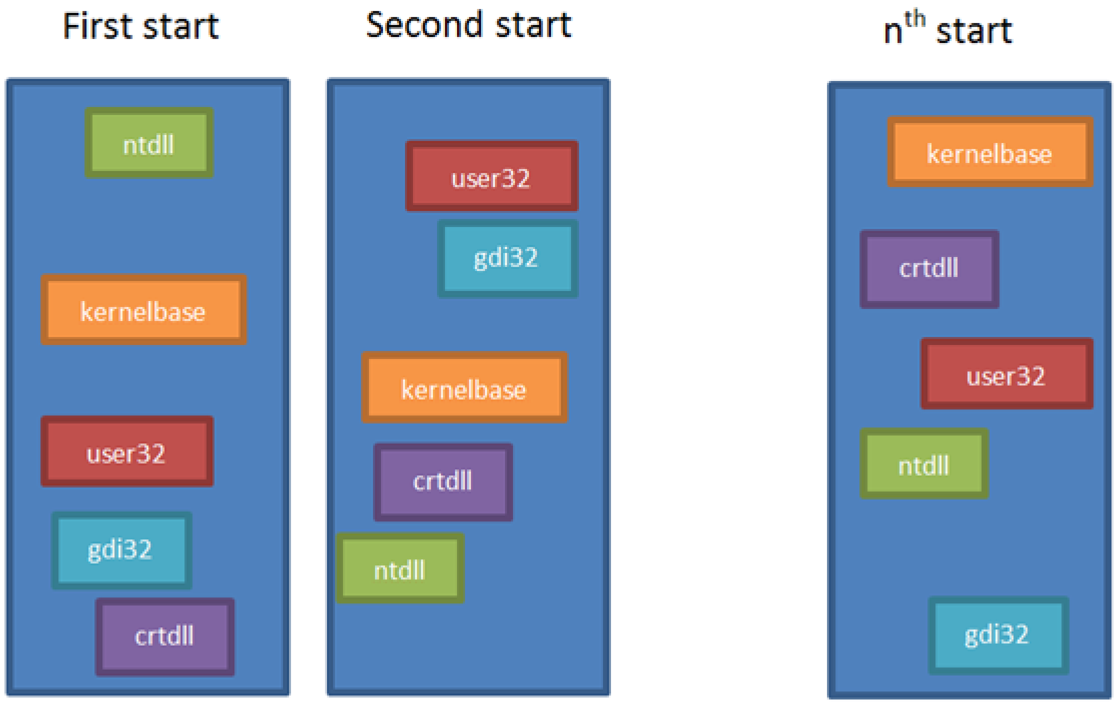
- The vulnerable code has no access directly to kernel space, but without DEP it can modify the code in the user space (even the OS API in user space) or can execute an attacker placed malicious code



- The OS can enforce DEP (i.e nxAlwaysOn settings on Windows)

OS protections – Address Space Layout Randomization (ASLR)

- Because of DEP the attackers turned into code reuse. To prevent it OS provides code randomization in the Virtual Address Space



OS advanced protections – new directions

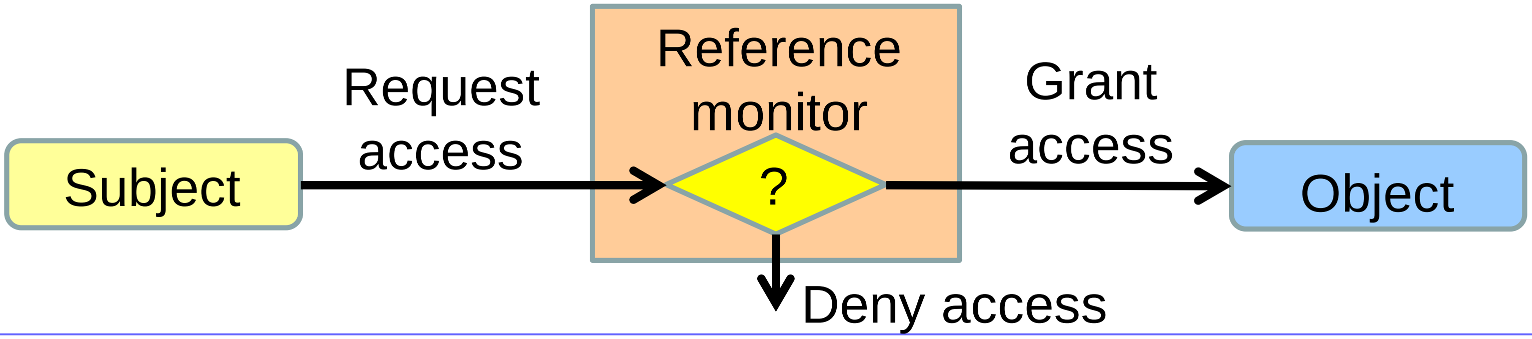
- Increasing the entropy of ASLR makes brute-forcing less effective (High Entropy ASLR)
- Address Space Layout Permutation (ASLP) the place of the libraries are randomized as well as the order of the methods in the Virtual Address Space
- Code diversity (OS protection ?)
- Multiple heaps and protected heap segments (e.g. late free, MS Edge)
- Execute no Read segments (XnR protection, hardware support later?)
- Control Flow Integrity (observation of unintended use-cases)

CPU protection

- Protection rings
- Hardware supported Data Execution Prevention
- Hardware supported Control Flow Integrity (not in use yet, e.g. Intel’s Control Flow Enforcement)
- Intel sgx (Protecting memory regions from higher privileged access)

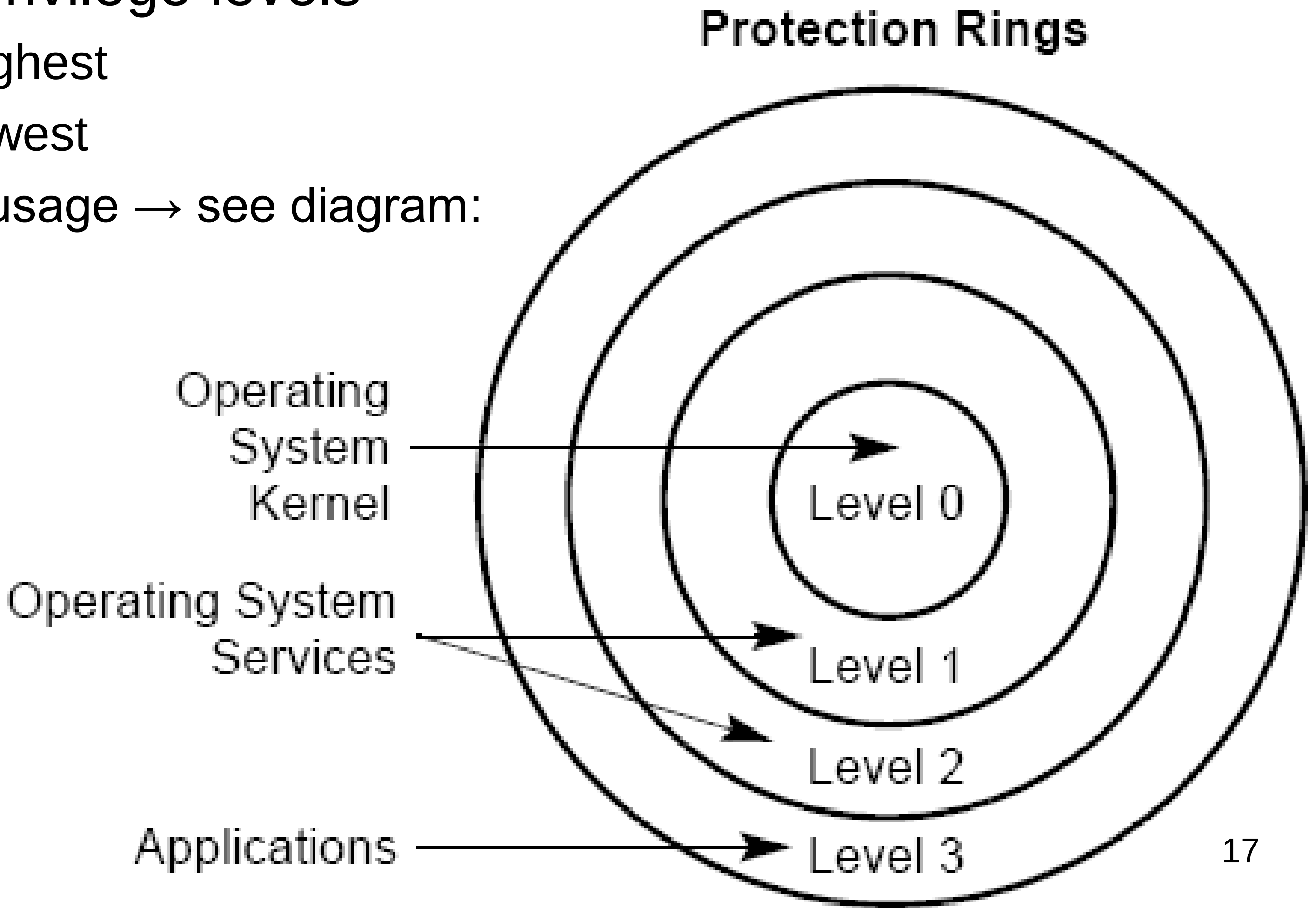
Reference Monitor

- A reference monitor is any security model for enforcing an access control policy over subjects' (e.g., processes and users) ability to perform operations (e.g., read and write) on objects (e.g., files, memory and sockets) on a system.
  - The reference monitor must always be invoked (complete mediation).
  - The reference monitor must be tamperproof (tamperproof).
  - The reference monitor must be small enough to be subject to analysis and tests, the completeness of which can be assured (verifiable).
- The security kernel of an OS is a low-level (close to the hardware) implementation of a reference monitor.



OS security kernel as reference monitor

- Hierarchic security levels were introduced in X86 CPU architecture in 1985 (Intel 80386)
- 4 ordered privilege levels
  - Ring 0: highest
  - Ring 3: lowest
  - Intended usage → see diagram:



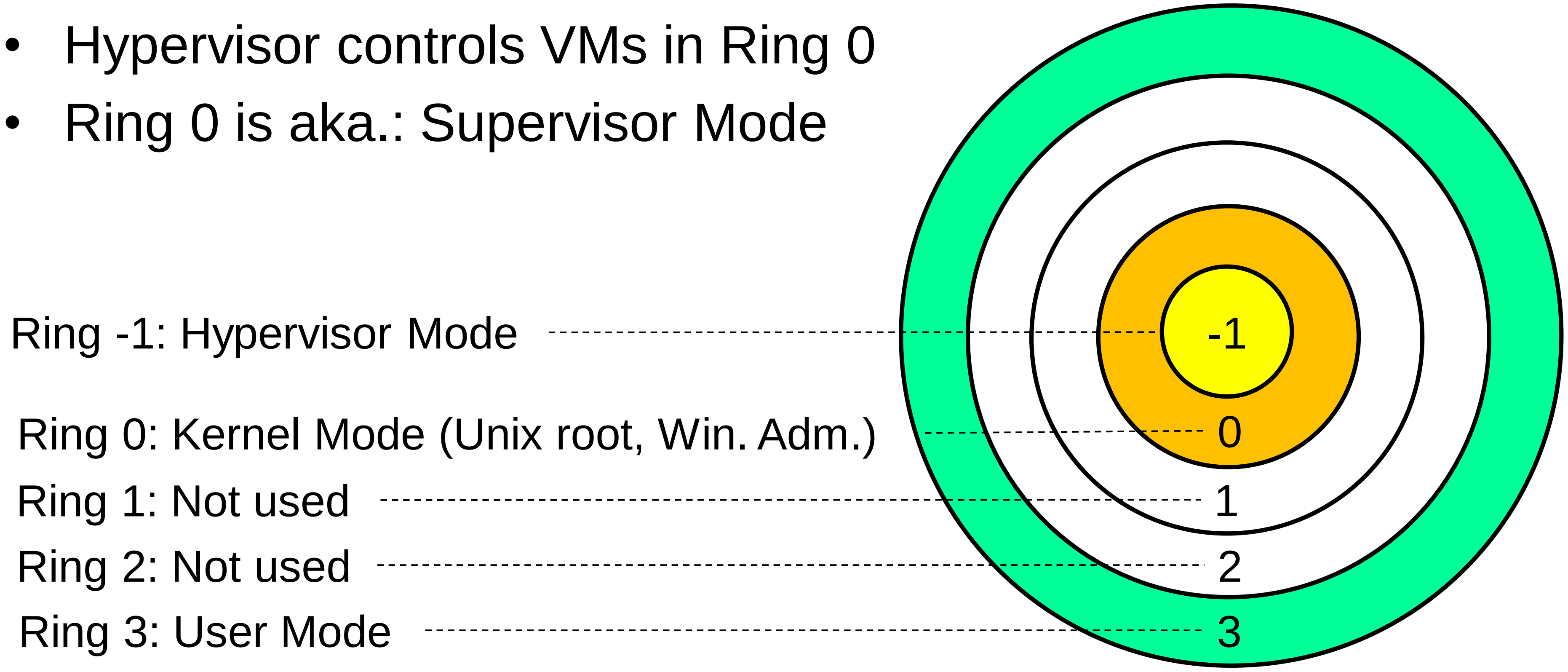
What happened to rings 1 & 2 ?

