White Paper

# AMD SEV-SNP: Strengthening VM Isolation with Integrity Protection and More

January, 2020

*This white paper is a technical explanation of what the discussed technology has been designed to accomplish. The actual technology or feature(s) in the resultant products may differ or may not meet these aspirations. Each description of the technology must be interpreted as a goal that AMD strived to achieve and not interpreted to mean that any such performance is guaranteed to be fully achieved. Any computer system has risks of security vulnerabilities that cannot be completely prevented or mitigated.*

DISCLAIMER

The information contained herein is for informational purposes only, and is subject to change without notice. While every precaution has been taken in the preparation of this document, it may contain technical inaccuracies, omissions and typographical errors, and AMD is under no obligation to update or otherwise correct this information. Advanced Micro Devices, Inc. makes no representations or warranties with respect to the accuracy or completeness of the contents of this document, and assumes no liability of any kind, including the implied warranties of noninfringement, merchantability or fitness for particular purposes, with respect to the operation or use of AMD hardware, software or other products described herein. Any computer system has risks of security vulnerabilities that cannot be completely prevented or mitigated. No license, including implied or arising by estoppel, to any intellectual property rights is granted by this document. Terms and limitations applicable to the purchase or use of AMD's products are as set forth in a signed agreement between the parties or in AMD's Standard Terms and Conditions of Sale.

# Introduction

In 2016, AMD introduced Secure Encrypted Virtualization (SEV), the first x86 technology designed to isolate virtual machines (VMs) from the hypervisor. While hypervisors have traditionally been trusted components in the virtualization security model, many markets can benefit from a different VM trust model. In the cloud for instance, customers may want to secure their VM-based workloads from the cloud administrator to keep their data confidential and minimize their exposure to bugs in the cloud provider's infrastructure. This leads to a desire to isolate VMs at a hardware level from the hypervisor and other code that may happen to coexist on the physical server.

AMD began tackling this challenge through the use of main memory encryption in SEV. With this technology, individual VMs could be assigned a unique AES encryption key that is used to automatically encrypt their in-use data. When a component such as the hypervisor attempts to read memory inside a guest, it is only able to see the encrypted bytes.

In 2017, AMD introduced the SEV-ES (Encrypted State) feature which added additional protection for CPU register state. In SEV-ES, the VM register state is encrypted on each hypervisor transition so that the hypervisor cannot see the data actively being used by the VM. Together with SEV, SEV-ES can reduce the attack surface of a VM by helping protect the confidentiality of data in memory.

This white paper introduces the next generation of SEV called **SEV-SNP (Secure Nested Paging)**. SEV-SNP builds upon existing SEV and SEV-ES functionality while adding new hardware-based security protections. SEV-SNP adds strong memory integrity protection to help prevent malicious hypervisor-based attacks like data replay, memory re-mapping, and more in order to create an isolated execution environment. Also, SEV-SNP introduces several additional optional security enhancements designed to support additional VM use models, offer stronger protection around interrupt behavior, and offer increased protection against recently disclosed side channel attacks.

This white paper refers to SEV, SEV-ES, and SEV-SNP collectively as AMD SEV technologies.

# The Case for Integrity

AES encryption, as used with AMD SEV technologies, provides increased confidentiality protection of memory. An attacker without knowledge of the encryption key cannot decipher VM data that is stored in DRAM. The SEV memory encryption key itself is generated from a hardware random number generator and is stored in dedicated hardware registers where software cannot directly read it. Additionally, the hardware is designed so that identical plaintext at different memory locations will encrypt differently.

Despite the encryption, a motivated attacker may attempt to change values in memory, even without knowing the encryption key. These types of attacks are referred to as integrity attacks because the values in memory are not the same as what the VM intended. While an attacker cannot easily put known data into a VM's memory without knowledge of the encryption key, they may be able to corrupt memory so that the VM sees random values or conduct replay attacks. In a replay attack, an attacker captures ciphertext at one point in time and later replaces memory with the earlier captured data. This type of attack is more effective if the attacker knew what the original data was.

Integrity attacks by themselves do not directly compromise a VM. Software inside the VM must utilize the incorrect data, leading to a compromise or information disclosure. Whether such an attack is successful or not is dependent on the software inside the VM and how it behaves when encountering this compromised data. As the software in the VM is generally not aware if its memory integrity has been compromised, its behavior in this situation can be challenging to predict.

SEV-SNP is designed to prevent software-based integrity attacks and reduce risk associated with compromised memory integrity. **The basic principle of SEV-SNP integrity is that if a VM is able to read a private (encrypted) page of memory, it must always read the value it last wrote.** This means that if the VM wrote a value A to memory location X, whenever it later reads X it must either see the value A or it must get an exception indicating the memory could not be read. SEV-SNP is designed so that the VM should not be able to see a different value from memory location X.

To support standard VM tasks, this guarantee must hold regardless of what happens to memory in between the read and the last write. If that memory page is swapped to a disk, or even if the entire VM is migrated to a new host, the integrity guarantee must still hold. Enforcing this integrity guarantee requires a combination of new CPU hardware and firmware discussed later in this white paper.

In typical use cases, VMs must both execute their own tasks and communicate with outside entities via I/O. This may include communication over a network link, with a storage server, or with other components. In the SEV architecture, this communication is done using shared (unencrypted) memory. Any outgoing data that a VM desires to make available is placed into a shared page of memory, and any incoming data must similarly be placed into a shared page. Because shared memory is not encrypted with the VM's specific key, appropriate software encryption protocols like HTTPS should be used for security of I/O traffic.

