Microsoft

Azure Confidential Computing powered by AMD SEV-SNP Workshop

V1.3 – July 2023
aka.ms/Azure-CC-AMD-Workshop

Workshop overview

Azure Confidential Computing powered by AMD SEV-SNP - Workshop

> In this technical workshop, you will gain a comprehensive understanding of Azure's Confidential Computing capabilities. We will take you on a journey through deploying confidential VMs, generating attestations, validating report authenticity, and deploying confidential containers on Azure Container Instances and Azure Kubernetes Services. To ensure you are well-prepared, we have a dedicated section that walks you through the installation of all necessary requirements.

Workshop agenda

🌅 Morning (9:00 – 12:00)

> *Focus: Introduction and first steps*

- 09:00 – 09:30: Introduction to Confidential Computing
- 09:45 – 10:00: Azure Confidential Computing Technologies
- 10:00 – 10:15: Azure Confidential VMs with AMD SEV-SNP
- 10:15 – 10:30: Break
- 10:30 – 11:00: Hands-on lab: Deploying a confidential VM
- 11:00 – 11:30: Overview of Attestation in AMD SEV-SNP
- 11:30 – 12:00: Hands-on lab: Generating and validating attestation report

🍽️ Lunch Break (12:00 – 13:00)

🌇 Afternoon (13:00 – 16:30)

- 13:00 – 13:45: Confidential Containers with ACI
- 13:45 – 14:00: Break
- 14:00 – 15:00: Hands-on lab: Deploying confidential containers on ACI
- 14:45 – 15:00: Break
- 15:00 – 15:30: Confidential Containers on AKS

Preparation

> This section is crucial to ensure a smooth run of the hands-on labs. If you're only here for the presentations, feel free to skip this.

Azure subscription and deployments

Please ensure that you have an active Azure subscription. If not, you can create one here.

Environment setup

Below we list all the requirements for this workshop. Be sure to set up everything in advance to avoid delays.

1️⃣ Installation Requirements

Please follow the steps outlined in the requirements notebook to set up your environment, which includes:

- Installing Azure CLI on Windows.
- Installing Visual Studio Code.
- Installing Docker Desktop and creating a Docker account.
- Setting up Windows Subsystem for Linux (WSL). You can follow these instructions from Microsoft.

Notebooks

- 💪 Confidential VM - Deploying a confidential VM on Azure.
- 💪 Confidential VM Attestation - Generating an attestation and validating its authenticity.
- 💪 Confidential VM Attestation Deepdive - Fetching and verifying a raw sev-snp report
- 💪 Confidential ACI - Deploying a confidential container on ACI.
- 💪 Confidential ACI Deepdive - Checking CCE policy enforcement on ACI.
- 💪 Confidential ACI Attestation Deepdive - Verifying sev-snp report using virtee/sevtool

aka.ms/Azure-CC-AMD-Workshop

Goal for today

Theory

- Introduction to confidential computing
- Azure Confidential Computing Technologies
- Azure Confidential VM with AMD SEV-SNP
- Overview of Attestation in AMD SEV-SNP
- Confidential Containers with ACI (Azure Container Instances)
- Confidential Containers on AKS (Azure Kubernetes Service)

Hands-on Lab

- Deploying a confidential VM on Azure
- Generating and validating attestation report
- Deploying confidential containers on ACI enforced with a CCE policy

Plan for today

Time (CET)	Topic
09:00 – 09:30	Introduction to Confidential Computing
09:45 – 10:00	Azure Confidential Computing Technologies
10:00 – 10:15	Azure Confidential VMs with AMD SEV-SNP
10:15 – 10:30	Break
10:30 – 11:00	Hands-on lab: Deploying a confidential VM
11:00 – 11:30	Overview of Attestation in AMD SEV-SNP
11:30 – 12:00	Hands-on lab: Generating and validating attestation report • **Part 1: Using Microsoft Azure Attestation** • **Part 2: Using AMD SEV-Tool**
12:00 – 13:00	Lunch break
13:00 – 13:45	Confidential Containers with ACI
13:45 – 14:00	Break
14:00 – 15:00	Hands-on lab: Deploying confidential containers on ACI • **Part 1: Generating a CCE policy** • **Part 2: Validating attesting report using VirTEE SNP-Guest tool** • **Part 3: Demonstrating the CCE policy**
15:00 – 15:30	Confidential Containers on AKS

Acronym

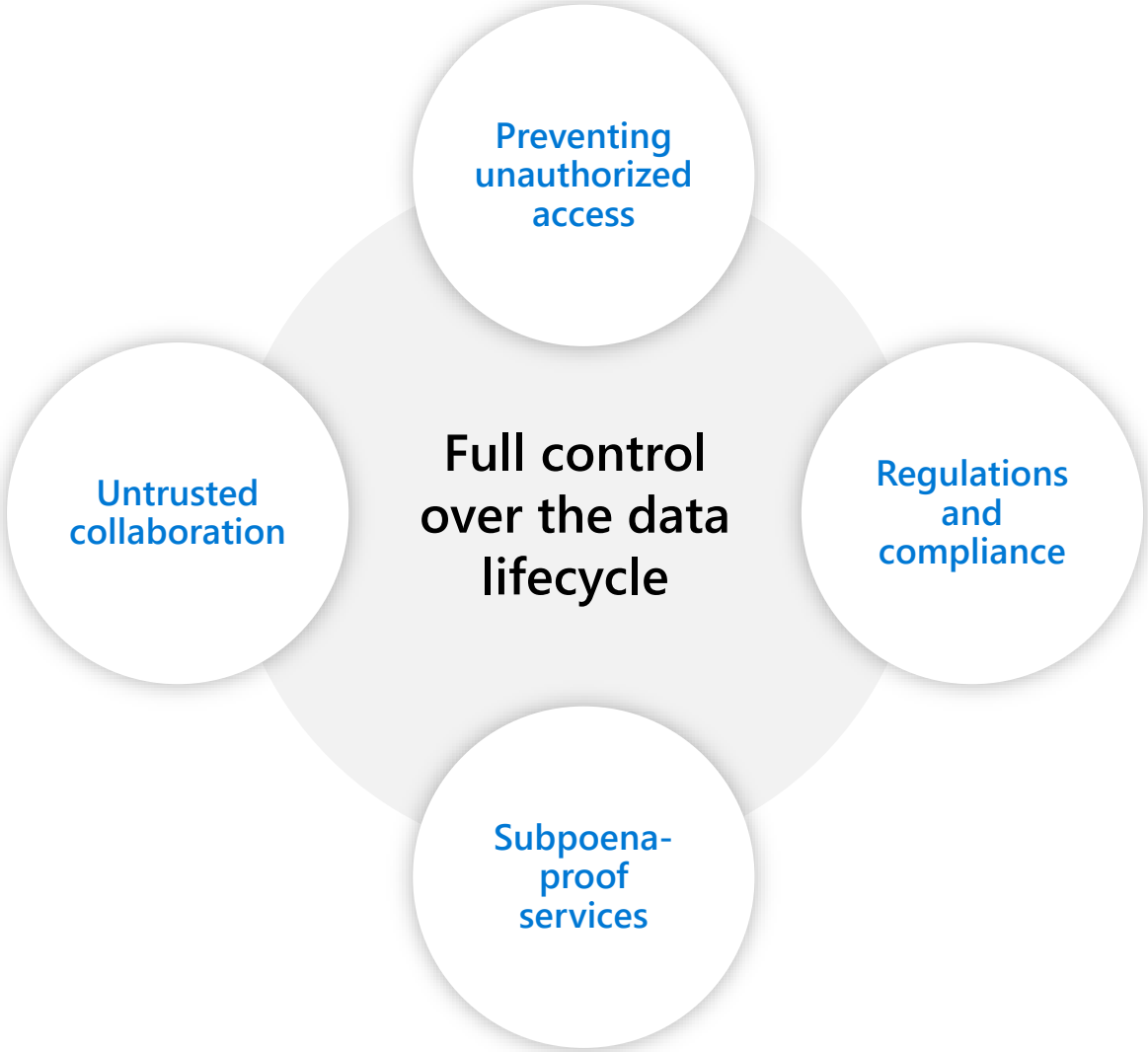
- **HCL**: Host Compatibility Layer
- **OCI**: Open Container Initiative
- **Open GCS**: Open Guest Compute Service
- **PSP**: Platform Security Processor
- **RoT**: Root of Trust
- **SEV**: Secure Encrypted Virtualization
- **SKR**: Secure Key Release
- **SNP**: Secure Nested Paging
- **SVSM**: Secure Virtual Machine Module
- **TCB**: Trusted Computing Base
- **TEE**: Trusted Execution Environment
- **THIM**: Trusted Hardware Identity Management
- **TPM**: Trusted Platform Module
- **UVM**: Utility VM
- **VCEK**: Versioned Chip Endorsement Key
- **VMPL**: Virtual Machine Privilege Levels

Introducing Confidential Computing

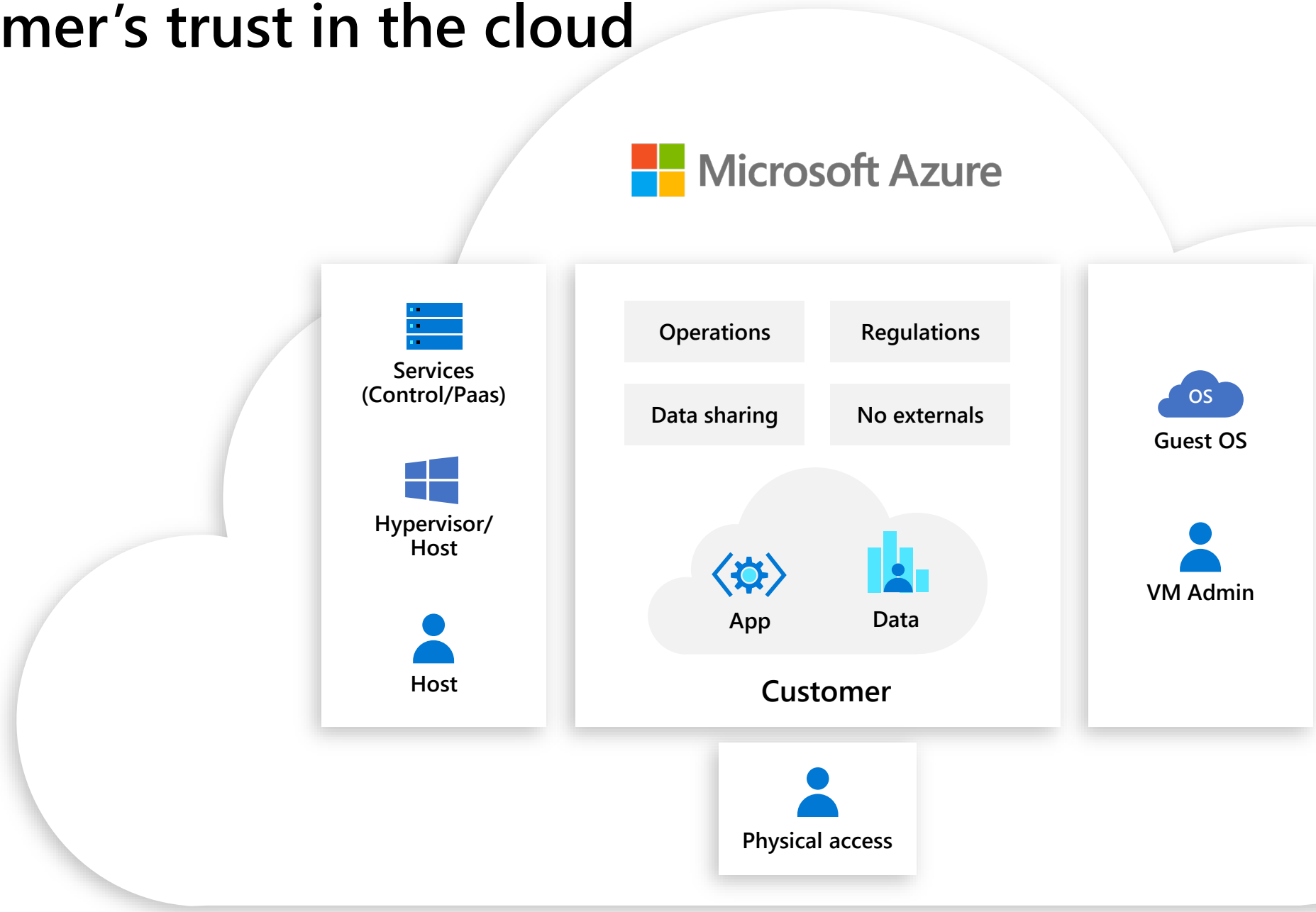
Customer motivations

Customers are increasingly looking for ways to trust as little as possible

Many workloads will not move to cloud due to lack of trust



Customer's trust in the cloud



Data protection

EXISTING ENCRYPTION

Data at rest

Encrypt inactive data when stored in blob storage, database, etc.

Data in transit

Encrypt data that is flowing between untrusted public or private networks

CONFIDENTIAL COMPUTING

In use

Protect/encrypt data that is in use, while in RAM and during computation

Confidential Computing Consortium

The Confidential Computing Consortium (CCC) brings together hardware vendors, cloud providers, and software developers to accelerate the adoption of Trusted Execution Environment (TEE) technologies and standards

A community focused on projects securing data in use and accelerating the adoption of confidential computing through open collaboration

Microsoft is a founding member, and chairs both the governing board and the Technical Advisory Council (TAC) of this open source community

@ConfidentialC2

PREMIER

Microsoft
intel
Red Hat
arm
HUAWEI
ORACLE
Google
Alibaba Cloud
FACEBOOK

GENERAL

AMD
ByteDance 字节跳动
cosmian
swisscom
Anqlave
Baidu 百度
Fortanix
OASIS LABS
anjuna
decentriq
NVIDIA
KINDITE
Tencent 腾讯
vmware

*Other names and brands may be claimed as the property of others.

Azure confidential computing portfolio

Item	Description	Purpose
Virtual machines	Multiple options from AMD and Intel support confidential computing.	Create confidential computing from raw virtual machines.
Microsoft Azure Attestation	The remote attestation service in Azure	Use for validating the trustworthiness of multiple TEEs and verifying the integrity of the binaries running inside the TEEs.
Azure Key Vault Managed HSM	A cloud service that enables you to safeguard cryptographic keys for your cloud applications	Store and manage your private keys in a fully managed, highly available, single-tenant, standards-compliant service that uses FIPS 140-2 Level 3 validated hardware security modules (HSM).
Trusted Launch	A feature set in Generation 2 VMs that brings hardened security features to virtual machines in Azure	Protect against boot kits, rootkits, and kernel-level malware with a secure boot and virtual trusted platform module with boot integrity monitoring.
Confidential Containers on ACI	Use AMD SEV-SNP with ACI	Protect the confidentiality of the container's processes and data, in case someone gains unauthorized access to the underlying host operating system.
App-enclave aware containers	Use Intel SGX with Azure Kubernetes Service (AKS)	Create confidential computing nodes on AKS to isolate applications within an enclave environment.
Confidential VM node pools on AKS	Container node pools that take advantage of VMs that use a hardware-based TEE	Confidential VMs using AMD container applications deny the hypervisor and other host-management code access to VM memory and state and add defense in depth protections against operator access.
THIM	The Trusted Hardware Identity Management service handles cache management of certificates for all trusted execution environments (TEEs) that reside in Azure.	It provides trusted computing base (TCB) information to enforce a minimum baseline for attestation solutions.

Azure Confidential Computing Technologies

Azure confidential computing

Existing encryption

Data at rest

Encrypt inactive data when stored in blob storage, database, etc.

Data in transit

Encrypt data that is flowing between untrusted public or private networks

Confidential computing

Data in use

Protect/encrypt data that is in use, while in RAM, and during computation

Protect Against

Insider Threat

Privileged admins abusing rights

Cloud Components

Removing need to trust Hypervisor/OS

Third Parties

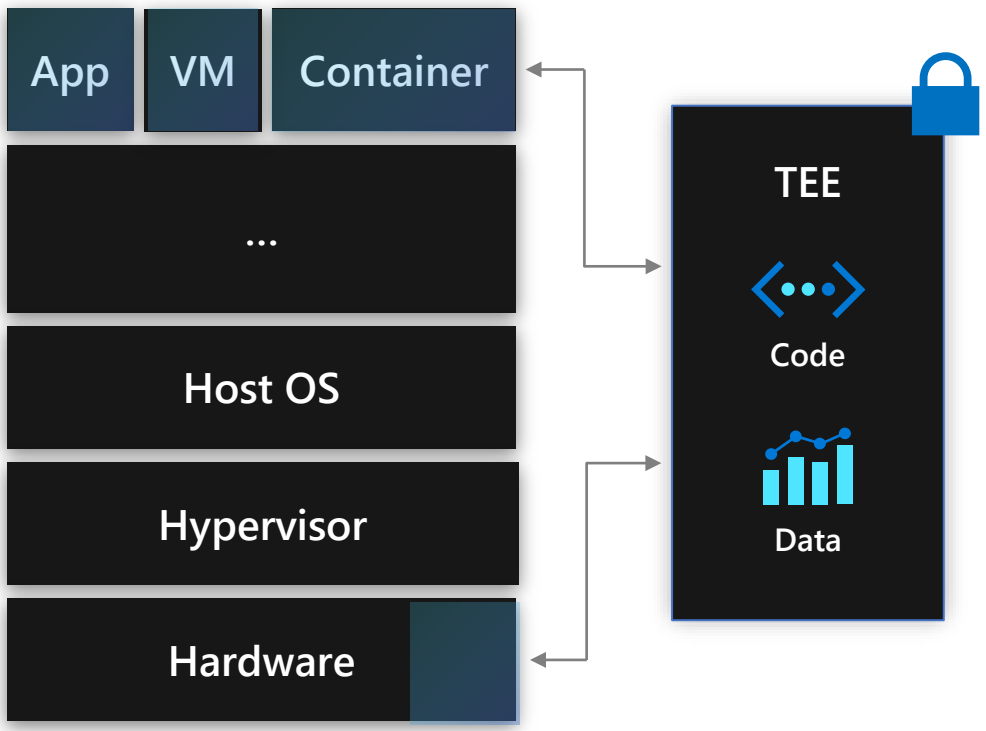
Accessing data without customer consent

In Azure, confidential computing means…

A ***hardware root-of-trust***, ***customer verifiable remote attestation***, and ***memory encryption***

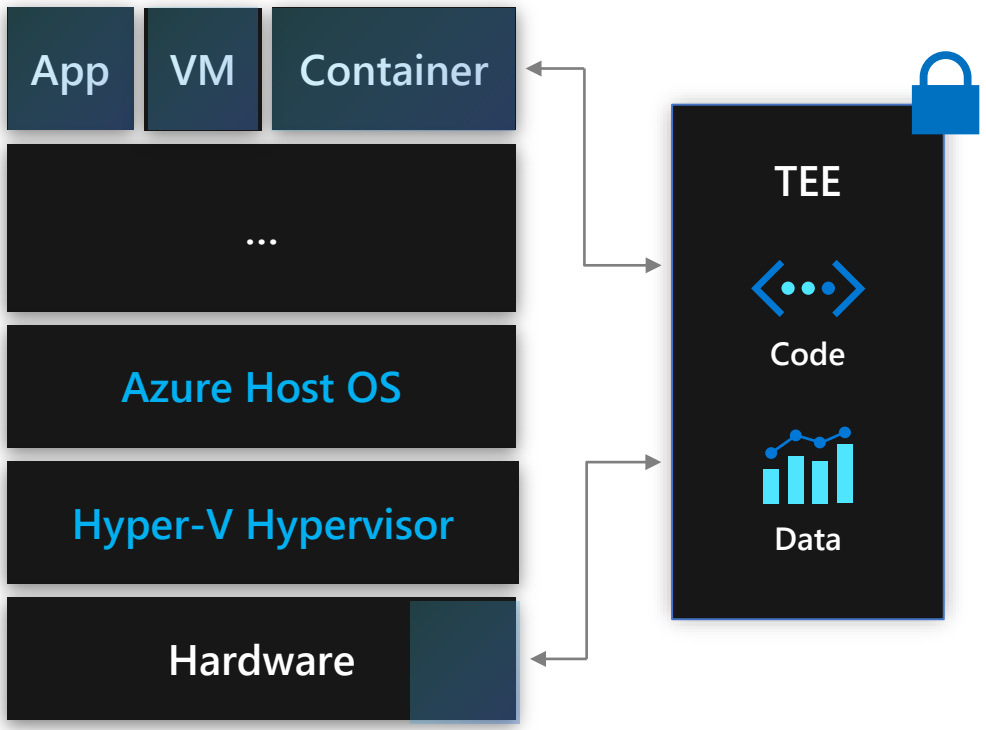
What is Confidential Computing?

The ***protection of data in use*** by performing computation in a hardware-based, attested Trusted Execution Environment (TEE) which provides data integrity, data confidentiality, and code integrity.



What is Confidential Computing?

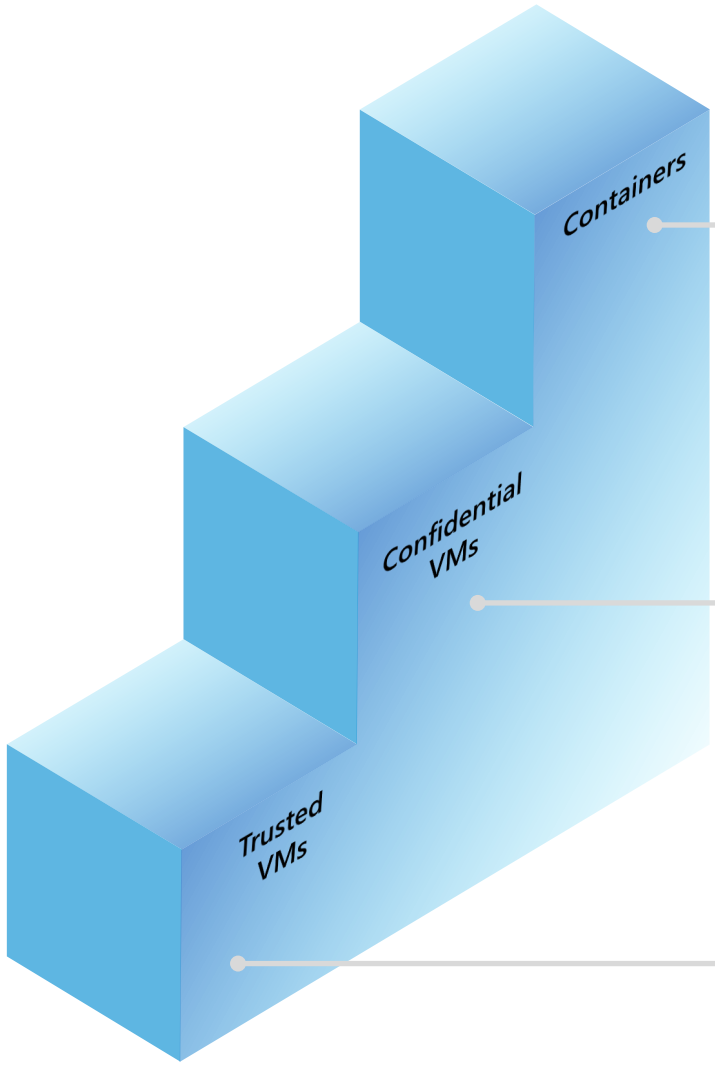
The ***protection of data in use*** by performing computation in a hardware-based, attested Trusted Execution Environment (TEE) which provides data integrity, data confidentiality, and code integrity.



At Microsoft, we evolved our virtualization stack so that a Confidential VM can be:

executed inside a TEE, whereby code and data within the entire VM is protected from the **hypervisor** and the **host OS**, as well as from other confidential VMs and **any hosting environment** in the TEE.

Azure confidential computing and AMD SEV-SNP

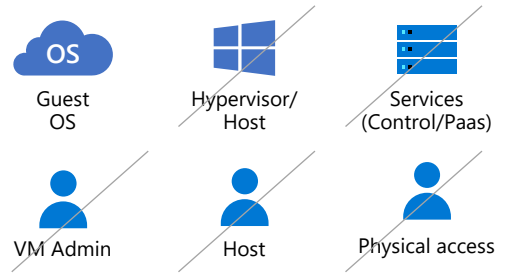


AVAILABLE TODAY

Confidential Containers on ACI
Confidential Pods (Containers) on AKS (*Preview*)

Technology: Attestation, Secure Key Release, Cloud sealing

"I just trust my app code and the chip."

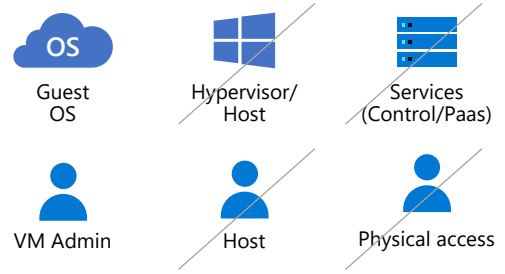


AVAILABLE TODAY

Confidential VMs with AMD SEV-SNP
Confidential VM node pool support on AKS

Technology: (Trusted VM), Secure Key Release, Blind Hypervisor

"Microsoft cannot touch my stuff in my VM."

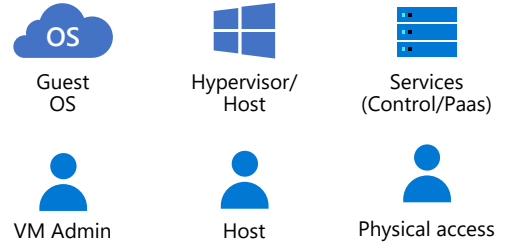


AVAILABLE TODAY

Trusted Launch VMs as default

Technology: VM Attestation, VM Secure Boot, vTPM, Virtualization-Based Security

"Only known, trusted code is running on my VM."