... it eventually became clear that the hierarchical protection that rings provided did not closely match the requirements of the system programmer and gave little or no improvement on the simple system of having two modes only. Rings of protection lent themselves to efficient implementation in hardware, but there was little else to be said for them. [...]. This again proved a blind alley...

Maurice Wilkes (1994)

CPU Protection Ring structure from 2006

- New Ring -1 introduced for virtualization.
- Necessary for protecting hypervisor from VMs (Virtual Machines) running in Ring 0.
- Hypervisor controls VMs in Ring 0
- Ring 0 is aka.: Supervisor Mode

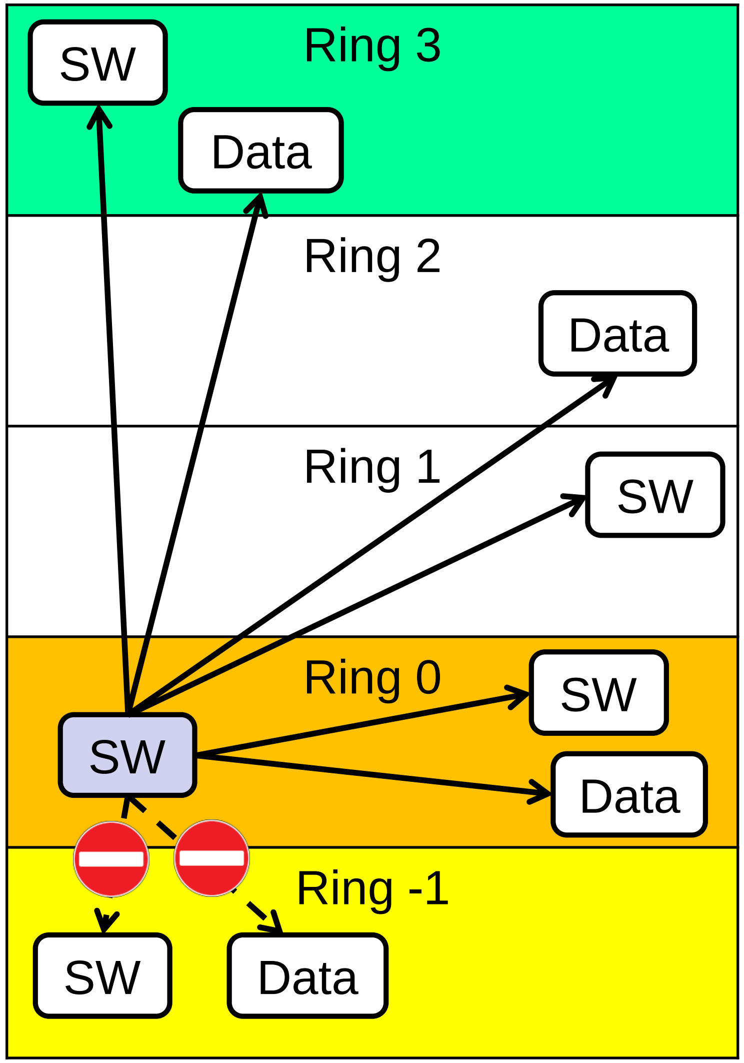


Privileged Instructions

- Some of the system instructions (called “privileged instructions”) are protected from use by application programs.
- The privileged instructions control system functions (such as the loading of system registers). They can be executed only when the Privilege Level is 0 or -1 (most privileged).
- If one of these instructions is attempted when the Privilege Level is not 0 or -1, then a general-protection exception (#GP) is generated, and the program crashes.

Principle of protection ring model

- A process can access and modify any data and software at the same or less privileged level as itself.
- A process that runs in kernel mode (Ring 0) can access data and SW in Rings 0, 1, 2 and 3
  - but not in Ring -1
- The goal of attackers is to get access to kernel or hypervisor mode.
  - through exploits
  - by tricking users to install software

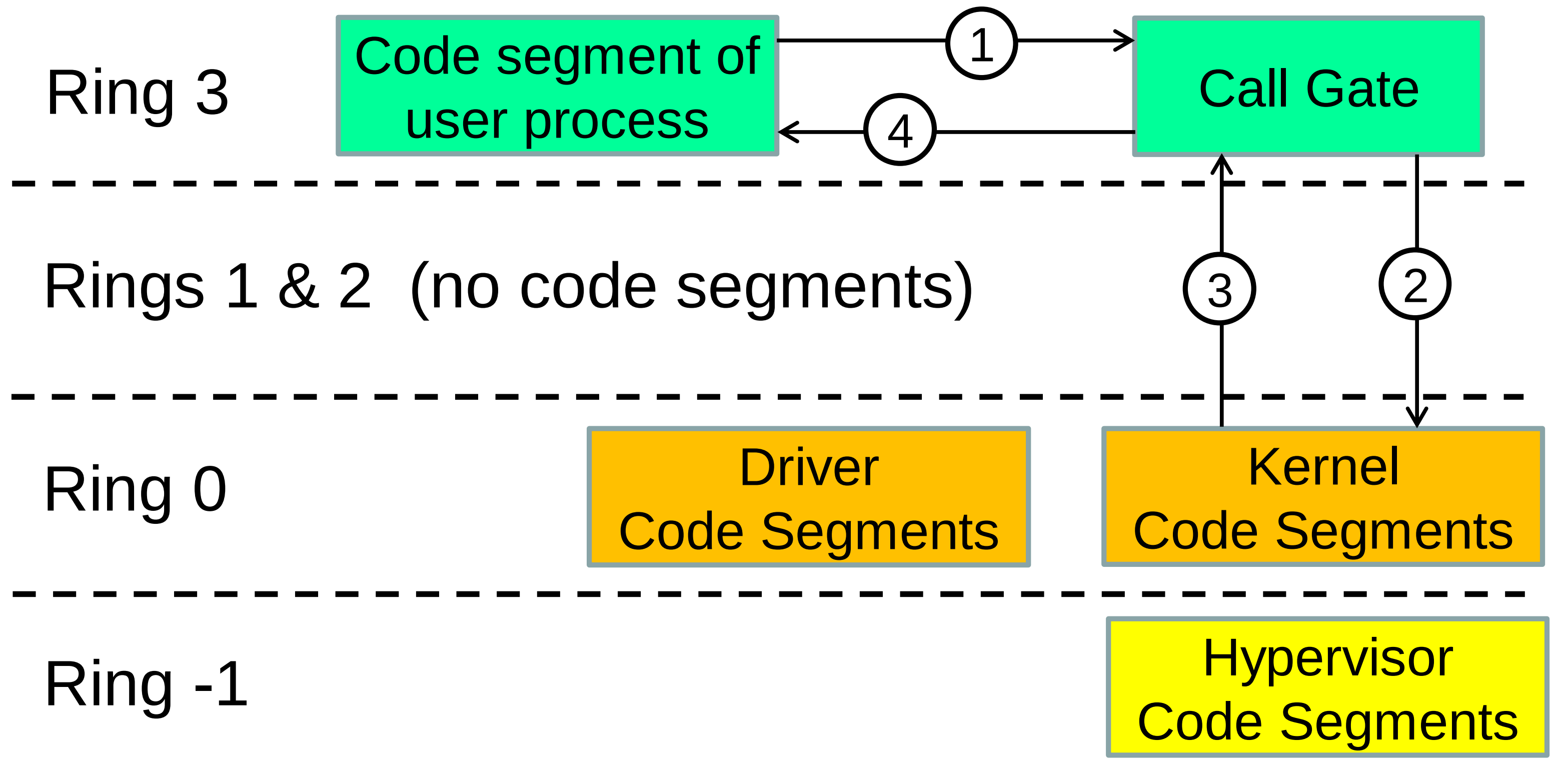


User processes access to system resources

- User processes need to access system resources (memory and drivers)
- User application processes should not access system memory directly, because they could corrupt memory.
- The CPU must restrict direct access to memory segments and other resources depending on the privilege level (Current Privilege Level, CPL).

- Question 1: How can a user process execute instructions that require kernel mode, e.g. for writing to memory ?
  - Answer: The CPU must switch between privilege levels
- Question 2: How should privilege levels be switched?
  - Answer: Through Controlled invocation of code segments

Controlled Invocation of code segments



Controlled Invocation

- The user process executes code in specific code segments.
- Each code segment has an associated mode which dictates the privilege level the code executes under.
- Simply setting the mode of user process code to Kernel would give kernel-privilege to user process without any control of what the process actually does. Bad idea!
- Instead, the CPU allows the user process to call kernel code segments that only execute a predefined set of instructions in kernel mode, and then returns control back to the user-process code segment in user mode.
- We refer to this mechanism as controlled invocation.

Hardware supported Data Execution Prevention

- Intel Processors with XD support
- AMD processors with NX support

Hardware-enforced DEP marks all memory locations as non-executable (you cannot execute code in this portion of memory) unless the location explicitly contains executable code.

Hardware-enforced DEP relies on processor hardware to mark memory with an attribute that indicates that code should not be executed from that memory.

