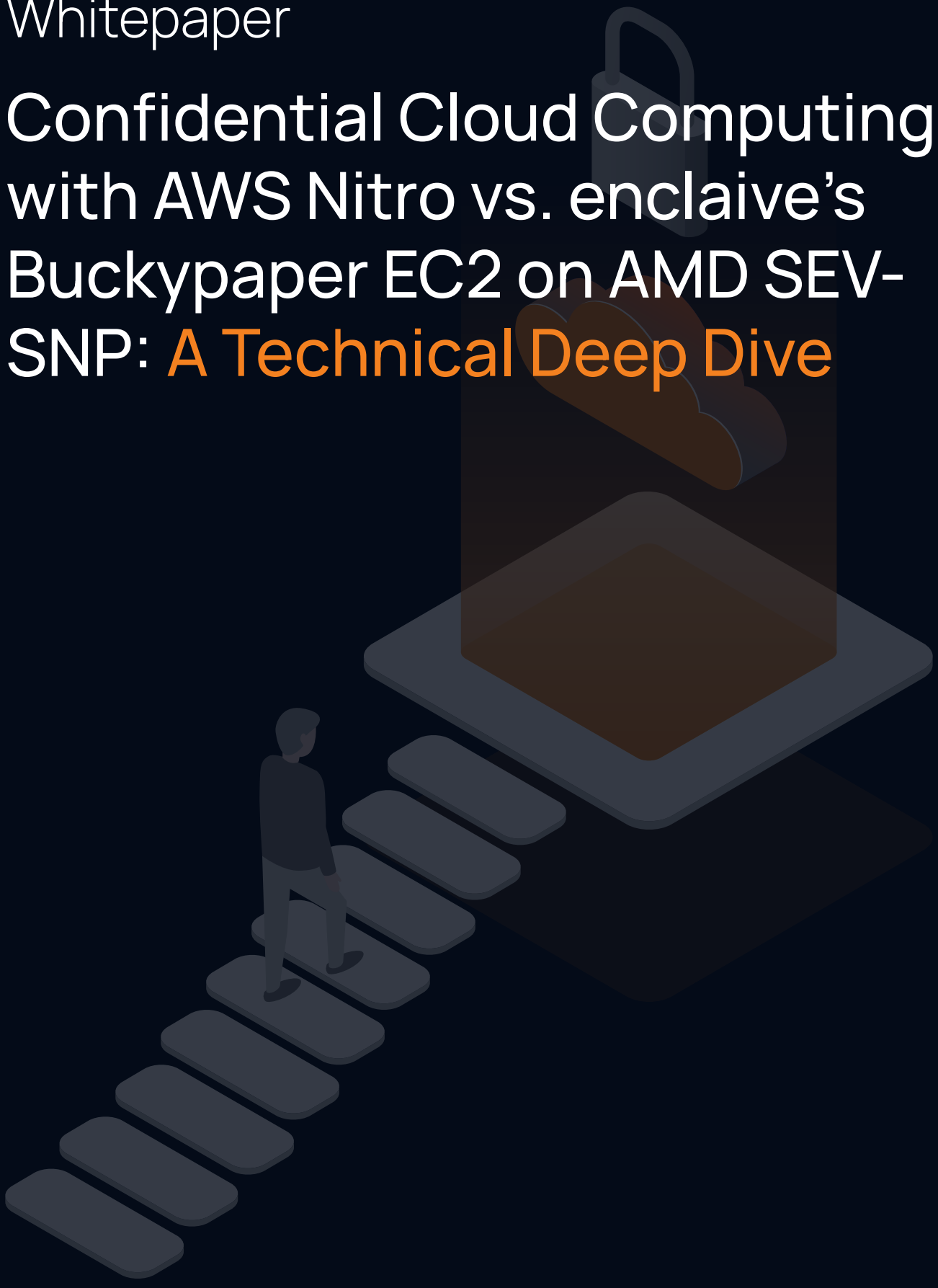
enclaive

Whitepaper

Confidential Cloud Computing with AWS Nitro vs. enclaive's Buckypaper EC2 on AMD SEV-SNP: A Technical Deep Dive



Extended Summary

As organizations increasingly shift sensitive workloads to the cloud, the demand for **confidential computing**—the protection of data in use—has grown sharply. This whitepaper provides a comprehensive technical comparison between two distinct confidential computing models available on Amazon Web Services: the **EC2 instances with Nitro Enclaves** and **with AMD SEV-SNP** enabled. We contrast the confidential computing offerings with **enclaive's EC2 instances with AMD SEV-SNP** enabled (called **"Buckypaper" VMs**).

We begin by contextualizing the urgent need for enhanced data privacy in light of tightening regulatory landscapes and the limitations of traditional cloud security models. These models, while effective in transit and at rest, fall short when it comes to runtime data protection, leaving gaps that can be exploited in multi-tenant environments.

The paper then introduces **Confidential Execution Environments (CEEs)**, tracing their origins from **Trusted Execution Environments (TEEs)** and detailing key architectural principles, including **remote attestation**, **secure boot**, and **hardware-based memory encryption**.

The core of this analysis contrasts the **two mutually exclusive confidential computing models** offered by AWS along enclaive's Buckypaper VM on AWS.

In summary, the findings are:

- **AWS Nitro Enclaves**, which enable enclave-style secure execution within EC2 instances, with strict isolation via vsock-based communication and a narrow application profile (e.g., cryptographic operations).
- **AWS EC2 instances with AMD SEV-SNP**, which build upon the off-the-shelf confidential computing technology of AMD, however do not fully implement the confidential computing premise of excluding the hyperscaler from controlling the workload.
- **enclaive's EC2 instances with AMD SEV-SNP (called Buckypaper VMs)**, which run fully isolated workloads in standard Linux environments, combining strong confidentiality, integrity and zero-trust, and thus completely implement the confidential computing premise to exclude the hyperscaler from the inside of the EC2 instance.

Finally, the paper provides a detailed **side-by-side technical comparison** of these models, highlighting trade-offs in security guarantees, performance, developer experience, networking, and attestation capabilities. It concludes with guidance on selecting the right model based on workload requirements, trust boundaries, and ecosystem fit.

Acknowledgement

This whitepaper borrows much information from the AWS web sites and documentation. Images relating to Nitro, Nitro enclaves and Nitro with AMD SEV-SNP are courtesy of AWS.

Table of Contents

1 Introduction

The digital economy's reliance on global data flows has amplified the complexities of data governance, leading to the proliferation of diverse and stringent data privacy regulations worldwide. Frameworks like the EU-U.S. Data Privacy Framework (DPF) aim to establish legal certainty for transatlantic data transfers, yet they inherently face scrutiny regarding the adequacy of protections, particularly concerning access by foreign governments and intelligence agencies. Traditional cloud security models, while strong in protecting data at rest and in transit, offer limited guarantees for data when it is actively being processed in memory (data in use), leaving a significant attack surface.

This vulnerability undermines the spirit of data privacy regulations, creating compliance gaps and fostering a climate of uncertainty for organizations. Confidential computing emerges as a transformative paradigm that addresses this fundamental gap by enabling data processing within hardware-isolated, verifiable environments. This paper will delve into the technical underpinnings of hardware-assisted confidential computing, specifically leveraging the AWS Nitro Enclaves architecture, to demonstrate its potential in enhancing data security and facilitating more robust compliance with stringent data privacy frameworks.