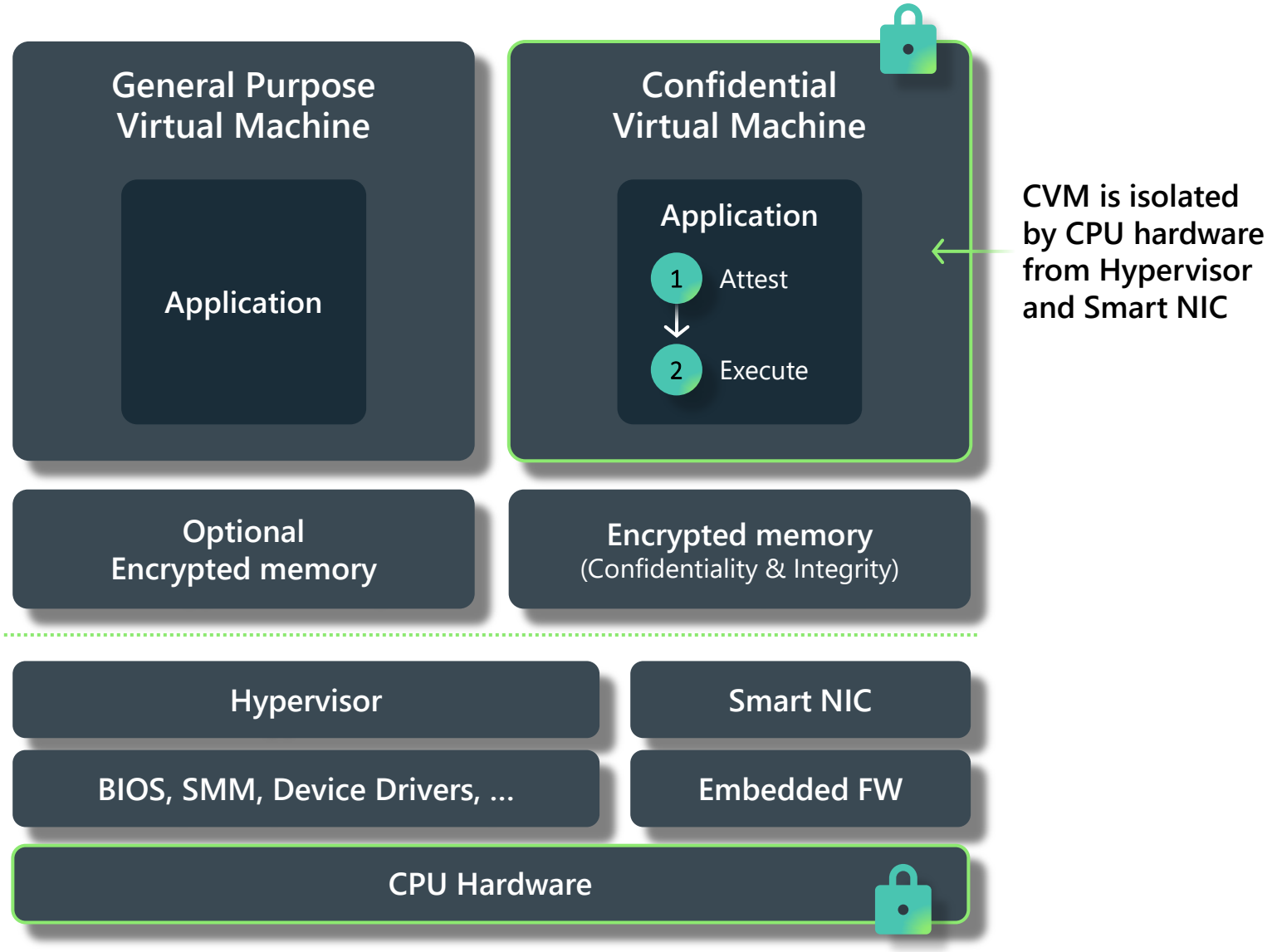
TRUST

Azure Confidential VMs with AMD SEV-SNP

Confidential Virtual Machine with AMD SEV-SNP

Generally available

AMD SEV-SNP



Azure Confidential VMs Powered by AMD SEV-SNP

Make your applications confidential without any code changes.

Availability: GA

DCasv5 and ECasv5 VM Series

VM memory encrypted, and integrity protected by CPU-created keys

Full OS disk encryption with keys sealed in vTPM

Guest attestation capability rooted to AMD's RoT

3rd Generation AMD EPYC™ 7763v (Milan) processors with SEV-SNP technology

Azure Confidential VMs Powered by AMD SEV-SNP

OS Support and Region Availability

VM Sizes

General Compute
DCasv5, DCadsv5

Memory Optimized
ECasv5, ECadsv5

Operating Systems

CANONICAL
Ubuntu Server 22.04
Ubuntu Server 20.04

Windows Server 2022
Windows Server 2019
Windows 11 22H2
SQL server on WS2022/2019

Now in Preview

Red Hat
Red Hat Enterprise Linux 9.2

Regions

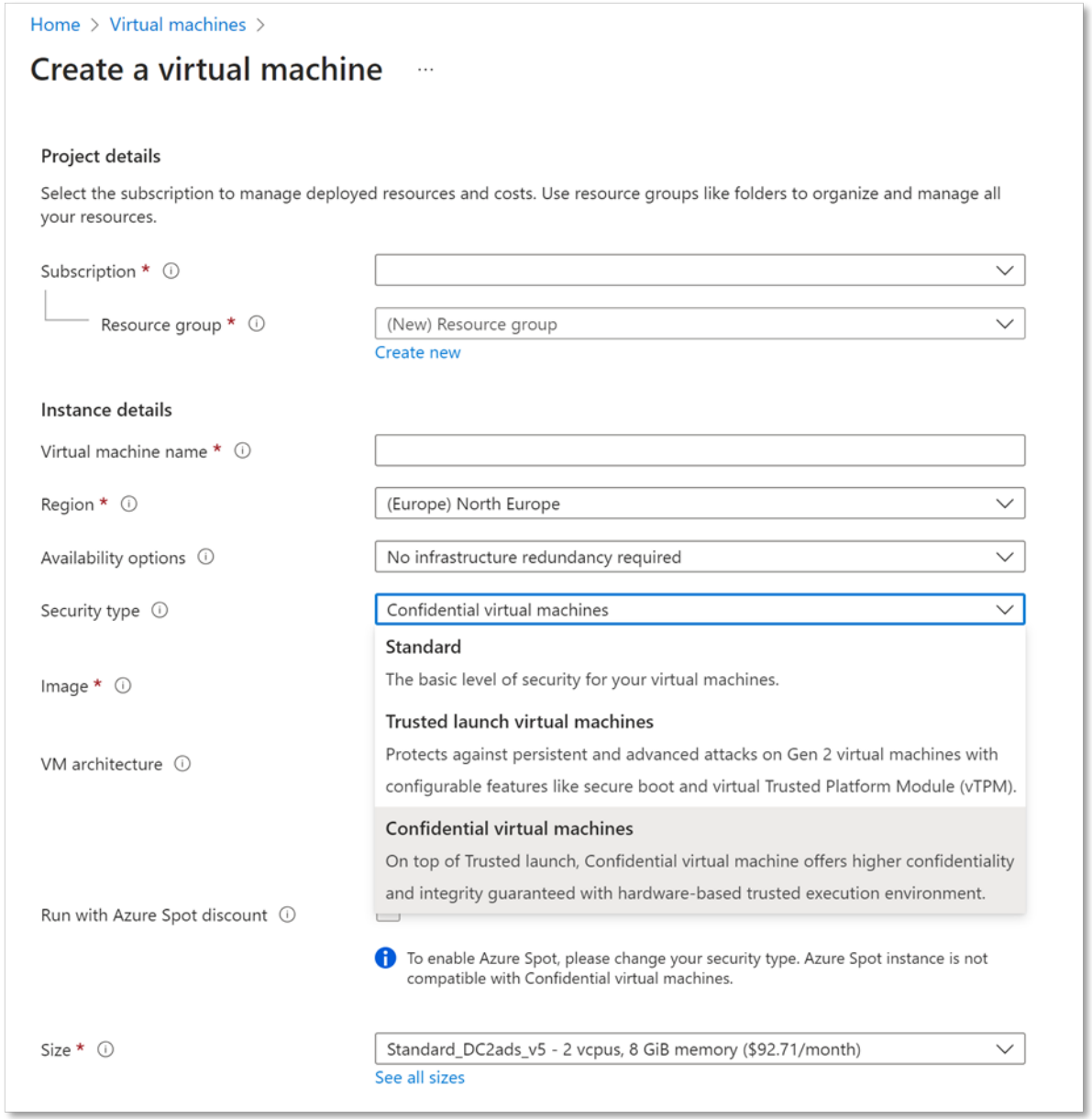
United States
West US, East US

Europe
North Europe, West Europe

Coming this Summer 2023

Southeast Asia, Japan,
Italy, Switzerland

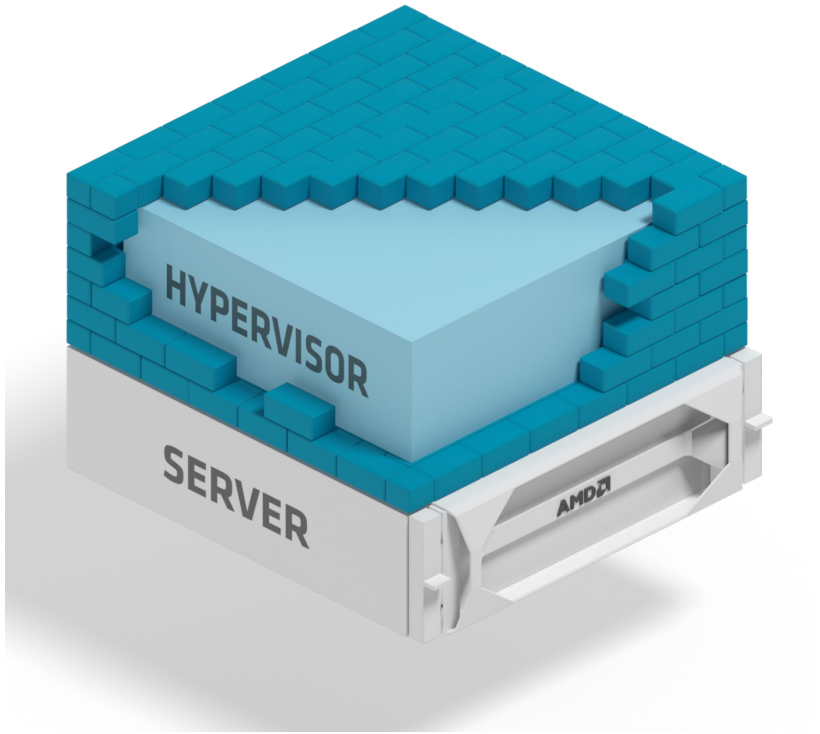
How to create a Confidential Virtual Machine



To create a Confidential Virtual Machine, login to and navigate to “Virtual Machines” under Azure services. On the Create a virtual machine page, select Confidential virtual machines.

It’s that easy!

The foundation of Confidential VM's - AMD Infinity Guard



SEV

SECURE ENCRYPTED VIRTUALIZATION
Encrypt Each VM with Unique Keys

SEV-ES

ENCRYPTED STATE
VM Integrity with Protected CPU Registers

SEV-SNP

SECURE NESTED PAGING
Hardware Protection Against Malicious Hypervisors

Leveraged by Azure confidential computing products

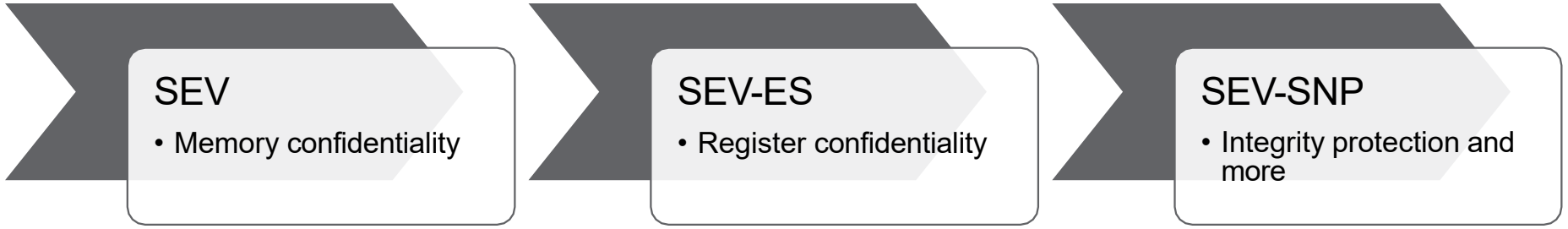
For more info, go to: https://developer.amd.com/sev

History of AMD SEV-SNP

- ***Secure Nested Paging*** (SEV-SNP) is the latest generation of AMD Secure Encrypted Virtualization (SEV) technology designed for Confidential Computing

- SEV-SNP builds on existing AMD SEV and AMD SEV-ES (Encrypted State) features to provide ***stronger security, additional use models, and more*** to protected VMs
  - SEV and SEV-ES supported in 1st and 2nd Generation AMD EPYC™ Processors (2017)
  - **SEV-SNP** supported starting in 3rd Generation AMD EPYC™ Processors (2021)

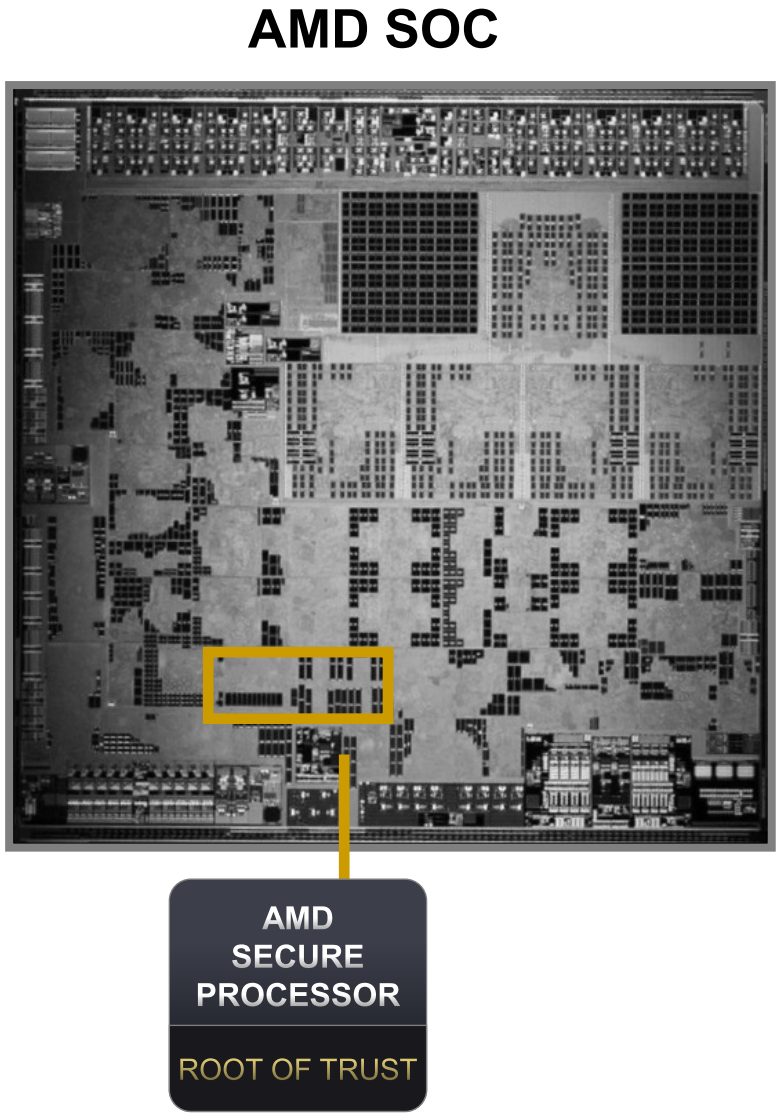
- SEV-SNP is designed to protect a VM from a malicious hypervisor in specific ways
  - Useful in public cloud and any scenario where the hosting environment cannot be trusted



AMD - A Dedicated Security Subsystem

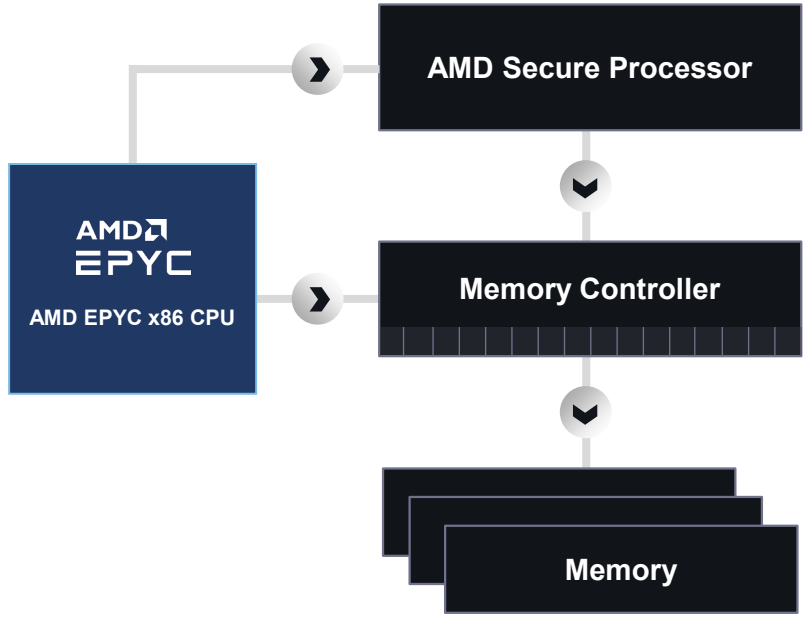
- AMD Secure Processor integrated within SoC
  - 32-bit microcontroller
- Runs an OS/kernel with improved security
- Off-chip NV storage to help protect firmware and data (i.e., SPI ROM)
- Provides cryptographic functionality for key generation and key management
- Enables hardware validated boot

Hardware Root of Trust Provides Foundation for Platform Security



EPYC™ Hardware based Memory Encryption

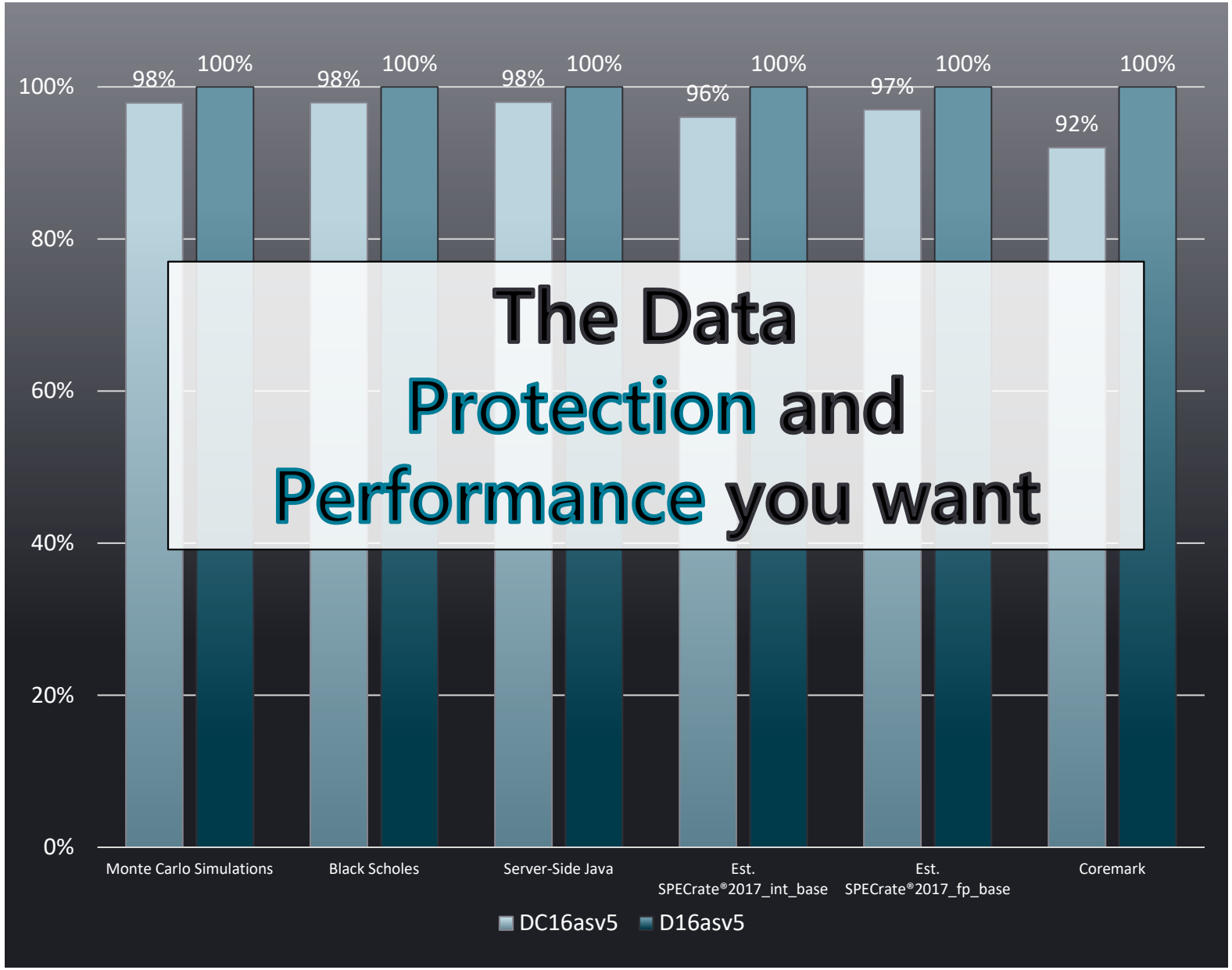
- AES-128 engine in the memory controller
  - Encryption keys managed by AMD Secure Processor / not exposed to x86 CPU
  - Guest OS chooses pages to encrypt via page tables
  - No changes to end user applications needed
- AMD Secure Memory Encryption (SME)
  - All system memory is encrypted using randomly generated key on each system reset
  - Transparent SME is OS agnostic and not visible to OS
- AMD Secure Encrypted Virtualization (SEV)
  - Provides strong cryptographic isolation between the VMs, as well as between the VMs and the hypervisor
  - Active encryption key selected by virtual machine ID



AMD EPYC	UNIQUE MEMORY KEYS
7001	16
7002	509
7003	509

AMD Infinity Guard features vary by EPYC™ Processor generations. Infinity Guard security features must be enabled by server OEMs and/or Cloud Service Providers to operate. Check with your OEM or provider to confirm support of these features. Learn more about Infinity Guard at https://www.amd.com/en/technologies/infinity-guard . GD-183

Enterprise Workloads on Azure Confidential Computing VMs



Take-aways:

1. **Minimal performance impact** running on Azure Confidential Computing VMs vs. standard general purpose VMs

2. No need to recompile applications which **saves time and money**

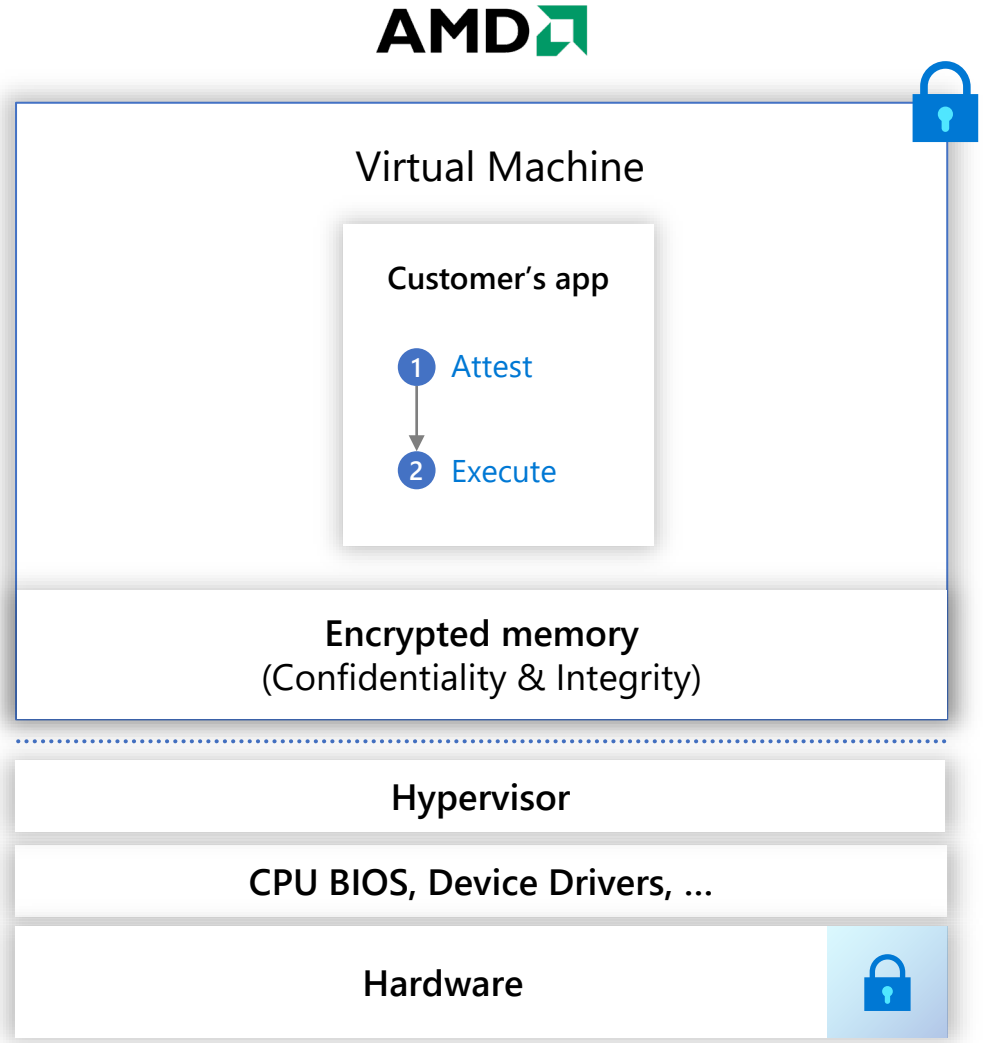
3. Confidential virtual machines offer the **same specs as general purpose** Dasv5/Dadsv5 and Easv5/Eadsv5

*MLNC-016 through 21. See Endnotes

Confidential Virtual Machines

Benefits

- A VM that is Confidential
- Isolation from Azure as the CSP
- No code changes required
- Independent hardware Root Of Trust (RoT)
- Customer verifiable attestation
- Confidential key management
- Achieve near general purpose performance



Azure Confidential VMs Powered by AMD SEV-SNP

General Available

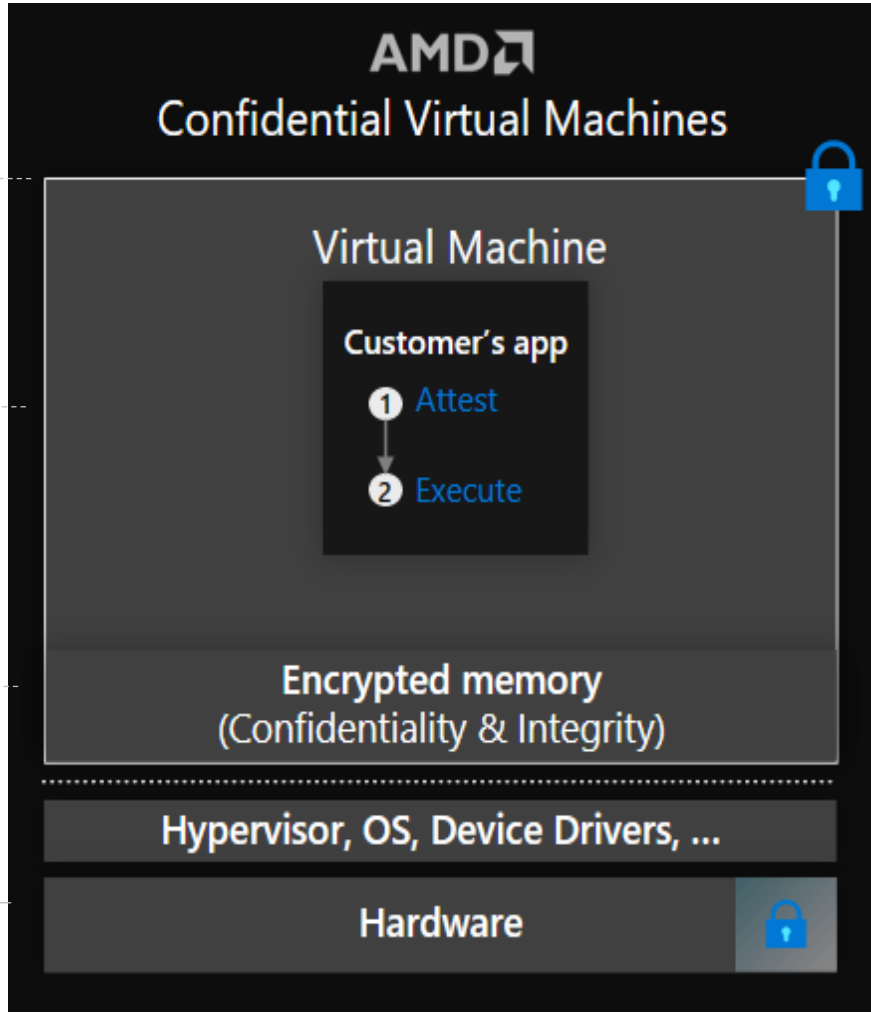
Azure confidential VMs harden guest protection without app code changes.