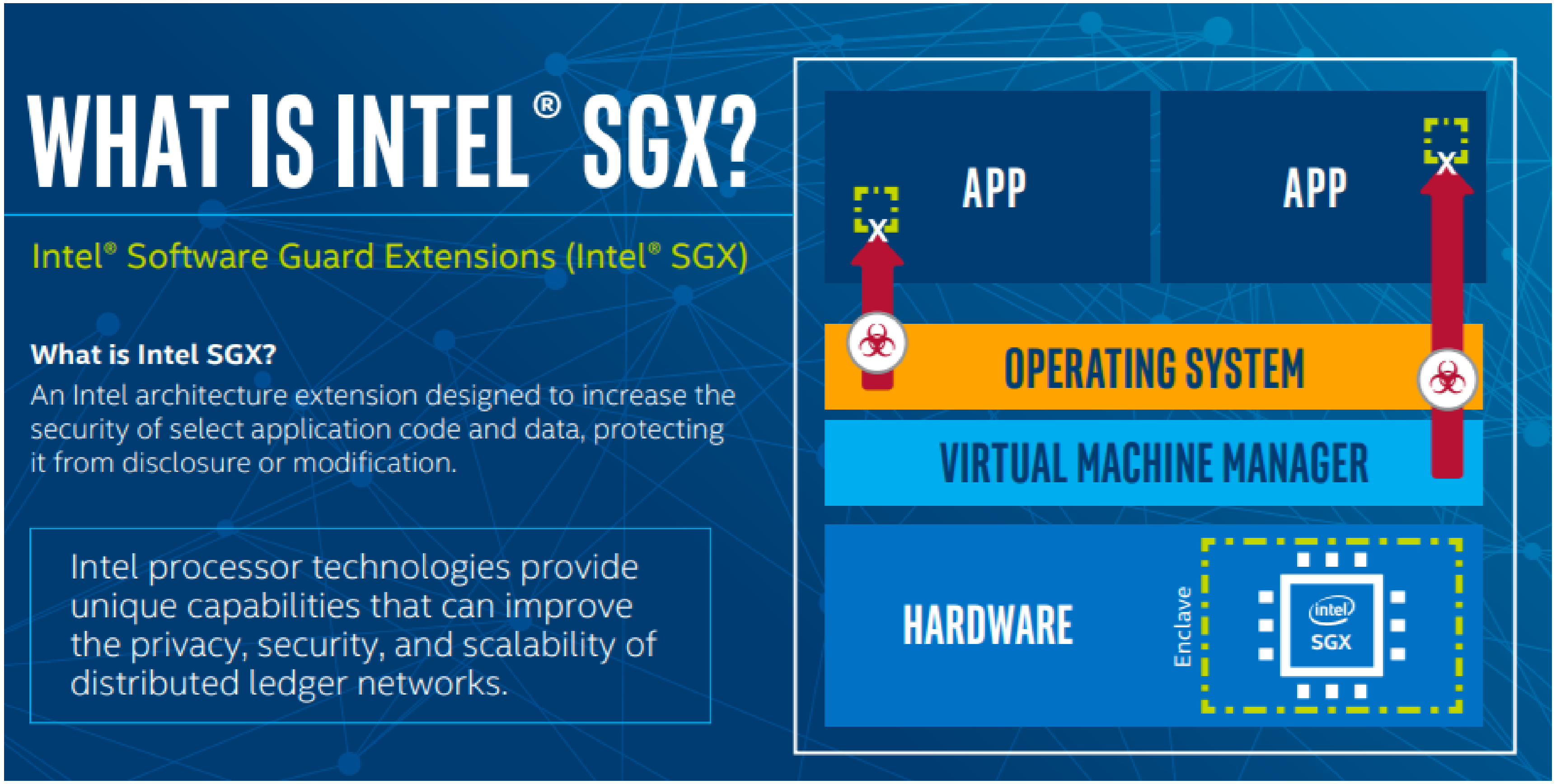
Processors that support hardware-enforced DEP are capable of raising an exception when code is executed from a memory location where it should not be executed.

To use these processor features, the processor must run in Physical Address Extension (PAE) mode. HP ships Windows XP with PAE enabled.

Control Flow Enforcement (in plan)

- Processors will directly support: Shadow (call) stack tracking. Method return addresses are stored in data stack and the shadow stack too. Shadow stack is not accessible, the processor checks the return addresses, if there's un-matching return then control protection exception is raised
- Processors also support: indirect branch tracking. After each legitimate indirect branch instruction the code must contain a *nop-like* special instruction. If the program execution is redirected by an attacker, the nop-like instruction is missing and a control flow protection error is raised.
- Control Flow Enforcement Technology is announced by Intel in June 2016. Release date is unknown.

Intel Software Guard Extension (SGX)



WHAT IS INTEL® SGX?

Intel® Software Guard Extensions (Intel® SGX)

What is Intel SGX?
An Intel architecture extension designed to increase the security of select application code and data, protecting it from disclosure or modification.

Intel processor technologies provide unique capabilities that can improve the privacy, security, and scalability of distributed ledger networks.

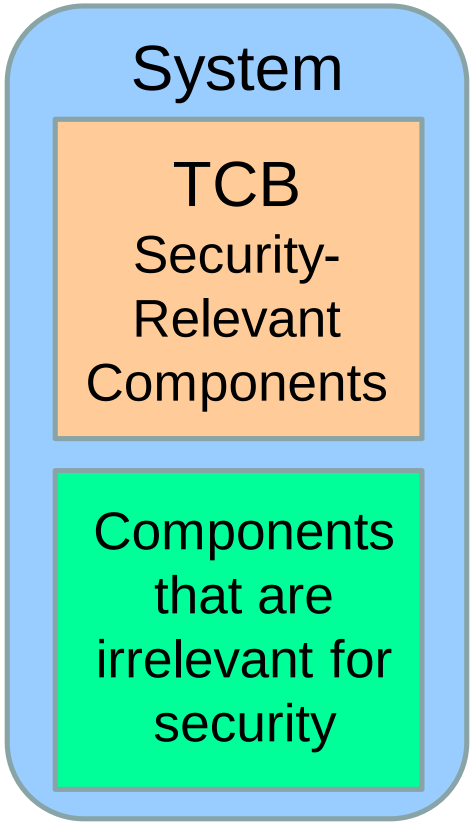
https://newsroom.intel.com/news/intel-microsoft-enterprise-blockchain-service/

TCB – Trusted Computing Base

- The trusted computing base (TCB) of a computer system is the set of all hardware, firmware, and/or software components that are critical to its security, in the sense that bugs or vulnerabilities occurring inside the TCB might jeopardize the security properties of the entire system.
- By contrast, parts of a computer system outside the TCB must not be able to breach the security policy and may not get any more privileges than are granted to them in accordance to the security policy

From TCSEC

Trusted Computer Evaluation Criteria, 1985.


System
TCB
Security-Relevant Components
Components that are irrelevant for security

Platform Virtualization

System software

- Operating system
- Virtual Machine Manager
- BIOS/UEFI
- GPU/NIC/SATA/HDD/EC firmware.

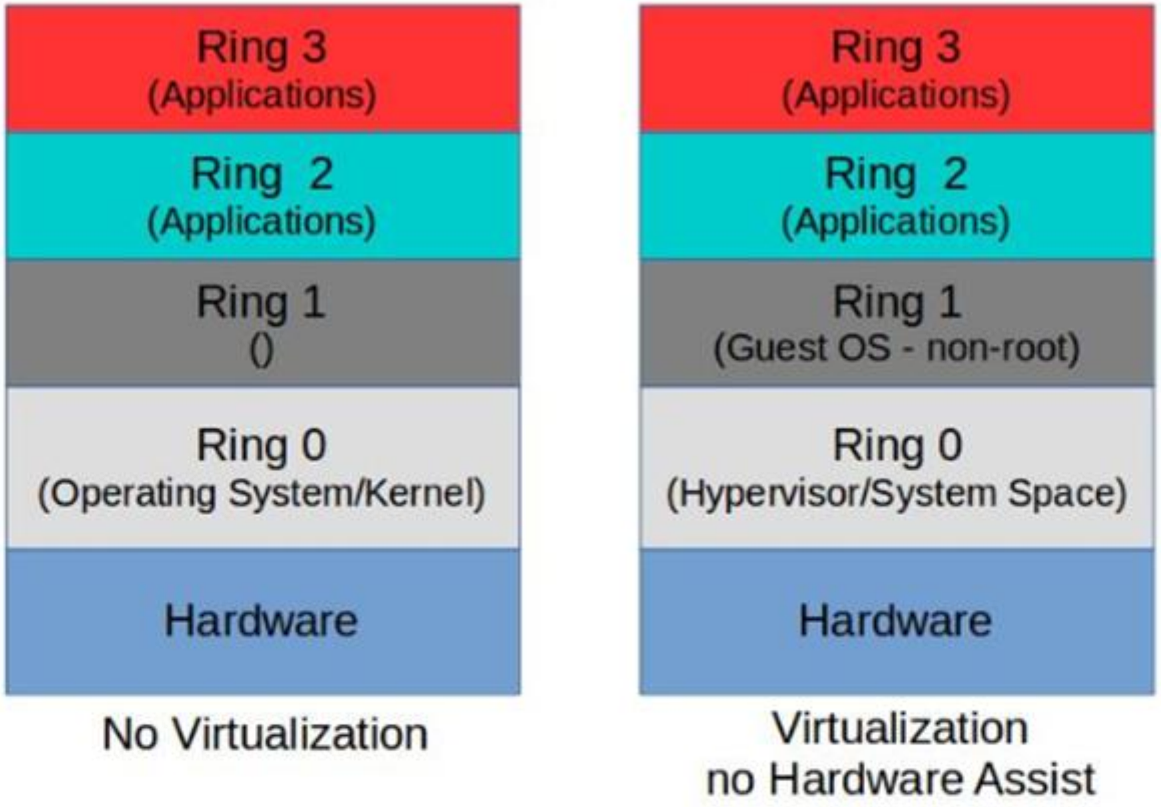
Hypervisor examples of use

- Cloud providers run large server parks
  - Each customer gets its own VM
  - Many customers share the same hardware
  - Migrated VMs between servers to increase/reduce capacity

- Testing and software analysis
  - Potentially damaging experiments can be executed in isolated environment
  - Take a snapshot of the current state of the OS
  - Use this later on to reset the system to that state
  - Malware Analysis