1.1 The Data Privacy Landscape and Its Challenges

International data transfer mechanisms, such as adequacy decisions (e.g., GDPR Article 45) and specific contractual clauses (e.g., Standard Contractual Clauses), depend on an assessment of the recipient country's data protection regime. The EU-U.S. DPF, an adequacy decision by the European Commission, seeks to provide a streamlined, legally sound mechanism for data transfers to certified U.S. organizations. Its effectiveness, however, hinges on the assurance that U.S. law provides "essential equivalence" to EU law regarding data protection principles, including government access to data.

Challenges to such frameworks often stem from:

- **Government Surveillance:** Concerns over broad governmental access to data, particularly under U.S. Section 702 of the Foreign Intelligence Surveillance Act (FISA), pose an ongoing threat to the perceived adequacy of U.S. data protection.
- **Third-Party Access:** The risk of unauthorized access by cloud service provider (CSP) administrators, malicious insiders, or sophisticated external attackers to data while it is in volatile memory.
- **Lack of Verifiability:** Difficulty for data exporters to cryptographically verify the runtime environment where their data is processed in the cloud.

These challenges highlight the need for technologies that offer stronger, verifiable guarantees of data confidentiality and integrity, irrespective of the underlying infrastructure's administrative access or the legal jurisdiction.

1.2 Traditional Cloud Security Limitations

Conventional cloud security architectures predominantly focus on protecting data at rest (e.g., encryption of storage volumes with KMS) and data in transit (e.g., TLS/SSL for network communication). While essential, these measures leave data exposed during its most vulnerable phase: **data in use**.

In a typical virtualized cloud environment, the hypervisor (or host operating system for bare metal) has privileged access to the memory of guest virtual machines (VMs). This architectural reality means that:

- **Hypervisor/Host Compromise:** A compromise of the hypervisor or the host operating system could theoretically expose all data being processed by guest VMs.
- **CSP Administrator Access:** While CSPs employ strict access controls, the fundamental design typically grants CSP administrators technical access to the underlying physical servers, which could theoretically lead to access to guest VM memory, even if policy prohibits it.
- **Side-Channel Attacks**: In multi-tenant environments, even without direct memory access, sophisticated side-channel attacks could potentially infer information about data being processed by co-located VMs.

These limitations underscore the demand for an additional layer of security that protects data within the CPU's execution environment itself, independent of the operating system or hypervisor.

2 Introducing Confidential Execution Environments

2.1 A Primer

Confidential computing is an industry initiative focused on protecting data and code in 3 dimensions, namely in use, at rest, and in transit, by performing computation in a hardware-graded Confidential Execution Environment (CEE). A CEE is a secure, isolated processing environment that guarantees:

- **Confidentiality:** Data loaded into the CEE's memory, network and storage is encrypted and inaccessible to any external entity, including the host OS, hypervisor, or other VMs.
- **Integrity:** The code executing within the CEE cannot be tampered with. Any unauthorized modification to the CEE's code or data results during execution in a failure.
- **Attestation:** The CEE provides cryptographic proof (an "attestation report") to remote parties that their code is running unmodified within a genuine CEE, providing verifiable trust in the execution environment.

This approach creates a "hardware-enforced fortress" around sensitive code and data, shielding it from external software, including privileged software like hypervisors.

2.2 The Evolution: Relation to Trusted Execution Environments

The concept of CEEs evolved from **Trusted Execution Environments (TEEs)**. Earlier TEE implementations primarily focused on guaranteeing the **integrity** of code and data execution. The fundamental promise of a TEE was that code would execute as intended, free from unauthorized modification, and that data processed by this trusted code would remain untampered.

Key characteristics of these foundational TEEs often include:

- **Code Integrity:** Assurance that the loaded code is genuine and has not been altered.
- **Data Integrity (within the execution flow):** Protection against tampering with data while it's being manipulated by the trusted code.
- **Isolation:** A hardware-enforced separation from the host OS, hypervisor, and other software components, preventing external observation or interference with the TEE's internal execution state.
- **Hardware Root of Trust:** A cryptographic anchor in hardware that validates the authenticity and integrity of the TEE itself.

However, a crucial distinction is that many initial TEEs did not inherently encrypt the data in the TEE's memory when it resided outside the immediate CPU caches and registers. A second crucial distinction is that—in virtualized environments—the host operator was assumed to be trusted. While the TEE itself was isolated, the memory pages holding its data might have been visible (though integrity-protected) to a sufficiently privileged and malicious observer on the host. This meant that while execution integrity was strong, full data confidentiality against a compromised lower layer was not always guaranteed purely by the TEE's isolation.

CEEs address this limitation by adding mandatory hardware-level memory encryption and process integrity control, ensuring that data is encrypted throughout its lifecycle within the confidential boundary, even when in physical memory, on the network or storage device.

2.3 Common Principles: Remote Attestation and Secure/ Measured/Trusted Boot

Both TEEs and CEEs fundamentally rely on **attestation** to enable verifiable trust in their execution environments. **Remote attestation** is the process by which a TEE/CEE can cryptographically prove its identity, configuration, and the integrity of the code and data loaded within it to a remote relying party. This proof, typically a cryptographically signed report, provides objective, verifiable assurance that the environment is genuine and untampered.

This attestation process is intricately linked with Secure Boot, Measured Boot, and Trusted Boot techniques:

- **Secure Boot:** During the system's startup, Secure Boot ensures that only firmware and software components signed by trusted authorities are allowed to execute. The system firmware verifies the digital signatures of each bootloader and OS kernel component before loading them. This prevents the execution of tampered or unauthorized code at boot time and establishes a root of trust from the very first instruction executed.
- **Measured Boot:** Complementing Secure Boot, Measured Boot involves hashing each critical software component (e.g., firmware, bootloader, kernel, hypervisor) before it is executed. These cryptographic measurements are securely recorded in tamper-evident hardware registers—such as Platform Configuration Registers (PCRs) in a TPM or secure elements in modern CPUs (e.g., Intel TXT, AMD PSP, or AWS Nitro). These measurements can later be queried and verified by a remote party during remote attestation to ensure the boot sequence has not been altered.
- **Trusted Boot:** Building upon Measured Boot, Trusted Boot introduces policy enforcement. If the measured values do not match a known-good reference set, the system refuses to proceed with the boot process. This prevents compromised or untrusted code from executing, ensuring the platform does not reach an operational state if its integrity cannot be assured.

Together, Secure Boot, Measured Boot, and Trusted Boot establish a layered trust chain from firmware through to the operating system, enabling robust remote attestation.

In the context of CEEs (and the more advanced TEEs that incorporate memory encryption), these boot processes establish the initial chain of trust. The measurements generated during boot are included in the attestation report, allowing a remote verifier to cryptographically confirm that the entire trusted computing base (TCB)—from the deepest hardware layer up to the CEE's runtime environment—is in an expected and secure state before sensitive data is entrusted for processing. This verifiable chain of trust is crucial for enabling secure, confidential computation in multi-tenant or potentially untrusted infrastructure, such as public cloud environments.

3 The Two, Mutually Exclusive AWS Confidential Computing Technologies

AWS has introduced distinct, mutually exclusive features, framed under the umbrella of confidential computing: AWS Nitro Enclave and AMD-SEV-SNP. Both EC2 features leverage the core capabilities of the AWS Nitro System, a re-architected virtualization platform that offloads host functions to dedicated hardware, to provide highly isolated and secure execution environments.