Hardware-based **VM memory encryption** with **integrity protection**, and keys are generated and safeguarded by AMD processors.

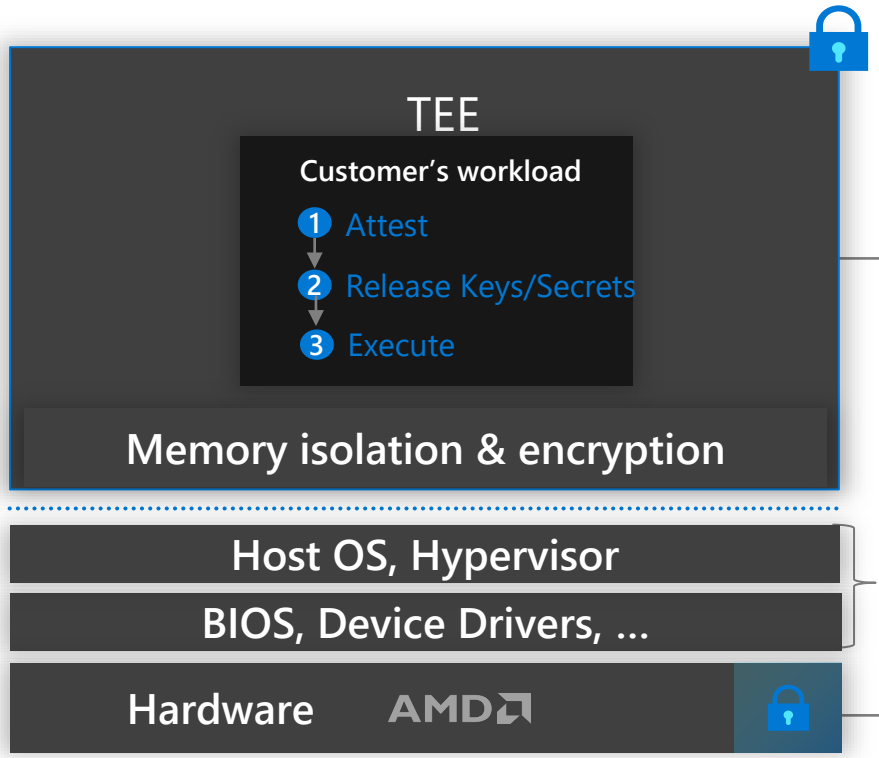
OS full disk encryption can be turned on with customer own keys, which is cryptographically bonded with customized release policies, and keys can reside in Azure Key Vault or Azure Managed HSM.

Verifiable attestation reports reflect status of the AMD root-of-trust information, VM guest firmware measurements, VM configuration such as serial console, secure boot state, virtual TPM identity, and more.

Dedicated virtual TPM to seal secrets/keys.
Secure boot capability protects entire boot path.



ACC Zero Trust Architecture



Properties and Proofs

Confidential Workload (VM or container):

Attestation of software and hardware

Persistent state and associated keys are protected within TEE

Code and data integrity in memory

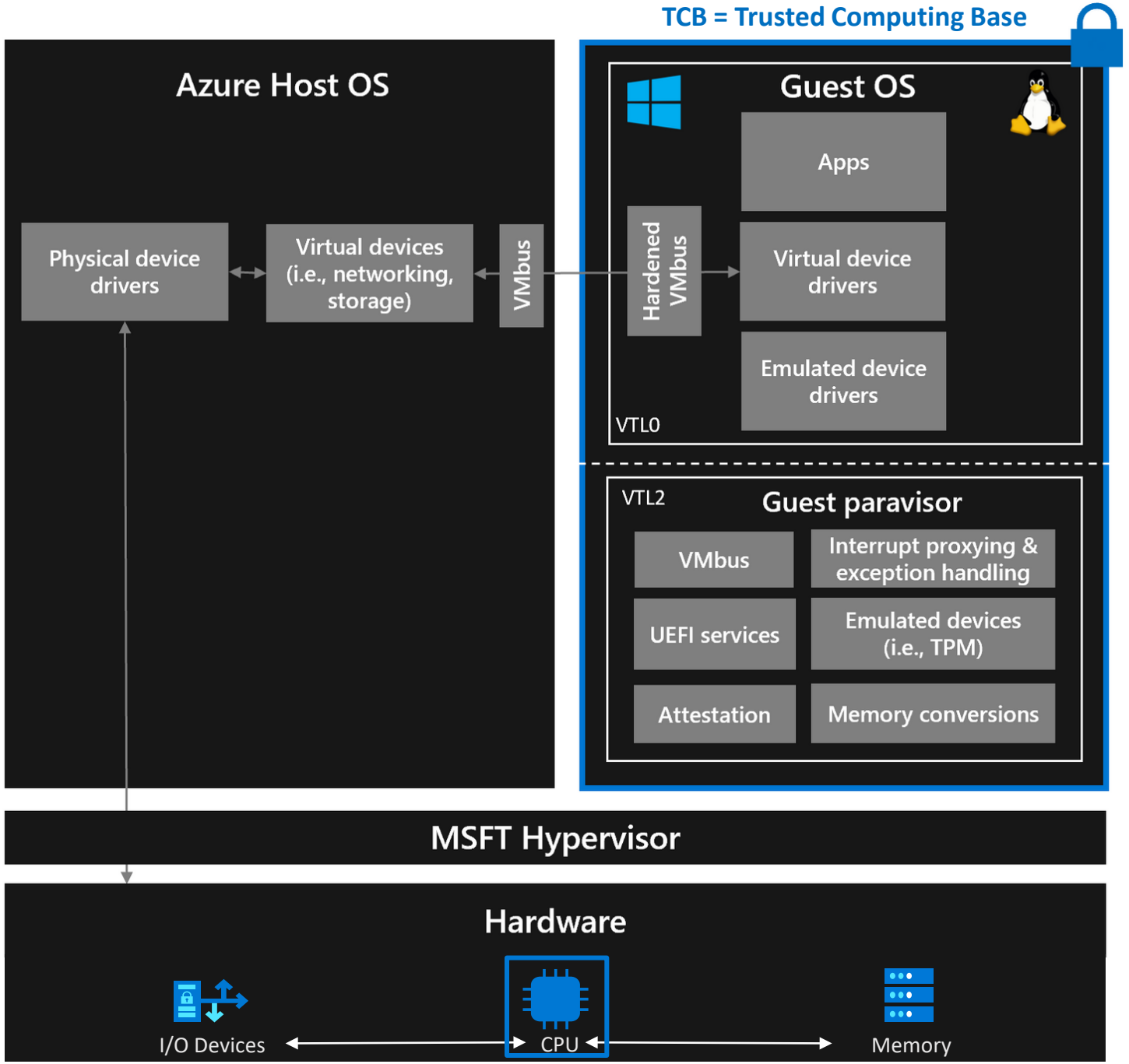
Outside of Trusted Computing Base (TCB)

H/W Root-of-trust → AMD

Architecture of Confidential VMs on Azure

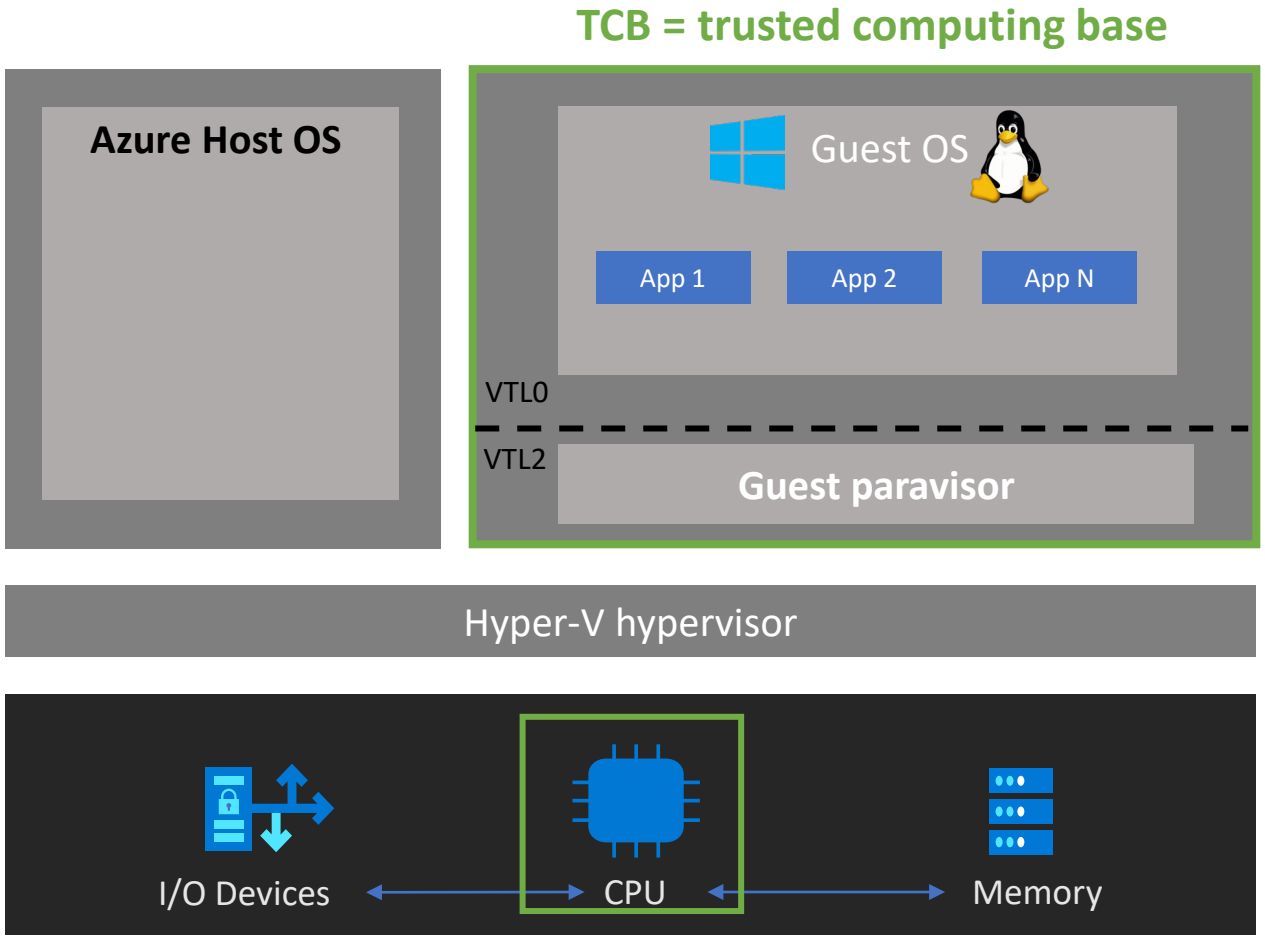
How did we do it?

In our mission to build the Azure Confidential Cloud and enable customers to lift and shift their sensitive workloads, we evolved our virtualization stack (i.e., paravisor) to enable both Windows and Linux operating systems to easily run inside Confidential VMs across hardware architectures.



Guest paravisor

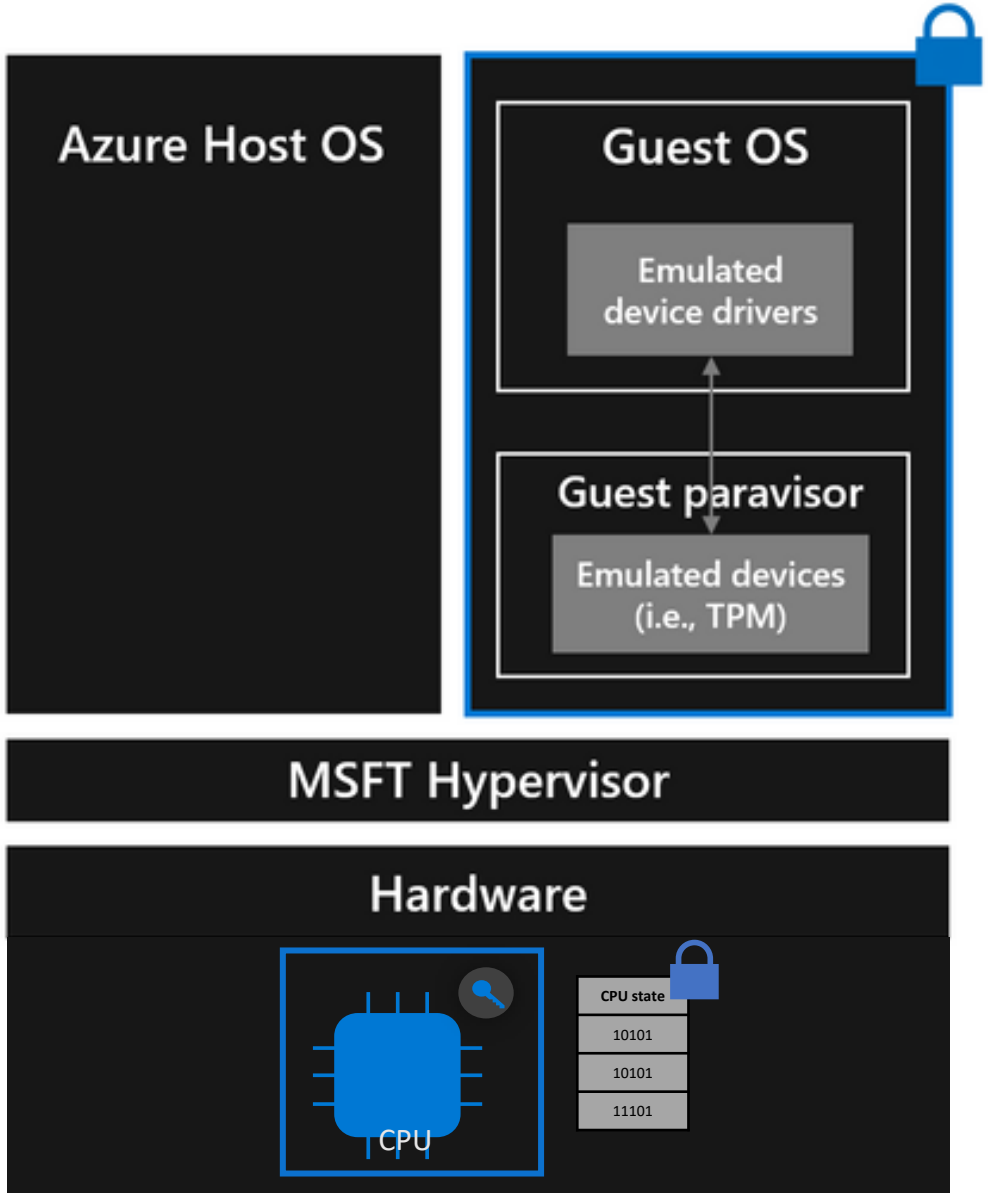
The **guest paravisor** implements the TEE enlightenments on behalf of the guest OS so the guest OS can run mostly unmodified.



Emulated Devices

We moved **emulated devices** (like TPM, RTC, serial, etc.) from the Host OS to the guest paravisor which gave us the ability to provide Confidential VMs a TPM and Secure Boot.

The guest paravisor is running in the guest but isolated from the guest OS



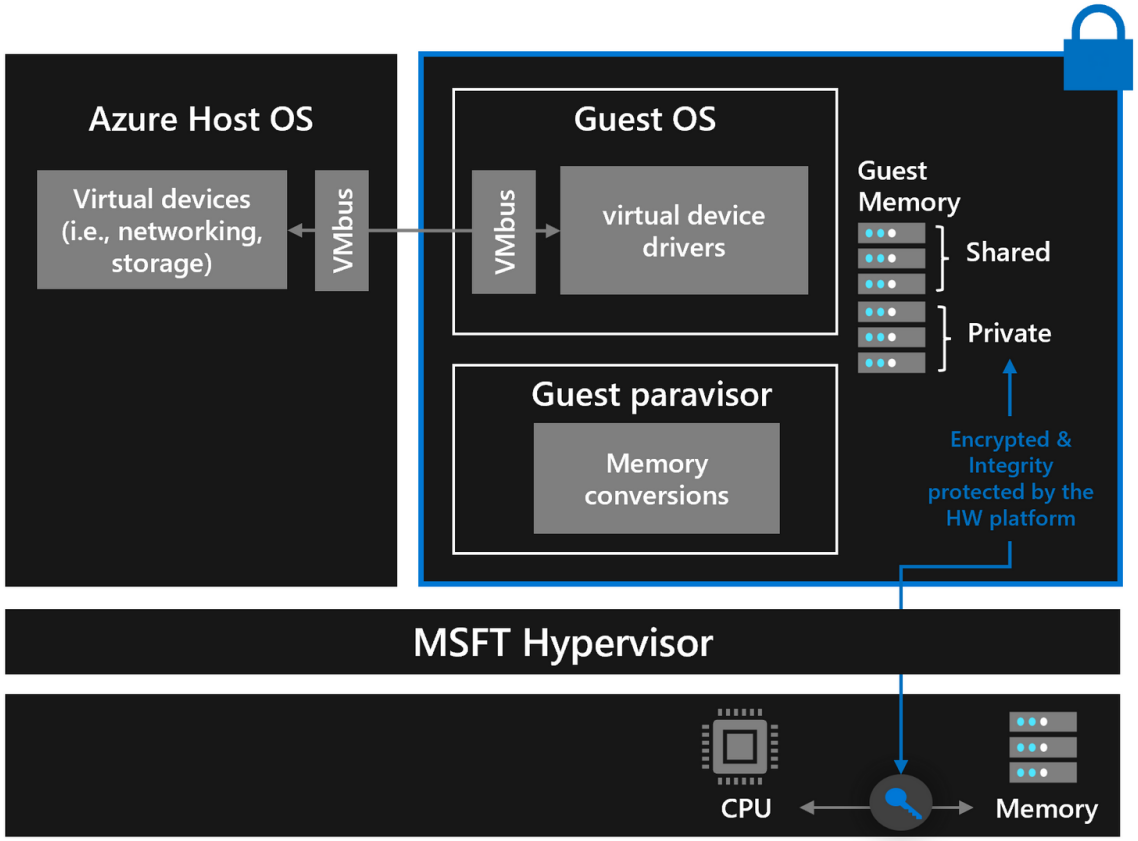
Memory Protections

Two types of memory:

- **Private Memory:** Default computation memory for Confidential VMs.
- **Shared Memory:** Used for communication with the virtualization stack, e.g., device IO.

Page Visibility:

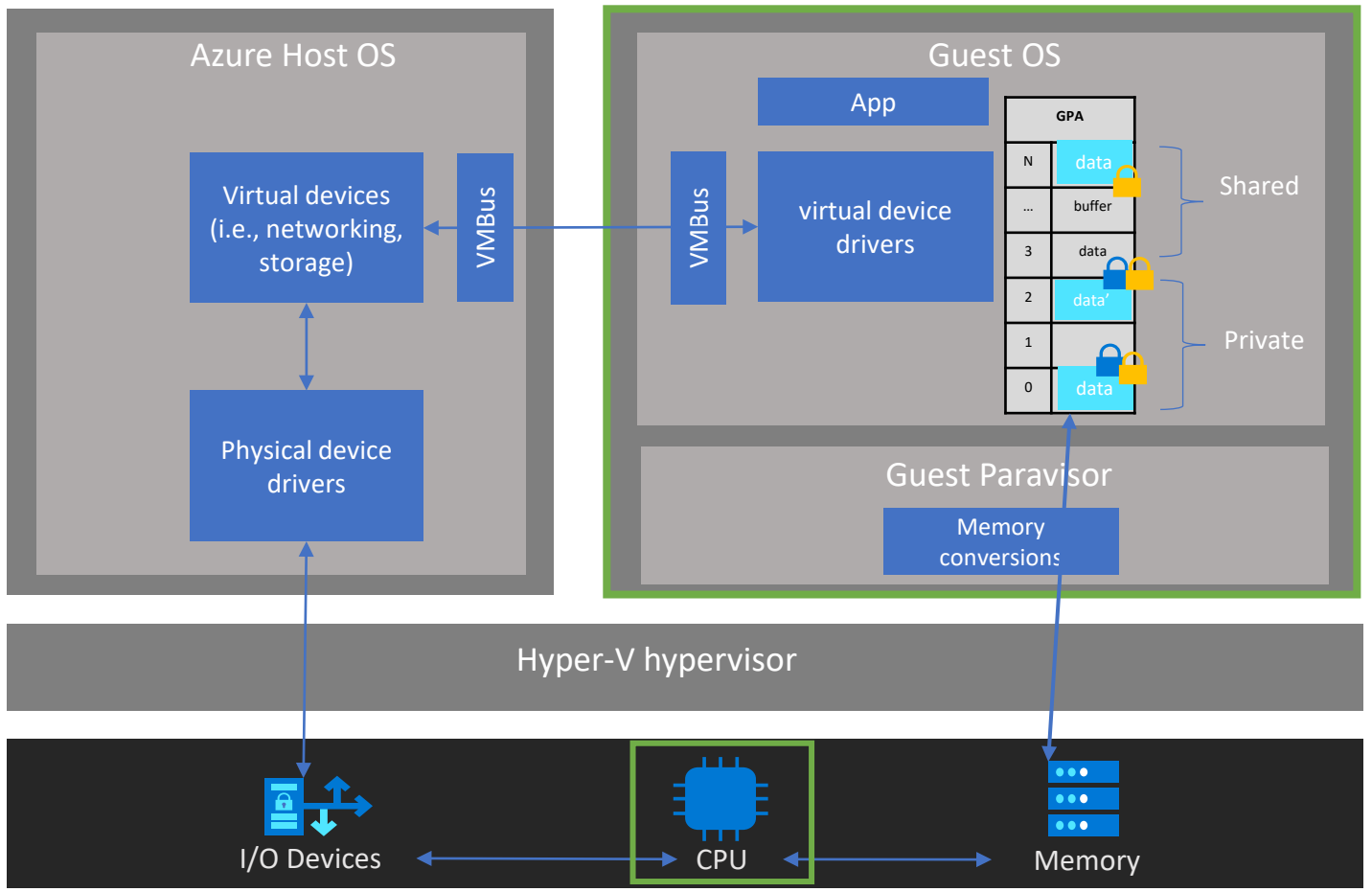
- Memory can be either **private** or **shared**, referred to as its page visibility.
- The guest can convert memory between private and shared as needed.
- Private pages are used by default.
- To use a shared page, the guest must manage page visibility.
- Confidential VMs need to protect secret data in shared memory.

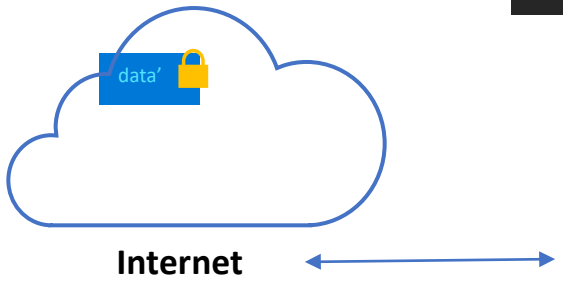


An example

Hardware key

SSL





End-to-end protection

Execution State Integrity

The guest paravisor can do **interrupt proxying, validating interrupts coming** from the hypervisor and handle the new special CVM exception type on behalf of the guest OS.

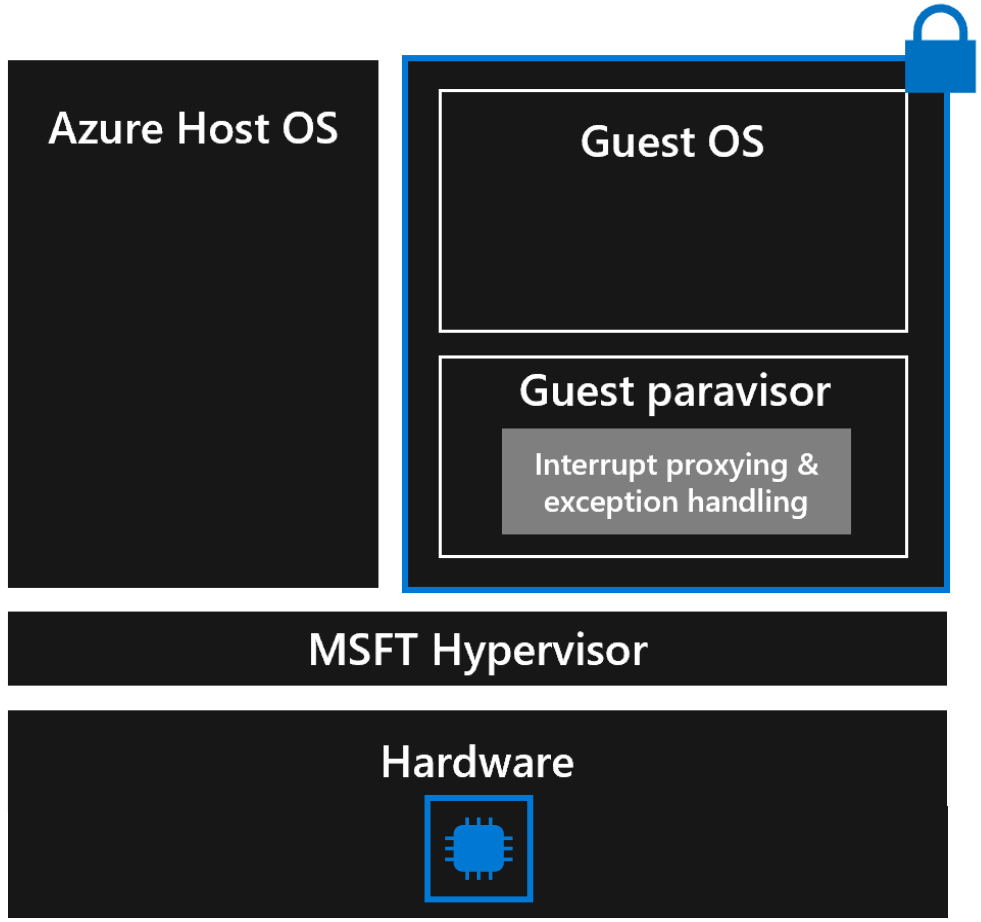
Guest Paravisor Role:

A crucial component within a Confidential VM, the guest paravisor, acts as an intermediary, ensuring secure interrupt handling.

- **Interrupt Proxying:** The paravisor validates interrupts coming from the hypervisor, thereby shielding the guest OS.
- **Re-injection of Valid Interrupts:** After validation, the paravisor re-injects the interrupt into the guest OS.

New Exception Type with Confidential VMs:

- This is a hardware-specific exception that is generated solely by the hardware.
- Instead of the virtualization stack, the guest VM is responsible for handling this new exception type.
- The paravisor can handle this exception on behalf of the guest OS, providing an extra layer of security.



Guest firmware protection

Guest firmware can **store and access guest state and guest secrets** that are inaccessible to the Host OS.