Amazon EC2 Data Centre

Early day Virtualization

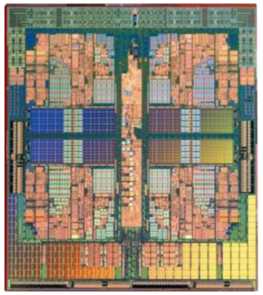


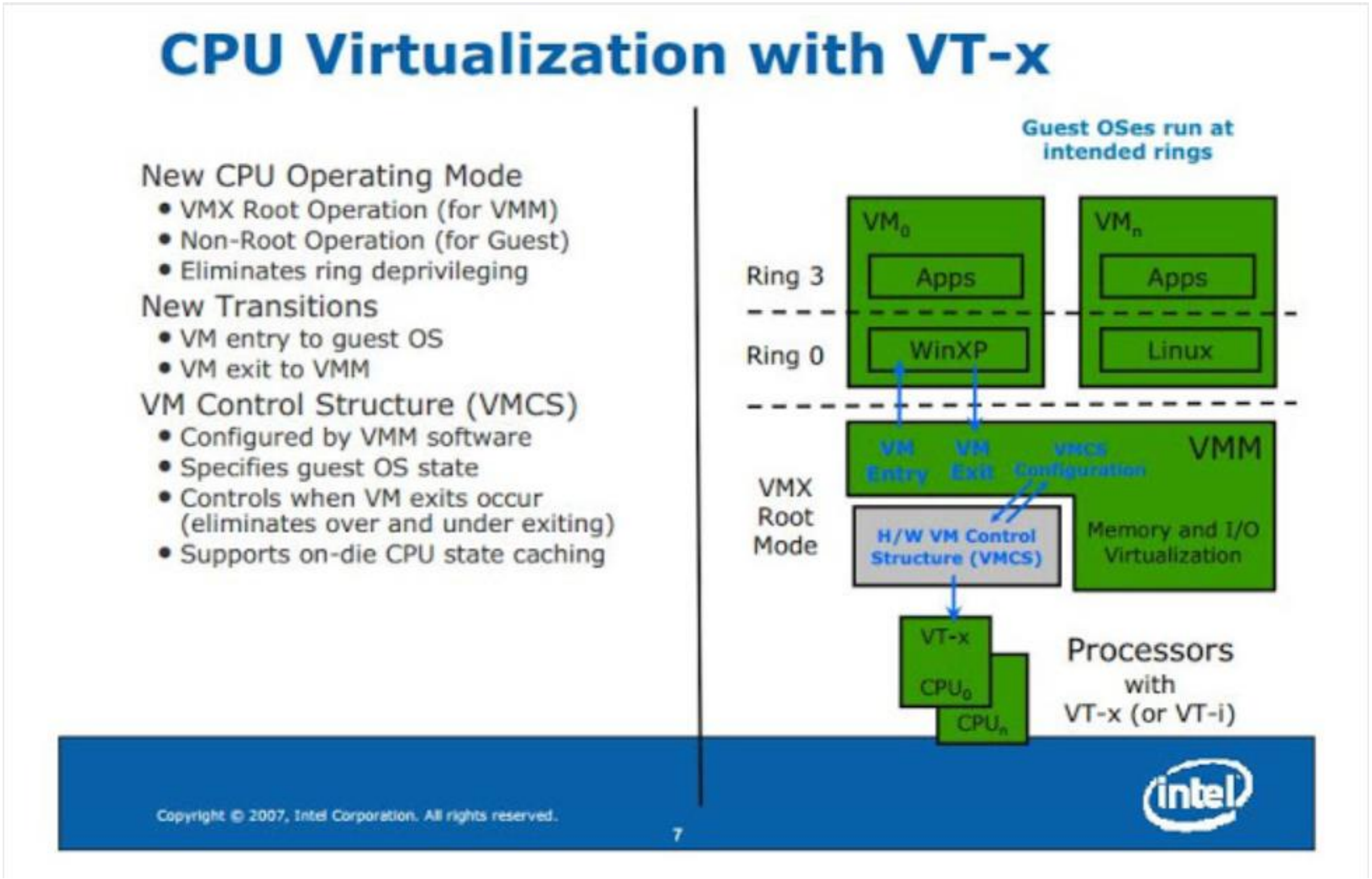
Hardware support for virtualization

- Modern Intel and AMD X86 CPUs support virtualization
  - Intel-VT (Intel Virtualization Technology)
  - AMD-V (AMD Virtualization)
- Must be enabled in BIOS
  - Can be enabled and disabled
  - Computers with single OS typically have virtualization disabled
- Access to data- and code segments for hypervisor can be restricted to processes running in hypervisor mode
- Some instructions are reserved for hypervisor mode

Intel Core i7 CPU

AMD Phenom CPU





CPU Virtualization with VT-x

New CPU Operating Mode
- VMX Root Operation (for VMM)
- Non-Root Operation (for Guest)
- Eliminates ring deprivileging

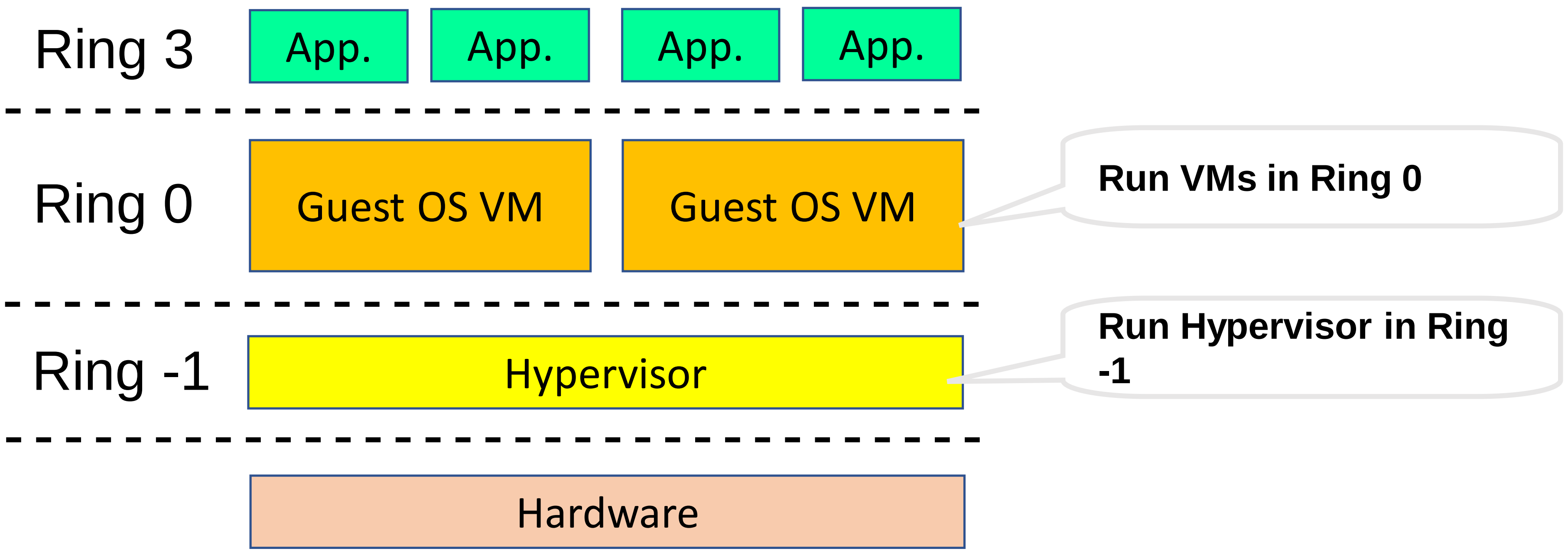
New Transitions
- VM entry to guest OS
- VM exit to VMM

VM Control Structure (VMCS)
- Configured by VMM software
- Specifies guest OS state
- Controls when VM exits occur (eliminates over and under exiting)
- Supports on-die CPU state caching

Guest OSes run at intended rings
Ring 3
Ring 0
VMX Root Mode
VM_0
VM_n
Apps
Apps
WinXP
Linux
VM Entry
VM Exit
VMCS Configuration
VMM
H/W VM Control Structure (VMCS)
Memory and I/O Virtualization
VT-x
CPU_0
CPU_n
Processors with VT-x (or VT-i)

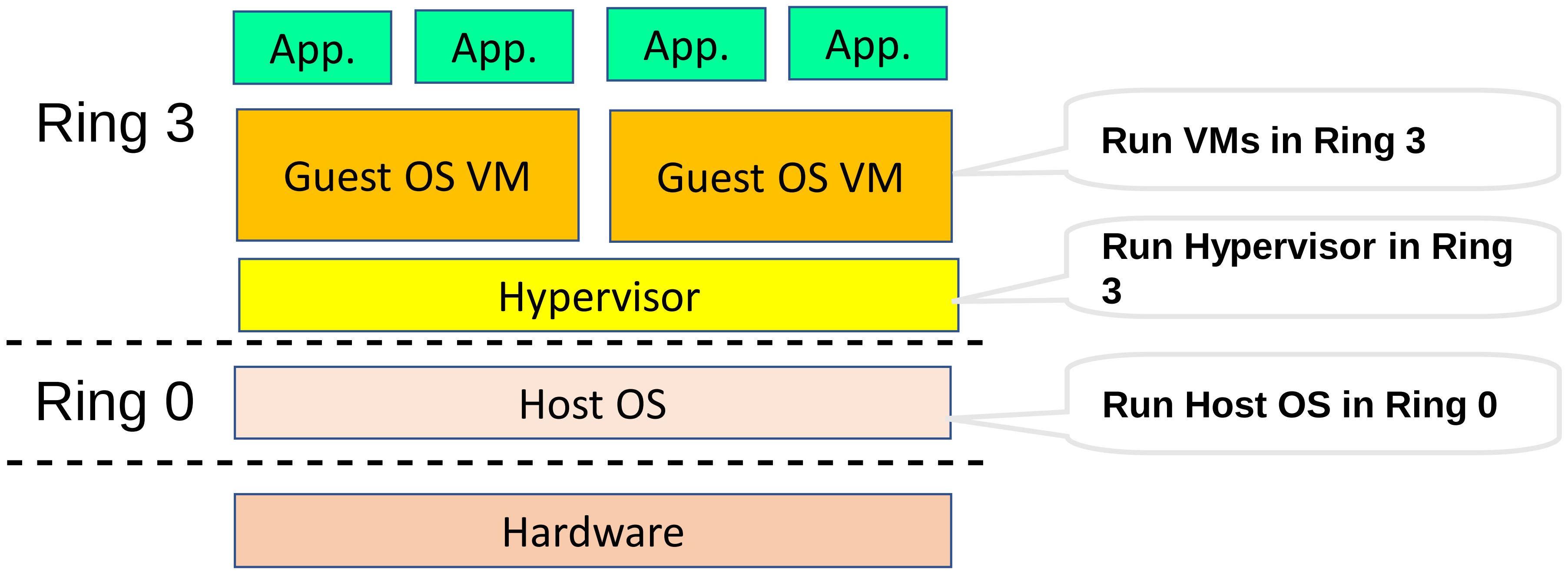
7

Type 1 VM Architecture Ring Allocation



- No host OS
- Hypervisor runs directly on hardware
- High performance
- Traditionally limited GUI, but is improved in modern versions
- HW support can be an issue

Type 2 VM Architecture Ring Allocation



- Hypervisor runs on top of host OS
- Performance penalty, because hardware access goes through 2 OSs
- Traditionally good GUI
- Traditionally good HW support, because host OS drivers available
- Slow performance!

Platform Virtualization Products

Hypervisor types

- Type 1 (Native)
  - Microsoft Hyper-V
  - VMWare ESX
  - Citrix XenServer (Xen)
  - ORACLE VM Server (Xen)
  - Amazon EC2 (Xen)
  - IBM System Z Hypervisor
- Type 2 (Hosted)
  - Microsoft Virtual PC
  - VMWare Workstation
  - Citrix XenClient
  - VirtualBox

Why use platform virtualization

- Efficient use of hardware and resources
  - Improved management and resource utilization
  - Saves energy
- Improved security
  - Malware can only infect the VM
  - Safe testing and analysis of malware
  - Isolates VMs from each other
- Distributed applications bundled with OS
  - Allows optimal combination of OS and application
  - Ideal for cloud services
- Powerful debugging
  - Snapshot of the current state of the OS
  - Step through program and OS execution
  - Reset system state

Hardware-Enabled Security

Motivation

Suppose you have developed a bug free banking user level application-BankApp:

Is it enough guarantee to tell your customers that they will be completely secure?

Part 1: OS level security services.

Once you run BankApp on a platform, its security relies on System software security:

- Operating system
- Virtual Machine Manager
- BIOS/UEFI
- GPU/NIC/SATA/HDD/EC firmware.