> ⓘ **Remark**
>
> Nitro is AWS's virtualization technology underlying EC2 instances, including compute, networking and storage. AWS Nitro Enclaves is an **Amazon EC2 feature** that allows you to create isolated execution environments, called enclaves, from Amazon EC2 instances. Nitro Enclaves is processor agnostic and it is supported on most Intel, AMD, and AWS Graviton-based Amazon EC2 instance types built on the AWS Nitro System.

3.1 The Nitro System

3.1.1 Traditional Virtualization: The Hypervisor Paradigm

In the realm of cloud computing and data centers, **traditional virtualization** relies on a software layer called a **hypervisor** to abstract physical hardware resources. This abstraction enables multiple isolated **Virtual Machines (VMs)**, each with its own guest operating system (OS) and applications, to run concurrently on a single physical host server.

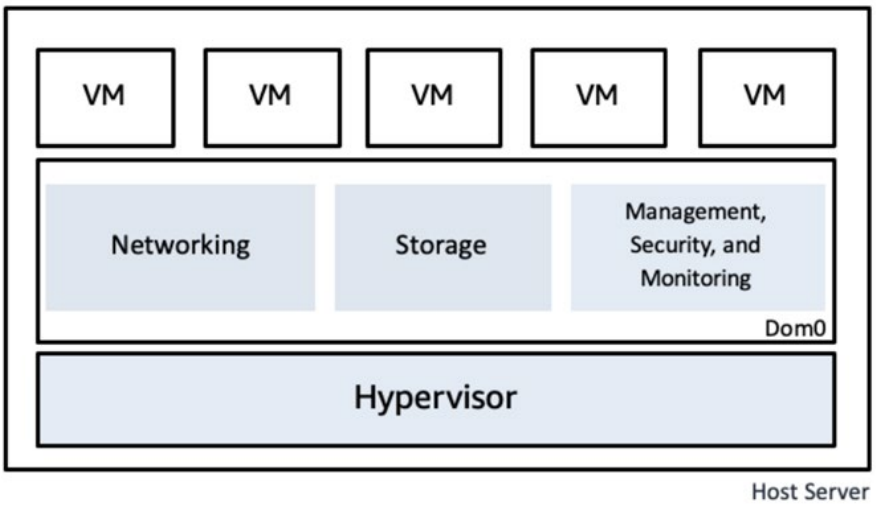


Fig. 1: Classical virtualization architecture

The principle of virtualization, illustrated in Fig. 1, works as follows:

1. **Host and Guest:** A physical server (the "host") runs the hypervisor. On top of the hypervisor, multiple VMs (the "guests") operate.
2. **Resource Abstraction:** The hypervisor creates virtualized versions of the physical hardware components—CPU, memory, storage, and networking interfaces—and presents them to each guest VM. Each VM perceives these virtual resources as if they were dedicated physical hardware.
3. **Resource Management and Scheduling:** The hypervisor is responsible for:
   - **CPU Virtualization:** Multiplexing the physical CPU cores among the vCPUs of the guest VMs, often leveraging hardware-assisted virtualization extensions (e.g., Intel VT-x, AMD-V) for efficient instruction execution.
   - **Memory Management:** Allocating and isolating virtual memory for each VM, mapping it to physical RAM. Techniques like memory overcommitment and page sharing are used for efficiency.
   - **I/O Emulation/Paravirtualization:** Handling I/O requests (disk, network) from guest VMs. This typically involves either full emulation of hardware devices (which can incur significant overhead) or paravirtualization, where guest OSes use specialized drivers (e.g., Virtio drivers) that communi cate directly with the hypervisor for more efficient I/O.
4. **Isolation:** The hypervisor enforces strong isolation boundaries between VMs, ensuring that a problem in one VM does not affect others on the same host.

3.1.2 Motivation for AWS Nitro Virtualization

Despite their revolutionary impact, traditional hypervisor-based architectures presented several challenges, particularly at the scale and security demands of a hyperscale cloud provider like AWS:

1. **Performance Overhead ("Hypervisor Tax"):**
   - **Resource Consumption:** The hypervisor and its associated management OS (e.g., Dom0) consume a significant portion of the host's CPU and memory resources for their own operations (e.g., scheduling, I/O processing, monitoring). This "hypervisor tax" directly reduces the amount of resources available to customer instances.
   - **I/O Bottlenecks:** Even with paravirtualization, I/O operations (networking and storage) are often processed by the hypervisor or the privileged domain. This can lead to increased latency, reduced throughput, and performance variability (jitter) due to contention for shared resources and software-based processing. For demanding workloads like HPC, real-time analytics, or high-IOPS databases, this overhead can be prohibitive..
2. **Security and Attack Surface:**
   - **Large Attack Surface:** Traditional hypervisors and their associated management OSes are complex software stacks, running a general-purpose Linux kernel and numerous drivers. This presents a larger attack surface for potential vulnerabilities and exploits that could compromise the hypervisor itself, potentially affecting all co-located customer VMs.

- **Administrator Access:** The necessity for cloud provider administrators to have privileged access (e.g., SSH, root access) to the hypervisor/host OS created a theoretical vector for unauthorized access to customer data or interference with instances, even if policies were stringent. This was a significant concern for customers with highly sensitive data or strict compliance requirements.
- **Firmware Vulnerabilities:** The host's firmware (BIOS/UEFI) was another potential point of compromise, and traditional systems had limited hardware-level verification.

3. **Limited Innovation and Agility:**
   - **Monolithic Design:** The tightly coupled nature of traditional hypervisors with host hardware and management functions made it challenging to rapidly introduce new hardware features, CPU generations, or instance types. Changes to one part of the system could have ripple effects across the entire hypervisor stack.
   - **Hardware Lock-in:** Integrating new hardware (e.g., faster NICs, NVMe SSDs, specialized accelerators) often required complex driver development and modifications within the hypervisor, slowing down innovation cycles.

3.1.3 Architectural Foundations: The AWS Nitro System

AWS's motivation for developing the Nitro System was to fundamentally reinvent their virtualization stack to overcome these inherent limitations. The goal was to:

- **Eliminate the "Hypervisor Tax":** Deliver virtually all of the host's compute and memory resources directly to customer instances, providing near bare-metal performance.
- **Drastically Enhance Security:** Minimize the attack surface by offloading functionality and eliminating AWS operator access to the host, establishing a hardware root of trust.
- **Accelerate Innovation:** Create a modular, flexible platform that allows rapid integration of new hardware and development of diverse EC2 instance types.

By disaggregating and offloading hypervisor responsibilities to specialized hardware (Nitro Cards and the Nitro Security Chip) and drastically slimming down the software hypervisor (Nitro Hypervisor), AWS built a foundation that delivers superior performance, enhanced security, and unprecedented agility in cloud infrastructure. This allows EC2 instances to operate with performance and isolation characteristics previously only achievable on bare metal, while still benefiting from the elasticity and management capabilities of the cloud. The Nitro System, illustrated in Fig. 2, consists of:

- **Nitro Cards:** Custom-designed hardware cards that offload networking, storage, and management I/O from the main CPU. They expose standard interfaces (e.g., NVMe, ENA) to the guest OS, effectively acting as "software-defined hardware."
- **Nitro Security Chip (aka Trusted Platform Module):** A dedicated hardware chip on the motherboard that establishes a hardware root of trust, ensures secure boot, and protects the host system's firmware.
- **Nitro Hypervisor:** A minimalist, lightweight KVM-based hypervisor that solely manages CPU and memory allocation. It's designed with an extremely small attack surface, having no persistent storage, no SSH access, and no general-purpose operating system.