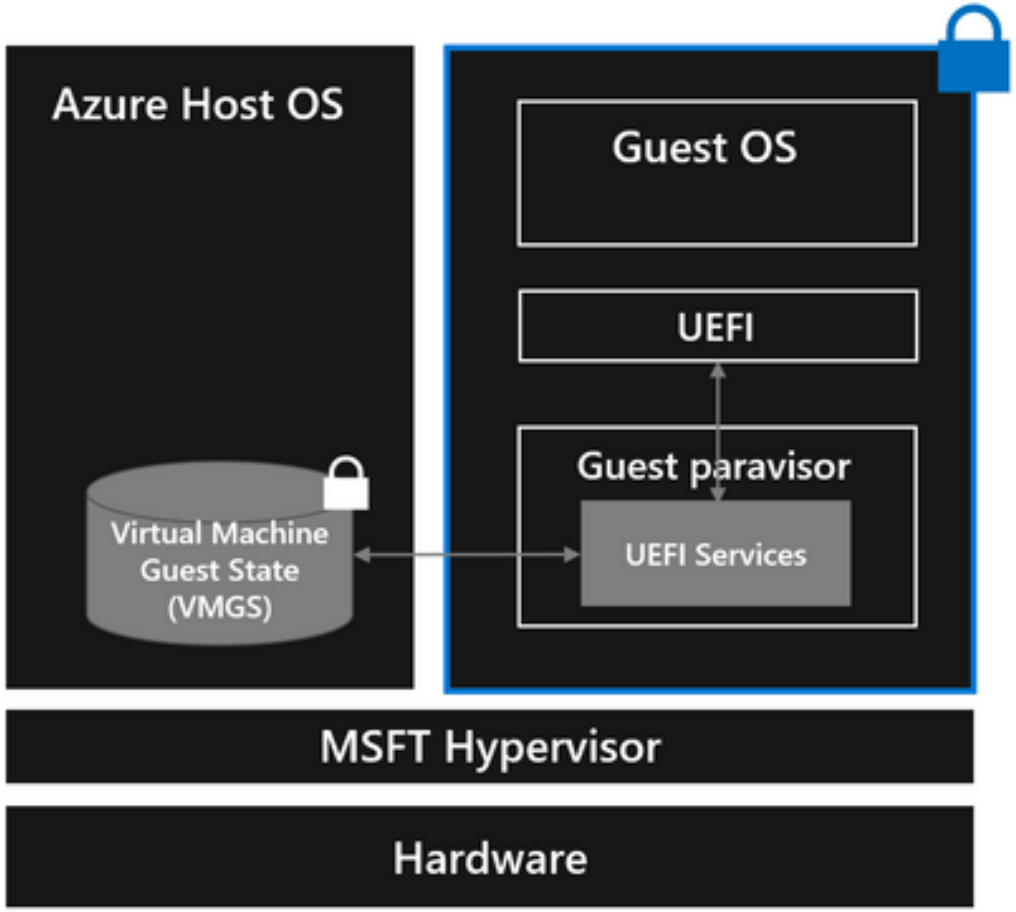
UEFI Firmware Evolution:
The guest UEFI firmware has been updated to retrieve trusted UEFI attributes from a new Virtual Machine Guest State (VMGS) file, packaged as a VHD, instead of from the host.

Secure Storage:
The VMGS.VHD file is encrypted, providing Confidential VM's firmware access to persistent storage that remains inaccessible to the host. The host interacts only with the encrypted VHD.

Authenticated UEFI Variables:
Managed by the guest paravisor, authenticated UEFI variables offer enhanced security from both the host and guest OS.

Variable Writing Process:
When a Confidential VM employs UEFI runtime services to write a variable, the guest paravisor processes and persists the authenticated variable rights in the VMGS file.

Persistent Storage & Access:
The updated design allows a Confidential VM to use the VMGS file for persistent storage and access to VM guest state and secrets, such as UEFI state and TPM state if desired by the customer.



Remote Attestation

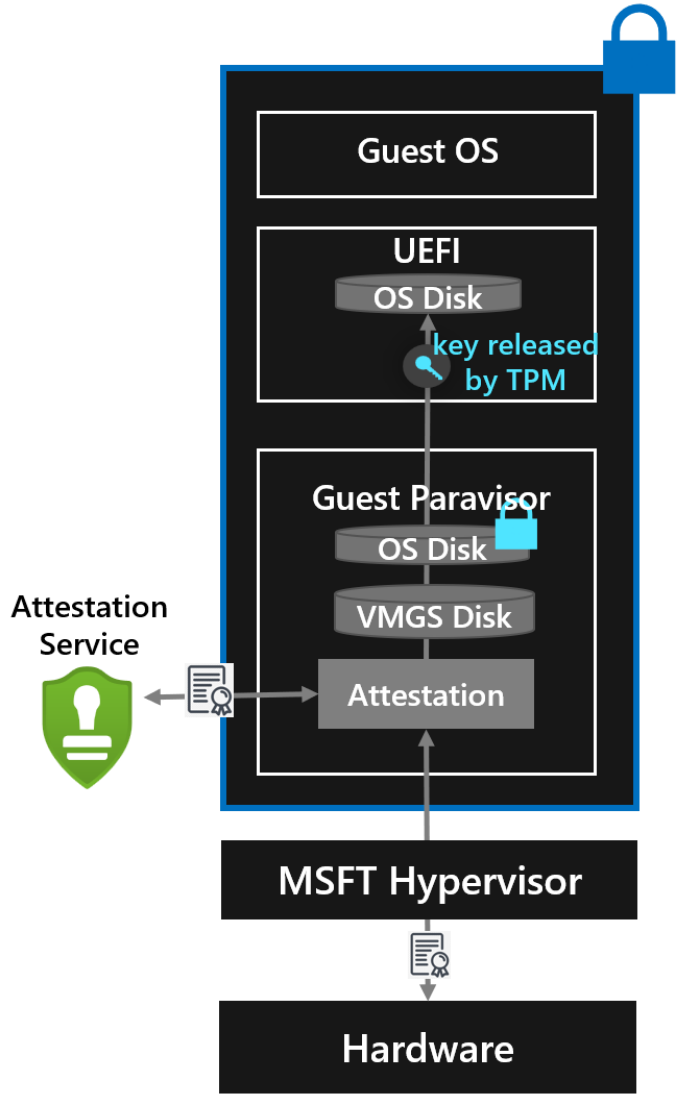
Attestation for a Confidential VM **"conceptually authenticates the VM and/or the virtual firmware used to launch the VM".**

Initialization of a Confidential VM:
A partition for the Confidential VM on Azure is created and started.
The hardware seals the partition to prevent modification by the host.
Hardware platform provides a measurement of the guest's launch context.

Guest Paravisor Boot and Attestation:
The guest paravisor boots first, performing attestation for the guest OS.
A signed attestation report is requested from the hardware platform and sent to an attestation verification service. Any failures in attestation verification prevent the VM from booting.

Control Transfer and Secure Boot:
The guest paravisor transfers control to UEFI.
Secure Boot verifies the startup components, checking their signatures before loading.

Role of TPM Measured Boot:
UEFI accumulates measurements of startup components into the TPM's PCRs as they load.
For encrypted OS disks, the TPM only releases the decryption key if the VM's firmware code, original boot sequence, and boot components remain unaltered.



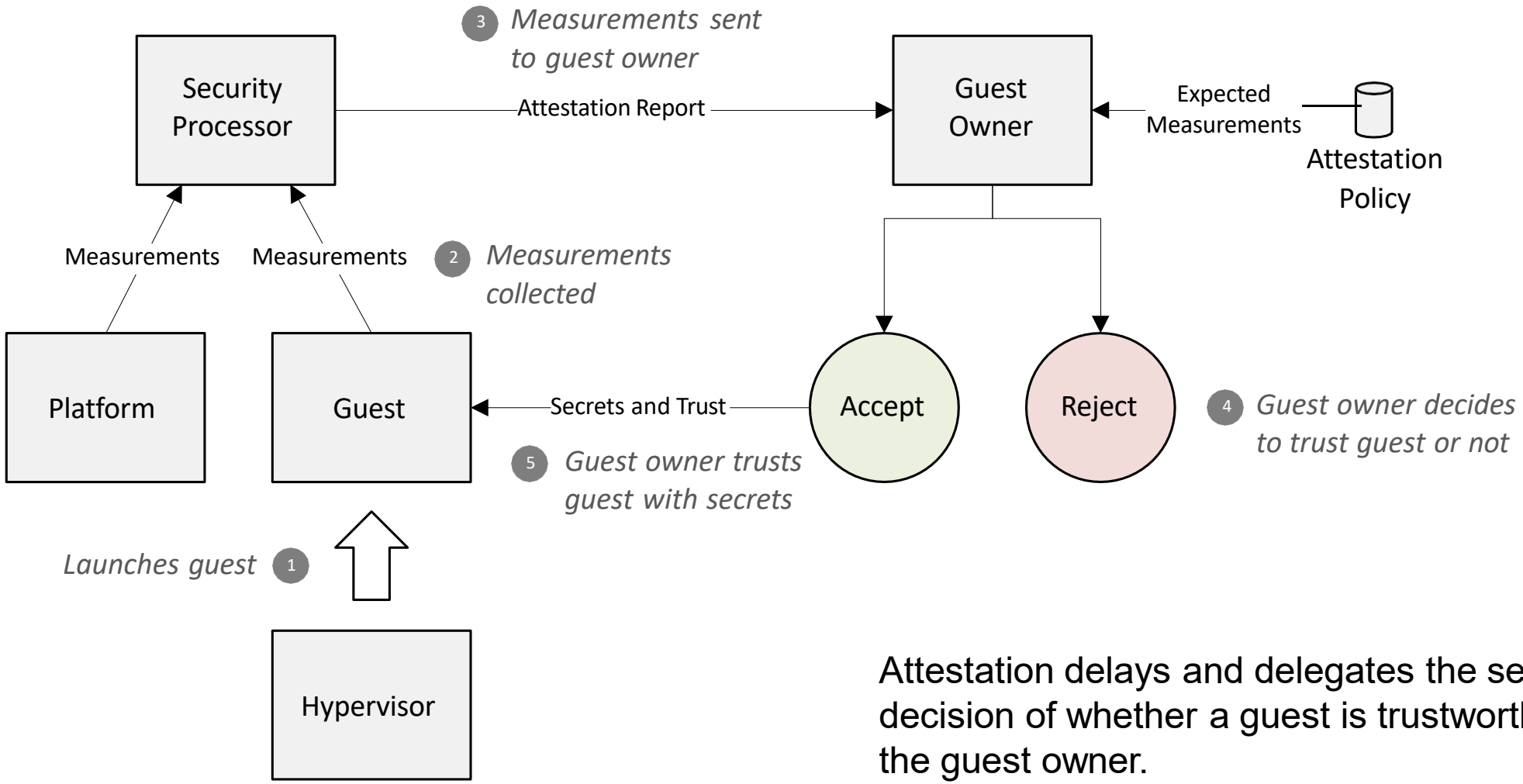
Hands-on lab: Deploying Confidential VM

Hands-on lab will cover - cvm_deployment.md

- Deploying a confidential VM on Azure
- Building the sample application by using https://github.com/Azure/confidential-computing-cvm-guest-attestation
- Requesting attestation from the VM by using the sample application
- Displaying the JWT token generated by Microsoft Azure Attestation

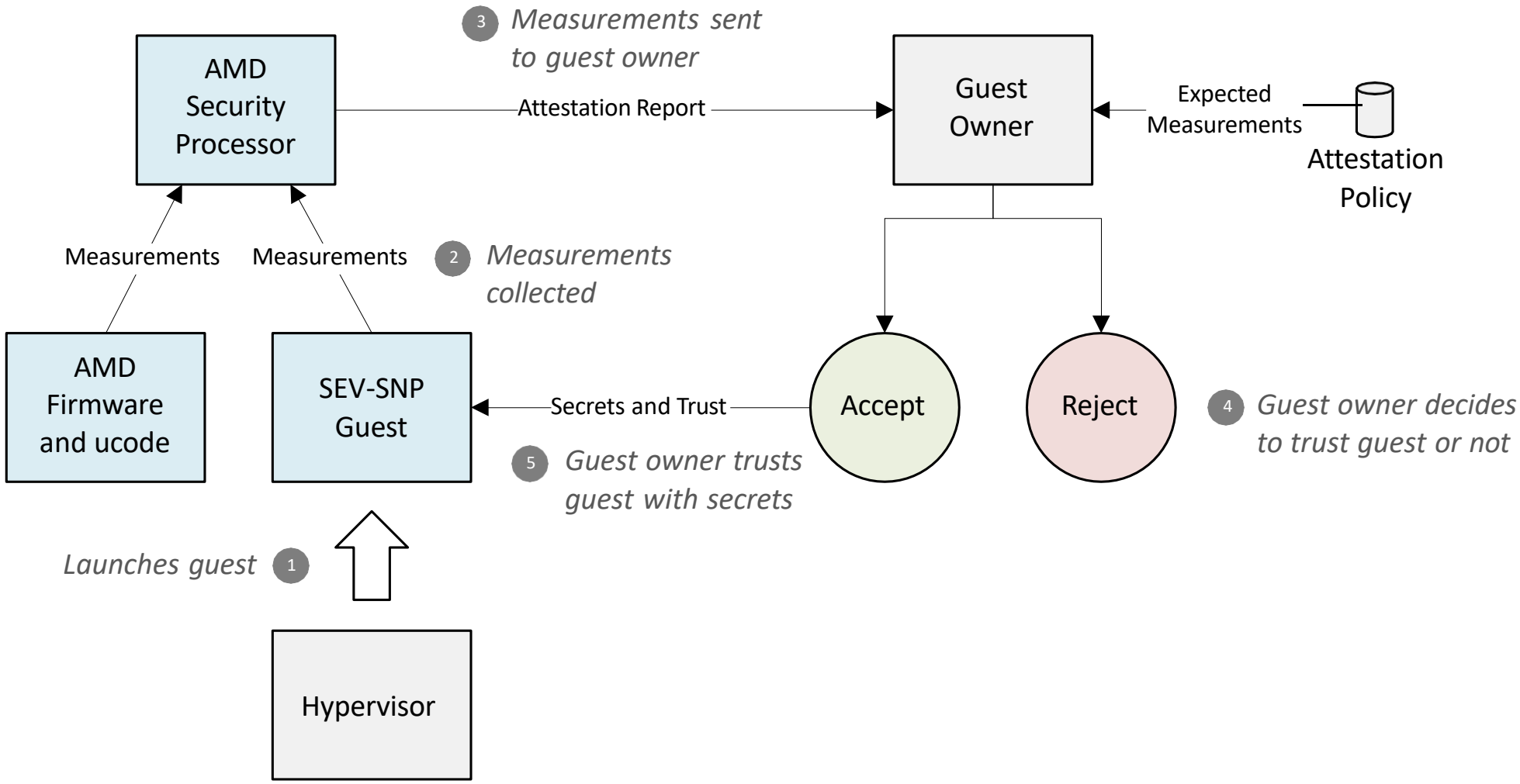
Overview of Attestation in AMD SEV-SNP

Overview of Remote Attestation



Attestation delays and delegates the security decision of whether a guest is trustworthy to the guest owner.

Overview of Attestation in AMD SEV-SNP



Overview of an AMD SEV-SNP attestation report

Table 8. Guest Policy Structure

Bit(s)	Name	Description
63:20	-	Reserved. MBZ.
19	DEBUG	0: Debugging is disallowed. 1: Debugging is allowed.
18	MIGRATE_MA	0: Association with a migration agent is disallowed. 1: Association with a migration agent is allowed.
17	-	Reserved. Must be one.
16	SMT	0: SMT is disallowed. 1: SMT is allowed.
15:8	ABI_MAJOR	The minimum ABI major version required for this guest to run.
7:0	ABI_MINOR	The minimum ABI minor version required for this guest to run.

Table 21. ATTESTATION_REPORT Structure

Byte Offset	Bits	Name	Description
00h	31:0	VERSION	Version number of this attestation report. Set to 1h for this specification.
04h	31:0	GUEST_SVN	The guest SVN.
08h	63:0	POLICY	The guest policy. See Table 8 for a description of the guest policy structure.
10h	127:0	FAMILY_ID	The family ID provided at launch.
20h	127:0	IMAGE_ID	The image ID provided at launch.
30h	31:0	VMPL	The request VMPL for the attestation report.
34h	31:0	SIGNATURE_ALGO	The signature algorithm used to sign this report. 102h indicates ECDSA P-384 with SHA-384. All other encodings are reserved.
38h	63:0	PLATFORM_VERSION	The install version of firmware.
40h	63:0	PLATFORM_INFO	Information about the platform. See Table 22.
48h	31:1	-	Reserved. Must be zero.
	0	AUTHOR_KEY_EN	Indicates that the digest of the author key is present in AUTHOR_KEY_DIGEST. Set to the value of GCTX.AuthorKeyEn.
4Ch	31:0	-	Reserved. Must be zero.
50h	511:0	REPORT_DATA	Guest-provided data.
90h	383:0	MEASUREMENT	The measurement calculated at launch.
C0h	255:0	HOST_DATA	Data provided by the hypervisor at launch.
E0h	383:0	ID_KEY_DIGEST	SHA-384 digest of the ID public key that signed the ID block provided in SNP_LANUNCH_FINISH.
110h	383:0	AUTHOR_KEY_DIGEST	SHA-384 digest of the Author public key that certified the ID key, if provided in SNP_LAUNCH_FINSIH. Zeroes if AUTHOR_KEY__EN is 1.
140h	255:0	REPORT_ID	Report ID of this guest.
160h	255:0	REPORT_ID_MA	Report ID of this guest's migration agent.
180h	63:0	REPORTED_TCB	Reported TCB version used to derive the VCEK that signed this report.
188h – 19Fh		-	Reserved.
1A0h-1DFh	511:0	CHIP_ID	Identifier unique to the chip.
1E0h-29Fh		-	Reserved.
2A0h-49Fh		SIGNATURE	Signature of this report. The format of the signature is described in Table 23.

For more info, go to: https://developer.amd.com/sev

AMD
together we advance_

Content of an AMD SEV-SNP attestation report

Field	Description
Version	The version number of the report. At time of writing, this value will always be the hexadecimal value of 2.
Guest SVN	The Security Version Number (SVN) of the SNP Firmware.
Policy	The Guest Policy. This includes the minimum required ABI major and minor versions, whether or not SMT is allowed, if a migration agent is allowed, if debugging is allowed, and if it may be activated on single or multiple sockets.
Family ID	The family ID provided at VM launch as part of the Identity Block.
Image ID	The image ID provided at VM launch as part of the Identity Block.
VMPL	The requested VM Permission Level (VMPL) for the attestation report.
Signature Algorithm	The algorithm used to sign the generated report. At time of writing, this is ECDSA P-384 with SHA-384.
Current TCB	Security Version Numbers (SVNs) of the currently executing platform firmware and microcode.
Platform Info	Information about the system, namely if TSME or SMT is enabled on the system.
Signing Key	Encodes the key used to sign this report; either VCEK or VLEK.
Mask Chip Key	Reports if the MaskChipKey is set. Defaults to 0 (disabled), but when set to 1 (enabled), it prevents the attestation report from being signed and prevents the guest from using the VCEK in guest key derivations.
Author Key En	Indicates if the author key is present in AUTHOR_KEY_DIGEST.
Report Data	The 512-bit value provided by the guest as part of the guest request.
Measurement	The launch measurement calculated by the AMD Secure Processor.
Host Data	Unique data provided by the hypervisor (host) at launch.
ID Key Digest	SHA-384 digest of the ID public key that signed the ID block provided in SNP_LAUNCH_FINISH
Author Key Digest	SHA-384 digest of the Author public key that certified the ID key, if provided in SNP_LAUNCH_FINISH. All zeroes if AUTHOR_KEY_EN is 1.

Field	Description
Report ID	The unique report ID of this guest.
Report ID MA	The unique report ID of the guest's migration agent.
Reported TCB	Reported TCB version used to derive the VCEK that signed this report.
Chip ID	If MaskChipId is set to 0, the unique ID of the chip, otherwise 0.
Committed TCB	SVNs of the anti-rollback minimum of the platform firmware and microcode.
Current Build	The build number of the current TCB Version.
Current Minor	The minor version number of the current TCB Version.
Current Major	The major version number of the current TCB Version.
Committed Build	The build number of the committed TCB Version.
Committed Minor	The minor version of the committed TCB Version.
Committed Major	The major version ofthe committed TCB Version.
Launch TCB	SVNs of the version of the platform firmware and microcode at time of launch of this guest.
Signature	The signature from the AMD Secure Processor including all of the aforementioned fields.

For more info, go to:
https://developer.amd.com/sev

Attestation Report: Platform Measurements

Report Field	Description	Usage and Verification
CHIP_ID	Unique chip identifier	Connects the report to the certificate of the attestation key that signed this report
PLATFORM_INFO	Indicates properties of the platform configuration, such as whether whole system memory encryption or Simultaneous Multithreading (SMT) is enabled	Do you require whole system memory encryption to be on? Do you require SMT to be off?
CURRENT_TCB	Security Version Numbers (SVNs) of the currently executing platform firmware and microcode	Informational
COMMITTED_TCB	SVNs of the anti-rollback minimum of the platform firmware and microcode	Does this TCB address all the vulnerabilities you care about?
REPORTED_TCB	Hypervisor has option to report a lower version to ease continuity on TCB update	Connects the report to the certificate of the attestation key that signed this report
LAUNCH_TCB	SVNs of the version of the platform firmware and microcode at time of launch of this guest	Do you care if the guest *once* executed with a particular TCB version?

For more info, go to: https://developer.amd.com/sev

Attestation Report: Guest Measurements

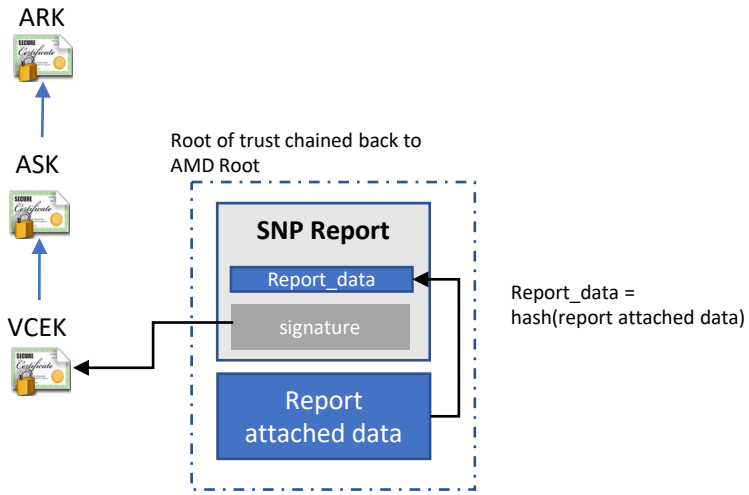
Report Field	Description	Usage and Verification
FAMILY_ID IMAGE_ID GUEST_SVN	Guest image identification information	Is this the image you expected?
MEASUREMENT	Measurement of the guest's address space, including page types as defined in SNP_LAUNCH_UPDATE	Did the hypervisor set the guest up as you expected?
ID_KEY_DIGEST AUTHOR_KEY_DIGEST	Owner's ID and Author key digests, identifying the owner	Is this you?
POLICY	SEV-SNP guest policy restricting various things like SEV-SNP debug commands	Is the policy what you expected?
REPORT_ID MA_REPORT_ID	Connects this attestation report to its migration agent's report, if one is present.	Did you expect a migration agent to be bound to this guest? Does the migration agent's attestation report check out?