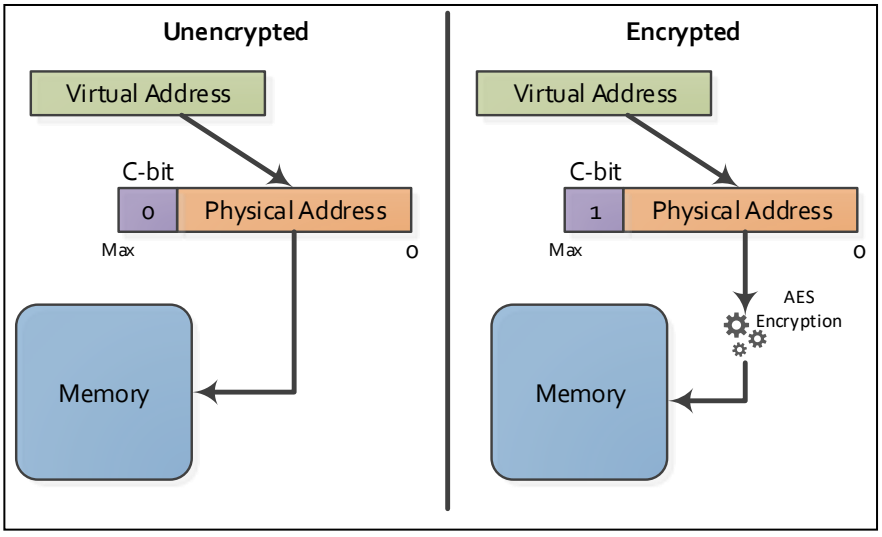


FIGURE 1: ENCRYPTION CONTROL

AMD SEV VMs control whether a memory page is private or shared using the enCrypted bit (C-bit) in the guest page tables. The location of the C-bit is implementation defined and may be the top physical address bit as shown in Figure 1. Shared (unencrypted) memory is marked C=0 by the VM, indicating it does not have to be encrypted with the VM's memory encryption key. Private (encrypted) memory pages are for the exclusive use of that VM and are marked as C=1. In a typical VM, most pages are marked private, and only the select pages needed for outside communication are marked shared. As with the SEV confidentiality guarantees, the SEV-SNP integrity guarantees only apply to private guest pages.

# Threat Model Details

As with the previous SEV and SEV-ES features, under SEV-SNP the AMD System-On-Chip (SOC) hardware, the AMD Secure Processor (AMD-SP), and the VM itself are all treated as fully trusted. The VM is responsible for protecting itself and its interfaces, and it should follow standard best practices for protecting any I/O data it uses such as network traffic, hard disk data, etc. To this end, AMD highly recommends using a Full Disk Encryption (FDE) solution with protected VMs since all SEV technologies only protect data in-use. FDE protects data-at-rest and many popular commercial solutions exist.

Under SEV-SNP, all other CPU software components and PCI devices are treated as fully untrusted as shown in Figure 2. This includes the BIOS on the host system, the hypervisor, device drivers, other VMs, etc. Fully untrusted means these components are assumed to be malicious, potentially conspiring with other untrusted components in an effort to compromise the security guarantees of an SEV-SNP VM.

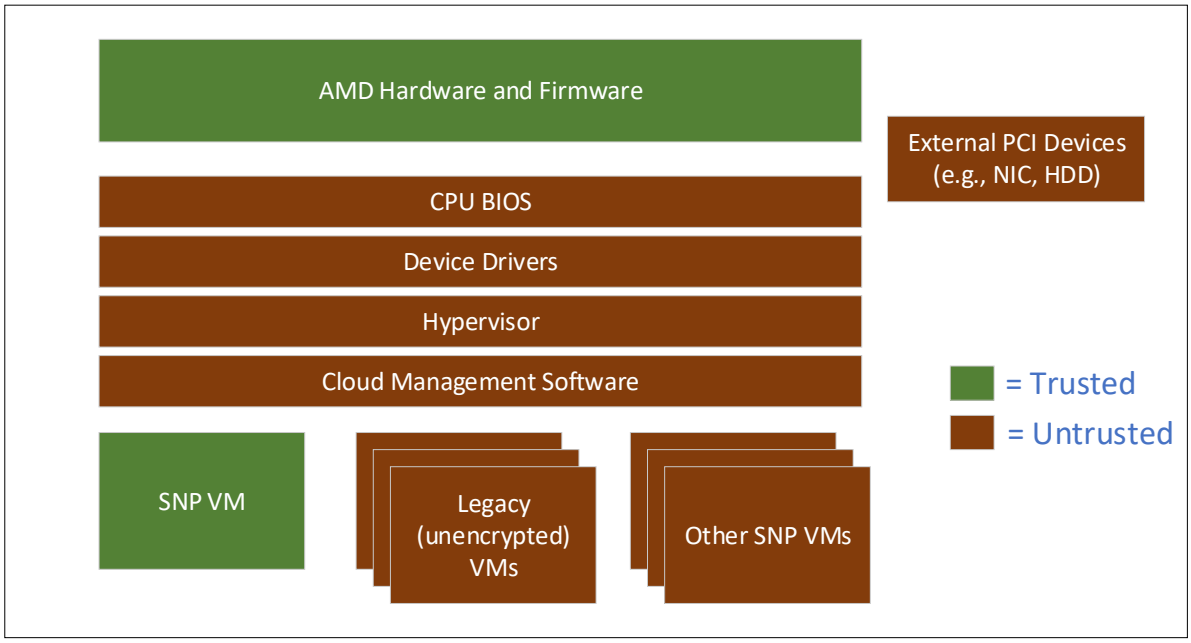


**FIGURE 2: SEV-SNP THREAT MODEL**

The SEV-SNP threat model includes features that are designed to protect against additional threats than previous AMD SEV technologies. SEV and SEV-ES use the threat model of a "benign but vulnerable" hypervisor. In this threat model, the hypervisor is not believed to be 100% secure, but it is trusted to act with benign intent. Meaning that while the hypervisor was not actively trying to compromise the SEV VMs underneath it, it could itself have exploitable vulnerabilities. By either blocking or making certain attacks more difficult, SEV and SEV-ES technologies can help limit the potential exposure of certain classes of hypervisor bugs or raise the difficulty of exploitation significantly. SEV-SNP addresses additional attack vectors and potential threats to VM security. The threats which are and are not addressed by various SEV technologies are summarized in Table 1.

**Confidentiality:** As noted, confidentiality threats are handled through the hardware-based memory encryption present in all current SEV technologies. This prevents an untrusted component, such as the hypervisor or a DMA-capable device, from being able to directly read the plaintext inside a

VM (except of course in cases where the VM has opted to allow untrusted access to a page). The SEV-ES technology added confidentiality protection for VM register state, encrypting this state when the VM exits back to the hypervisor. This protection exists in SEV-SNP as well.