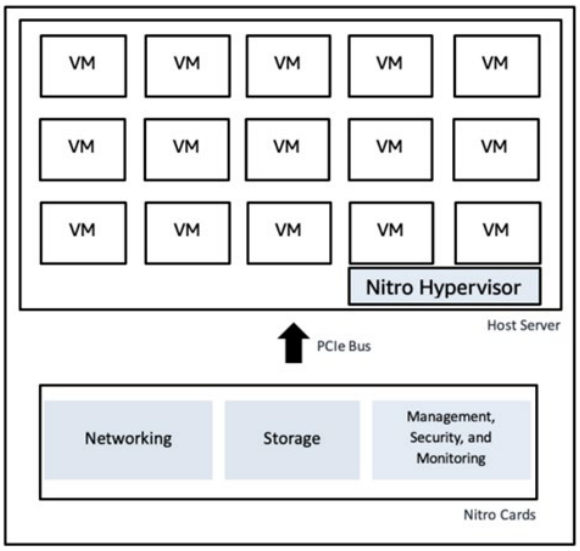


Fig. 2: Nitro virtualization architecture

This architecture ensures that the host CPU's resources are almost entirely dedicated to guest instances, and management functions are isolated on dedicated hardware, significantly reducing the attack surface.

3.2 AWS EC2 Nitro Enclaves

Nitro Enclaves (see Fig. 3) are purpose-built virtual machines launched from a parent EC2 instance. They leverage the same Nitro Hypervisor technology that provides CPU and memory isolation for standard EC2 instances but take isolation to an extreme for sensitive workloads.

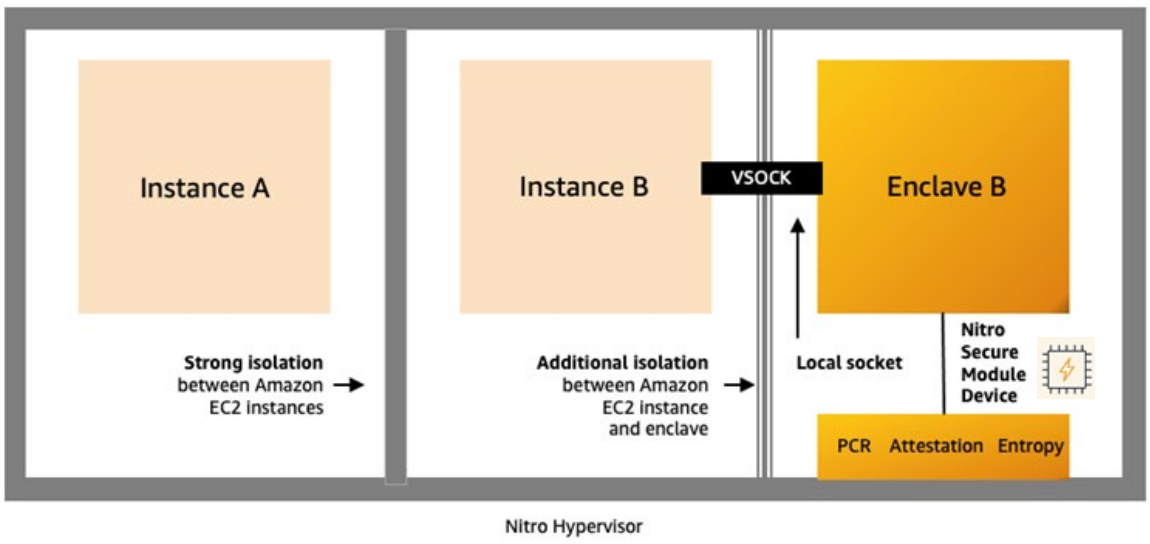


Fig. 3: Nitro enclave architecture

3.2.1 Isolation Mechanisms

Isolation of compute, network and storage works as follows:

- **CPU and Memory Dedication:** An enclave is allocated dedicated CPU cores and memory from the parent EC2 instance. This allocation is hardware-enforced and completely isolated from the parent instance's operating system, other processes, and even the Nitro Hypervisor itself.
- **No Persistent Storage:** Enclaves do not have direct access to persistent storage. All data is processed in ephemeral memory.
- **No Network Interface:** Critically, enclaves do not have network interfaces. They cannot initiate or receive network connections directly. This eliminates a vast array of network-based attack vectors.

3.2.2 Attestation Process

Attestation is the cornerstone of trust. For Nitro Enclaves, it works as follows:

1. **Enclave Image Hashing:** When an enclave is created, the Nitro System calculates a cryptographic hash (SHA-256) of the enclave's entire software image (the kernel, application code, and any libraries). This hash serves as a unique digital fingerprint.
2. **Attestation Document Generation:** The Nitro System generates an "attestation document" that includes:
    - The cryptographic hash of the enclave's code.
    - Metadata about the enclave's configuration (e.g., CPU, memory allocated).
    - A unique hardware-bound nonce.
    - This document is then cryptographically signed by a private key unique to the Nitro Hypervisor on that specific host.
3. **Remote Verification:** A remote party (e.g., a client, a data owner, or a key management service) can receive this attestation document. They can then:
    - Verify the signature using AWS's public key to confirm it originated from a genuine Nitro Hypervisor.
    - Compare the reported cryptographic hash of the enclave's code against a known, trusted hash.
    - Verify the nonce to ensure freshness and prevent replay attacks. If all checks pass, the remote party gains cryptographic assurance that their code is running unmodified within a legitimate, secure Nitro Enclave.

3.2.3 Secure Communication

Since enclaves lack network interfaces, all communication with the outside world must be mediated via the parent EC2 instance. This is achieved through a secure local channel:

- **vsock (Virtio Socket):** Enclaves communicate with the parent instance using a virtual socket interface (vsock). This provides a secure, low-latency communication channel within the same host.

- **Cryptographic End-to-End Encryption:** For sensitive data exchange with external services (e.g., an external KMS for decrypting keys, or a data source), communication typically involves cryptographic challenges and responses facilitated through the parent instance. The sensitive data itself is often encrypted before leaving the enclave and only decrypted inside the enclave, ensuring end-to-end confidentiality. The parent instance acts merely as a secure proxy, unable to inspect the encrypted payload.