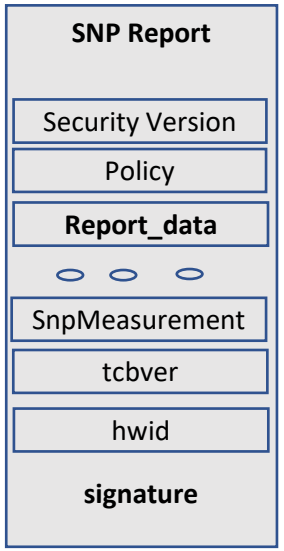
For more info, go to: https://developer.amd.com/sev

Authenticity of Attestation Report

- Attestation report contains
  - identity information about the guest (from the launch sequence),
  - migration and policy information.
  - a block of arbitrary data supplied by the guest VM

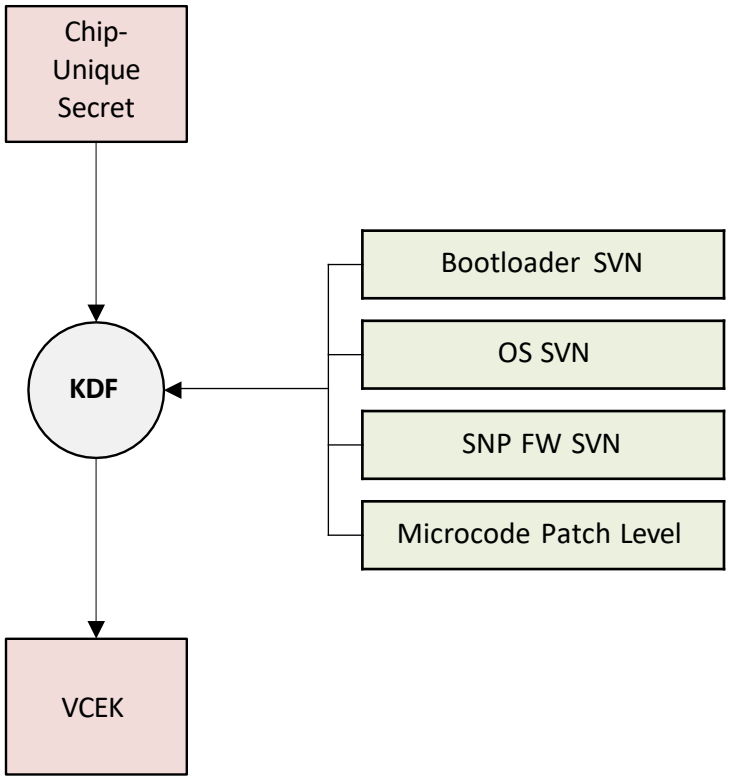
- Generated by AMD Milan Platform Security Processor
- Signed by AMD-SP firmware using VCEK (Versioned Chip Endorsement Key) private key
- Enable a thirty party to conduct remote attestation by validating a SNP report



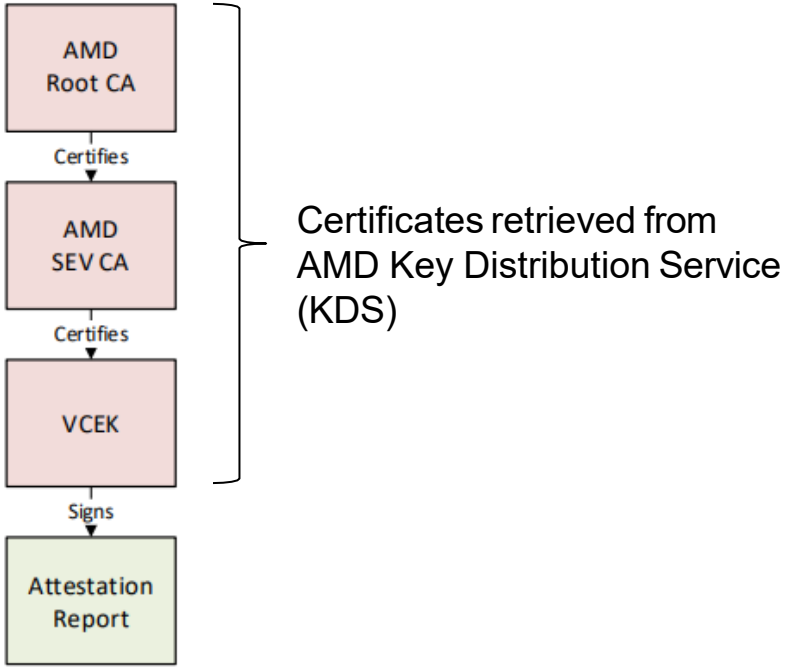


For more info, go to: https://developer.amd.com/sev

Authenticity of Attestation Report



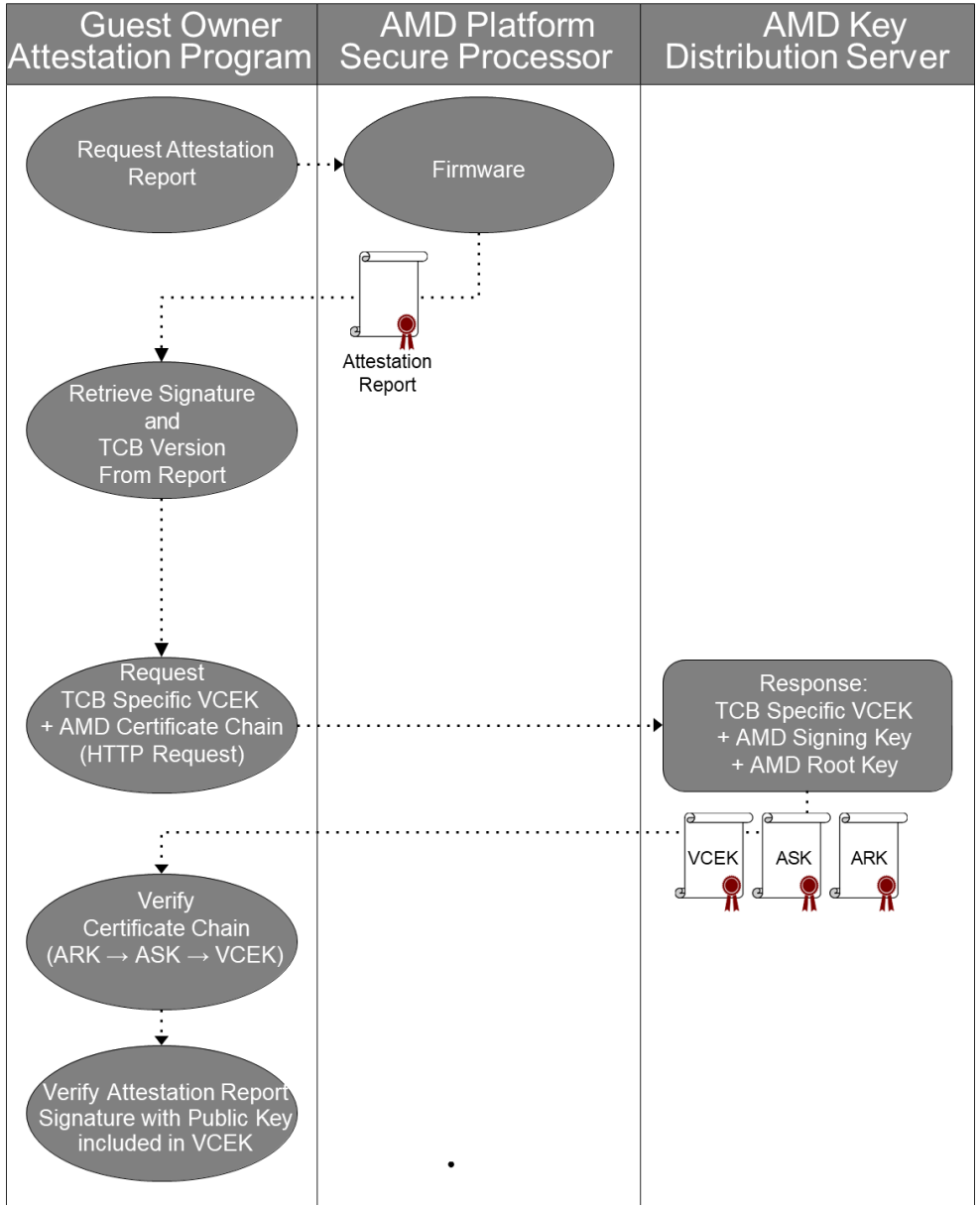
REPORTED_TCB is mixed into the chip unique secret to derive the Versioned Chip-Endorsement Key (VCEK), which is used as the attestation key



AMD Certificate Authority certifies the VCEK

VCEK signs the attestation report

Regular Attestation Workflow



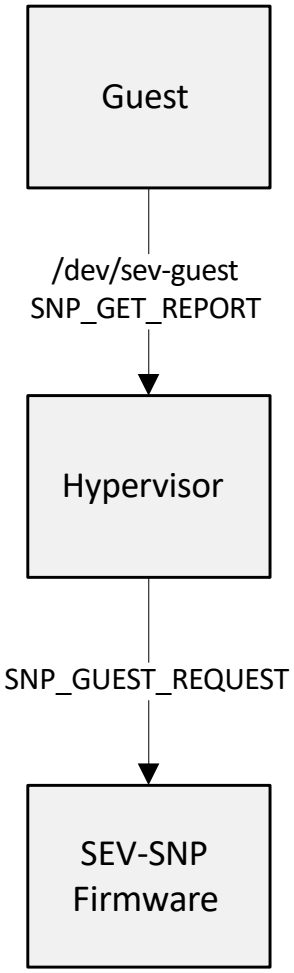
A valid signed attestation report guarantees:

- The contents of report have not been modified or tampered with.
- The report was generated by a specific, genuine, AMD EPYC Processor; which is correctly configured for SEV-SNP.
- Virtual machine's memory is in a safe state.

Retrieving Attestation Reports

- Reports retrieved via guest message
  - Guest construct MSG_REPORT_REQ message
  - Guest encrypts and integrity protects message with key shared at guest launch
  - Guest sends wrapped request to hypervisor
  - Hypervisor invokes SNP_GUEST_REQUEST on wrapped request
  - SEV-SNP firmware returns attestation report through same channel
- Linux guest retrieves reports via IOCTLs on /dev/sev-guest
  - SNP_GET_REPORT – retrieves report
  - SNP_GET_EXT_REPORT – retrieves report and certificates (see below)
- Linux host can provide attestation key certificates via IOCTLs on /dev/sev
  - SNP_SET_EXT_CONFIG – stores attestation key certificates for retrieval by /dev/sev-guest

    E.g., VCEK certificate chain is retrieved from AMD Key Distribution Service (KDS) and stored in the host via SNP_SET_EXT_CONFIG. The guest retrieves the report and certificate chain via SNP_GET_EXT_REPORT.



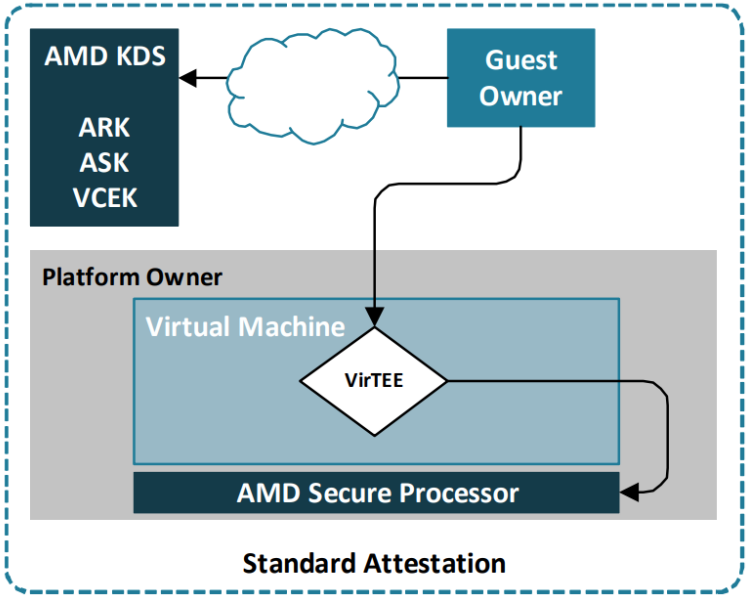
For more info, go to: https://developer.amd.com/sev

Project VirTEE

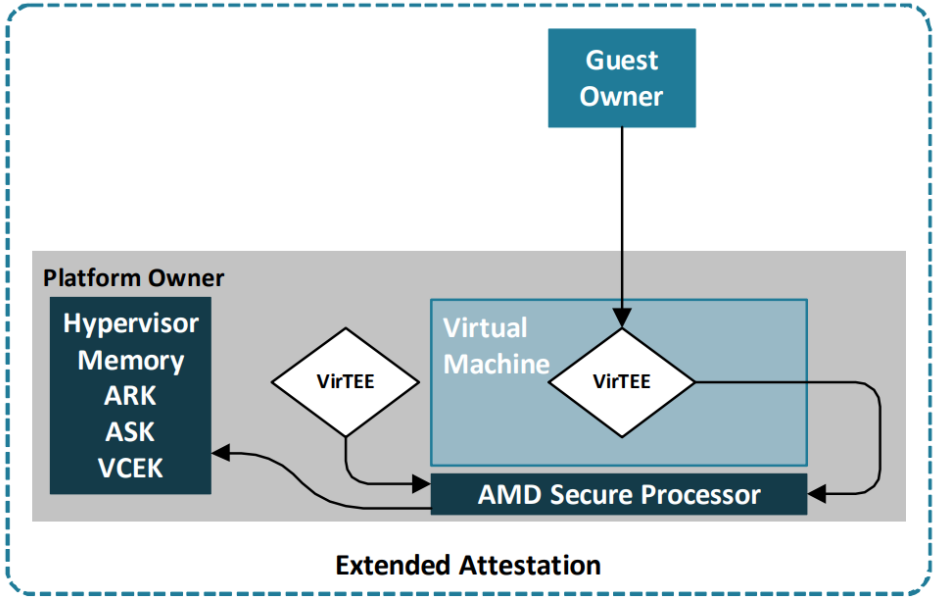
https://github.com/virtee/snpguest

- Rust library and Multitool for AMD SEV-SNP
- For more info, go to: https://developer.amd.com/sev

Standard Attestation: Guest owners may request a standard attestation report. This process involves pulling the certificate-chain from the AMD Key Distribution Server (KDS) in accordance with the VCEK Specification.

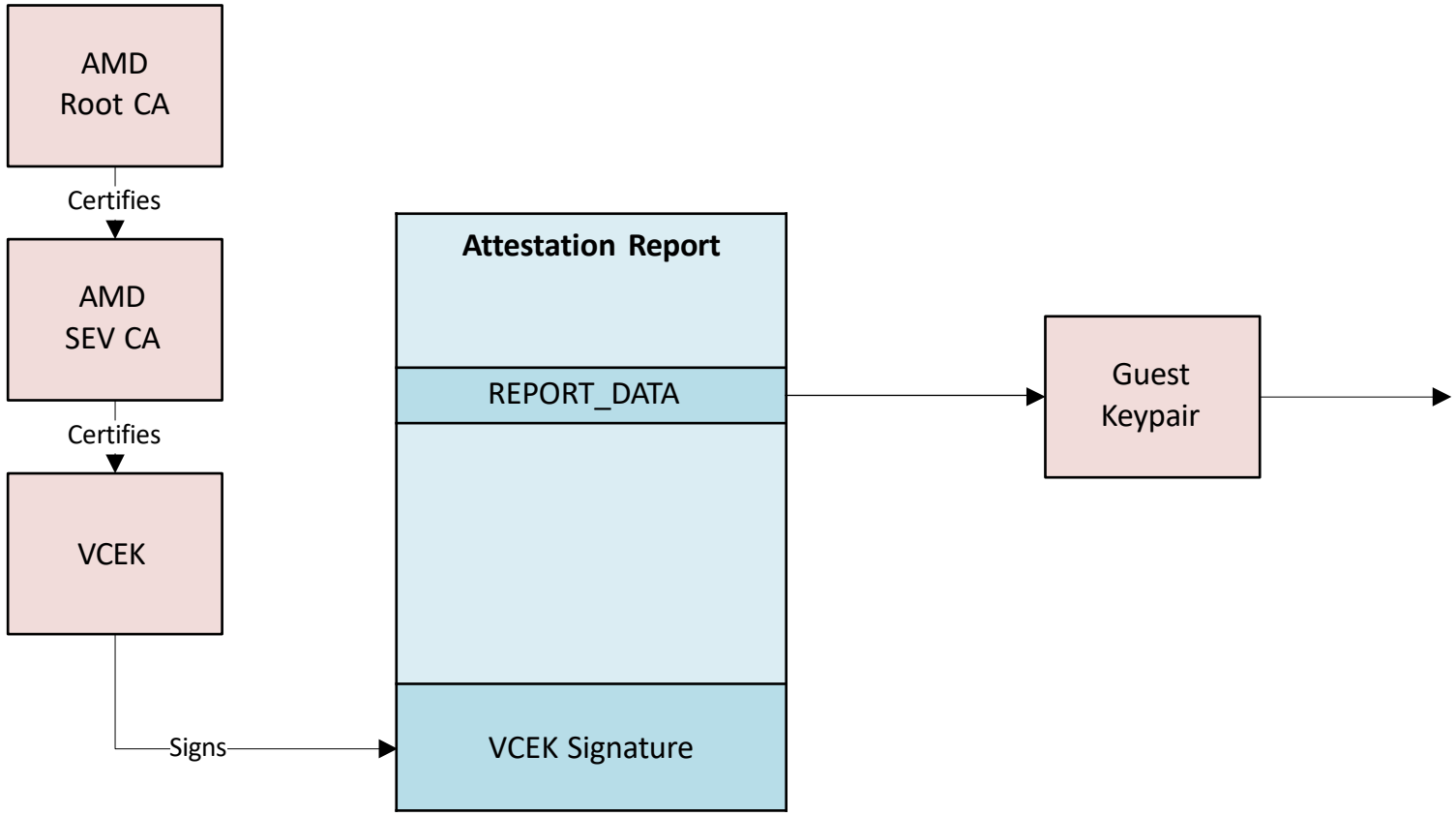


Extended Attestation: Guest owners may request an extended attestation report. In this case, virTEE takes over the task of storing certificate-chains in the hypervisor memory. This eliminates the need for manual requests to the AMD KDS, reduces dependence on external operations.



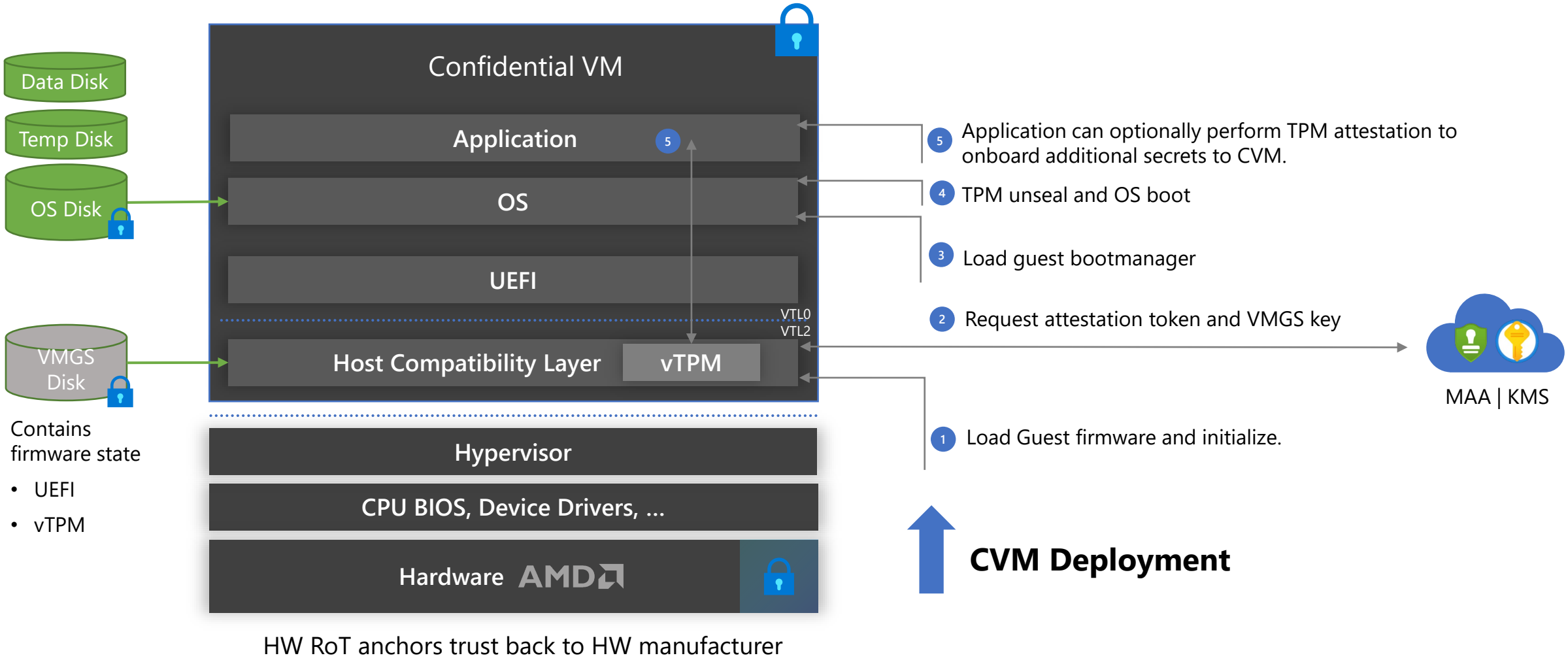
Binding Guest Credentials to Attestation Report

- REPORT_DATA provided by guest
  - 64 btyte field
  - Not interpreted by firmware
  - Placed into the attestation report
- Guest can…
  - Generate a key pair
  - Calculate digest of public key
  - Place digest in REPORT_DATA
  - Send report and public key to relying party to use
  - Key could establish trusted channel
  - Key could chain trust to a virtual TPM in guest

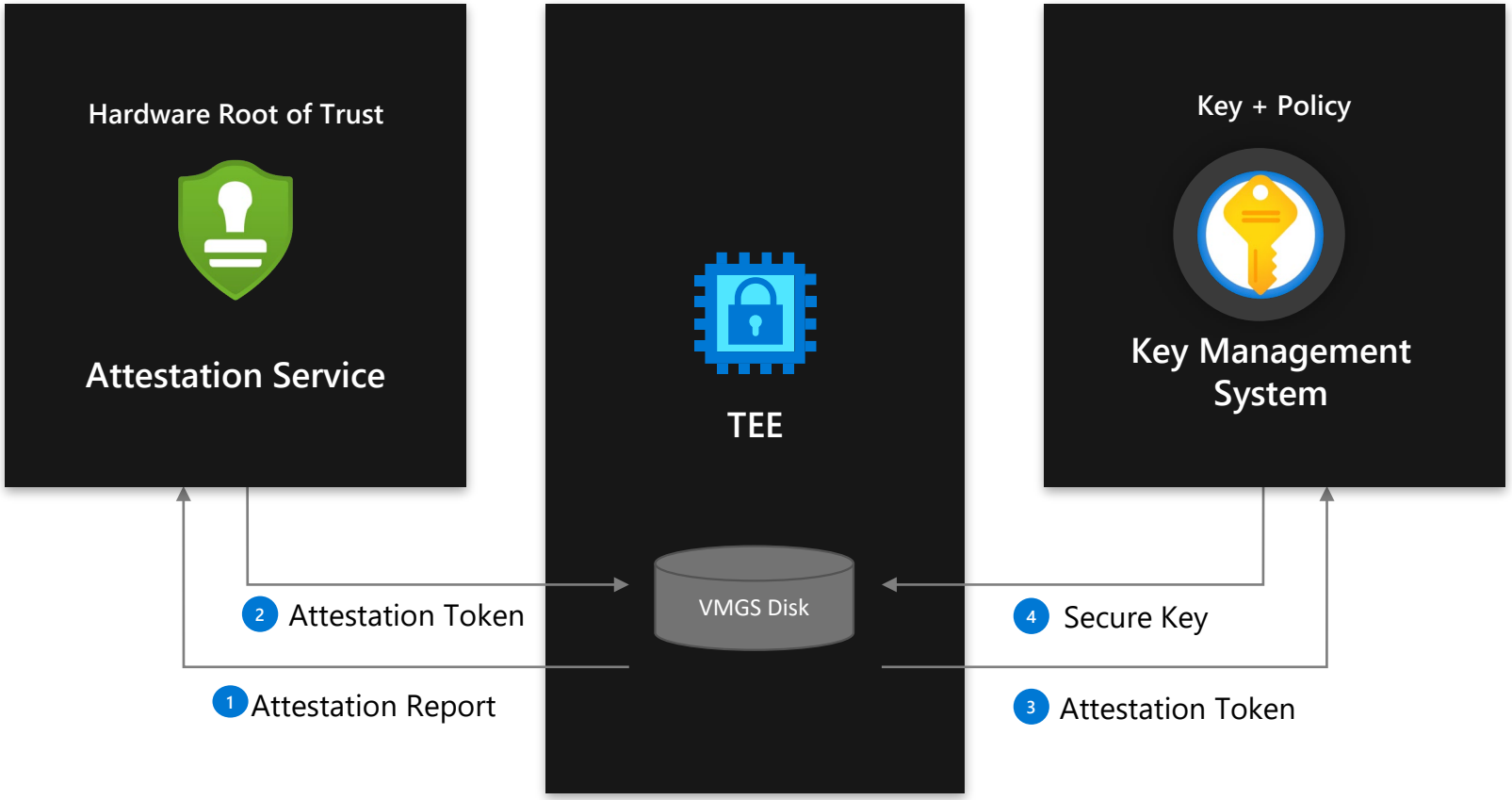


Confidential VMs – Hardware based isolation

Provisioning and deployment flows.



Remote attestation and Secure Key Management



Verify identity and integrity → release keys and secrets

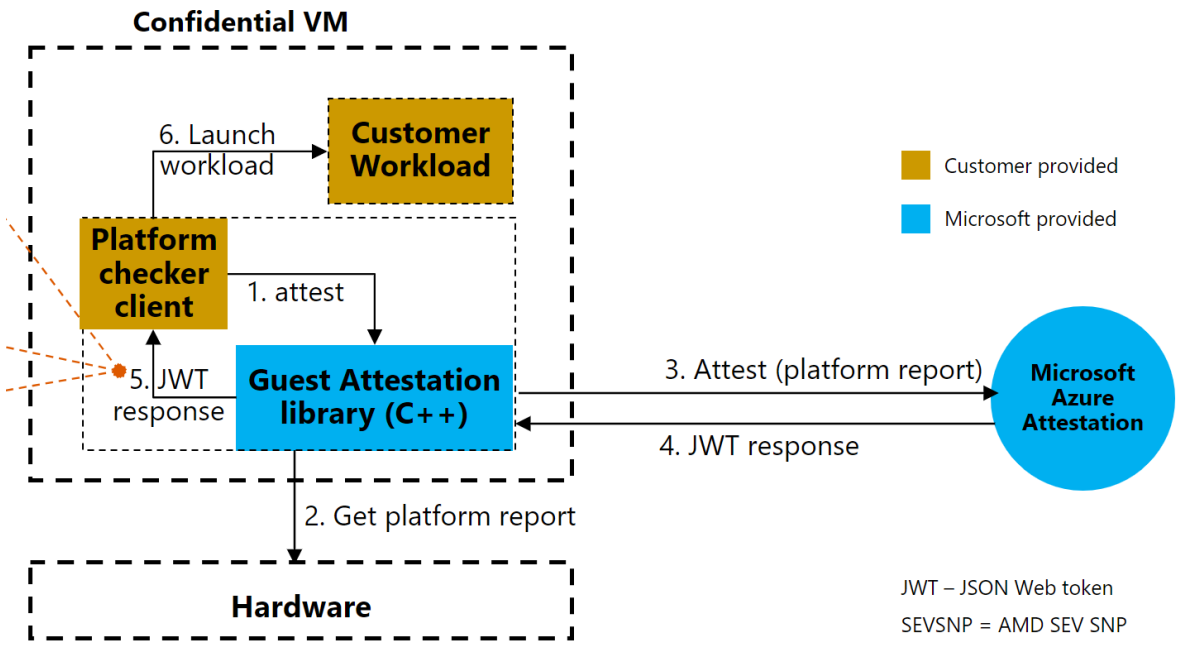
What is guest attestation for confidential VMs?

Scenario Overview:
In this specific scenario, attestation requests are isolated in a separate workload. These requests aim to verify if the confidential VM is running on the accurate hardware platform **prior to launching any workload**.

The Role of Workload:
A workload, represented as **Platform checker client** in the scenario diagram, is required to integrate with the attestation library. This workload runs within the confidential VM and carries out the attestation.

Attestation Request and Response:
The platform checker client initiates a request to the **attestation library**. Post this, the response received is parsed by the workload.