**Integrity:** SEV-SNP technology is designed to protect against integrity attacks, which include data replay, corruption, re-mapping, and aliasing based attacks. The guarantee that a VM always sees the data it last wrote implies that all these attack vectors must be prevented.

**Availability:** There are two aspects of availability with any virtualization platform. The first is ensuring that the hypervisor retains control of the system, and the guest VM is not able to deny the hypervisor from running or otherwise render the physical machine unusable. All SEV technologies support this level of availability and guarantee that the hypervisor can always regain control when it desires (e.g., via a physical timer interrupt) or terminate a guest at any time without the consent of that VM. The second aspect of availability is whether the guest enjoys any guarantees of availability such as a minimum run-time. This is not part of any of the SEV technology threat models as a malicious hypervisor can always choose never to run some or all of a guest VM.

**Physical Access Attacks:** While certain physical attacks such as DRAM cold boot attacks (where DRAM chips are analyzed off-line) can be blocked by these technologies, on-line DRAM integrity attacks, such as attacking the DDR bus while the VM is actively running, are out of scope. These attacks are very complex and require a significant level of local access and resources to perform.

**Miscellaneous:** There are several other types of potential attacks against secure VMs, some of which are in-scope in this threat model. For instance, SEV-SNP includes features to help prevent Trusted Computing Base (TCB) rollback attacks. As discussed later, this enables a cryptographic means to verify that the AMD-SP firmware and other trusted components in the system meet the policy of the VM.

Additionally, SEV-SNP optionally supports the ability to restrict how interrupts and exceptions can be injected into a VM. It can also support Branch Target Buffer (BTB) protection against certain types of side channel attacks. Both protections are discussed later in this white paper.

Lastly, there are certain classes of attacks that are not in scope for any of these three features. Architectural side channel attacks on CPU data structures are not specifically prevented by any hardware means. As with standard software security practices, code which is sensitive to such side channel attacks (e.g., cryptographic libraries) should be written in a way which helps prevent such attacks. Fingerprinting attack protection is also not supported in the current generation of these technologies. Fingerprinting attacks attempt to determine what code the VM is running by monitoring its access patterns, performance counter information, etc. While fingerprinting can sometimes provide information about the code being run inside a VM, typically the most sensitive information is the data itself (e.g., data in the database), not the code being run (e.g., which version of the database software is being used). The current set of SEV technologies therefore focuses primarily on protecting the sensitive VM data contents. Additional protection against certain fingerprinting attacks may be offered in future SEV technologies.

| ✓ = Mitigated ★ = Optionally Mitigated ⊘ = Not Mitigated | *SEV* | *SEV-ES* | *SEV-SNP* |
|---|---|---|---|
| **Potential Threats** | | | |
| **Confidentiality** | | | |
| VM Memory<br>*Example attack: Hypervisor reads private VM memory* | ✓ | ✓ | ✓ |
| VM Register State<br>*Example attack: Read VM register state after VMEXIT* | ⊘ | ✓ | ✓ |
| DMA Protection<br>*Example attack: Device attempts to read VM memory* | ✓ | ✓ | ✓ |
| **Integrity** | | | |
| Replay Protection<br>*Example attack: Replace VM memory with an old copy* | ⊘ | ⊘ | ✓ |
| Data Corruption<br>*Example attack: Replace VM memory with junk data* | ⊘ | ⊘ | ✓ |
| Memory Aliasing<br>*Example attack: Map two guest pages to same DRAM page* | ⊘ | ⊘ | ✓ |
| Memory Re-Mapping<br>*Example attack: Switch DRAM page mapped to a guest page* | ⊘ | ⊘ | ✓ |
| **Availability** | | | |
| Denial of Service on Hypervisor<br>*Example attack: Malicious guest refuses to yield/exit* | ✓ | ✓ | ✓ |
| Denial of Service on Guest<br>*Example attack: Malicious hypervisor refuses to run guest* | ⊘ | ⊘ | ⊘ |
| **Physical Access Attacks** | | | |
| Offline DRAM analysis<br>*Example attack: Cold boot* | ✓ | ✓ | ✓ |
| Active DRAM corruption<br>*Example attack: Manipulate DDR bus while VM is running* | ⊘ | ⊘ | ⊘ |
| **Misc.** | | | |
| TCB Rollback<br>*Example attack: Revert AMD-SP firmware to old version* | ⊘ | ⊘ | ✓ |
| Malicious Interrupt/Exception Injection<br>*Example attack: Inject interrupt while RFLAGS.IF=0* | ⊘ | ⊘ | ★ |
| Indirect Branch Predictor Poisoning<br>*Example attack: Poison BTB from hypervisor* | ⊘ | ⊘ | ★ |
| Secure Hardware Debug Registers<br>*Example attack: Change breakpoints during debug* | ⊘ | ⊘ | ★ |
| Trusted CPUID Information<br>*Example attack: Hypervisors lies about platform capabilities* | ⊘ | ⊘ | ★ |
| Architectural Side Channels<br>*Example attack: PRIME+PROBE to track VM accesses* | ⊘ | ⊘ | ⊘ |
| Page-level Side Channels<br>*Example attack: Track VM access patterns through page tables* | ⊘ | ⊘ | ⊘ |
| Performance Counter Tracking<br>*Example attack: Fingerprint VM apps by performance data* | ⊘ | ⊘ | ⊘ |

**TABLE 1: THREAT MODEL**

| THREAT | DESIRED SECURITY PROPERTY | SEV-SNP ENFORCEMENT MECHANISM |
|---|---|---|
| **REPLAY PROTECTION** | Only the owner of a memory page can write that page | Reverse Map Table (RMP) |
| **DATA CORRUPTION** | Only the owner of a memory page can write that page | Reverse Map Table (RMP) |
| **MEMORY ALIASING** | Every physical memory page can map only to a single guest page at one time | Reverse Map Table (RMP) |
| **MEMORY RE-MAPPING** | Every guest page can map only to a single physical memory page at one time | Page Validation |

**TABLE 2: INTEGRITY THREATS**

# Integrity Threats

The previous section highlighted four unique types of integrity threats: Replay Protection, Data Corruption, Memory Aliasing, and Memory Re-Mapping. Protecting against these threats requires the enforcement of different security properties as shown in Table 2. In the case of Replay Protection and Data Corruption based attacks, these attacks rely on untrusted code being able to write to the memory of a protected VM. SEV-SNP addresses this by enforcing that only the owner of a memory page (e.g., the SEV-SNP VM to which the page was assigned) may write to that page. This enforcement is done using the Reverse Map Table (RMP) mechanism described in the following section.