Assumptions Vs. Reality

- Assume that the millions of line of code of system software are:
  - Perfectly secure
  - Never backdoored
- What do we know about system software and its security?
  - Complex:
    - Linux: 12 million lines of code
  - vulnerable
  - Backdoored:
    - BIOS
    - Keyboard firmware

Platform security use cases

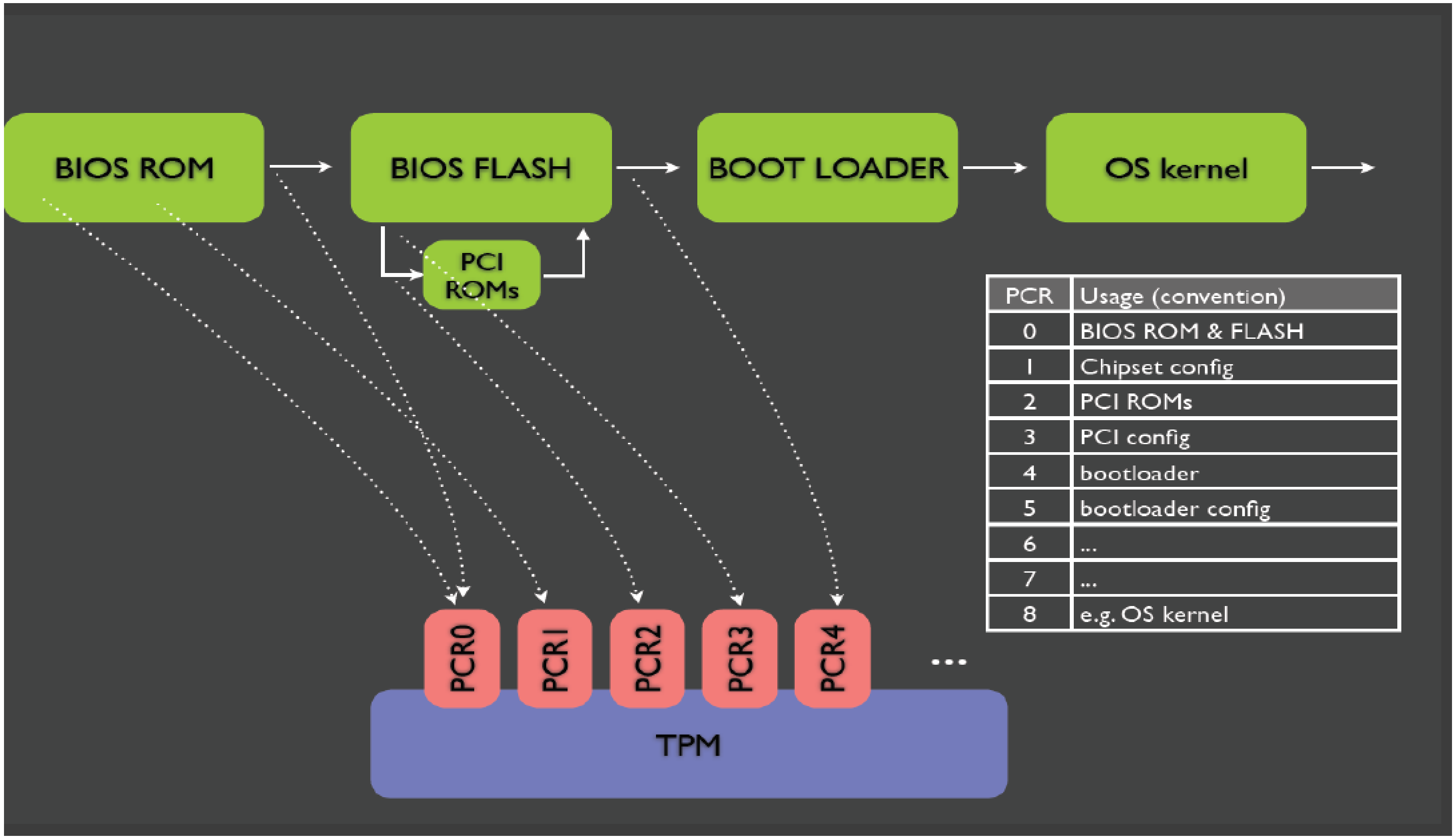
- Local computation:
  - Establish trust in your platform

- Remote computation:
  - Third parties want to establish trust in your platform:
    - Netflix wants to ensure you are not duplicating the content.
    - Health Services to ensure that the doctor’s platform is not going to leak patient’s data.
    - Bank to ensure that it is you who is approving the credit card transaction and not malware.
  - You want to establish trust in service provider platforms:
    - Secure Storage.
    - Secure Computation.

“Computer and Network security, in practice, starts at the hardware and firmware underneath the end points.”

“Their security provides an upper bound for the security of anything built on top”

Trusted Computing



PCR	Usage (convention)
0	BIOS ROM & FLASH
1	Chipset config
2	PCI ROMs
3	PCI config
4	bootloader
5	bootloader config
6	...
7	...
8	e.g. OS kernel

Trusted Computing

- What do we need to implement this securely?
  - Secure implementation of the hashing algorithm
  - A secure immutable storage for the measurements
  - A way to report the final measurement
  - An immutable first code

- Software Vs. Hardware

Trusted Computing

- We don’t trust software, thus:
- To implement such measurement of firmware, some help from hardware is needed for storing and reporting of the measurements reliably, as otherwise malicious software could have forged all the results.

- Hence: Hardware-enabled security Or Trusted Computing

Trusted Computing

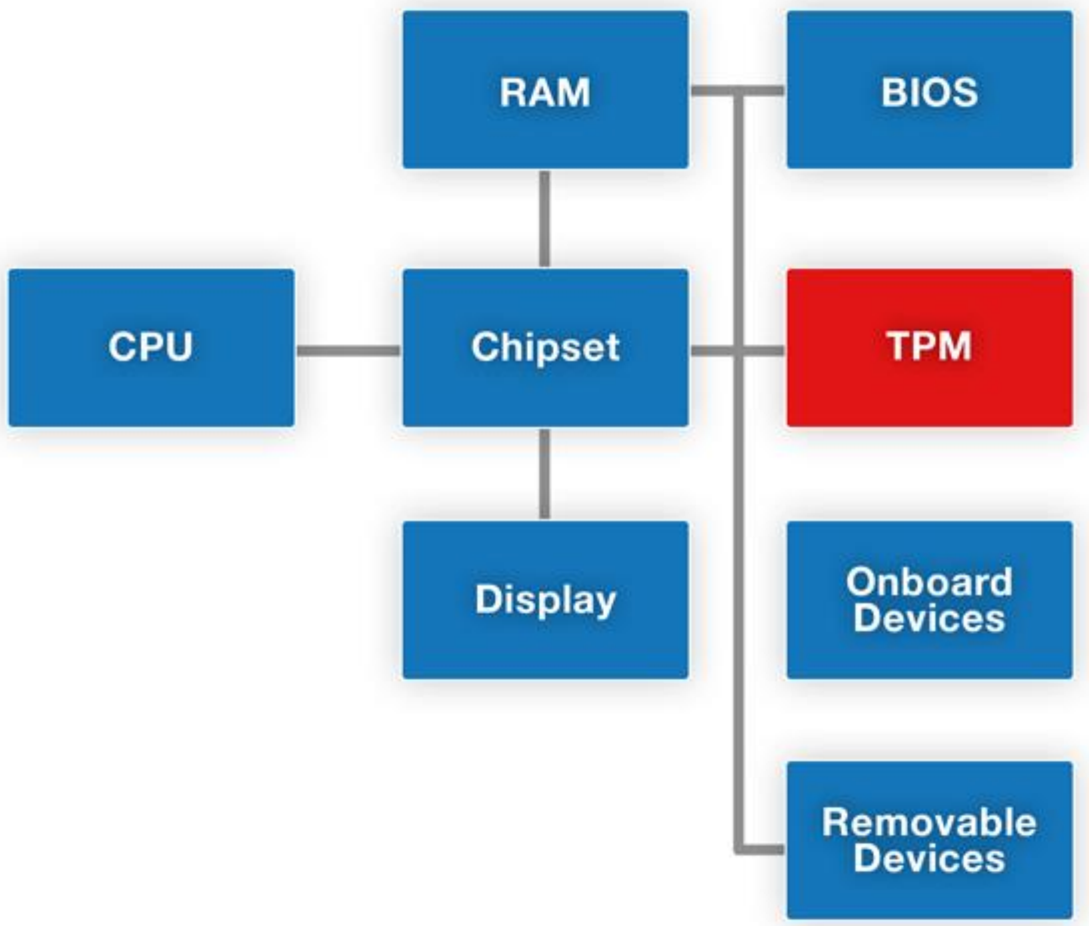
Hardware-rooted mechanism that can allow us to know the state of the platform?

A number that can tell us whether this is a “Good” or “Bad” Platform

Trusted Platform Module

- Typically this is implemented by the TPM
- A passive device usually connected to the southbridge via LPC or a similar bus.
- A TPM device plays the role of a Root of Trust for such measurements .
  - It offers an API to send measurements to it (implemented by so called “PCR Extend” operation) and then provides ways to either: reliably report these (e.g. via the “Quote” operation, which involves signing of the measurement with the known-to-the-tpm-only private key)
  - or conditional release of some secrets if the measurements match a predefined value (the “Seal”/“Unseal” operations).

Pervasiveness of the TPM

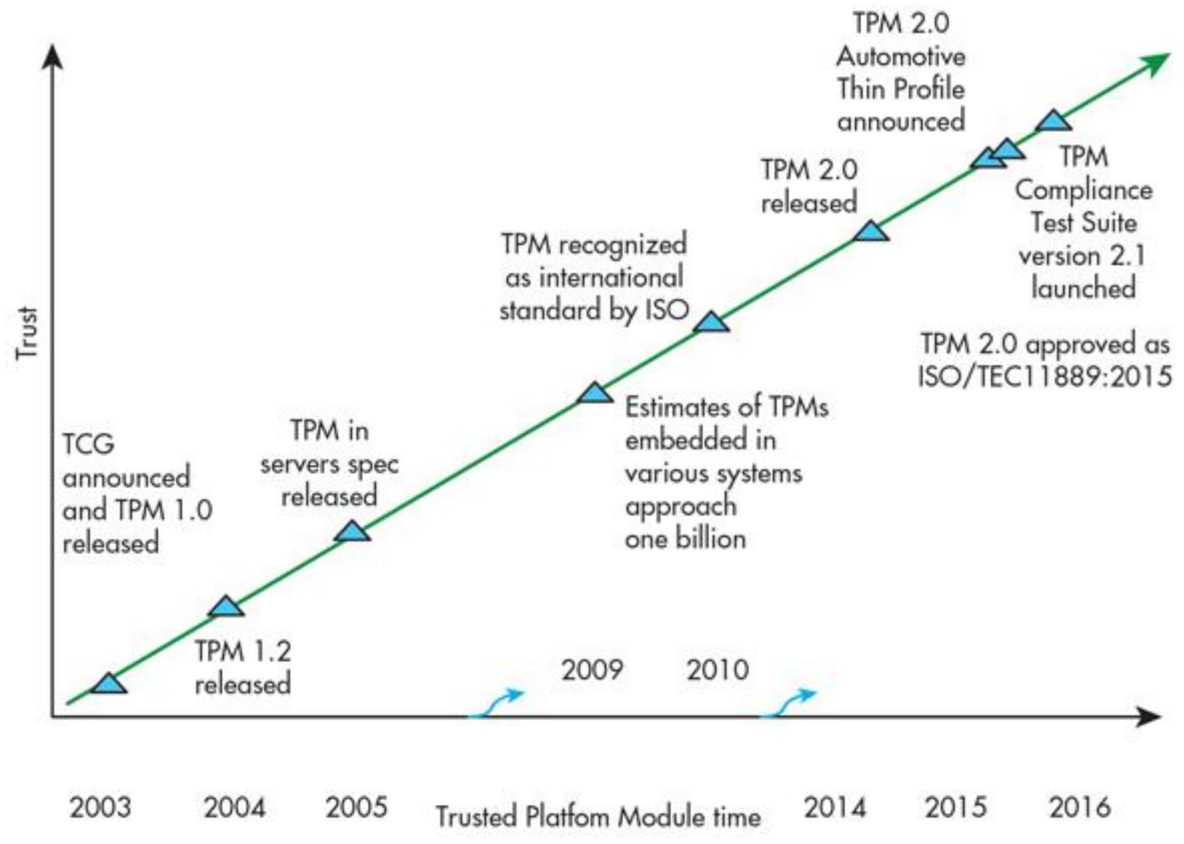


- The TPM chip sits on the motherboard
- Installed in 2 billion devices per 2015
- Relatively obscure technology for most people