Final Verification:
The parsing of the response allows the workload to **confirm if the VM** is operating on the appropriate hardware platform and/or secure boot setting. This verification is critical and must be completed prior to launching the sensitive workload.



https://github.com/Azure/confidential-computing-cvm-guest-attestation

SEV-SNP attestation example generated by Microsoft Azure Attestation

{
"iss": "https://sharedeus2.eus2.test.attest.azure.net",
"jti": "7afd308c493bac16b980be4f73d479be982306a6790fcf19a68a5f77528b5504",
"nbf": 1680702867,
"nonce": "13104315944768644014",
"x-ms-attestation-type": "sevsnpvm",
"x-ms-compliance-status": "azure-compliant-uvm",
"x-ms-policy-hash": "9NY0VnTQ-IiBriBplVUpFbczcDaEBUwsiFYAzHu_gco",
"x-ms-sevsnpvm-authorkeydigest": "0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000",
"x-ms-sevsnpvm-bootloader-svn": 3,
"x-ms-sevsnpvm-familyId": "01000000000000000000000000000000",
"x-ms-sevsnpvm-guestsvn": 2,
"x-ms-sevsnpvm-hostdata": "42d3ea021f6fbd2e1ce5bc1470ee127673e349b9be815ef87e585a8f8fe2e8d7",
"x-ms-sevsnpvm-idkeydigest": "cf7e12541981e6cafd150b5236785f4364850e2c4963825f9ab1d8091040aea0964bb9a8835f966bdc174d9ad53b4582",
"x-ms-sevsnpvm-imageId": "02000000000000000000000000000000",
"x-ms-sevsnpvm-is-debuggable": false,
"x-ms-sevsnpvm-launchmeasurement": "a1e1a4b64e8de5c664ceee069010441f74cf039065b5b847e82b9d1a7629aaf33d5591c6b18cee48a4dde481aa88d0fb",
"x-ms-sevsnpvm-migration-allowed": false,
"x-ms-sevsnpvm-reportdata": "7ab000a323b3c873f5b81bbe584e7c1a26bcf40dc27e00f8e0d144b1ed2d14f10000000000000000000000000000000000000000000000000000000000000000",
"x-ms-sevsnpvm-reportid": "aaf081e18d64bca741bc5bbcebb6f0493d014b442f030a709979442d4ebcae6a",
}

SEV-SNP attestation evidence from Microsoft Azure Attestation

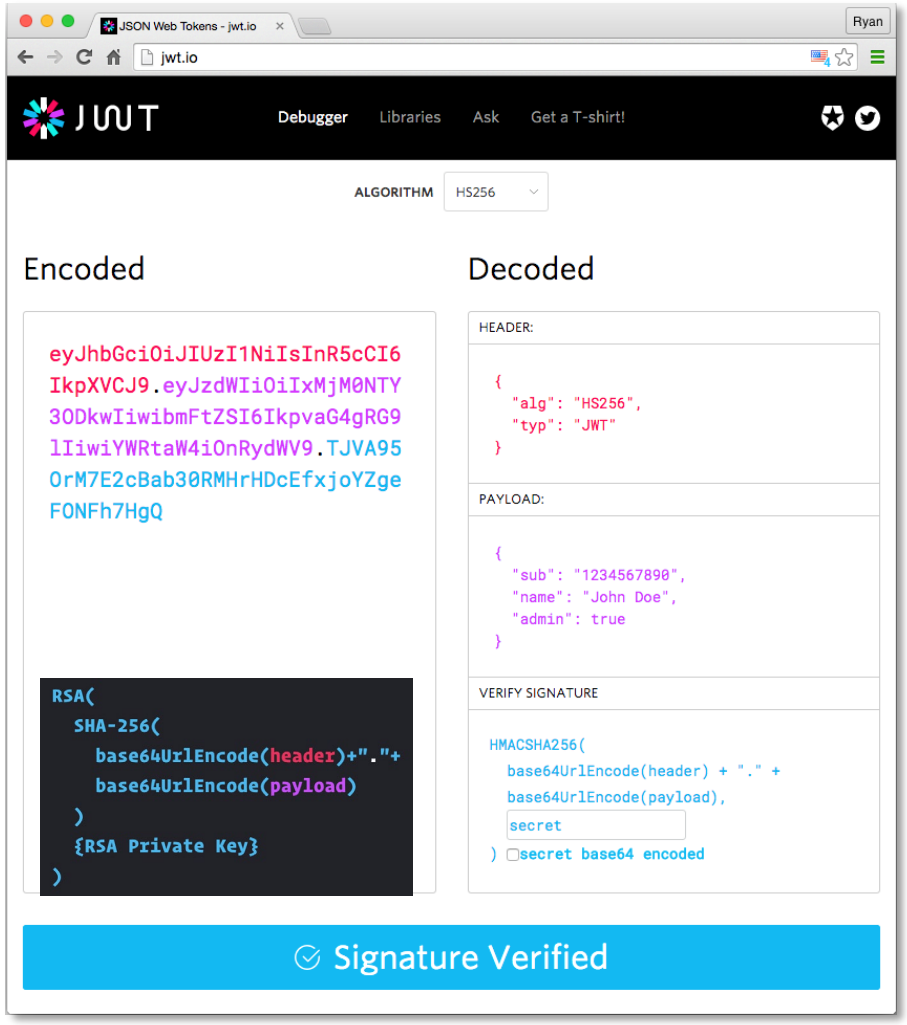
Name	Meaning
x-ms-sevsnpvm-authorkeydigest	The SHA384 hash of the author signing key
x-ms-sevsnpvm-bootloader-svn	The AMD boot loader security version number (SVN)
x-ms-sevsnpvm-familyId	The HCL family identification string
x-ms-sevsnpvm-guestsvn	The HCL security version number (SVN)
x-ms-sevsnpvm-hostdata	**Arbitrary data defined by the host at VM launch time**
x-ms-sevsnpvm-idkeydigest	The SHA384 hash of the identification signing key
x-ms-sevsnpvm-imageId	The HCL image identification
x-ms-sevsnpvm-is-debuggable	Boolean value indicating whether AMD SEV-SNP debugging is enabled
x-ms-sevsnpvm-launchmeasurement	**Measurement of the launched guest image**
x-ms-sevsnpvm-microcode-svn	AMD microcode security version number (SVN)
x-ms-sevsnpvm-migration-allowed	Boolean value indicating whether AMD SEV-SNP migration support is enabled
x-ms-sevsnpvm-reportdata	**CVM: Data passed by HCL to include with report, to verify that transfer key and VM configuration are correct**
x-ms-sevsnpvm-reportid	Report ID of the guest
x-ms-sevsnpvm-smt-allowed	Boolean value indicating whether SMT is enabled on the host
x-ms-sevsnpvm-snpfw-svn	AMD firmware security version number (SVN)
x-ms-sevsnpvm-tee-svn	AMD trusted execution environment (TEE) security version number (SVN)
x-ms-sevsnpvm-vmpl	VMPL that generated this report (0 for HCL)

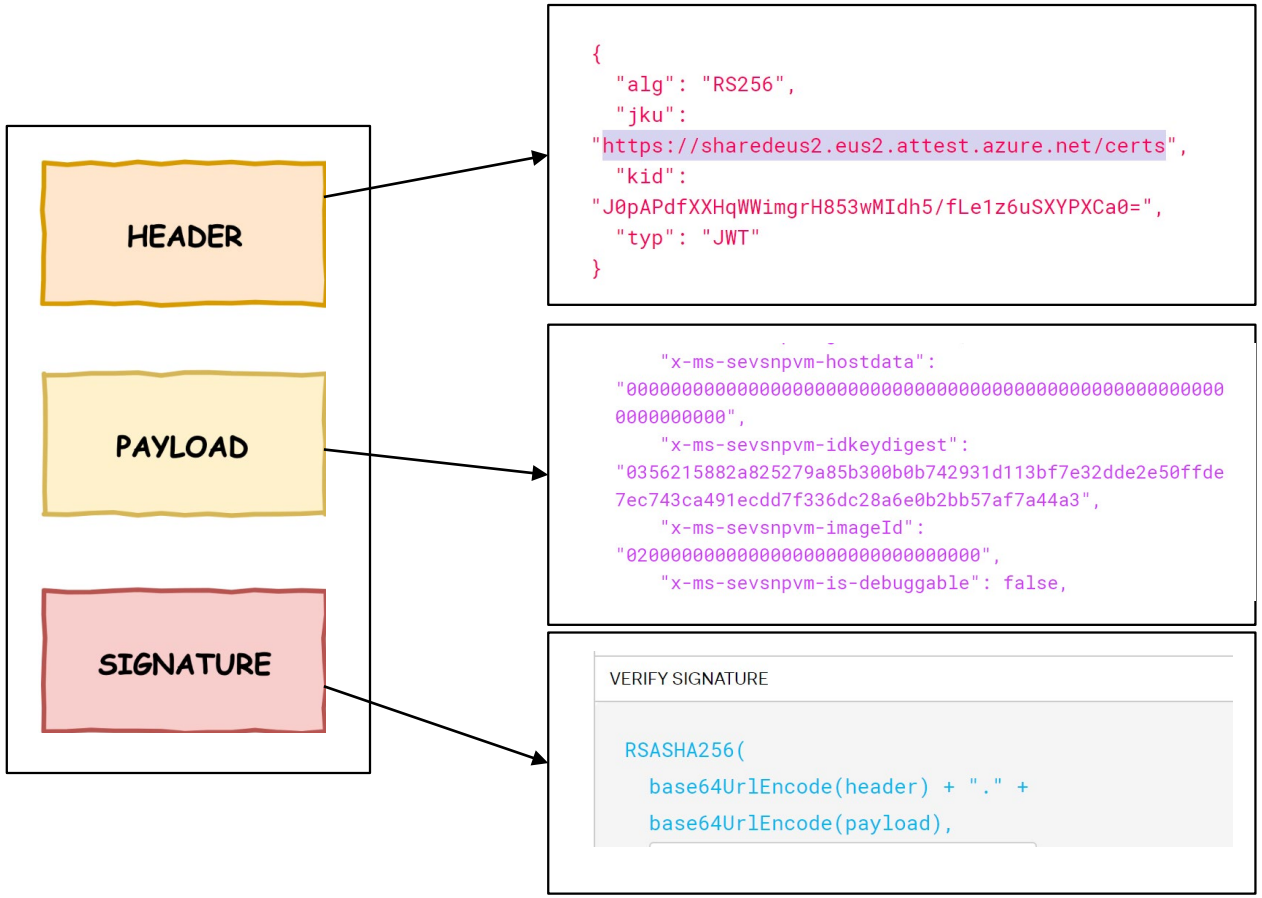
Note: Definitions based on Azure “confidential VM” use of SEV-SNP attestation

Attestation claim details

Claim	Description	Usage and verification	Data size/example
x-ms-attestation-type	Type of processor	Is it running on a genuine AMD sev-snp hardware?	sevsnpvm
x-ms-compliance-status	Type of vm	Is it an azure compliant confidential vm?	azure-compliant-cvm
x-ms-sevsnpvm-hostdata	sha256 hash digest of the security policy	Is the workload what I expect it to be?	256 bits
x-ms-sevsnpvm-**launchmeasurement**	sha-384 digest of utility VM's launch measurement	Is the initial state of the memory, correct?	384 bits
x-ms-sevsnpvm-reportdata	When the guest asks for a report, it supplies 512 bits of arbitrary data to be included in the report as ReportData.	Is the data provided by the hypervisor at launch, correct?	512 bits

Validating JSON Web Tokens (JWT)





Signature Verification steps

1. Extract the JWT header, payload, and signature segments
2. Base64url decode each segment and convert to bytes
3. Concatenate the JWT header and payload segments with a period delimiter
4. Calculate the SHA-256 hash of the JWT Signing Input
5. Load the public key from a the x509 certificate
6. Verify the RSA signature
7. Compare the hex representations of digest and decrypted signature

Hands-on lab:
Generating and validating attestation report

Hands-on lab will cover – cvm_attestation.ipynb

- Highlighting essential claims extracted from Microsoft Azure Attestation, including details of the Azure confidential VM and its configuration parameters.
- Validating the JWT's signature through attestation signature validation using a Python script. This validation involves comparing the calculated SHA-256 hash with the decrypted signature.
- Introducing a helper application for remote attestation using a web API. This application facilitates the setup and retrieval of attestation by requesting it from the VM and displaying the JWT token.

Hands-on lab will cover - cvm_attestation_deepdive.md

- Manually fetching and verifying a raw sev-snp report
- Installing the tpm2-tss firmware
- Extracting the VCEK certificate from a CVM
- Using the virtee/snpguest tool to validate the report

Confidential Containers on Azure Containers Instances

Confidential Containers on ACI

Confidential containers on Azure Container Instances (ACI)

GENERALLY AVAILABLE

- No infrastructure management
- Run any image or language framework
- Pay per second billing

General availability: Confidential containers on Azure Container Instances (ACI)

Published date: May 23, 2023

Confidential containers on Azure Container Instances (ACI), now generally available, enables you to run containers in a trusted execution environment (TEE) that provides hardware-based confidentiality and integrity protections for your container workloads while in use in memory.

Confidential containers on ACI is supported as a new SKU that you can select when deploying your workload and will provide you with the following benefits for workloads processing highly sensitive data:

Ability to lift and shift workloads to a confidential environment without needing to take any dependencies on any confidential computing libraries.

In-memory encryption of data with a hardware based dedicated key per container group helping to guard against attacks from a malicious OS, or Hypervisor components.

Support for remote attestation to enable a relying party to verify that a service is running in a TEE before processing any sensitive data. As part of confidential containers on ACI, an agent will validate the authenticity of the hardware and application components which can be verified through a remote attestation service before any sensitive data is released to the TEE.

To learn more, read the blog announcement and documentation.

ACI is 100% open-source, leveraging Open GCS.

Confidential containers should be?

- Transparent & Auditable environment
- Reproducible environment
- Container + Config integrity
- Immutable post launch
- Measure all and attest
- Smallest possible Trusted Computing Base (TCB)
- Data and code shielded from host/operator access
- A hardened env/locked down access

[Confidential Computing Consortium]

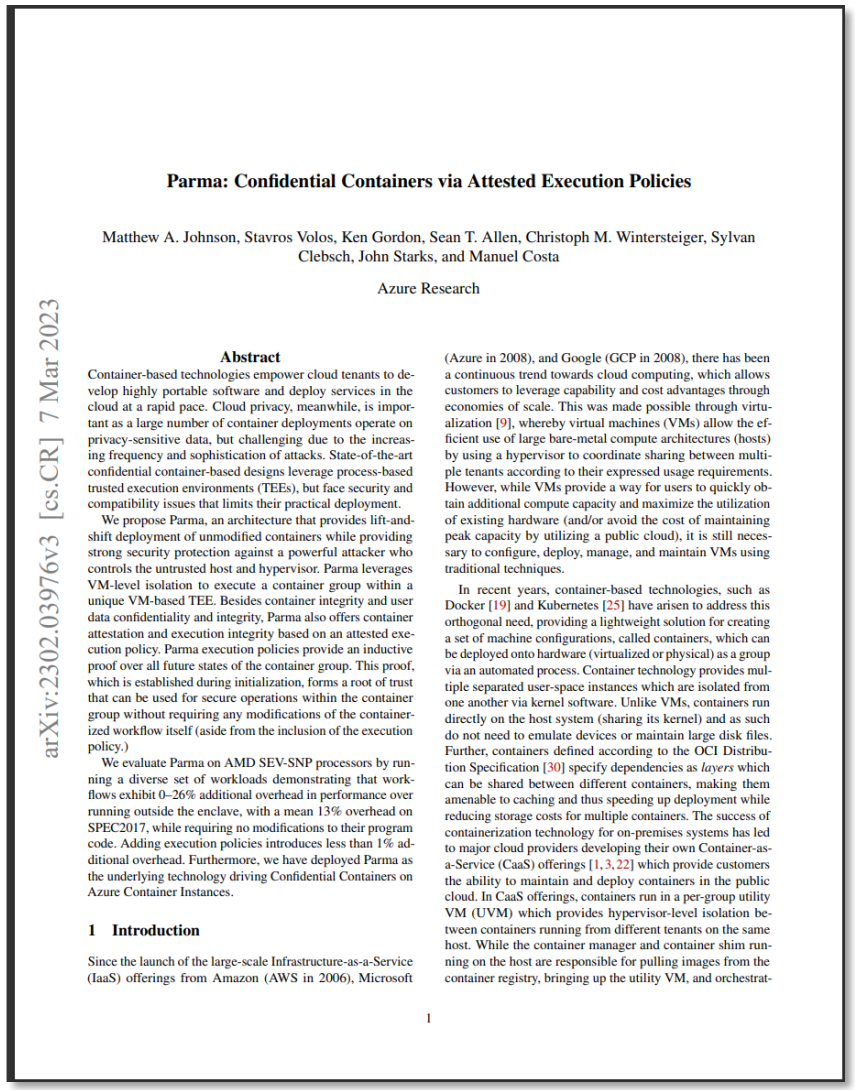
[Confidential Containers (github.com)]

[Kata Containers - Open Source Container Runtime Software | Kata Containers]

[Open Guest Compute Service – Azure Container Instances.]

Parma

1. Parma is an architecture that provides **lift-and-shift deployment of unmodified containers** while providing strong security protection against a powerful attacker who controls the untrusted host and hypervisor.
2. Parma, a novel security architecture for **confidential containerized workloads**, establishes security guarantees rooted in an inductive proof over all future states of the container group provided by the **introduction of an attested execution policy**.
3. The **execution policy** is a **component of the utility VM** that is **attested at initialization time**, describing **all of the actions the user has explicitly allowed the guest agent** to take within the container group. When a hash does not match, this action is denied.
4. The **guest agent** coordinates the container workflow as directed by the host-side container shim and is **augmented to enforce the execution policy** such that it only executes commands (submitted by the untrusted container shim) **explicitly allowed by the user.**
5. Parma leverages **VM-level isolation to execute a container group within a unique VM-based TEE**, offering container integrity and user data confidentiality and integrity, as well as container attestation and execution integrity based on an **attested execution policy**.
6. Parma provides strong protection for the container's root filesystem (comprised of the container image layers and writeable scratch space) and the **user's data through block-level encryption and integrity (using dm-crypt + dm-integrity**).
7. The implementation of **Parma forms the basis for Confidential Containers** on Azure Container Instances and is publicly available on GitHub **hcsshim system**, **neither requiring changes to existing containers nor container image signing**.



arXiv:2302.03976v3 [cs.CR] 7 Mar 2023

Parma: Confidential Containers via Attested Execution Policies

Matthew A. Johnson, Stavros Volos, Ken Gordon, Sean T. Allen, Christoph M. Wintersteiger, Sylvan Clebsch, John Starks, and Manuel Costa

Azure Research

Abstract

Container-based technologies empower cloud tenants to develop highly portable software and deploy services in the cloud at a rapid pace. Cloud privacy, meanwhile, is important as a large number of container deployments operate on privacy-sensitive data, but challenging due to the increasing frequency and sophistication of attacks. State-of-the-art confidential container-based designs leverage process-based trusted execution environments (TEEs), but face security and compatibility issues that limits their practical deployment.

We propose Parma, an architecture that provides lift-and-shift deployment of unmodified containers while providing strong security protection against a powerful attacker who controls the untrusted host and hypervisor. Parma leverages VM-level isolation to execute a container group within a unique VM-based TEE. Besides container integrity and user data confidentiality and integrity, Parma also offers container attestation and execution integrity based on an attested execution policy. Parma execution policies provide an inductive proof over all future states of the container group. This proof, which is established during initialization, forms a root of trust that can be used for secure operations within the container group without requiring any modifications of the containerized workflow itself (aside from the inclusion of the execution policy.)

We evaluate Parma on AMD SEV-SNP processors by running a diverse set of workloads demonstrating that workflows exhibit 0–26% additional overhead in performance over running outside the enclave, with a mean 13% overhead on SPEC2017, while requiring no modifications to their program code. Adding execution policies introduces less than 1% additional overhead. Furthermore, we have deployed Parma as the underlying technology driving Confidential Containers on Azure Container Instances.

1 Introduction

Since the launch of the large-scale Infrastructure-as-a-Service (IaaS) offerings from Amazon (AWS in 2006), Microsoft (Azure in 2008), and Google (GCP in 2008), there has been a continuous trend towards cloud computing, which allows customers to leverage capability and cost advantages through economies of scale. This was made possible through virtualization [9], whereby virtual machines (VMs) allow the efficient use of large bare-metal compute architectures (hosts) by using a hypervisor to coordinate sharing between multiple tenants according to their expressed usage requirements. However, while VMs provide a way for users to quickly obtain additional compute capacity and maximize the utilization of existing hardware (and/or avoid the cost of maintaining peak capacity by utilizing a public cloud), it is still necessary to configure, deploy, manage, and maintain VMs using traditional techniques.

In recent years, container-based technologies, such as Docker [19] and Kubernetes [25] have arisen to address this orthogonal need, providing a lightweight solution for creating a set of machine configurations, called containers, which can be deployed onto hardware (virtualized or physical) as a group via an automated process. Container technology provides multiple separated user-space instances which are isolated from one another via kernel software. Unlike VMs, containers run directly on the host system (sharing its kernel) and as such do not need to emulate devices or maintain large disk files. Further, containers defined according to the OCI Distribution Specification [30] specify dependencies as *layers* which can be shared between different containers, making them amenable to caching and thus speeding up deployment while reducing storage costs for multiple containers. The success of containerization technology for on-premises systems has led to major cloud providers developing their own Container-as-a-Service (CaaS) offerings [1, 3, 22] which provide customers the ability to maintain and deploy containers in the public cloud. In CaaS offerings, containers run in a per-group utility VM (UVM) which provides hypervisor-level isolation between containers running from different tenants on the same host. While the container manager and container shim running on the host are responsible for pulling images from the container registry, bringing up the utility VM, and orchestrat-

1

Confidential Containers via Attested Execution Policies

ACI Trusted execution environment

Microsoft Azure

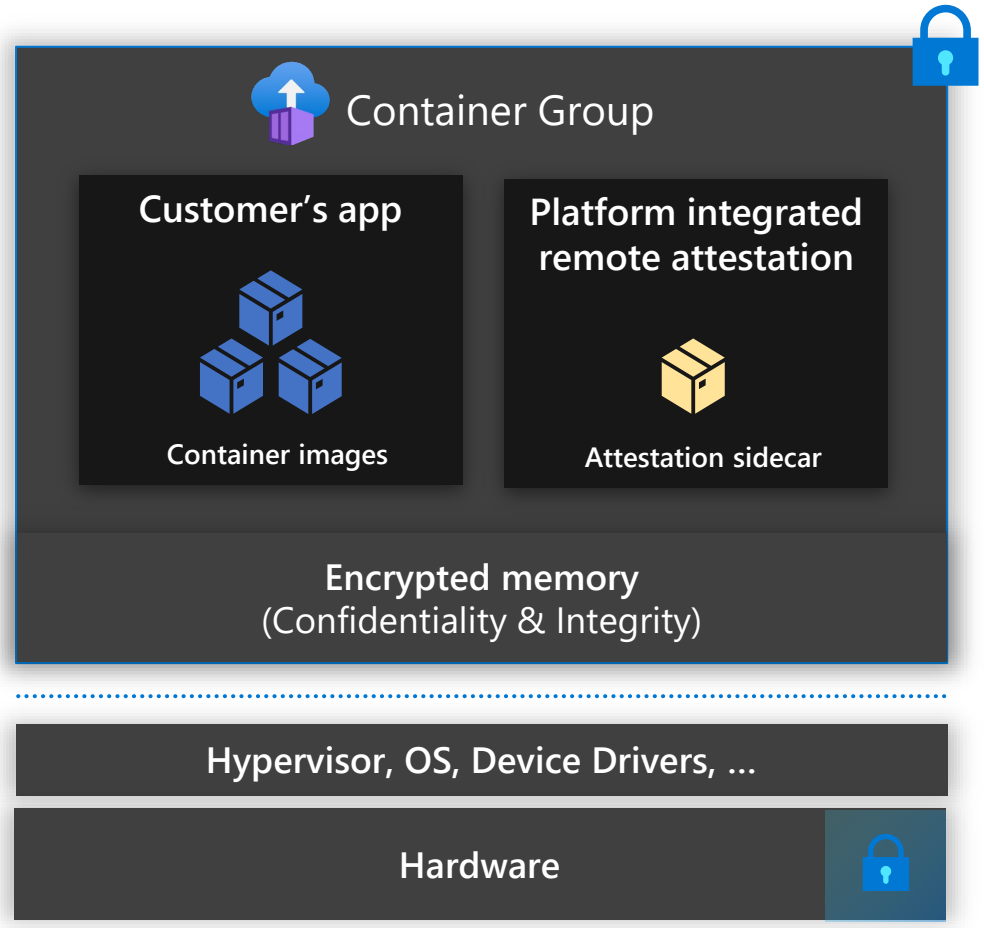
Customers can **lift and shift existing container images** while receiving in-memory encryption and attestation

Each container group is **isolated through Hyper-V**

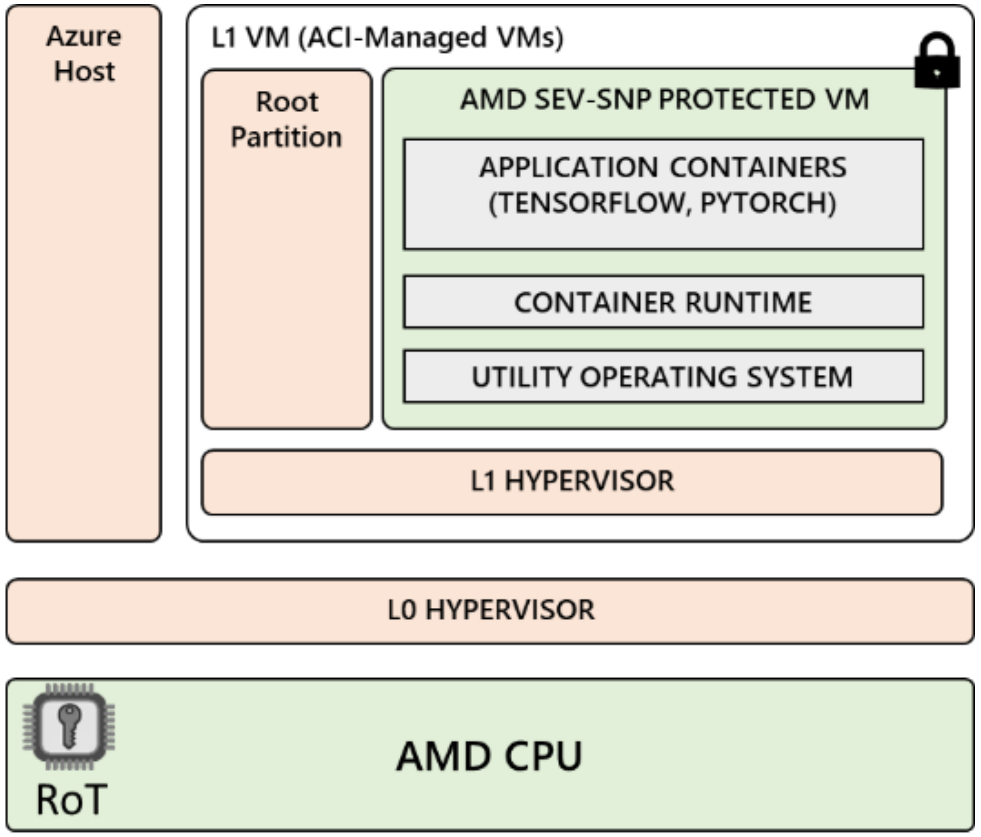
Hardware-based memory encryption with a key per container group

Container code and config **integrity protection** and immutable container groups

Integrated platform sidecars to make it **easy to perform remote attestation** measuring all hardware and software components



Confidential Containers on ACI



Common Confidential ACI workloads

CONFIDENTIAL DATA PROCESSING

CONFIDENTIAL ML TRAINING (NON-AML)

CONFIDENTIAL MICRO-SERVICE API/FUNCTION

CRYPTO OPERATIONS CONTAINER SECURING

MULTI-PARTY CONFIDENTIAL ANALYTICS

[Confidential containers on ACI]

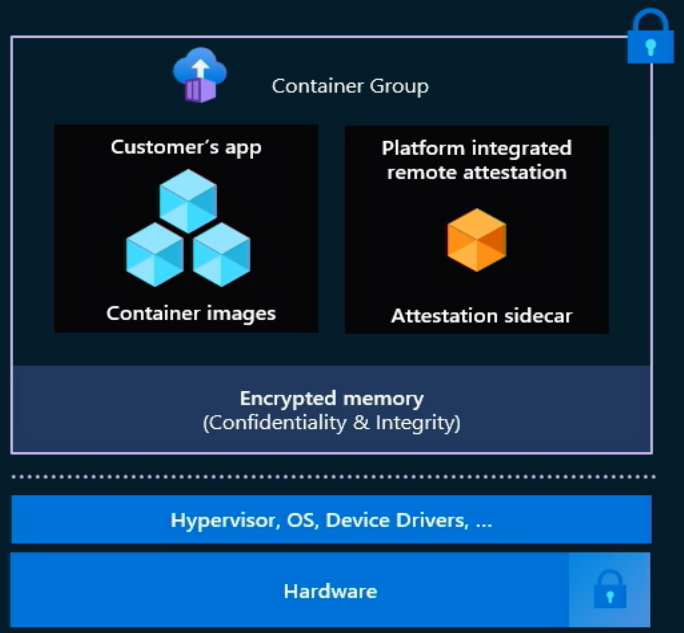
Confidential Containers on ACI

Customer motivation

Containers have become widely adopted for running container-based applications within Azure and customers are deploying more scenarios that are focused on data privacy.

40% of customers using containers for most of their production workloads cited security as their top challenge.*

Confidential computing provides mechanisms to support applications processing highly sensitive data but can be complex to use due to specialized programming models and infrastructure management.



Serverless Solution

Ability to lift and shift existing applications without requiring modifications making ACI the quickest and easiest way to leverage confidential computing.