Memory Aliasing attacks involve the hypervisor maliciously simultaneously mapping two different guest pages to the same physical memory page. A guest naturally expects that different pages in its guest physical address space map to different memory so any aliasing could lead to unintentional data corruption. Addressing this threat requires ensuring that every physical page of memory can only be mapped to one guest page at a time. Again, the RMP structure is used to enforce this property.

The final integrity threat, Memory Re-Mapping, involves the hypervisor maliciously re-mapping a single guest page to multiple different physical memory pages. In this threat, the guest might see an inconsistent view of memory where only a subset of data it wrote appears in memory. Addressing this threat requires ensuring that every guest page only maps to a one page of physical memory at a time, and that this mapping cannot be changed except by trusted entities like the AMD-SP. SEV-SNP uses a mechanism called Page Validation to address this threat. Page Validation relies on a combination of the new RMP mechanism with new VM code to manage the injective relationship between guest memory and system memory.

# Reverse Map Table

As mentioned, many of the integrity guarantees of SEV-SNP are enforced through a new structure called the Reverse Map Table (RMP). The RMP is a single data structure shared across the system that contains one entry for every 4k page of DRAM that may be used by VMs. The goal of the RMP is simple: it tracks the owner for each page of memory. Pages of memory can be owned by the hypervisor, owned by a specific VM, or owned by the AMD-SP. Access to memory is controlled so only the owner of that page can write it. The RMP is used in conjunction with standard x86 page tables to enforce memory restrictions and page access rights.

The RMP is indexed by system physical address and is checked at the end of CPU and IOMMU table-walks. For example, in native (non-VM) mode, virtual addresses are translated into physical addresses using the standard x86 page tables. After that translation, the final physical address is used to index the RMP. The RMP entry is read out and checked. If the RMP entry indicates that the page is a hypervisor-owned page, then the checks pass and a new TLB entry is created. If the RMP entry indicates that the page is not a hypervisor-owned page though, the table-walk faults (#PF) and the access is denied.

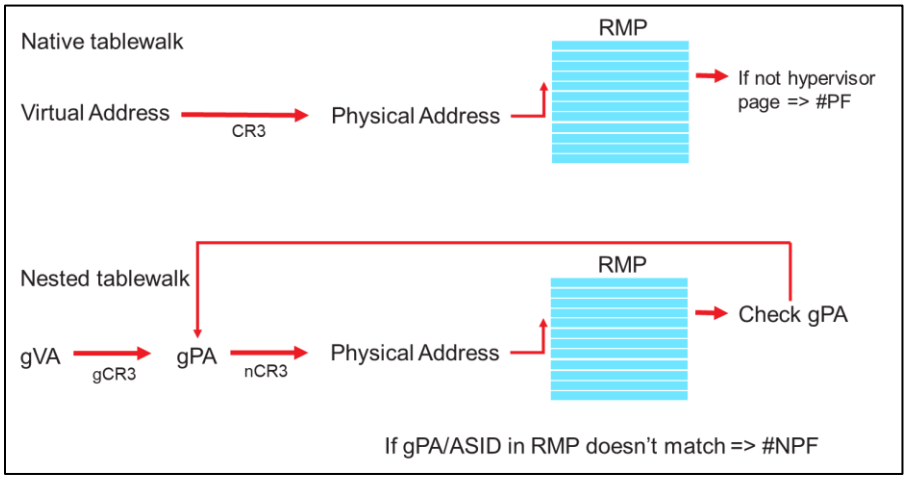


FIGURE 3: RMP CHECKS

When running in an SEV-SNP VM, the RMP check is slightly more complex. As with native mode, the virtual address is first translated into a system physical address. In this case, AMD-V 2-level paging[1] is used to translate a Guest Virtual Address (GVA) to a Guest Physical Address (GPA) and finally to a System Physical Address (SPA). The SPA is used to index the RMP and the entry is checked. This RMP entry should contain information indicating that the page is a guest-owned page, assigned to this specific guest, and mapped at this specific GPA. That is, the RMP entry contains the GPA where it should be mapped, and the hardware verifies that this GPA matches the GPA of the current table-walk. If this or any other check fails, an exception is generated, and the access is denied.

Not every memory access requires an RMP check. In particular, read accesses from the hypervisor (or non-SEV-SNP guests) do not require RMP checks because data confidentiality is already protected via the AES memory encryption. Otherwise, all write accesses in any mode require an RMP check, and both read and write accesses to private memory pages inside an SEV-SNP require RMP checks. Write accesses include both standard memory writes as well as A/D-bit updates as part of the page table walk. Like with standard x86 paging, the results of the RMP check are cached in the CPU TLB and related structures.

Because the RMP is used to enforce access control to memory, the table itself is not directly writeable by software. New CPU instructions exist to enable manipulation of RMP entries, allowing the hypervisor to assign pages to specific guests, take pages back, etc. When required, hardware automatically performs TLB invalidations to ensure that all processors in the system see the updated RMP entry information.

[1] Also known as Nested Paging, as described in section 15.25 of the AMD64 Programmer's Manual Volume 2.

# Page Validation

As mentioned earlier, each RMP entry contains the GPA at which a particular page of DRAM should be mapped. This ensures by construction that every SPA can only be mapped to a single GPA at one time. The inverse, a single GPA mapping to more than one SPA, also cannot be allowed to meet the SEV-SNP integrity guarantee. While the nested page tables ensure that each GPA can only map to one SPA, the hypervisor may change these tables at any time. SEV-SNP integrity requires that manipulating the tables in this way cannot break the desired integrity and this is addressed through the concept of Validation.