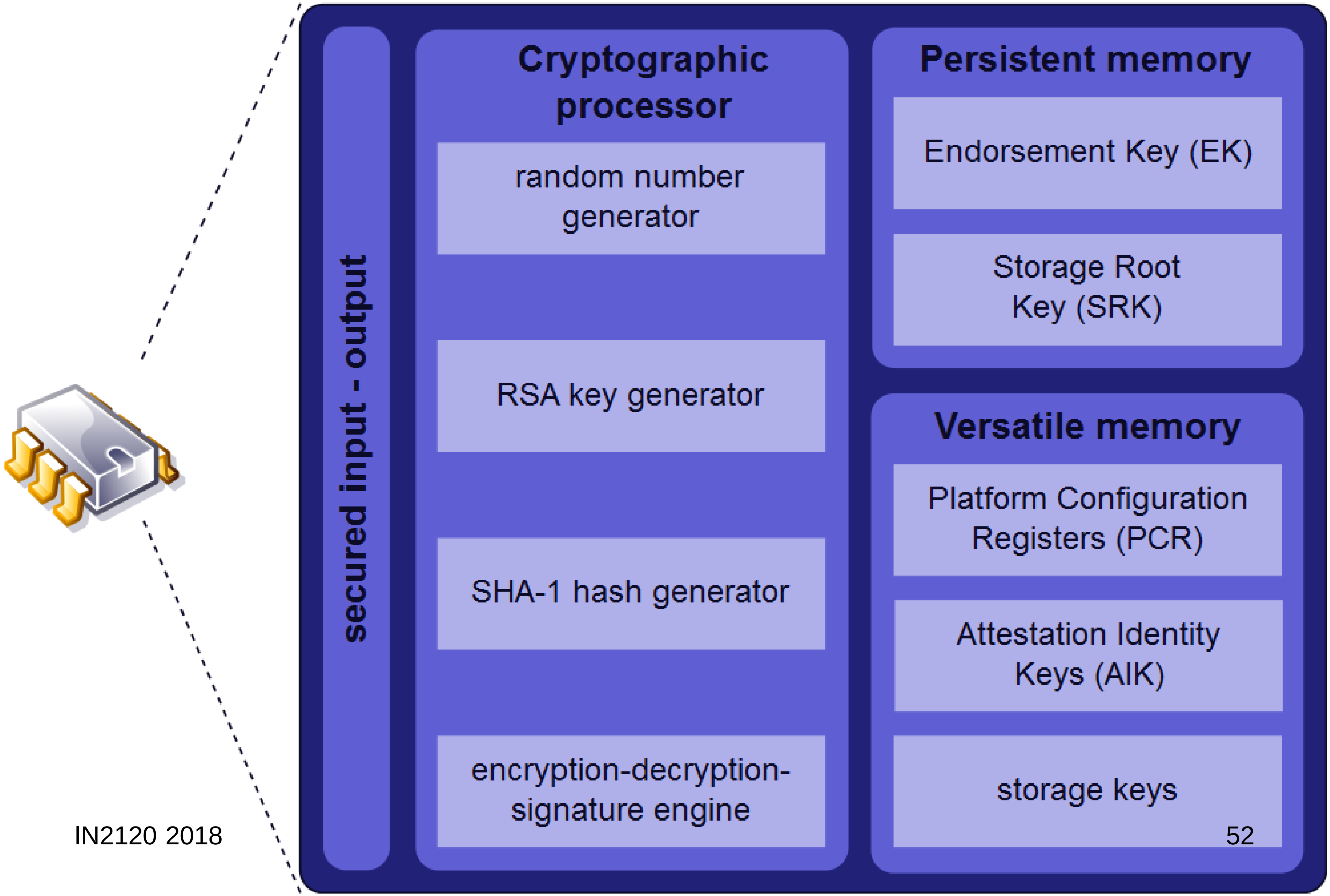
Trusted Computing Group
TCG History & Evolution

- October 1999: TCPA formed
  - Trusted Computing Platform Alliance
  - Founders: IBM, HP, Compaq, Intel and Microsoft
- 2001: 1st TPM specification released
  - Trusted Platform Module
- 2002: TCPA changes its name to TCG
  - Trusted Computing Group
  - Industry standards organization
- 2003: TCPA TPM spec. adopted by TCG as TPM 1.2
- 2012: Draft TPM Specification 2.0 published
  - TPM 2.0 spec. not compatible with TPM 1.2 spec.
- 2015: Official TPM specification 2.0

TPM History
Trusted Platform Module

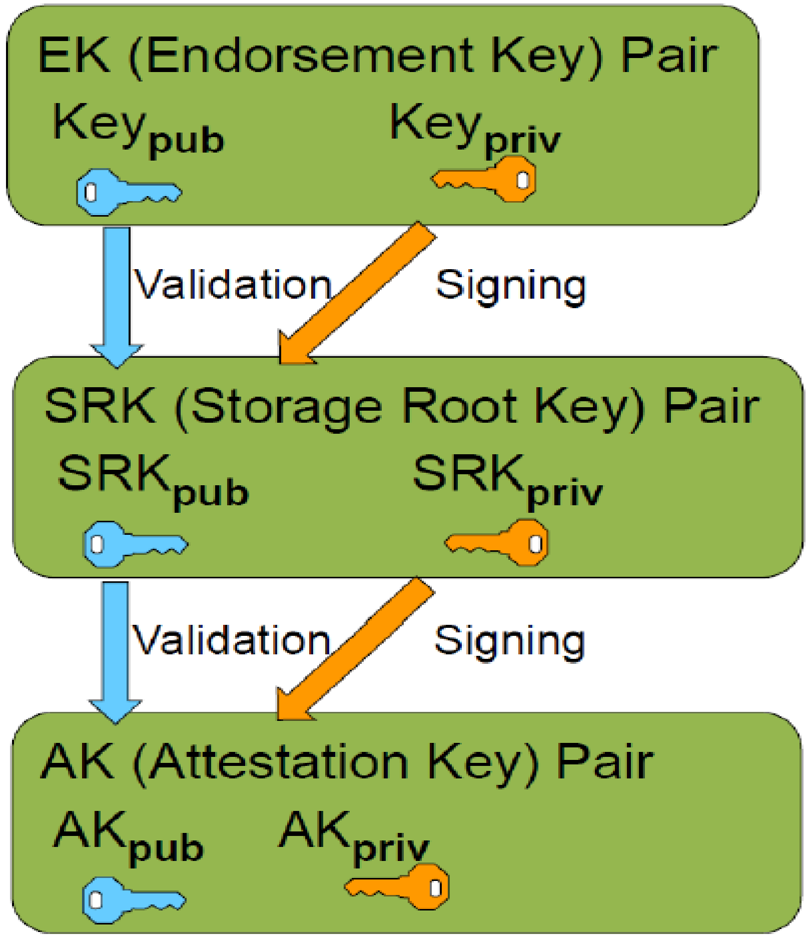

TPM 2.0 Automotive Thin Profile announced
TPM 2.0 released
TPM Compliance Test Suite version 2.1 launched
TPM recognized as international standard by ISO
TPM 2.0 approved as ISO/TEC11889:2015
Trust
Estimates of TPMs embedded in various systems approach one billion
TCG announced and TPM 1.0 released
TPM in servers spec released
TPM 1.2 released
2009
2010
2003
2004
2005
Trusted Platfom Module time
2014
2015
2016

TPM 1.2 Functionality



Key Management

- Endorsement Key (EK)
  - Created once
  - At manufacture time
  - Stored securely in non-volatile memory
- Storage Root Key (SRK)
  - Stored security in non- volatole memory
  - Validated by EK
  - At initiation time
- Attestation Key (AK)
  - Used for remote attestation
  - Validated by SRK
- Custom keys
  - Possible to create additional keys and validate that they are created under SRK.



TPM Keys

- **Endorsement Key (EK):** a pair of RSA keys that is installed when the TPM is manufactured. The public EK value is used to uniquely identify a TPM and will not change during the TPM's lifetime.
- **Storage Root Key (SRK)** is also a pair of RSA keys that is used to encrypt other keys stored outside the TPM. SRK is in effect the *Root of Trust for Storage* (explained later). SRK can change when a new user *takes ownership* of the TPM.

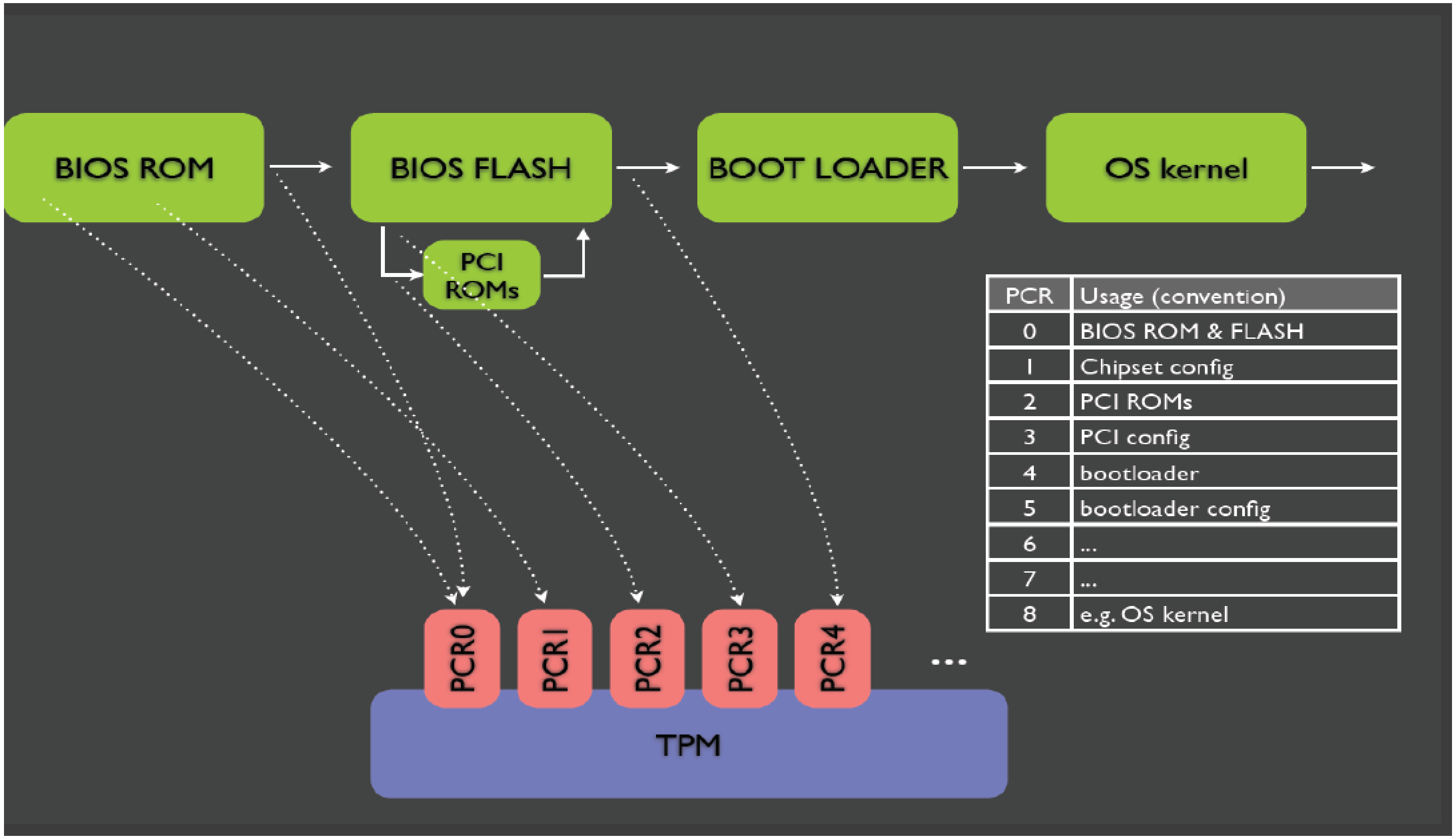
TPM Keys

- **Platform Configuration Register (PCR)**: There are at least 16 PCRs in a TPM. They store platform configuration measurements. These measurements are normally hash values (SHA-1) of entities (applications) running on the platform. PCRs can not be written directly; data is stored by a process called extending the PCR.
- **Attestation Identity Key (AIK)**: In remote attestation, it is important to know that you are communicating with a valid TPM-enabled platform. EK can be used, but it would undermine the privacy of the owner (because of the uniqueness of the EK). Hence an AIK is used instead.