The use of vsock as the sole communication channel between an AWS Nitro Enclave and its parent EC2 instance offers distinct advantages that enhance the overall security and operational model of confidential computing workloads:

- **Enforced Local Communication:**
  - **Reduced Attack Surface:** By eliminating direct external network connectivity from the enclave, vsock inherently prevents a vast array of network-based attacks (e.g., denial-of-service, network scanning, remote code execution vulnerabilities in network services) from reaching the highly sensitive code within the enclave.
  - **Isolation Integrity:** The vsock communication is entirely local to the physical host, remaining within the trusted boundary of the Nitro System. This means that data exchanged over vsock does not traverse external networks, reducing the risk of eavesdropping or interception by external adversaries.
- **Trusted Channel for Sensitive Operations:**
  - **Secure Proxies:** The parent EC2 instance can act as a secure proxy, mediating communication between the enclave and external AWS services (e.g., AWS KMS for key management, S3 for data ingress/egress, external APIs). The parent instance handles the network connectivity, but the sensitive data itself remains encrypted and unintelligible to the parent until it's securely processed within the enclave.
  - **Attestation-Driven Trust:** Communication over vsock can be coupled with cryptographic attestation. For instance, when an enclave requests a cryptographic operation from AWS KMS via its parent, the enclave's signed attestation document is automatically included in the request. This allows KMS to verify the enclave's identity and ensure only authorized code is running before performing the sensitive operation, creating a robust, attestation-driven trust chain.
- **Performance Efficiency for Local Interprocess Communication:**
  - **Low Latency and High Throughput:** vsock is designed for efficient inter-VM communication within the same host. Compared to traditional network stacks that involve multiple layers of processing and potential network hops, vsock offers significantly lower latency and higher throughput, making it suitable for frequent, high-volume data exchange between the parent and the enclave.
  - **Optimized Resource Utilization:** Since vsock operates at a virtualized socket layer without full network stack overheads, it consumes fewer system resources (CPU, memory) for communication, allowing more of the parent and enclave resources to be dedicated to the actual workload.

- **Simplified Security Posture:**
  - **Clear Trust Boundary:** The vsock mechanism clearly defines the communication trust boundary. Developers and security teams know precisely how data enters and exits the enclave, simplifying threat modeling and security audits.
  - **Controlled Data Flow:** All data entering or leaving the enclave must flow through the vsock inter face to the parent. This provides a single, well-defined point for controlling and auditing data flows, enforcing strict data governance policies.

The exclusive use of vsock for communication with the parent VM in AWS Nitro Enclaves is a deliberate architectural choice that reinforces the enclave's stringent isolation at the network layer. It ensures that while the enclave can interact with necessary external services, it does so through a secure, high-performance, and auditable local channel, significantly enhancing the confidentiality and integrity of the sensitive workloads processed within.

3.3 AWS EC2 Instances With AMD SEV-SNP Enabled

AWS offers EC2 instances that leverage AMD's Secure Encrypted Virtualization - Secure Nested Paging (SEV-SNP) technology. This capability provides a robust hardware-level Confidential Execution Environment (CEE) that enhances the security posture of an entire EC2 virtual machine, primarily against the cloud provider's hypervisor and other co-resident VMs.

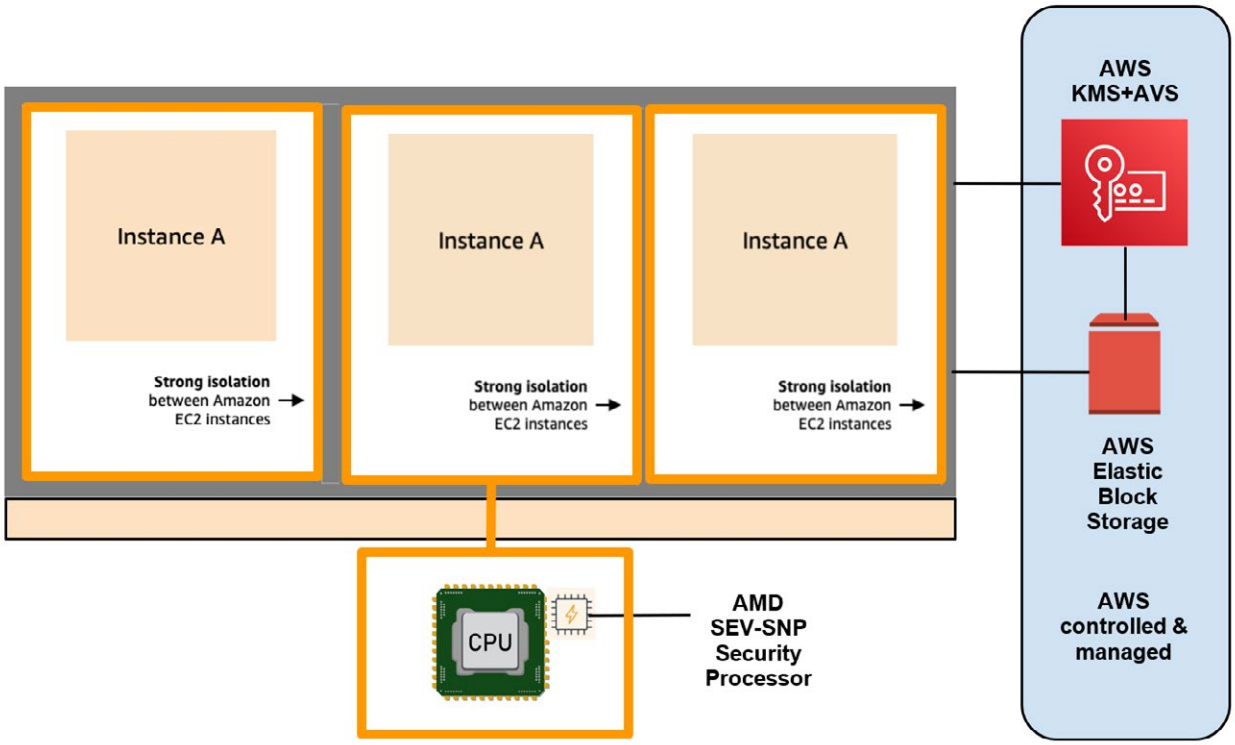


Fig. 4 EC2 architecture with AMD SEV-SNP enabled

3.3.1 Isolation Mechanism

AMD SEV-SNP operates by introducing hardware-enforced memory encryption and integrity protection for the guest VM. This is managed by the **AMD Platform Security Processor (PSP)**, a dedicated security co-processor integrated into the AMD EPYC CPU.

- **Memory Encryption (Confidentiality):**
  - Each SEV-SNP enabled VM is assigned a unique, hardware-managed AES encryption key. This key is generated by PSP and is never exposed to the hypervisor, guest OS, or any other software component.
  - As data is written from the CPU to DRAM, it is automatically encrypted using this VM-specific key. Conversely, data is decrypted as it is read back into the CPU.
  - This ensures that even if the hypervisor or another malicious entity were to gain access to the physical DRAM, they would only observe encrypted ciphertext, preserving the confidentiality of the VM's memory contents.
  - AMD SEV-SNP encrypts both memory and CPU state, similar to its predecessor SEV-ES, but adds hardware-enforced memory integrity to prevent tampering, remapping, or replay attacks. It eliminates the need to trust the hypervisor by enforcing guest page table ownership in hardware and validating the integrity of guest memory. SEV-SNP also introduces cryptographically signed attestation reports with measured boot support and allows the use of guest owner-supplied certificates for third-party attestation.
  - Newer CPU versions allow for secure software-based interrupt handling and PCI bus encryption. The latter is of importance to establish secure communication with GPU hardware.
- **Memory Integrity Protection (Integrity):**
  - Beyond confidentiality, SEV-SNP introduces strong memory integrity guarantees through mechanisms like the **Reverse Map Table (RMP)**, managed by the PSP.
  - The RMP tracks the ownership and state of each memory page within the VM's address space. It ensures that only the rightful owner (the VM itself) can write to its private memory pages.
  - This protection is designed to prevent various hypervisor-based attacks, including:
    - **Data Replay:** Preventing an attacker from substituting old copies of memory pages.
    - **Memory Re-mapping/Aliasing:** Preventing the hypervisor from maliciously re-mapping or aliasing memory pages to other locations or VMs.
    - **Data Corruption:** Detecting unauthorized modifications to memory contents.
  - If any unauthorized write or manipulation of a VM's private memory is detected by the hardware, an exception is triggered, preventing the VM from operating on compromised data.