Inside each RMP entry is a Validated bit, and this bit is automatically cleared to 0 by CPU hardware when a new RMP entry is created for a guest. Pages that are assigned to guests but have the Validated bit clear are not useable by the hypervisor or as a private guest page since the page is not validated. The guest can only use the page after it sets the Validated bit through a new CPU instruction, PVALIDATE. Only the guest is able to use PVALIDATE, and each guest VM can only validate its own memory.

Adding a new page to a guest VM therefore requires a 2-step process as shown in Figure 4. First, the hypervisor assigns the page to the guest using the new RMPUPDATE instruction. This transitions the page into the Guest-Invalid state. Second, the guest validates the page using the new PVALIDATE instruction to transition the page to the Guest-Valid state, from where it can be used.

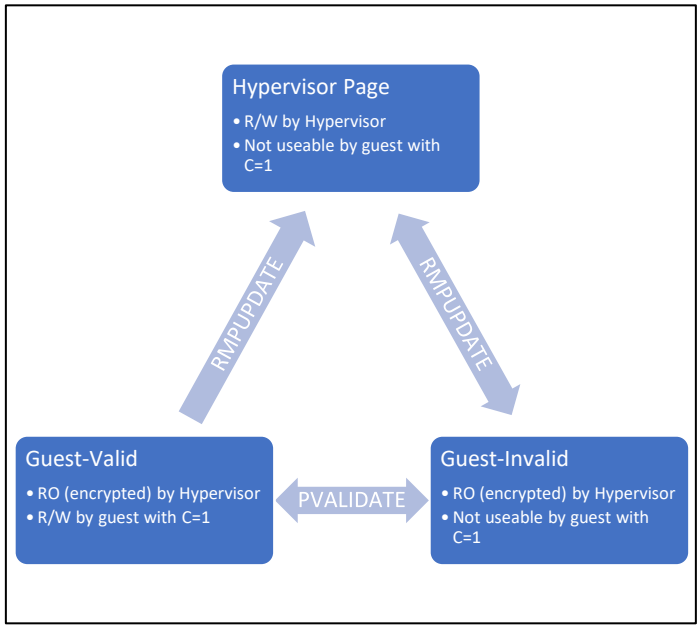


FIGURE 4: BASIC PAGE STATES

In order to meet the desired integrity of SEV-SNP, **the guest VM should never validate memory corresponding to the same GPA more than once**. This can be accomplished simply by the guest VM validating all of its memory at boot and refusing to ever validate additional memory (other than as part of hot-plug events). Alternatively, the guest VM can track memory locations it has validated and refuse to ever validate the same one twice.

Assuming the guest VM correctly validates its memory, this guarantees the injective mapping between GPAs and SPAs. The guest will validate each GPA only once, and the RMP table, by construction, ensures that each SPA can only map to one GPA.

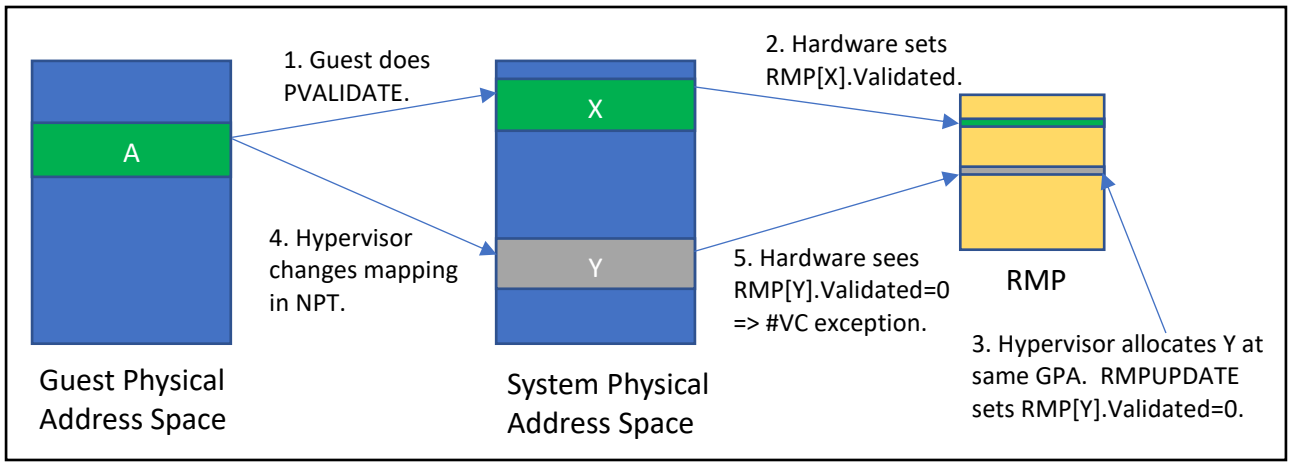


**FIGURE 5: PAGE RE-MAPPING ATTACK**

When done correctly, page validation can block re-mapping attacks like the one shown in Figure 5. In this example, GPA A is initially mapped to SPA X. The guest does a PVALIDATE to validate this translation, which causes the Validated bit to be set in the RMP entry corresponding to SPA X. If the hypervisor then maliciously attempts to remap A to a different SPA Y, it will start by creating an RMP entry for SPA Y attempting to map the same GPA A using the RMPUPDATE instruction. The hypervisor then maliciously modifies the nested page table (NPT) to re-map GPA A to Y. When the guest accesses Y however it will get a #VC (VMM Communication) exception. This exception occurs because the Validated bit in the RMP entry corresponding to SPA Y was clear (as when RMPUPDATE was executed to assign a new page to the guest, it initially cleared the Validated bit). As the guest knows it had already validated GPA A, it knows it should not be receiving a validation error and therefore it is under attack and the hypervisor is not behaving correctly. In response, the guest can terminate or take other steps to protect itself.

## Page States

As shown, the RMP in SEV-SNP tracks the state of each page of memory. These states dictate what the memory can be used for, who is allowed to read/write it, and to what states the page can later be transitioned. For instance, pages in the Hypervisor state can be read/written by the hypervisor, or by SEV-SNP VMs accessing the memory with C=0 (shared pages). Pages in the Guest-Valid state by contrast can be read/written by SEV-SNP VMs but cannot be written by the hypervisor.