TPM measurement

- A measurement is stored by *extending* a particular PCR:
- The ***extend*** operation is:
- **PCR := SHA-1(PCR + measurement)**
  - It is unfeasible to find 2 different measurement values such that when extended returns the same value.
  - It preservers order in which entities' measurement were extended (extending A then B results in a different value than extending B then A).
  - The operation allows unlimited number of measurement to be stored

Trusted Computing



PCR	Usage (convention)
0	BIOS ROM & FLASH
1	Chipset config
2	PCI ROMs
3	PCI config
4	bootloader
5	bootloader config
6	...
7	...
8	e.g. OS kernel

CRTM

“The Core Root of Trust for Measurement (CRTM) MUST be an immutable portion of the Host Platform’s initialization code that executes up on a Host Platform Reset” TCG

Until recently the CRTM was implemented by the BIOS using .

The normal SPI flash memory. The same flash memory the attacker can usually modify after successfully attacked the BIOS. This has only changed with the latest Intel processors which implement the so called Boot Guard technology.

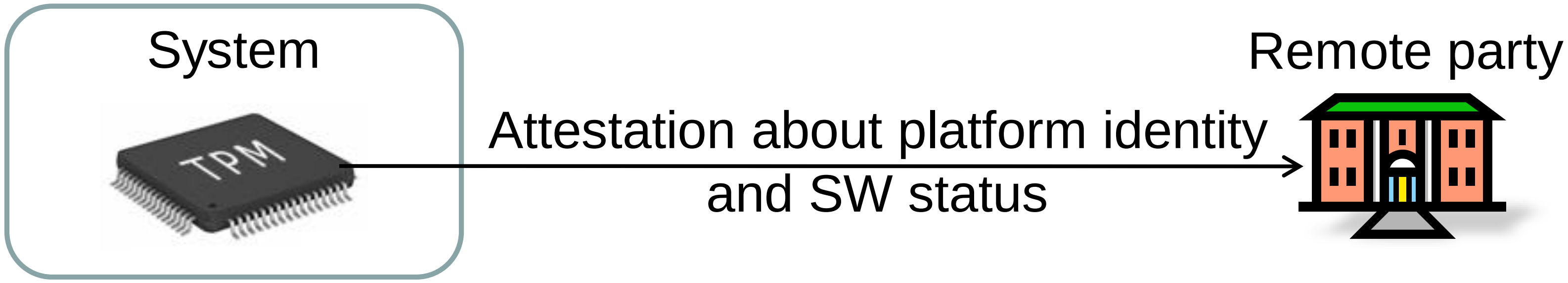
CRTM

- Does this integrity measurement mechanism prevent an entity from misbehaving or being malicious?

- Short answer: No.
- What does TPM Integrity provide?
- An unforgeable record of all the entities that have been loaded. One can choose whether to trust the system based on this record

Remote Attestation

- TPM can certify configuration to others
  - with a digital signature in configuration info
  - giving another user confidence in it
  - Based on Attestation Key (AK)
- Remote parties can validate signature based on a PKI
- Provides hierarchical certification approach
  - trust TPM, then trust the OS, then trust applications

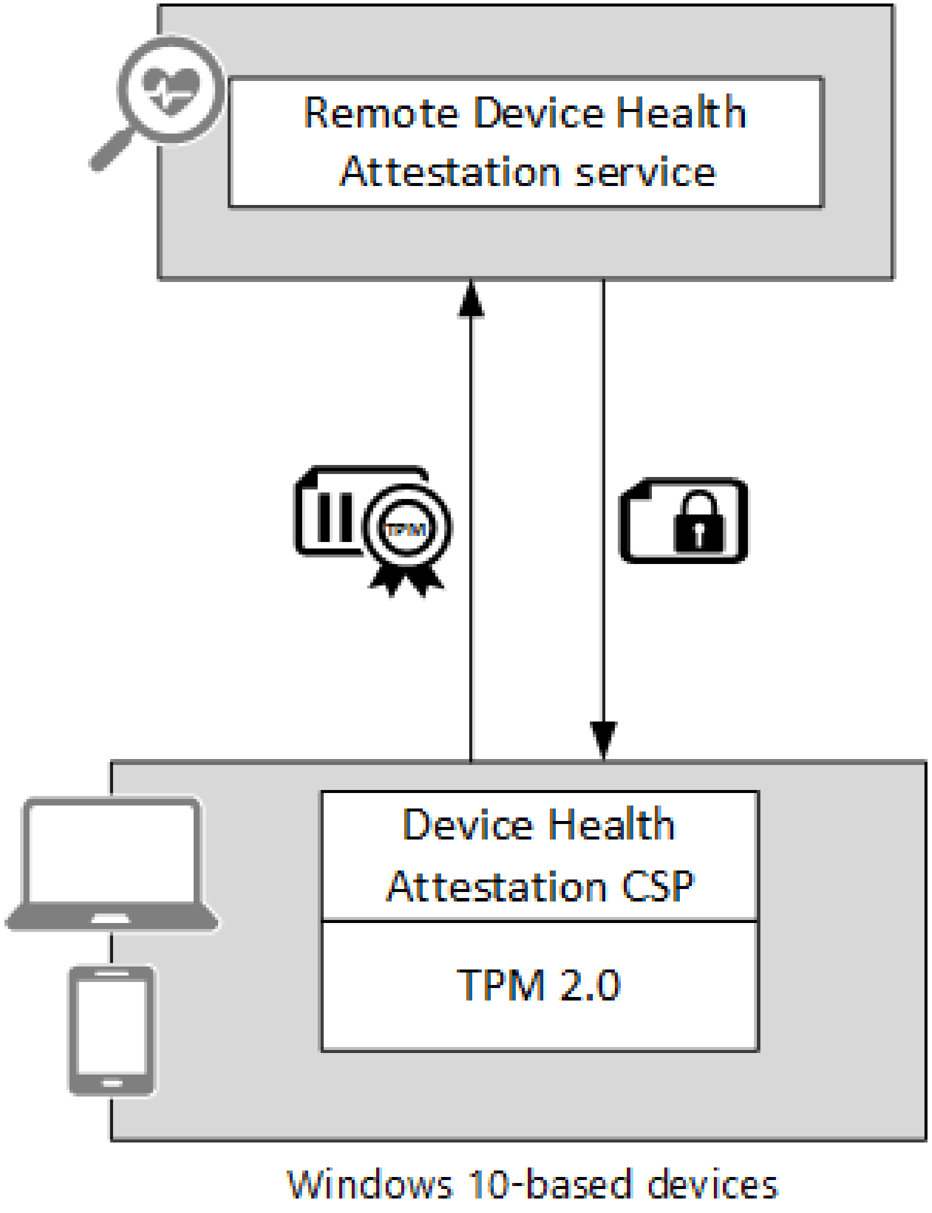


Remote Attestation

- Privacy concerns
  - Benefits of using the AIK for signing: Help prevent cryptanalysis of the EK; Somewhat addresses the privacy issues, since the AIK is not directly associated with the hardware.
  - Risks of using the AIK for singing: It Reveals the unique hardware key (EK) and therefore the identity of the platform.
  - This enables a remote computer to *link* different sessions to the same trusted computer.

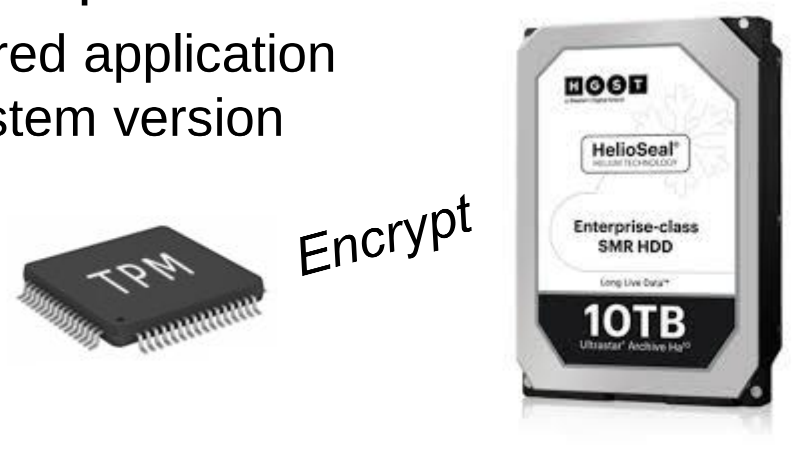
Remote Attestation

- Boot measurement with remote attestation


Remote Device Health Attestation service
Device Health Attestation CSP
TPM 2.0
TPM
Windows 10-based devices

Sealed Storage / Encryption

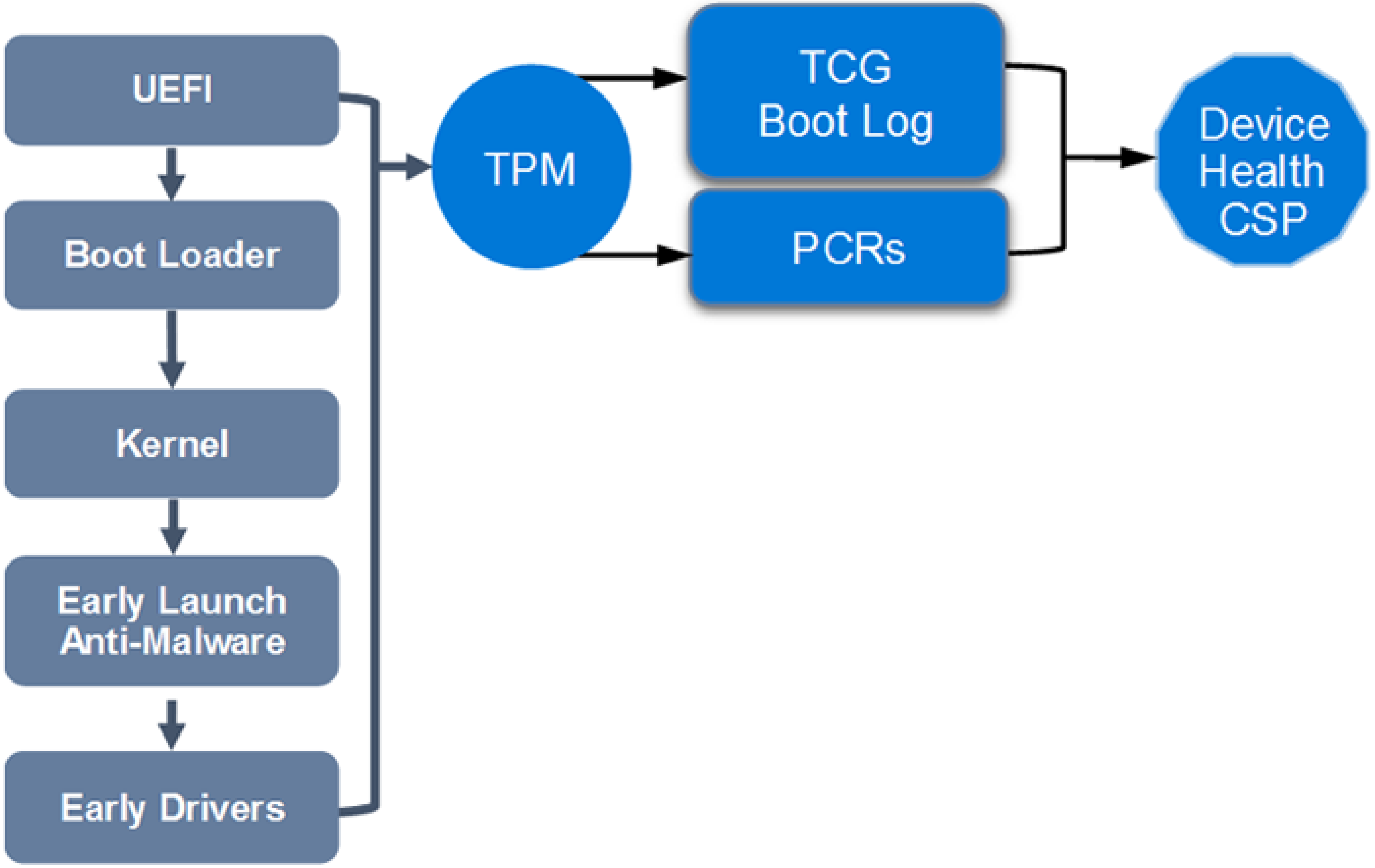
- Encrypts data so it can be decrypted
  - by a certain machine in given configuration
- Depends on
  - Storage Root Key (SRK) unique to machine
  - Decryption only possible on unique machine
- Can also extend this scheme upward
  - create application key for desired application version running on desired system version
- Supports disk encryption