The overall effect is that the entire VM, including its guest operating system, applications, and all data in memory, is protected from the untrusted hypervisor and other VMs on the same physical host.

3.3.2. Attestation Process

Attestation in AMD SEV-SNP provides a cryptographic assurance of the VM's integrity and authenticity to a remote relying party. This process verifies that the VM is running on a genuine AMD platform with SEV-SNP enabled and that its initial state (the "launch measurement") is as expected.

The key steps involve:

1. **Launch Measurement (Initial Attestation):**
   - During the VM's launch process, the PSP calculates a cryptographic hash (the "launch measurement") of the initial content and layout of the guest VM's memory and initial vCPU state.
   - This measurement captures the integrity of the VM's initial boot code and configuration.
   - The launch measurement is an integral part of the attestation report.
2. **Attestation Report Generation (Guest-Initiated):**
   - An application or service within the running SEV-SNP VM can request an attestation report from the PSP. This request is typically made via a secure, hardware-protected interface. An agent (e.g. enclavelet) is necessary to interact with the PSP API and an attestation verification service (e.g. enclave's Nitride).
   - The PSP generates a report that includes:
     - The **launch measurement** (from boot).
     - Additional platform details (e.g., CPU microcode version, SEV-SNP firmware version) comprising the Trusted Computing Base.
     - Optionally, an arbitrary data block provided by the guest VM (e.g., a hash of a public key, OVMF variables).
   - The entire attestation report is then cryptographically signed by the **Versioned Loaded Endorsement Key (VLEK)**. The VLEK is a unique, hardware-bound key derived from the PSP and its firmware version.
3. **Remote Verification:**
   - A remote relying party (aka the attestation verification service) operated by, e.g., a data owner, an application server, or a key management service receives the signed attestation report.
   - They perform several critical checks:
     - **Signature Verification:** The report's signature is verified using AMD's public key infrastructure (Root of Trust and the specific VCEK certificates obtained from AMD's Key Distribution Service—KDS). This confirms the report's authenticity and that it originated from a genuine AMD processor with a verified TCB version.
     - **Launch Measurement Verification:** The relying party compares the launch measurement in the report against a known, trusted hash of the expected VM image. This ensures that the VM started with the correct and untampered software.

- **Policy Enforcement:** The relying party can also verify other policy-related fields in the report (e.g., whether the VM's policy allowed debugging, whether it required specific security features) to ensure it aligns with their trust policy.

By successfully completing this attestation process, the relying party gains strong cryptographic assurance that their confidential workload is executing on an authentic AMD platform with SEV-SNP enabled, with its memory encrypted and integrity-protected against the underlying hypervisor. This makes AMD SEV-SNP a powerful solution for protecting sensitive workloads in multi-tenant cloud environments.

4 enclaive Buckypaper VMs on AMD SEV-SNP Enabled EC2 Instances

While cloud providers offer native confidential computing capabilities, third-party platforms such as enclaive build solutions that enhance the developer and operational experience, often adding further layers of security and compliance features. enclaive's Buckypaper VMs, depicted in Fig. 5, are a prominent example, leveraging AMD SEV-SNP enabled EC2 instances to deliver a highly confidential, integrity-protected, and verifiable virtual machine environment with the premise AWS has no access to data and code in the instance. In other words, Buckypaper implements the confidential computing premise!

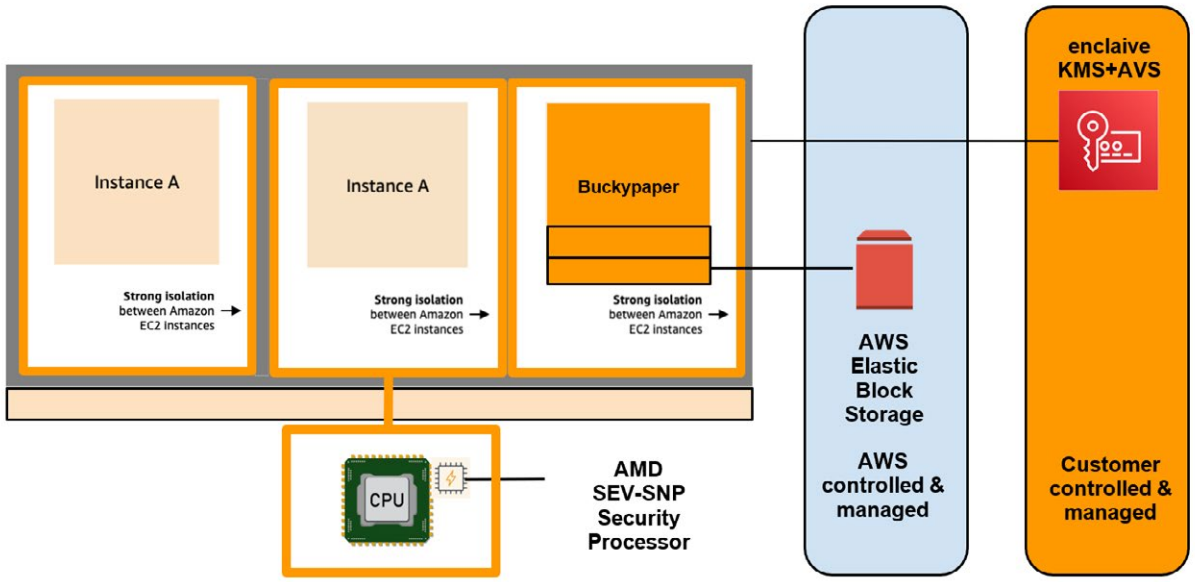


Fig. 5 EC2 Buckypaper architecture with AMD SEV-SNP enabled

4.1 Architectural Foundations

enclaive Buckypaper VMs are designed to run on public cloud infrastructure that provides underlying hardware support for AMD SEV-SNP, such as specific Amazon EC2 instance types (e.g., M6a, C6a, R6a families). The core of their architecture is the integration with the **AMD Secure Processor (PSP)**, which is the hardware root of trust on AMD EPYC CPUs.

Beyond the hardware-level protections offered by AMD SEV-SNP, Buckypaper VMs integrate additional software and service components to form a comprehensive confidential computing solution:

- **Guest OS with Disk Encryption:** The guest operating system running within a Buckypaper VM is configured to utilize its own disk encryption engine. This ensures that data stored at rest on virtual disks (e.g., EBS volumes in AWS) remains encrypted, complementing the memory encryption provided by AMD SEV-SNP.
- **Independent Attestation Verification and Key Management Services:** The design promotes the use of independent, external services for attestation verification (referred to as an Attestation Verification Service, or AVS) and key management (referred to as a Secret Provisioning Service, or SPS). This architectural choice enhances the overall trust model by allowing critical security functions to reside outside the immediate cloud provider's control plane.

- **Guest OS with concealed Endpoint keys, secrets, and certs:** Thanks to the SPS, keys, passwords or certificates are securely provisioned into the EC2 instance from a AWS-independent key management service, typically running in the customers's trusted execution environment (e.g. on-premise).

The bottom line is that when keeping the AVS and KMS away from the AWS perimeter, then encrypting the compute, storage and network data flow of the EC2 makes sure the EC2 is fully confidential and integrity protected against the hypervisor. As mentioned above this premise is what confidential computing stands and has been designed for.