The diagram in Figure 4 describes three of the basic page states: the Hypervisor, Guest-Valid, and Guest-Invalid states. In total, there are eight main page states defined by the SEV-SNP architecture as listed in Table 3. Page state transitions are shown in Figure 6 and can occur via the new CPU RMPUPDATE instruction (red), the new PVALIDATE instruction (blue), or through the VM management API in the AMD-SP (green).

As with previous SEV technologies, SEV-SNP implements a VM management API in the AMD-SP. The hypervisor calls this interface to assist with VM lifecycle tasks and page management. For security reasons, any pages the AMD-SP will manipulate must be placed into special states, called Immutable states, prior to issuing the necessary API call. Pages in the Immutable states cannot be written by any software on the CPU (hypervisor or guest) and cannot have their RMP entry modified by anyone other

| STATE | DESCRIPTION | NOTES |
|---|---|---|
| **HYPERVISOR** | Default state for otherwise unassigned memory | Used for hypervisor memory, non-SNP-VM memory, and shared (C=0) memory |
| **GUEST-INVALID** | Page is assigned to a guest but not ready to be used | Not useable by SEV-SNP VMs until validation has occurred |
| **GUEST-VALID** | Page is assigned to a guest and useable | Page may be used as private (C=1) memory by the assigned SEV-SNP VM |
| **PRE-GUEST** | Page is Immutable and not validated | Used when initially launching SEV-SNP VMs |
| **PRE-SWAP** | Page is Immutable and validated | Used when swapping guest pages to disk |
| **FIRMWARE** | Page is Immutable and reserved for AMD-SP use | Typically used as transitory state until AMD-SP has configured the page |
| **METADATA** | Page is Immutable and used for metadata | Metadata is used when swapping guest pages to disk |
| **CONTEXT** | Page is Immutable and used for context information | Context pages are used by the AMD-SP to identify individual VMs and hold per-VM data |

TABLE 3: SEV-SNP PAGE STATES

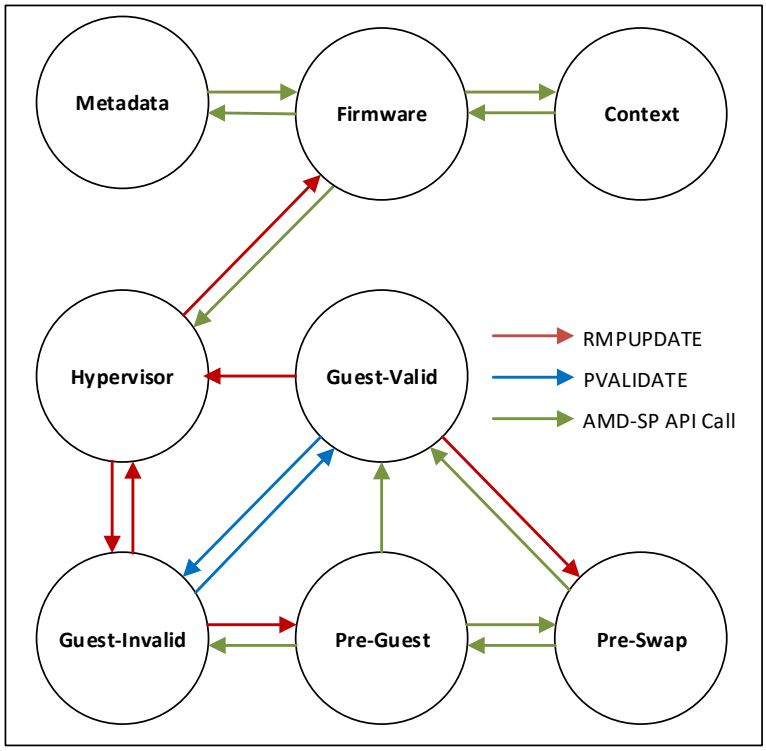


FIGURE 6: PAGE STATE TRANSITIONS

than the AMD-SP. When the AMD-SP is finished processing a page in one of these Immutable states, it may transition it to a different state as defined by the specific API call.

For example, 'Metadata' pages are a type of Immutable pages. These pages are only writeable by the AMD-SP and are used to hold metadata entries associated with guest pages that have been swapped to disk. Because of the SEV-SNP integrity guarantees, any pages that are swapped to disk must have their integrity confirmed before they can be swapped back into memory. When a page is swapped to disk, the AMD-SP creates a metadata entry containing an authentication tag (from AES-GCM), as well as data from the page's RMP entry, such as the GPA where it was located. As the Metadata page itself is not writeable by the hypervisor, the integrity of this information is guaranteed. When a page is swapped back into memory, the AMD-SP verifies the contents were unchanged and ensures the page enters the guest address space at the same location as it was before. Metadata pages themselves can also be swapped to disk in a similar fashion, allowing for the entire guest to be saved to disk if desired.

# Virtual Machine Privilege Levels

Virtual Machine Privilege Levels (VMPLs) are a new optional feature in the SEV-SNP architecture which allows a guest VM to divide its address space into four levels. These can be used to provide hardware isolated abstraction layers within a VM for additional security controls, as well as assistance with managing communication with the hypervisor.

These levels are hierarchical in nature where VMPL0 is the highest privilege and VMPL3 is the least privileged. When this feature is enabled, every vCPU of a VM is assigned a VMPL. The RMP entry for each page of private guest memory is also augmented with page access rights corresponding to each VMPL and are applied in addition to standard paging permissions. Specifically, individual guest pages can be marked as readable, writeable, supervisor-mode executable, and user-mode executable. By default, when a page is first validated by a guest, VMPL0 is granted full permissions to the page and all other VMPLs are granted no permissions. The guest can choose to modify VMPL permissions via the new RMPADJUST instruction.

The RMPADJUST instruction allows a given VMPL to modify permissions for a less privileged VMPL. For example, VMPL0 can grant read and write (but not execute) permissions on a page to VMPL1. This is restricted so one level cannot grant more permissions than it currently has. VMPLs are primarily used to set additional page permission checks and are otherwise orthogonal to other x86 security features.