Summary of TPM Services

- Measurement
- Sealing
- Remote Attestation

Trusted Boot



Secure Boot

- Secure Boot
- What is referred to as “UEFI Secure Boot” is not really a new hardware technology, but rather merely a way of writing the BIOS in such a way that it hands down execution only to code, typically OS loader or kernel, which meets certain requirements.
  - These requirements are checked by validating if the hash of this next block of code is signed with a select key, itself signed with a key provided by the OEM (called a Platform Key).

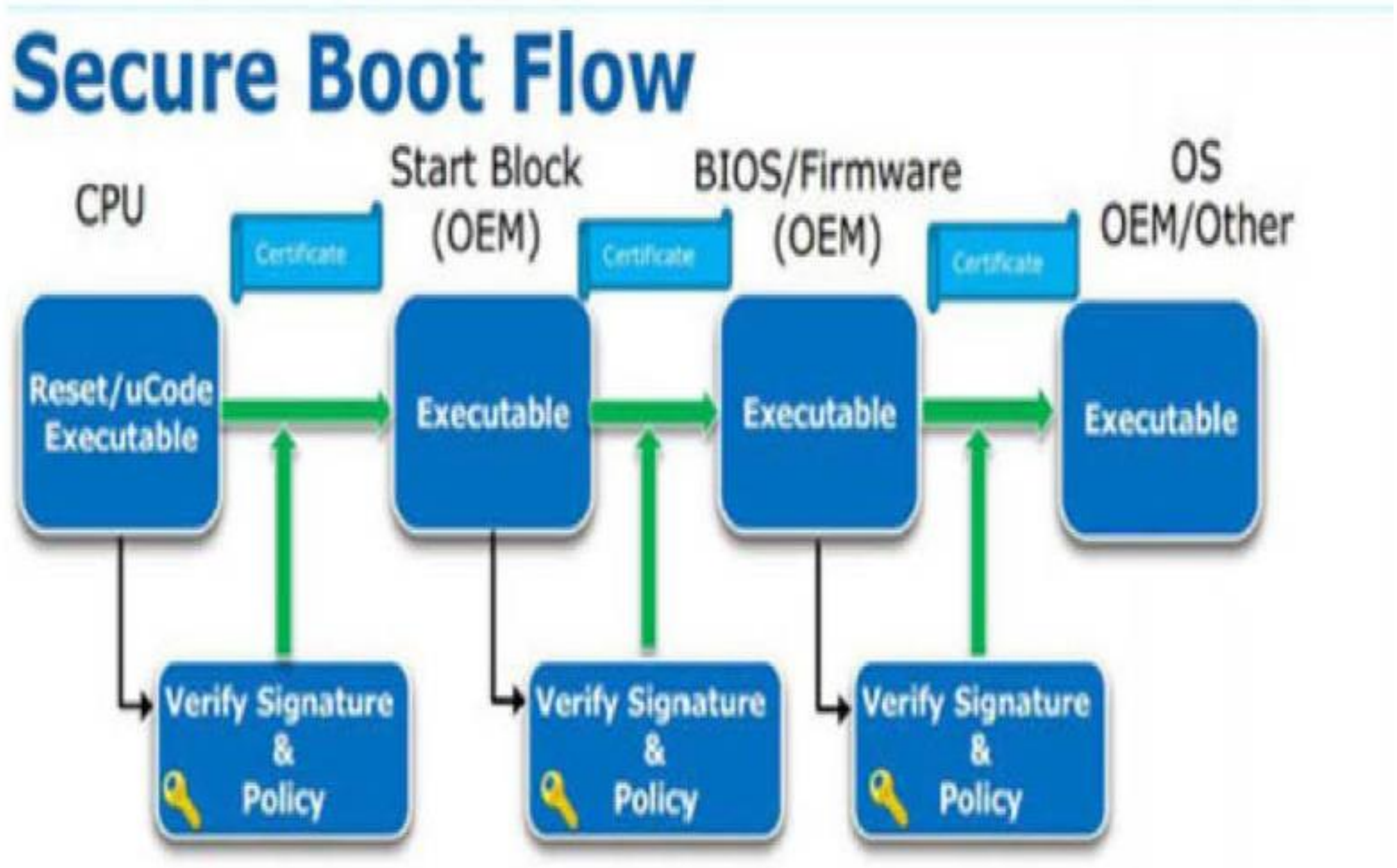
Secure Boot and Backdoors

- Imagine the BIOS vendor was asked to provide a backdoor to local law enforcement
- All the vendor would have to do would be to sign an additional certificate, thus allowing “alternative” code to be booted on the platform.
- Such “additionally” authorized code could then be immediately used to implement perhaps an Evil Maid attack against the user’s laptop.

Secure Boot and Backdoors

A way to protect against such backdooring would be to combine Secure Boot with measurements stored in a TPM, as discussed previously. But then it becomes questionable if we really need this whole complex UEFI secure boot scheme in the first place? Except, perhaps, to limit the user’s freedom of choice with regards to the OS she wants to install on her laptop.

Secure Booy


Secure Boot Flow
CPU
Start Block (OEM)
BIOS/Firmware (OEM)
OS OEM/Other
Certificate
Certificate
Certificate
Reset/uCode Executable
Executable
Executable
Executable
Verify Signature & Policy
Verify Signature & Policy
Verify Signature & Policy

Challenges of TPM Remote Attestation

Static Trusted Boot:

- The problem of maintaining a long chain of trust (discussed further down).

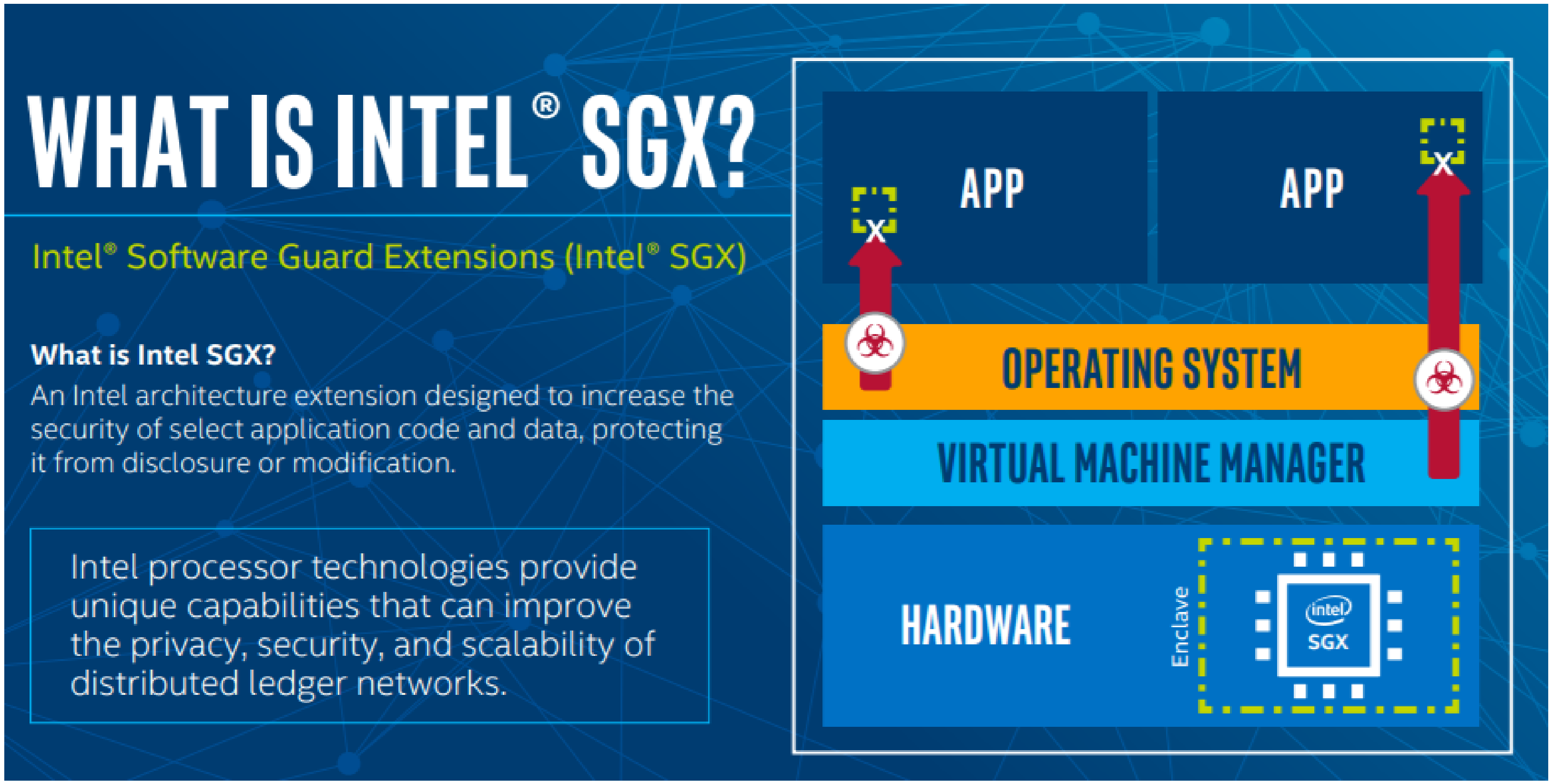
- Different vendors and configuration.
  - How can we interpret the final measurements.

→ Need to reduce the TCB

Reduce TCB

- ArmTrustZone
- Intel SGX
- OffPAD

Intel Software Guard Extension (SGX)



WHAT IS INTEL® SGX?

Intel® Software Guard Extensions (Intel® SGX)

What is Intel SGX?
An Intel architecture extension designed to increase the security of select application code and data, protecting it from disclosure or modification.

Intel processor technologies provide unique capabilities that can improve the privacy, security, and scalability of distributed ledger networks.

https://newsroom.intel.com/news/intel-microsoft-enterprise-blockchain-service/

Trusted Execution Environments

- Isolation
- Secrets
- Imutable memory

Hardware-enabled security

- Has been used since the 1970’s
  - Mobile ecosystem:
    - Secure elements:
      - SIM cards
      - Smart cards
  - Desktop ecosystem
    - Cryptographic co-processor

Hardware-enabled security

- Classification
  - Removable:
    - Secure elements
  - Embedded:
    - Trusted Platform Module
  - CPU-Based:
    - Intel SGX
    - ARM TrustZone

- Considerations
  - Cost
  - Performance
  - Usability
  - Isolation

End of lecture