4.2 Security Model

The fundamental security principle guiding enclaive Buckypaper and the key differentiator to AWS EC2 instances with AMD SEV-SNP support is **"do not trust the hypervisor."** This explicit threat model extends beyond typical cloud security by assuming that the underlying cloud infrastructure's hypervisor, host operating system, and even cloud administrators are potentially untrusted or compromised.

By adhering to this rigorous principle, Buckypaper VMs aim to implement the full security model envisioned by the Confidential Computing Consortium, providing:

- **Increased Confidentiality:** Protecting data and code not only at rest and in transit, but also crucially, in use within the VM's memory, making it inaccessible to the untrusted host.
- **Enhanced Integrity:** Ensuring that the VM's execution environment and data cannot be tampered with by the hypervisor or other external malicious entities.
- **Vendor-independent Attestation:** Providing cryptographic proof of the VM's genuine state and the integrity of its software stack to remote, independent parties.

This model enables organizations to process highly sensitive data in public cloud environments with a significantly reduced Trusted Computing Base (TCB) at the cloud provider layer, shifting trust to the hardware (PSP) and the customer-controlled software stack.

4.3. Isolation Mechanism

The isolation for enclaive Buckypaper VMs is primarily underpinned by the capabilities of **AMD SEV-SNP:**

- **Hardware-Enforced Memory Encryption:** As detailed in Section 4.2.1, AMD SEV-SNP ensures that all private memory pages of the Buckypaper VM are automatically encrypted by the PSP when written to DRAM. This renders the VM's memory unintelligible to the hypervisor and other VMs on the same physical host.
- **Memory Integrity Protection:** AMD SEV-SNP's Secure Nested Paging (SNP) feature also provides robust integrity protection. The PSP, through mechanisms like the Reverse Map Table (RMP), prevents unauthorized modification, replay attacks, or re-mapping of the VM's memory by the hypervisor. Any detected integrity violation triggers an immediate fault, safeguarding the VM's state.
- **Guest OS Disk Encryption:** Complementary to the in-memory protection, the use of a disk encryption engine within the guest OS provides a strong isolation layer for data at rest. This means that if the underlying virtual disk (e.g., an EBS volume) were to be detached and accessed independently, its contents would remain encrypted.

- **Secure Endpoint Provisioning:** Complementary to cloud-init based practices, secrets and certificates for endpoint authentication are not generated within the EC2 instance, rather they are securely injected into the EC2 instance. This includes, for example, SSH host key pairs or SSH pass words. Their secrecy and integrity is critical to establish encryption-at-transit and exclude network eavesdropping.

This multi-layered approach to isolation, starting from the AMD hardware up through the guest OS, provides a comprehensive shield for the VM's entire workload.

4.4 Attestation Process

The attestation process for enclaive Buckypaper VMs is a critical component for establishing verifiable trust. It leverages the inherent capabilities of AMD SEV-SNP while integrating with external services for enhanced security and independent verification:

1. **AMD SEV-SNP Attestation Report:** The primary attestation primitive is the attestation report generated by the PSP. This report, as described in Section 4.2.2, contains a cryptographic "launch measurement" of the VM's initial memory and vCPU state, along with platform details, and is signed by the PSP's hardware-bound Versioned Chip Endorsement Key (VCEK).
2. **PSP as Root of Trust for Secure Boot:** The PSP acts as the hardware root of trust for the secure boot process within the Buckypaper VM. It validates the integrity of the UEFI firmware and the subsequent boot chain components (e.g., bootloader, kernel) before they are loaded and executed. This ensures that the VM's software stack starts in a known, trusted state, and these measurements are included in the overall attestation.
3. **Independent Attestation Verification Service (AVS):** The attestation report from the PSP is then sent to an independent Attestation Verification Service (AVS). This AVS, which operates outside the direct control of the cloud provider, verifies the authenticity of the report (using AMD's public keys) and compares the included measurements against a known set of expected values (e.g., hashes of the trusted kernel, bootloader, and other critical components). This independent verification provides strong assurance to the VM owner that their Buckypaper VM is running on genuine hardware and with the intended software stack.
4. **Integration with Key Management Service (KMS):** Upon successful attestation, the AVS (or the VM itself, after verifying its own attestation) can securely release cryptographic keys from a Key Management Service (KMS). These keys are typically required to unlock the guest OS's disk encryption, or to enable confidential processing within applications running inside the VM. This "attestation-bound key release" mechanism ensures that sensitive keys are only provided to a VM once its trusted and verified state has been established.

By combining the hardware-level security of AMD SEV-SNP with a robust, independently verifiable attestation and key management workflow, enclaive Buckypaper VMs provide a comprehensive and trustworthy platform for deploying highly sensitive workloads in public cloud environments, aligned with the stringent demands of confidential computing.

5 Comparative Analysis of Confidential Computing Solutions on AWS

The choice of a confidential computing solution hinges on a nuanced understanding of its underlying hardware, trust model, and operational implications. This section provides a comparative analysis of three prominent approaches for deploying confidential workloads on AWS: AWS EC2 Nitro Enclaves, native AWS EC2 instances with AMD SEV-SNP enabled, and enclaive's Buckypaper VMs, which operate on top of AMD SEV-SNP enabled EC2 instances.

5.1 Solution Overview

- **AWS EC2 with Nitro Enclaves:** A granular Trusted Execution Environment provided by AWS's proprietary Nitro System. It creates isolated compute environments next to a parent EC2 instance, designed to protect sensitive data and applications from the parent instance, its administrators, and AWS operators. Communication is restricted to a local vsock channel.
- **AWS EC2 with AMD SEV-SNP Enabled:** Leverages the hardware capabilities of AMD EPYC processors on specific EC2 instance types. It provides a Confidential Execution Environment by encrypting and integrity-protecting the entire VM's memory from the hypervisor and other VMs on the same physical host.
- **enclaive's Buckypaper VMs (on AMD SEV-SNP Enabled EC2):** A managed confidential computing platform that builds a comprehensive solution atop AMD SEV-SNP enabled EC2 instances. enclaive aims to simplify the deployment and management of confidential VMs, abstracting underlying complexities and adding features like independent attestation verification and guest OS disk encryption.

5.2 Comparative Table

Feature	AWS EC2 (Nitro Enclaves)	AWS EC2	enclaive EC2
Underlying Hardware	AMD/Intel/Graviton CPU + AWS Nitro Security Chip	AMD EPYC CPUs (3rd Gen or newer)	AMD EPYC CPUs (3rd Gen or newer)
Root of Trust	AMD+AWS	AMD+AWS	AMD
Trusted Computing Base	AMD/Intel/Graviton CPU + AWS Nitro System (Nitro Security Chip & Hypervisor) + AWS KMS+AVS	AMD + AWS Nitro System (Nitro Security Chip & Hypervisor) + AWS KMS+AVS	AMD + enclaive KMS+AVS