The RMP page permission checks are performed at the time of the RMP lookup at the end of a table-walk. Page permissions checks are restrictive in nature, so for a guest page to be writeable for instance, it must be marked writeable in the guest-managed page tables (corresponding to the active vCPU), the nested page tables (managed by the hypervisor), as well as the RMP table (managed by a higher privileged VMPL).

VMPLs are in some ways like nested virtualization in that a guest may contain its own management layer running at high VMPL which controls permissions on its other pages. This enables use cases such as the secure virtualization of a security enforcing hypervisor. While on a bare-metal system, a standard hypervisor may be used to enforce that certain pages are read-only, not executable, etc., SEV-SNP enables that same use model in a cloud environment. In this case, VMPL0 inside the guest would enforce the required page permissions as the true hypervisor in the cloud is treated as untrusted.

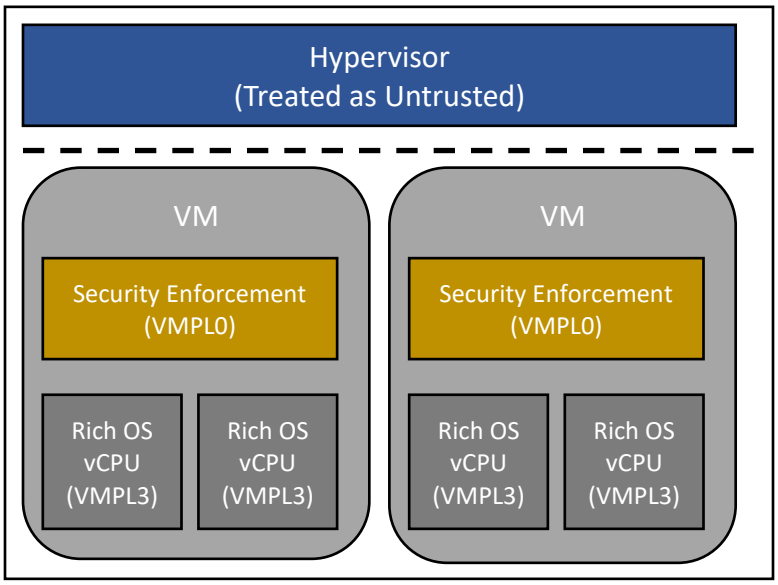


**FIGURE 7: VMPLS**

VMPLs are useful in several additional scenarios as well as an abstraction layer. For example, APIC emulation is a task traditionally handled by the hypervisor. In SEV-SNP, certain VMs may desire a more restrictive environment where APIC emulation is moved inside the trust domain of the guest. In this case, VMPL0 can be used to perform trusted APIC emulation while allowing the rest of the guest running in lower VMPLs, remaining unaware of the emulation.

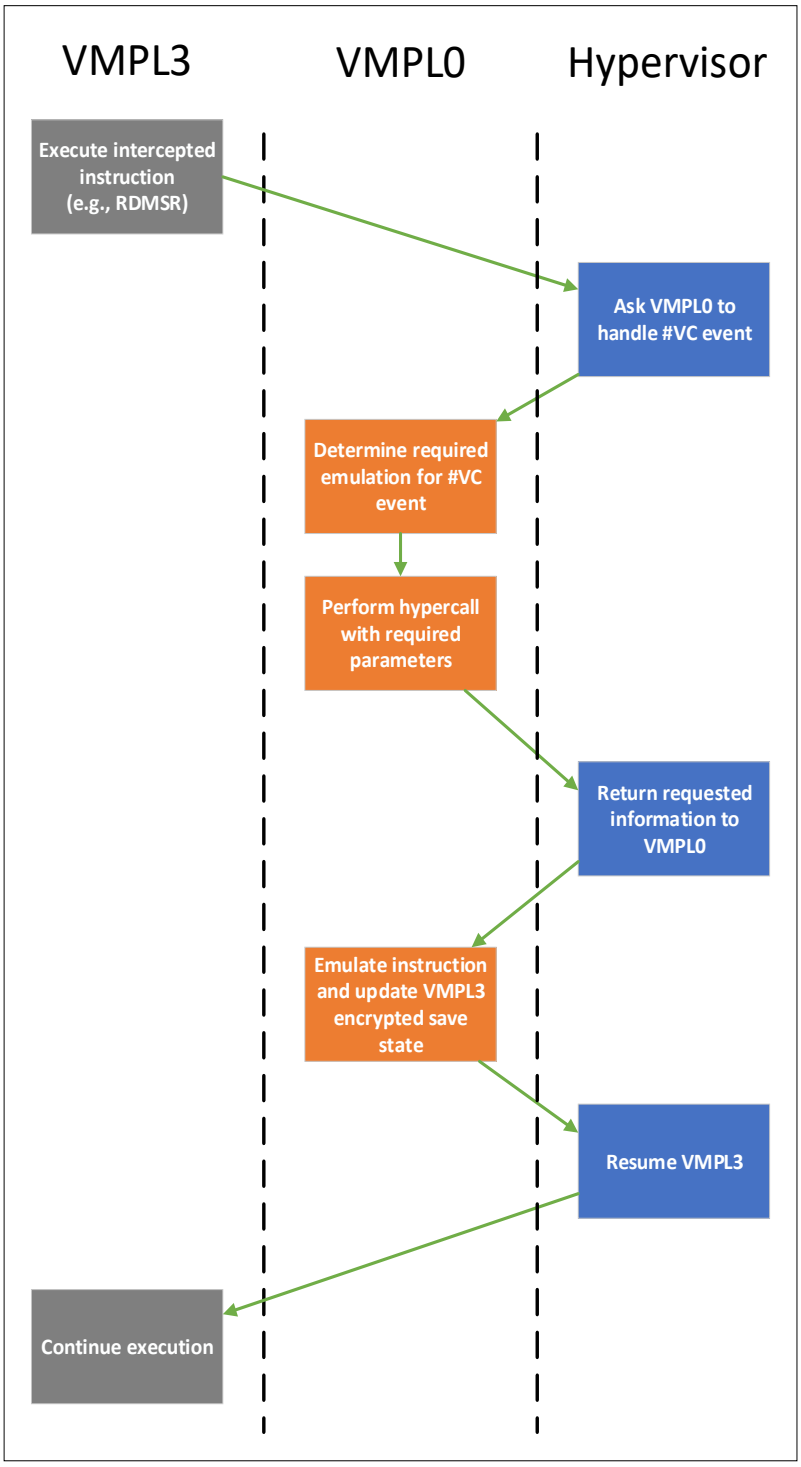


FIGURE 8: VMPL0 #VC EMULATION

VMPL0 can also be used to act as an intermediary for guest to hypervisor communications. Previously, SEV and SEV-ES technologies have required an enlightened guest OS which is aware of these security features. The guest OS was expected to do things like set the C-bit in page tables, handle #VC exceptions (in SEV-ES), and more. In the case of SEV-SNP, these tasks can be optionally delegated instead to VMPL0.