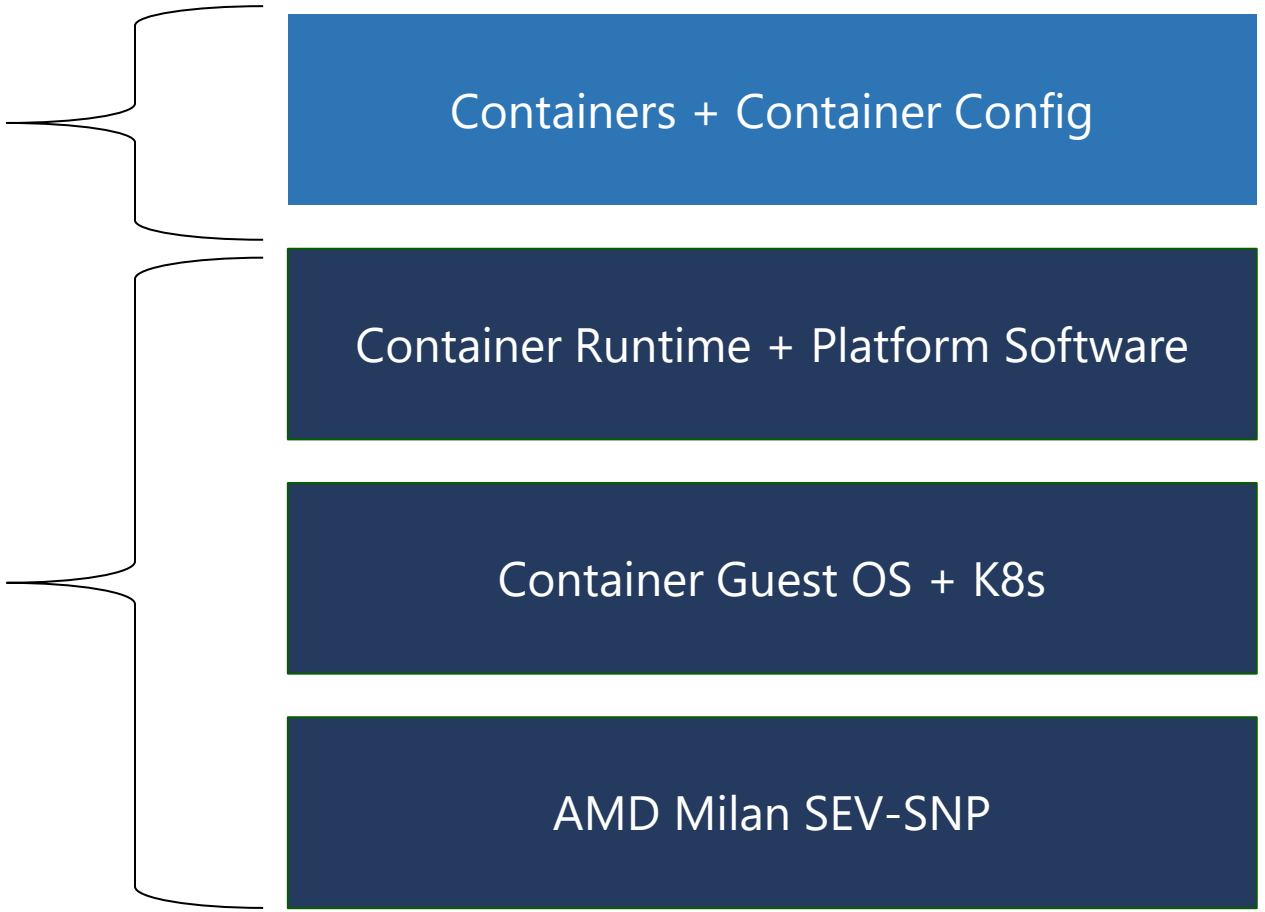
Increased security posture through data in use protection and encryption provided by the latest in confidential computing hardware.

Data integrity provided by platform integrated sidecars ensures only the code that a customer trusts is running within their container group.

Attestation evidence for verification

Report_Data:
Your code and configs

Host_Data:
Platform code and configs



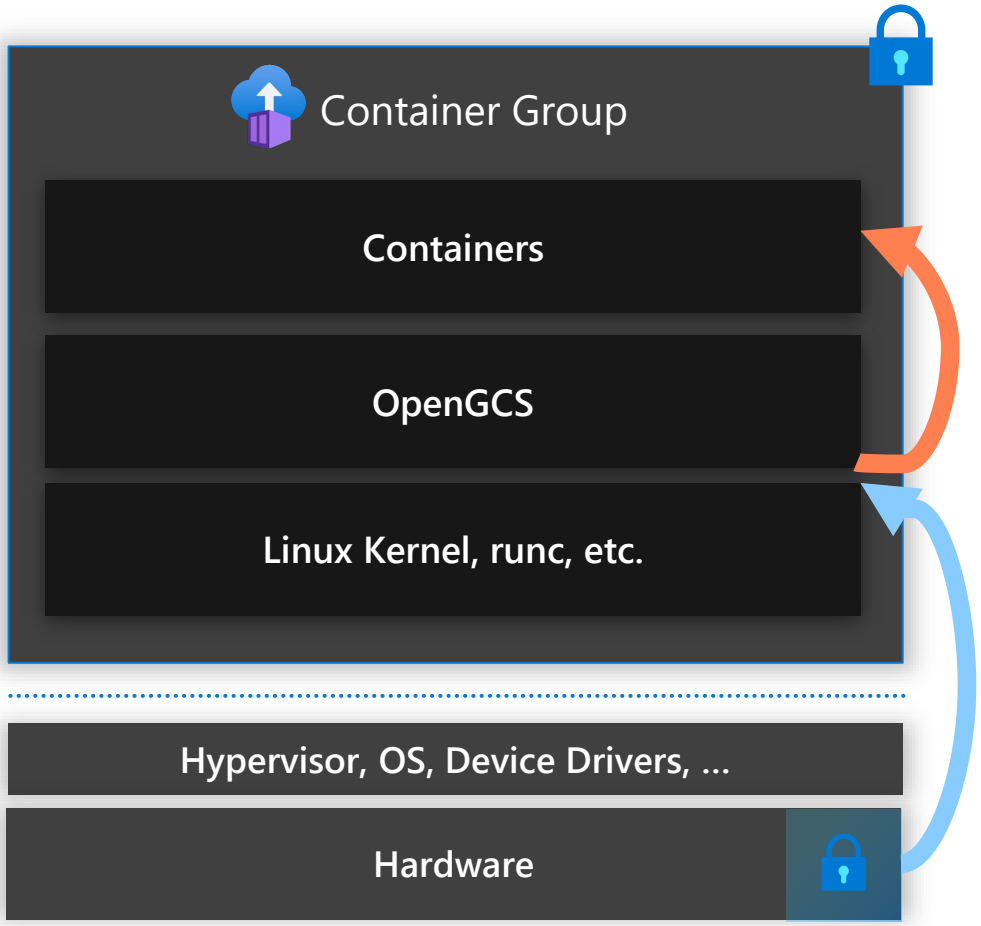
Example Raw SNP report & Microsoft Azure Attestation Quote

```
AMD SEV-SNP Attestation Report:

version:
00000002
guest_svn:
00000002
policy:
000000000003001f
family_id:
00000000000000000000000000000001
image_id:
00000000000000000000000000000002
vmpl:
00000000
signature_algo:
00000001
platform_version:
03000000000008ce
platform_info:
0100000000000000
author_key_en:
00000000
reserved1:
00000000
report_data:
0000000000000000000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000
measurement:
ccec9231a24445e962a56ca6f4844458 71e17540d4ab7b49a8c036681af07802
8ad6677879cd062c5922c7d56290e98a
host_data:
73973b78d70cc68353426de188db5dfc 57e5b766e399935fb73a61127ea26d20
id_key_digest:
f86e45cf4c1b89fd088c3fc1dbf020c7 15b5307f505655d16dc3bd4eb9e4ac20
3a1d1ce87813c6bf230b643314c4e0aa
author_key_digest:
0000000000000000000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000
report_id:
b40ee2826499c96361ecd071a2f138d3 0bdf52c532ae536245547b5f5381313e
report_id_ma:
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
reported_tcb:
7308000000000003
reserved2:
00000000000000000000000000000000000000000000000000000000000000000000000000000000
chip_id:
e3a0c8c2bd4bd931158b7f9203bb329b 9ee660ef9095c497da9b5d09a1cd43fe
b968eecd41bd84dab54095b53b929063 dce2a45908bff55df839b4ce3421570f
committed_svn:
0300000000000873
committed_version:
0434010004340100
launch_svn:
0300000000000873
reserved3:
0000000000000000000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000
0000000000000000
signature:
b897ee36c439ab742ffba12fcd9ef665 7b5f7ccdae540fd900afb7fde4386573
6c64768f612cf8e925211494ab0fa037 0000000000000000000000000000000000000000000000000000000000000000
000000000000000004a8746f7382a091 b9334b87b128c330caab95deaea1a218
e8d784a5deb766541595163104 6ac0cb 09abbf4d4a2015750000000000000000
0000000000000000000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000
```


```
{
  "alg": "RS256",
  "jku": "https://sharedeus2.eus2.attest.azure.net/certs",
  "kid": "J0pAPdfXXHqWWimgrH853wMIdh5/fLe1z6uSXYPXCa0=",
  "typ": "JWT"
}
{
  "exp": 1664193003,
  "iat": 1664164203,
  "iss": "https://sharedeus2.eus2.attest.azure.net",
  "jti": "768f851235c78616ee6a0f59a82dc69b8e70770c3339b34117b729cdc91ddebd",
  "nbf": 1664164203,
  "secureboot": true,
  "x-ms-attestation-type": "azurevm",
  "x-ms-azurevm-attestation-protocol-ver": "2.0",
  "x-ms-azurevm-attested-pcrs": [
    0,
    1,
    2    ],
  "x-ms-azurevm-bootdebug-enabled": false,
  "x-ms-azurevm-dbvalidated": true,
  "x-ms-azurevm-dbxvalidated": true,
  "x-ms-azurevm-debuggersdisabled": true,
  "x-ms-azurevm-default-securebootkeysvalidated": true,
  "x-ms-azurevm-elam-enabled": false,
  "x-ms-azurevm-flightsigning-enabled": false,
  "x-ms-azurevm-hvci-policy": 0,
  "x-ms-azurevm-hypervisordebug-enabled": false,
  "x-ms-azurevm-is-windows": false,
  "x-ms-azurevm-kerneldebug-enabled": false,
  "x-ms-azurevm-osbuild": "NotApplication",
  "x-ms-azurevm-osdistro": "Ubuntu",
  "x-ms-azurevm-ostype": "Linux",
  "x-ms-azurevm-osversion-major": 18,
  "x-ms-azurevm-osversion-minor": 4,
  "x-ms-azurevm-signingdisabled": true,
  "x-ms-azurevm-testsigning-enabled": false,
  "x-ms-azurevm-vmid": "B958DC88-E41D-47F1-8D20-E57B6B7E9825",
  "x-ms-isolation-tee": {
    "x-ms-attestation-type": "sevsnpvm",
    "x-ms-compliance-status": "azure-compliant-uvm",
    "x-ms-runtime": {
```

Trust Chain



Launch Process:
1. Hardware measures the container group software stack at container group creation time and captures the hash of the security policy.

2. The OpenGCS agent ensures that only "allowed" containers with "allowed" configurations can be loaded in reference to the security policy.

Attestation Process:
1. Software requests Attestation evidence from the hardware.

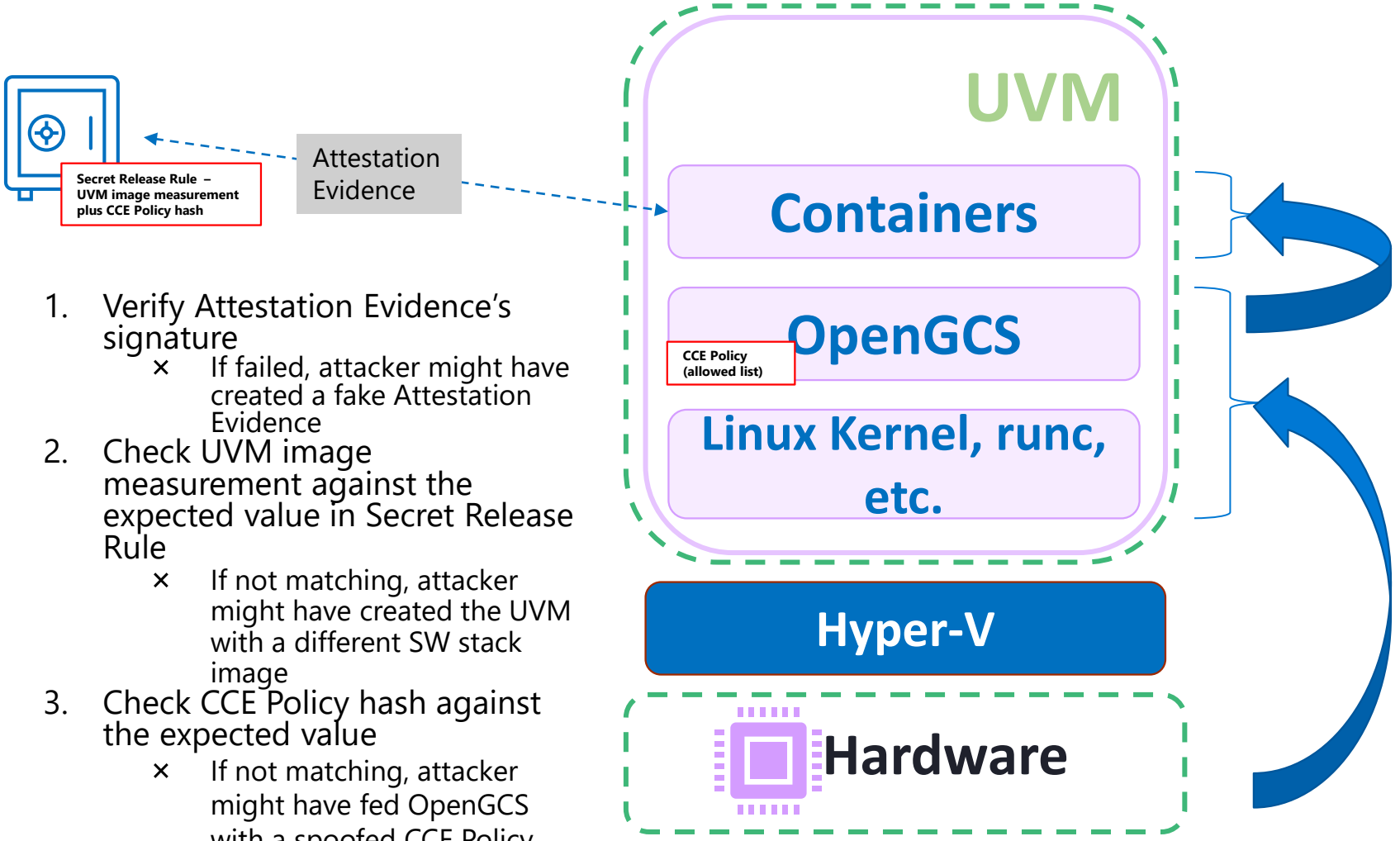
2. Hardware generates attestation evidence with the measurement and security policy hash which is signed by a private key that is only accessible by the TEE hardware.

3. Verify attestation evidence signature
- If failed, attacker might have created fake attestation evidence

4. Check UVM image measurement against the expected value in Secret Release Rule
- If not matching, attacker might have created the UVM with a different software stack image

5. Check security policy hash against the expected value
- If not matching, attacker might have fed OpenGCS with a spoofed security policy

Policy enforced trusted execution environment



1. Verify Attestation Evidence's signature
   - × If failed, attacker might have created a fake Attestation Evidence
2. Check UVM image measurement against the expected value in Secret Release Rule
   - × If not matching, attacker might have created the UVM with a different SW stack image
3. Check CCE Policy hash against the expected value
   - × If not matching, attacker might have fed OpenGCS with a spoofed CCE Policy

SW requests an Attestation Evidence from the HW

- HW generates Attestation Evidence, with the measurement and Security Policy hash, signed by a private key only accessible by the TEE HW

OpenGCS makes sure only "allowed" containers with "allowed" configuration can be loaded in reference to the CCE Policy (Allowed List)

HW measure the UVM SW stack at UVM creation time; the HW also captures the hash of the CCE Policy

Enforce code integrity through customer defined policies

Confidential computing enforcement (CCE) policy


Container images


Environment variables


Logging


Volume mounts


Container privileges

- Enforced by OpenGCS
- Confidential computing enforcement policies must be generated by the Azure CLI confcom extension and can't be manually created.

What is the content of a CCE policy?

```
},
"confidentialComputeProperties": {
  "ccePolicy": "<base64-standard-encoded-string of security policy>",
  "isolationType": "sevsnp"
}
```

- Which containers are permitted to run in the utility VM along with the hashes of each layer filesystem of the container image,
- What command lines can be used as entry point,
- The permitted environment variables and their values,
- Whether a container is allowed to run as privileged.

CCE Policy Validation in ACI Deployment

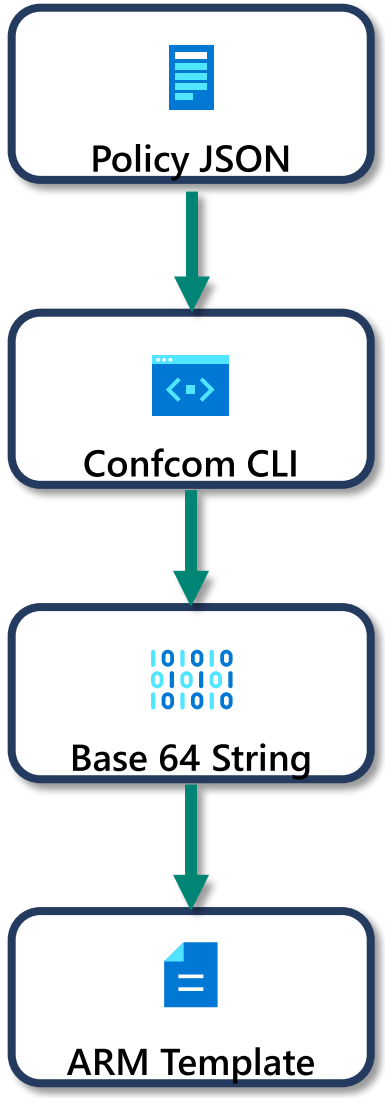
Customer runs a CLI tool on the deployment template to get an output of the CCE policy and then calls the create ACI call with the deployment template and policy.

The CCE policy is ultimately a representation the following:
- **Container image & configuration** parsed from the deployment template
- **Side Cars (fragments)** that ACI will insert next to their container based on the deployment template

At the time of deployment ACI will validate this incoming CCE policy with what it parses from the deployment template and only proceed if both match. If the customer tries to edit this policy (e.g. removing side cars) the deployment will fail. Which means the customer either takes the policy as is or leaves it.

Side cars are represented in the policy via **Fragments.**

Generating confidential computing enforcement policies



JSON document defines what is allowed to run within the TEE

CLI tooling takes the JSON document as input and calculates a policy which can be enforced by Open GCS

The policy is outputted as a base 64 string containing all the properties that need to be validated at container launch time

The base 64 string is added to the CCE property of the container group which is now ready for deployment

Sample CCE Policy from ConfCom CLI

```
package policy

api_svn := "0.7.0"

containers := [## Definitions for each container to create ##]
fragments := [## List of fragments that can be loaded ##]

mount_device := data.framework.mount_device
unmount_device := data.framework.unmount_device
mount_overlay := data.framework.mount_overlay
unmount_overlay := data.framework.unmount_overlay
create_container := data.framework.create_container
exec_in_container := data.framework.exec_in_container
exec_external := data.framework.exec_external
shutdown_container := data.framework.shutdown_container
signal_container_process := data.framework.signal_container_proces
plan9_mount := data.framework.plan9_mount
plan9_unmount := data.framework.plan9_unmount
load_fragment := data.framework.load_fragment
reason := {"errors": data.framework.errors}
```

Created by CLI tool, based on Container images and Conf. Specified in ARM Template

Inserted by CLI tool

Inserted by CLI tool

Sample CCE policy – More Details

```
"ccePolicy": "package policy

import future.keywords.every
import future.keywords.in

api_version := \"0.10.0\"
framework_version := \"0.2.3\„

fragments:= {\"feed\": \"mcr.microsoft.com/aci/aci-cc-infra-fragment\", \"includes\":
["containers\",\"fragments\"],\"issuer\":\"did:x509:0:sha256:I__iuL25oXEVEFdTP_aBLx_eT1RPHbCQ_ECBQbFYZpt9s::eku:1.3.6.1.4.1.311.76.
59.1.3\",\"minimum_svn\": \"1\"}

containers:= [
{"allow_elevated":false,
"command":["/bin/sh","-c","node /usr/src/app/index.js"],
"env_rules":[
            {"rule":"PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin","strategy":"string"},
            {"rule":"azurecontainerinstance_restarted_by=.+","strategy":"re2"}
            ],
"id":“samplesnpregistry.azurecr.io/custom-aci-hello:4g",
"layers":["c6d324ab16cc7589e917c706b19a1137f7835d2c77b4604c28f9494dd8b60959",”
c6d324ab16cc7589e917c706b19a1137f7835d2c77b4604c28f9494dd8b60959”}],
"working_dir":"/usr/src/app“}]
```

“name”

Signing key digest

Min. SVN

DM-Verity measurements of the container layers

CCE Policy

- CCE Policy is created from the deployment template to give customer full visibility into how ACI will provision the container from the deployment template
- The policy defines the entities and constraints that are allowed in a confidential execution environment. The policy contains the customer workload containers and their layers, mount information, and system sidecar's config/identity via fragment references. This policy is required for all confidential container groups created in ACI. The policy is enforced by the OpenGCS after the start of the UVM and helps in achieving full attestation of the customers TEE.
- CCE Policy contains the following main parts (all extracted from the deployment template)
  - Fragment REGO representing the side car
    - fragment feed (pointing to the MCR location containing the side car details)
    - SVN on the fragment (min version)
    - Public Key of the Fragment (to allow customer to audit the signing)
  - Customer Images & Layers
  - Environment Variables
  - Mounts
  - Execution Commands
- When the CCE Policy reaches the lower layers in ACI, OpenGCS will validate that the policy matches what ACI is attempting to execute. Only if the actions match the policy the container will be created. This means if the customers tries to edit the policy (add/ remove side cars as an example), it will be rejected.

What is Rego?

Fragments are represented by Rego in the CCE Policy

Rego Sample

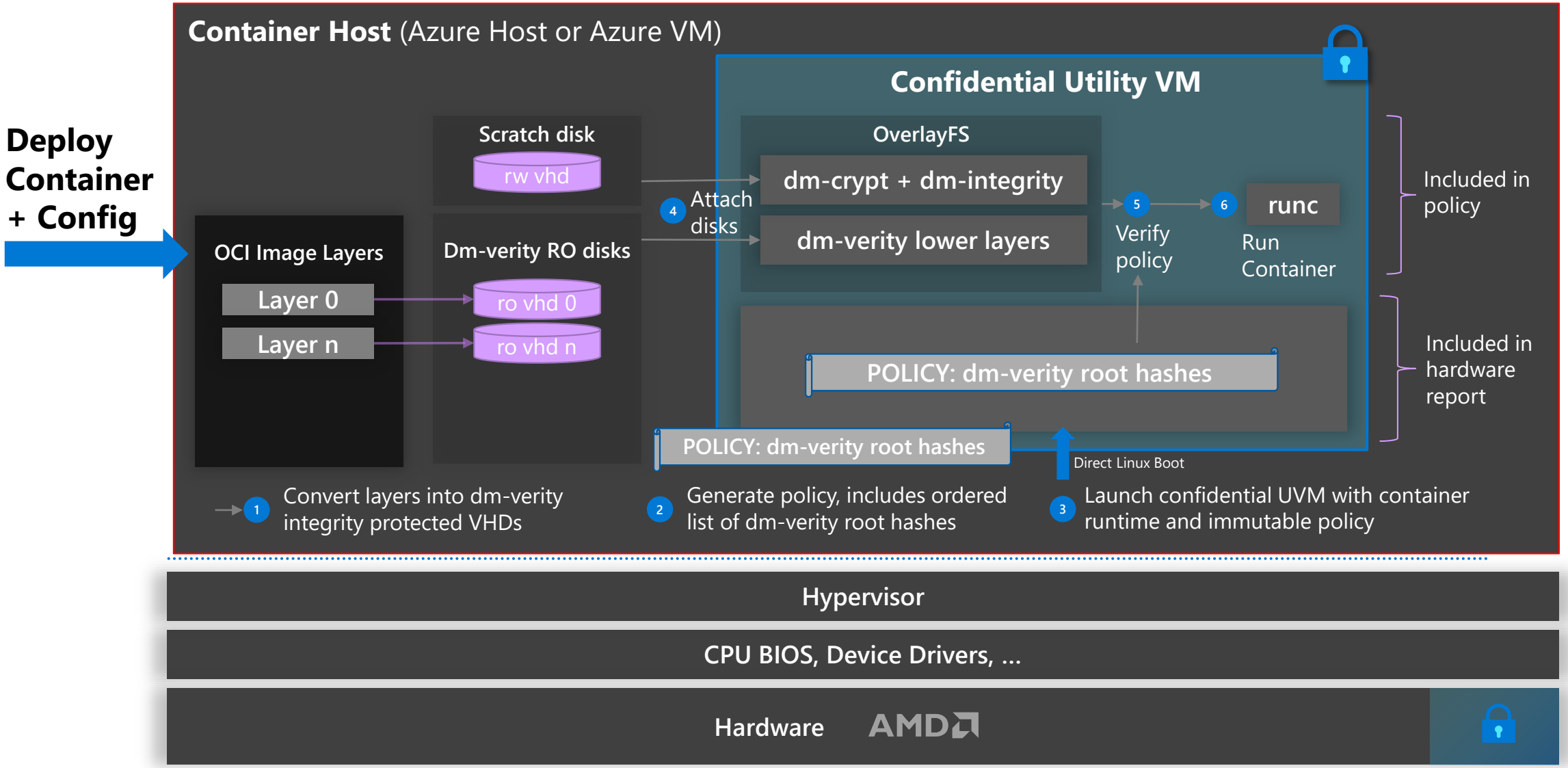
```
1  package example.rules
2
3  public_network[net.id] {      # net.id is in the public_network set if...
4
5      net := input.networks[_]  # some network exists and...
6
7      net.public                        # it is public.
8
9  }
```

Rego is the purpose-built declarative policy language that supports Open Policy Agent (OPA). It's used to write policy that is easy to read and easy to write. Fundamentally, Rego inspects and transforms data in structured documents, allowing OPA to make policy decisions.