Protection Granularity	Para-VM (isolated execution environment next to parent EC2)	Entire VM (including Guest OS, applications, and all memory)	Entire VM (including Guest OS, applications, and all memory, storage, network)
Cloud Provider Visibility	full	full	none
Memory Confidentiality	none	Entire VM memory encrypted by AMD hardware	Entire VM memory encrypted by AMD hardware
Memory Integrity	Enclave code/data integrity via Secure Boot	Entire VM memory integrity via AMD SEV-SNP attestation	Entire VM memory integrity via AMD SEV-SNP attestation
Attestation Root	AWS	AMD	AMD
Attestation Verification	AWS KMS integration (can verify against AWS policies)	AWS AVS and KMS integration (can verify against AWS policies)	enclaive's AVS and KMS integration (can verify against customer policies)
Persistent Storage	No direct persistent storage within enclave	Standard VM storage (e.g., EBS)	Standard VM storage (e.g., EBS)
Data at Rest Encryption	n/a	Outside enclave (performed by Nitro Card)	Inside enclave (performed by disk encryption engine)
Primary Communication	vsock (secure local channel to parent EC2 only)	Standard VM networking	Standard VM networking
External Network Access	None (only via parent EC2 proxy over vsock)	Full external network access	Full external network access

End-Point Confidentiality	No	No	Yes
Application Modification	Often required (to interface with vsock and attestation SDKs)	Minimal to none (standard VM applications)	Minimal to none (standard VM applications)
Deployment Complexity	Moderate (requires enclave image building, vsock integration)	Moderate (instance type selection, AMI with UEFI/SEV-SNP support)	Simplified (managed platform, abstraction over hardware details)
Verifiable Implementation	No	No	Yes / Open-Source
Transparent Build	No	No	Yes
Compliance Focus	General-purpose TEE	General-purpose CEE	Strong focus on EU compliance (GDPR, NIS2) and tech soverei-gnty
Vendor lock	Strong (AWS proprietary technology)	Medium	Vendor-lock free
Multi-Cloud Support	AWS-only	AWS-only	Designed for multi-cloud deployments

5.3 Discussion

The three solutions present distinct trade-offs in terms of security guarantees, ease of use, and target workloads:

- **AWS EC2 Nitro Enclaves** offer unparalleled isolation for a specific part of an application. Their highly constrained environment significantly reduces the attack surface from both the parent instance and AWS. However, this isolation often necessitates application re-architecture to fit the enclave's communication model (vsock) and lack of direct external access. They are ideal for securing small, critical components like cryptographic key operations.
- **AWS EC2 instances with AMD SEV-SNP enabled** provide comprehensive protection for an entire virtual machine's memory, ensuring confidentiality and integrity against the hypervisor. This makes them suitable for "lift-and-shift" scenarios where an existing confidential application or entire operating system needs protection without extensive modification. The trust shifts directly to the AMD CPU hardware. However, when using the VMs naively without proper data-at-rest, data-in-transit encryption, external authentication and secret provisioning, the merit of memory encryption is negligible.
- **enclaive's Buckypaper VMs**, building on AWS EC2 instances with AMD SEV-SNP, provide a significantly higher confidentiality and integrity experience. enclaive layers additional security features (like guest OS disk encryption, enhanced attestation chains, post-quantum security) and operational convenience (simplified deployment, multi-cloud management, multi-cloud secret management) over the raw AMD SEV-SNP capabilities. Their explicit **"do not trust the hypervisor"** security model and independent attestation service are particularly appealing for organizations with stringent compliance requirements or a strong desire for third-party verification of trust. This approach effectively abstracts much of the underlying confidential computing complexity, allowing users to focus on their applications while benefiting from robust hardware-backed security.

The selection among these options depends on the specific threat model, the granularity of protection required, the level of application re-architecture feasible, and the overarching compliance and operational strategy of the organization.

Appendix A

Requirements EC2 Nitro Enclaves

Nitro Enclaves have the following requirements:

- Parent instance requirements:
  - Virtualized Nitro-based instances
    - Intel or AMD-based instances with at least 4 vCPUs, excluding `c7i.24xlarge, c7i.48xlarge, G4ad, m7i.24xlarge, m7i.48xlarge, M7i-Flex, r7i.24xlarge, r7i.48xlarge, R7iz, T3, T3a, Trn1, Trn1n, U-*, VT1`
    - AWS Graviton-based instances with at least 2 vCPUs, excluding `A1, C7gd, C7gn, G5g, Hpc7g, Im4gn, Is4gen, M7g, M7gd, R7g, R7gd, T4g`
  - Linux or Windows (2016 or later) operating system
- Enclave requirements:
    - Linux operating system only

Keep the following in mind when using Nitro Enclaves:

- Nitro Enclaves is supported in the following regions: US East (Ohio), US East (N. Virginia), US West (N. California), US West (Oregon), Africa (Cape Town), Asia Pacific (Hong Kong), Asia Pacific (Hyderabad), Asia Pacific (Jakarta), Asia Pacific (Mumbai), Asia Pacific (Osaka), Asia Pacific (Seoul), Asia Pacific (Singapore), Asia Pacific (Sydney), Asia Pacific (Tokyo), Canada (Central), Europe (Frankfurt), Europe (Ireland), Europe (London), Europe (Milan), Europe (Paris), Europe (Stockholm), Middle East (Bahrain), South America (São Paulo), AWS GovCloud (US-East), and AWS GovCloud (US-West).
- You can create up to four individual enclaves per parent instance.
- Enclaves can communicate only with the parent instance. Enclaves running on the same or different parent instances cannot communicate with each other.
- Enclaves are active only while their parent instance is in the running state. If the parent instance is stopped or terminated, its enclaves are terminated.
- You cannot enable hibernation and enclaves on the same instance.
- Nitro Enclaves are not supported on Outposts.
- Nitro Enclaves are not supported in Local Zones or Wavelength Zones.

Requirements EC2 AMD-SEV-SNP

To use AMD SEV-SNP, you must do the following:

- Use one of the following supported instance types:
  - General purpose:
    `m6a.large | m6a.xlarge | m6a.2xlarge | m6a.4xlarge | m6a.8xlarge`

- Compute optimized:
  `c6a.large | c6a.xlarge | c6a.2xlarge | c6a.4xlarge | c6a.8xlarge | c6a.12xlarge | c6a.16xlarge`
- Memory optimized:
  `r6a.large | r6a.xlarge | r6a.2xlarge | r6a.4xlarge`

- Launch your instance in a supported AWS Region. Currently, only US East (Ohio) and Europe (Ireland) are supported.
- Use an AMI with `uefi` or `uefi-preferred` boot mode and an operating system that supports AMD SEV-SNP. For more information about AMD SEV-SNP support on your operating system, refer to the respective operating system's documentation. For AWS, AMD SEV-SNP is supported on AL2023, RHEL 9.3, SLES 15 SP4, and Ubuntu 23.04 and later.

You can only enable AMD SEV-SNP when you launch an instance. When AMD SEV-SNP is enabled for your instance launch, the following rules apply.

- After it is enabled, AMD SEV-SNP can't be disabled. It remains enabled throughout the instance life cycle.
- You can only change the instance type to another instance type that supports AMD SEV-SNP.
- Hibernation and Nitro Enclaves aren't supported.
- Dedicated Hosts aren't supported.
- If the underlying host for your instance is scheduled for maintenance, you'll receive a scheduled event notification 14 days before the event. You must manually stop or restart your instance to move it to a new host.

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) may represent or impact current enclaive product offerings and practices, which are subject to change without notice, and © does not create any commitments or assurances from enclaive and its affiliates, suppliers, or licensors. enclaive products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of enclaive to its customers are governed by enclaive agreements, and this document is not part of, nor does it modify, any agreement between enclaive and its customers.