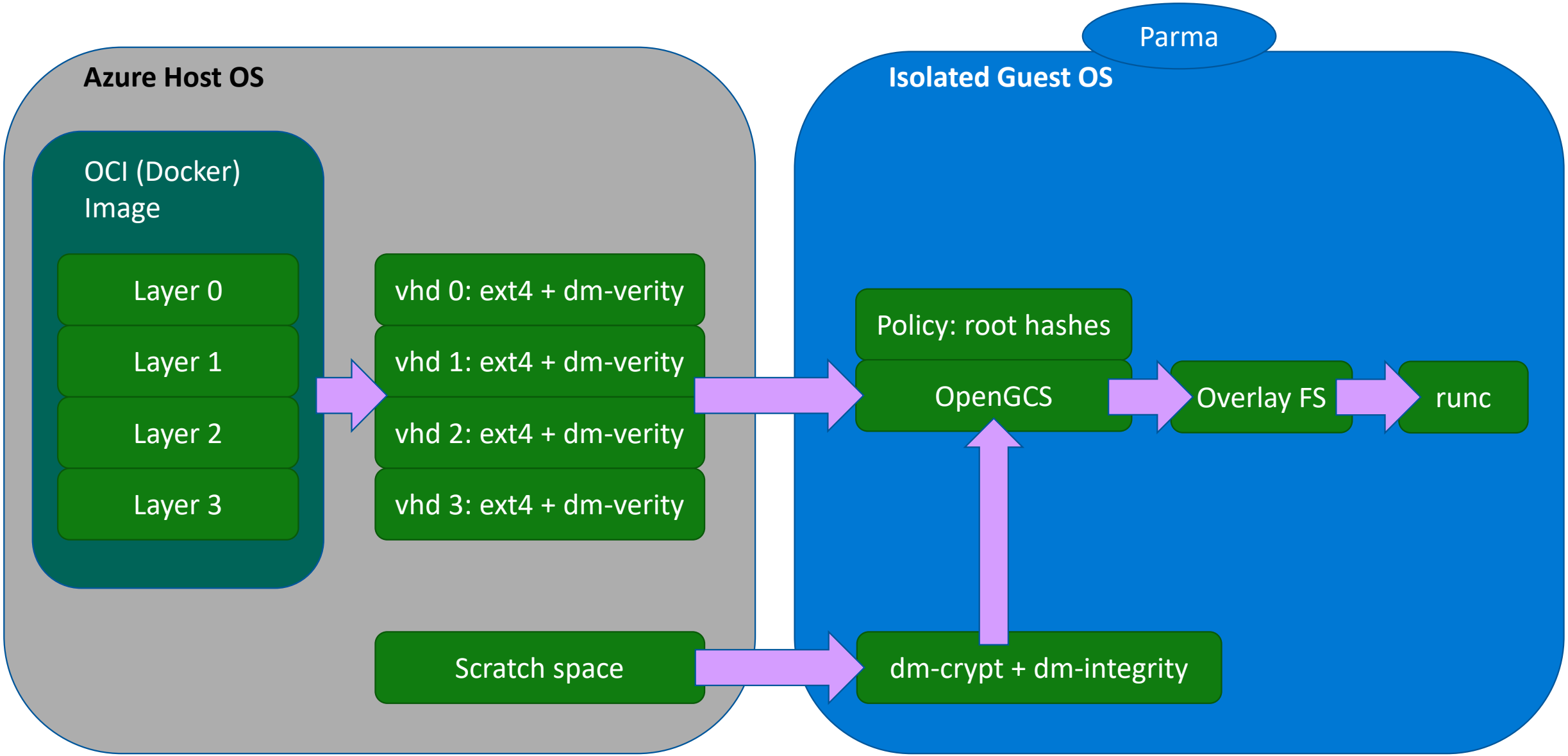
Rego was originally inspired by Datalog to support structured document models like JSON. , a common query language with a decades-long history, but extends its capabilities

Confidential containers on ACI Example

Microsoft Azure



Integrity Protection



Attestation schemes supported in ACI

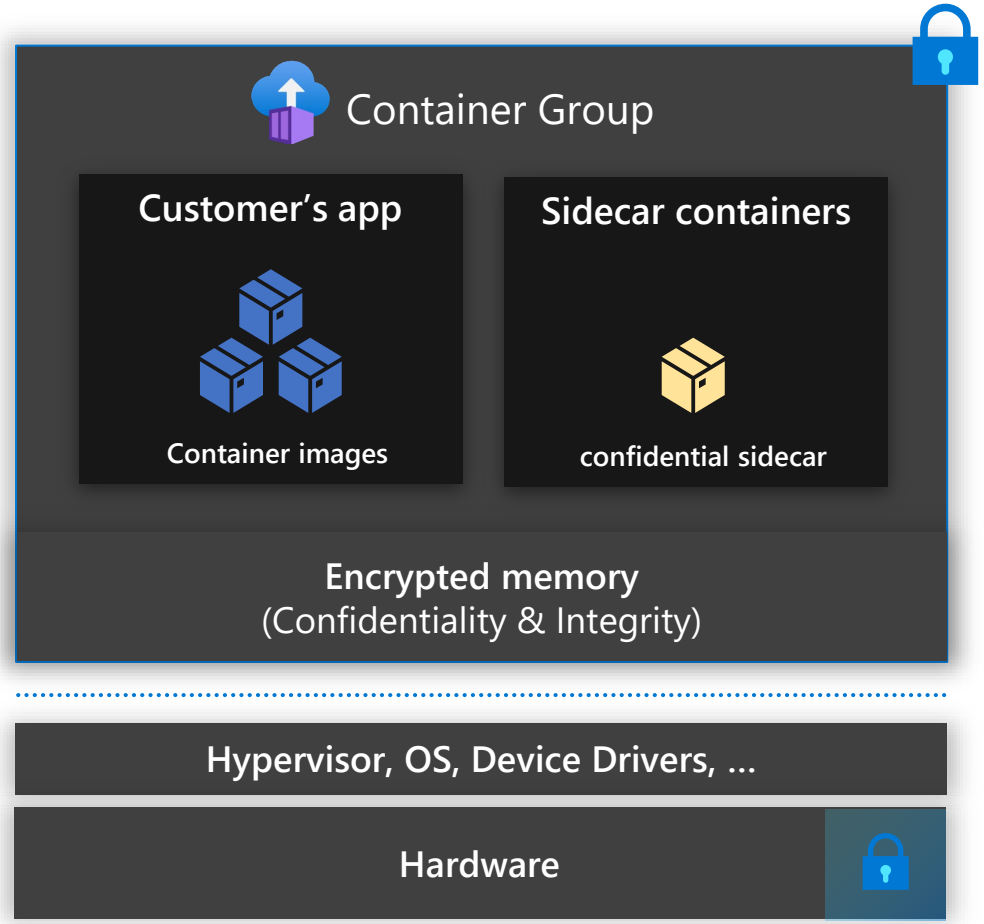
Platform Guest Attestation

- Remote attestation to help validate the hardware supporting the AMD SEV-SNP ACI instance
- Guest applications can tap into the Microsoft Azure Attestation service to attest the TEE's validity

Advanced Guest Workload Attestation

- Reflects customers workload in the attestation report
- TCB measures the launch policy which is then hardware report
- Code integrity is maintained through DM Verity hash and launch enforcements
- Additional claims around launch parameters etc. through CCE policy

Confidential Sidecar container



Integrated platform sidecars to make it **easy to perform remote attestation** measuring all hardware and software components

- **Secure key release (SKR) sidecar**
- **Encrypted filesystem sidecar**

https://github.com/microsoft/confidential-sidecar-containers/

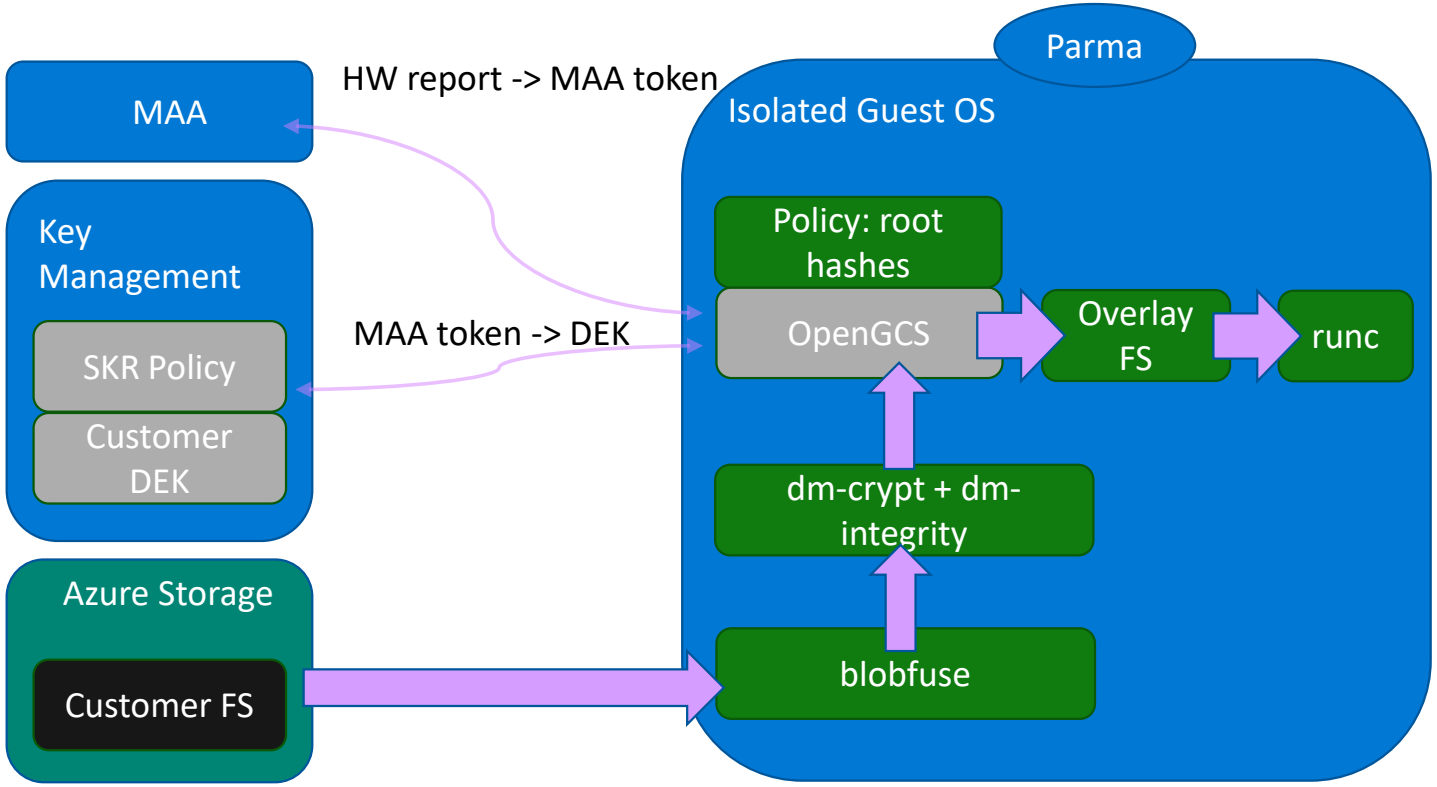
Secure Key Release (SKR) sidecar

REST API - https://github.com/microsoft/confidential-sidecar-containers/blob/main/cmd/skr/README.md

POST method	Description	JSON expected
attest/raw	The POST method **attest/raw** returns the hardware attestation report in hexstring format. It expects a JSON of the following format.	`{ "runtime_data": "<Base64-encoded blob; the hash digest of the blob will be presented as report data in the raw attestation }`
attest/maa	The POST method **attest/maa** interacts with Microsoft Azure Attestation service to receive a JSON Web Token that represents the attestation report as a collection of claims. It expects a JSON of the following format.	`{ "maa_endpoint": "<maa endpoint>", "runtime_data": "<Base64-encoded blob whose hash digest will be presented as runtime data in maa token>" }`
key/release	The POST method **key/release** interacts with AKV MHSM service to release a key previously imported to the managed HSM. It expects a JSON of the following format.	`{ "maa_endpoint": "<maa endpoint>", "akv_endpoint": "<akv endpoint>", "kid": "<key identifier>", "access_token": "optional aad token if the command will run in a resource without proper managed identity assigned" }`

Encrypted filesystem sidecar

- OCI images are integrity protected but not encrypted.
- Customers provide encrypted data (which may include code/binaries) by using tooling to:
  - Build an encrypted and integrity protected VHD containing their data.
  - Store the VHD in Azure Blob Storage.
  - Provision the customer DEK to an Azure key management system (MHSM, etc.) along with a tooling-generated secure key release (SKR) policy that allows only their VM Confidential Container to receive the DEK.



Hands-on lab
ACI

Hands-on lab will cover - confidential_containers_cce.md

- Creating an ARM template for deploying ACI with a confidential computing environment.
- Introducing the CCE policy with defined rules enforced in the utility VM.
- Installing the required Azure CLI extension: az confcom.
- Generating the CCE policy: az confcom acipolicygen -a ./template.json --print-policy.
- Adding the generated CCE policy to the ARM template under the ccePolicy property.
- Logging into Azure and accessing "Deploy a custom template".
- Observing the attestation report as proof of confidential computing deployment.

Hands-on lab will cover -

confidential_containers_attestation_deepdive.md

- Deploying a confidential container on ACI
- Connecting to the confidential container
- Using the sidecar-containers to retrieve a report attestation
- Using the virtee/snpguest to validate the signature of the report

Hands-on lab will cover – confidential_containers_cce_deepdive.ipynb

- Deploying the container group using the ARM template.
- Checking for successful deployment and obtaining the attestation token:
  - Verify the successful deployment.
  - Compare the security policy hash in the attestation token.
- Stopping the container.
- Deploying a different image than what is specified in the security policy:
  - Deploy a container image that calculates the product of two numbers.
  - The deployment will be denied by the security policy.
- Restarting the container:
  - The deployment will fail due to the image not matching the security policy.
- Deploying an updated image with an updated security policy:
  - Generate a new security policy and update the ARM template.
- Checking deployment success and obtaining the attestation token:
  - Verify the deployment success.
  - Compare the security policy hash in the attestation token.

Confidential Containers on AKS

Kata Confidential containers on AKS

Confidential Containers on AKS run in dedicated "*child VMs*" per pod.

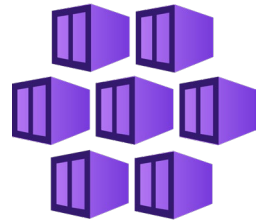
1. Enhanced Application Isolation: Running Kubernetes pods at this isolation level using nested virtualization provides app isolation from the parent VM and the tenant OS admin.

2. Flexible Workload Support: It accommodates the need of natively running any Linux container, promoting a diverse workload ecosystem.

3. Kata Confidential Containers (CoCo): This open-source sandboxed CNCF project provides an ideal set of building blocks and security constructs for broad Kubernetes workload support.

4. Augmented Data Protection:

- Protects data from potential threats from your Azure tenant admin or Kubernetes admin.
- Safeguards data from the cloud provider/operator, ensuring enhanced data security.



Kata Confidential containers on AKS

SIGN-UP for PREVIEW

Isolate containers from k8s admin

Run any containerized app

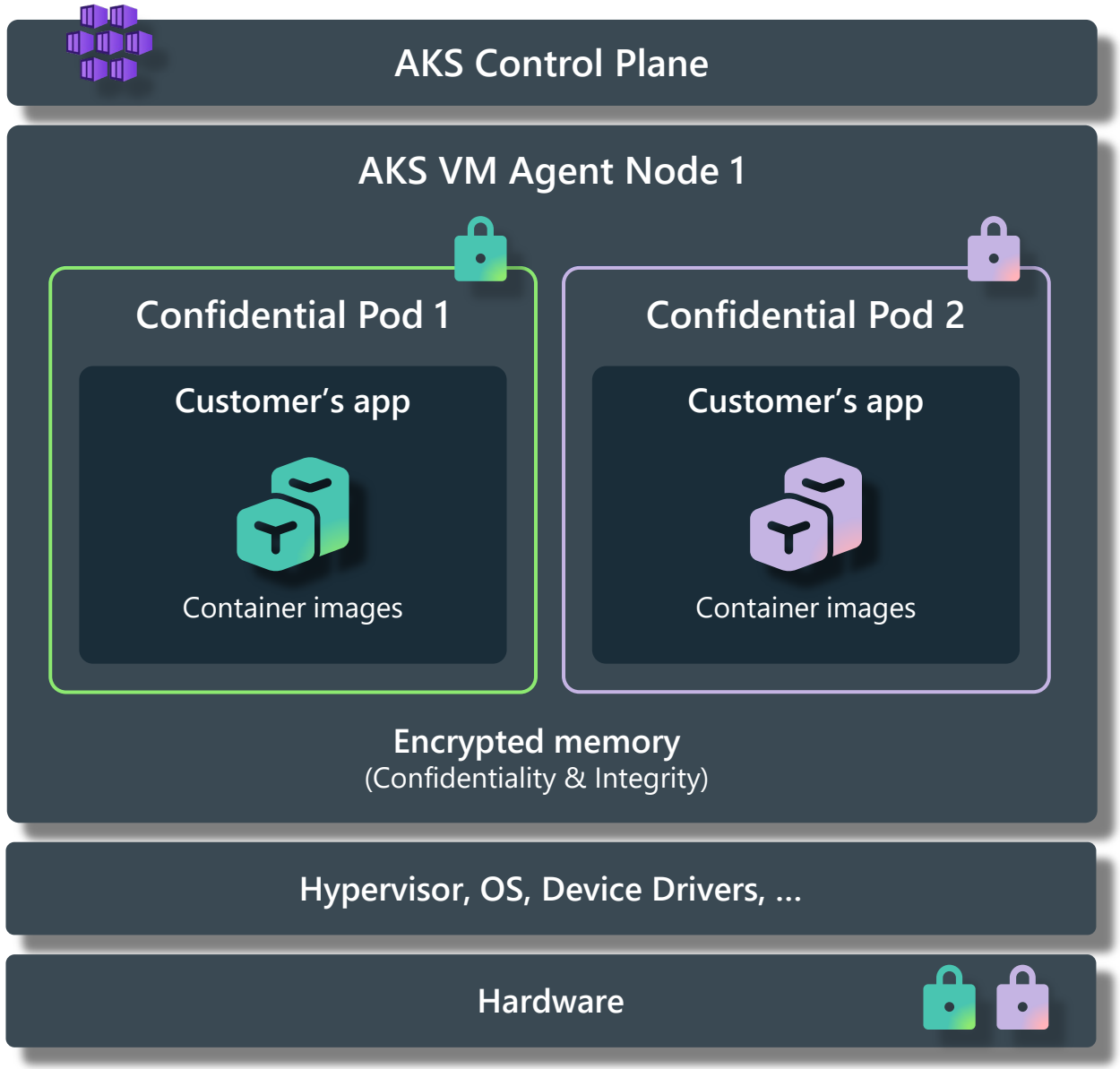
Open source, building on Kata containers

CONFIDENTIAL CONTAINERS

CLOUD NATIVE COMPUTING FOUNDATION

Aka.ms/**cocoakspreview**

Kata Confidential Containers



Scenarios/Use cases

1. **Big Data Analytics:** Employing privacy-preserving analytics in the banking sector, such as Apache Spark jobs for fraud pattern recognition, while ensuring data confidentiality.

2. **CI/CD DevOps Practices:** Running self-hosted GitHub runners for secure code signing as part of Continuous Integration and Deployment (CI/CD) processes.

3. **Machine Learning (ML):** Performing ML inferencing and training using an encrypted dataset from trusted sources. Data is only decrypted within a confidential container environment for privacy-preserving ML inference.

4. **Big Data Clean Rooms:** Building secure environments for ID matching as part of multi-party computation in industries like retail and digital advertising.

5. **Zero Trust Landing Zones:** Creating confidential computing zones that adhere to privacy regulations for application migrations to the cloud.

Product design principles

1. **Transparency:**
   - All components of the Trusted Computing Base (TCB) are to be open sourced

2. **Auditability:**
   - Verify that version of the Kata CoCo environment including Linux Guest OS is current
   - Microsoft-signed guest OS and container runtime
   - Secure hash algorithm (SHA) for audibility and control

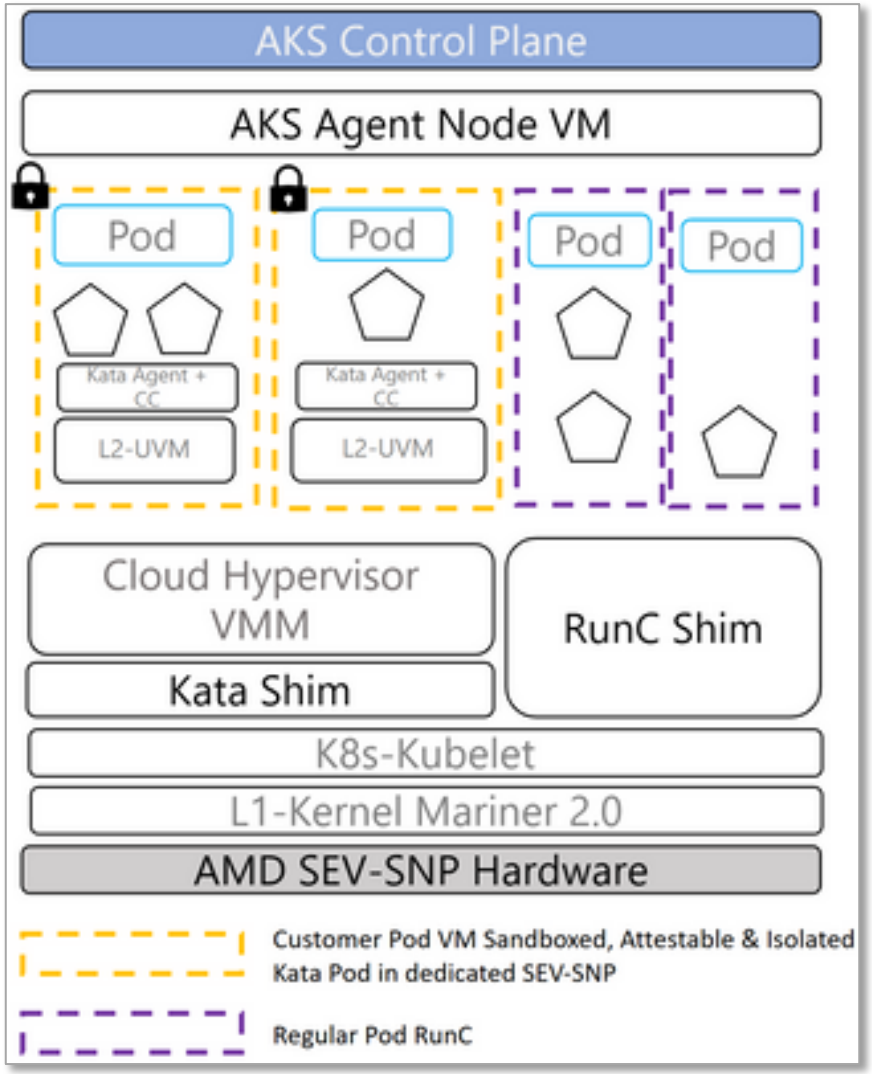
3. **Reproducibility:**
   - OSS first approach with well-documented build instructions

4. **Full attestation:**
   - Fetch hardware AMD SEV-SNP local report that reflects Guest OS image and container runtime

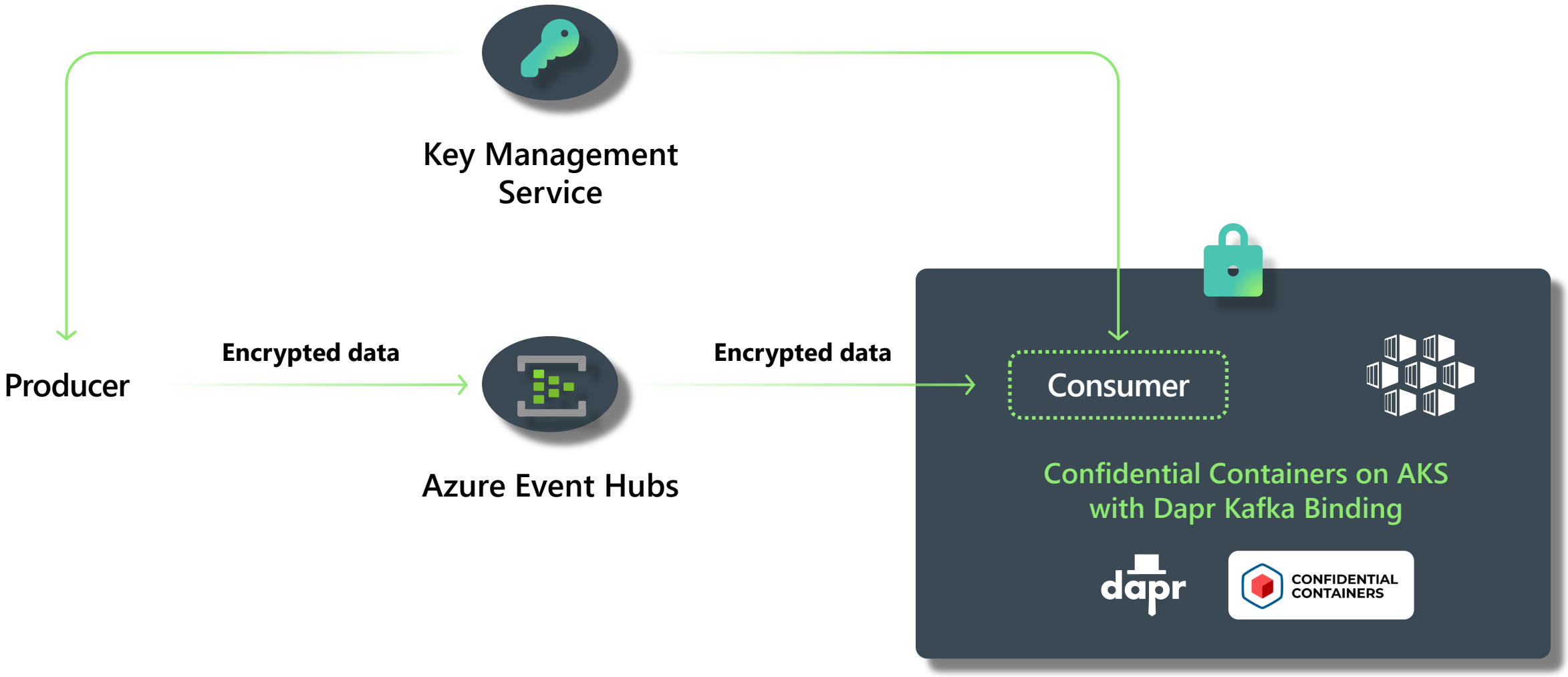
5. **Code integrity:**
   - Runtime enforcement through customer-defined policies
   - Immutable policies and container signing

6. **Isolation from operator:**
   - Least privilege and highest isolation security designs

7. **Ease of use:**
   - Support for unmodified Linux containers



End to end encryption with Confidential Containers



Demo
End to end encryption
with Kata Confidential Container

https://youtu.be/sgIBC3yWa-M?t=2164

Confidential Computing use cases

Government

- Digital identity
- Critical infrastructure
- Anti-corruption
- Cyber crime prevention
- Judicial proceedings and case management
- Deployed and disconnected operations
- Safeguarding / vulnerable population protection (including child exploitation, human trafficking, etc.)

Financial Services

- Anti-money laundering
- Digital currencies
- Secure Payment Processing including Credit Card and Bank Transactions
- Fraud prevention
- Credit risk assessment and qualification from combined bank records
- Capital Markets e.g.: Securing Quantitative Hedge Funds code and models
- Proprietary analytics / algorithms

Healthcare

- Disease diagnostic
- Insurance fraud prevention
- Drug development
- Contact tracing
- Records and evidence management
- Insurance fraud, waste, and abuse prevention

Confidential computing is
state-of-the-art,
and Azure is ready!

Microsoft

Thank you!