In this use model, VMPL0 can be used to configure which guest memory in another vCPU is private (C=1) versus shared (C=0) using a watermark called the Virtual Top of Memory (vTOM). Memory addresses below vTOM are automatically treated as private while memory above vTOM is treated as shared. Using vTOM to separate memory in this way avoids the need to augment the standard x86 page tables with C-bit markings, simplifying guest OS software.

Additionally, VMPL0 can be used to handle #VC events that occur in another vCPU. An SEV-SNP VM can be configured so that when an intercepted instruction is executed in one vCPU (e.g., RDMSR), that vCPU exits and VMPL0 can be invoked. VMPL0 can then view the intercepted information directly from

the encrypted save area of the original vCPU, perform any necessary hyper-calls, and emulate the instruction on behalf of the original vCPU as shown in Figure 8.

While not as performant as a natively enlightened guest, this behavior can enable VMPL0 to serve as the glue logic for running unenlightened (legacy) guest VMs. This has the potential to allow SEV-SNP to be used to secure older workloads that may not be easily upgrade-able to newer operating systems.

## Interrupt/Exception Protection

While almost all VM operating systems support interrupt and exception handling, some operating systems may have built-in assumptions about interrupt and exception behavior based on bare-metal hardware. If those assumptions can be violated by a malicious hypervisor, it is possible this behavior may violate the assumptions of the design of the operating system. For example, an operating system may not expect to take a low priority interrupt when their TPR is elevated or it may not expect to take a #UD exception after executing an ADD instruction.

To address these concerns, SEV-SNP adds two optional modes that VMs can choose to enable to support a more restrictive interface between the VM and hypervisor regarding interrupts and exceptions. The first mode, called Restricted Injection, disables the virtual interrupt queuing and partially disables the interrupt injection interface. In this mode, the hypervisor is only allowed to inject a single newly defined exception vector, #HV, to act as a doorbell. Restricted Injection assumes that the VM and hypervisor will communicate events in a para-virtualized manner, such as an event queue in shared memory. The #HV exception can be a signal to the guest to re-scan the event queue for new information.

A second mode, called Alternate Injection, allows for standard virtual interrupt queuing and injection interfaces, but these may only be controlled by the guest itself. New fields are added to the encrypted save area (called the Virtual Machine Save Area or VMSA) which allow for interrupt queuing and event injection. Being in the VMSA, these fields can only be manipulated by someone with access to the guest VMSA data, such as VMPL0. In the Alternate Injection mode, all interrupt related security sensitive state (such as TPR) is saved to the VMSA so it cannot be manipulated by a malicious hypervisor.

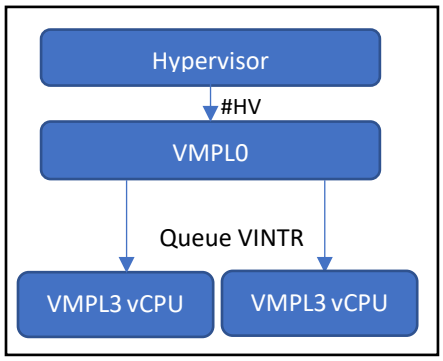


FIGURE 9: VMPL INTERRUPT HANDLING

Combined, these two modes enable VMPL0 to perform interrupt handling and APIC emulation. vCPUs that are used to run VMPL0 can be run with Restricted Injection enabled so they communicate with the hypervisor using a para-virtualized interface and the #HV exception. vCPUs used to run other VMPLs (where presumably the main OS of the guest runs) can be run with Alternate Injection enabled. In this way, VMPL0 can inject events and virtual interrupts to the main OS when it is safe to do so. From a software standpoint, the main OS can use a standard APIC or x2APIC interface and any APIC accesses it needs can be trapped and emulated in VMPL0.

## Trusted Platform Information

Platform features and capabilities are traditionally discovered via the CPUID instruction. Hypervisors typically trap and emulate the CPUID instruction for a variety of reasons, including limiting the features a guest may use to make it easier to migrate. In many cases, a malicious hypervisor can only cause denial-of-service on a guest by lying about CPUID features.

There are some cases however where incorrect CPUID information can potentially lead to a security issue. For instance, a hypervisor which lies about the size of the x86 Extended Save Area (used with the XSAVE/XRSTOR CPU instructions) may cause a guest to allocate too small a memory region and have a buffer overflow when it executes the hardware XSAVE instruction. While it is possible in many cases for guest VMs to attempt to validate the CPUID it receives (e.g., by checking that the reported XSAVE buffer size is correct), this may be challenging, especially during the boot process.

To simplify the checks that a guest must do, SEV-SNP supports an optional capability to filter CPUID results through the AMD-SP. The AMD-SP will verify that the CPUID results that the hypervisor is reporting are no greater than the capabilities of the platform and that security sensitive information, such as the x86 Extended Save Area size, is correct.

CPUID filtering may be done either on-the-fly, or as part of guest boot. For on-the-fly filtering, after receiving CPUID information the guest may ask the AMD-SP to verify that security sensitive information is correct. Alternatively, during VM launch a special 'CPUID page' can be created which contains pre-vetted CPUID information from the AMD-SP allowing the guest access to trusted CPUID information starting from early boot. In addition to the security benefits, this special page allows for faster access to CPUID information which can speed up the VM boot process.

## TCB Versioning

In the SEV-SNP architecture, there are several upgradeable firmware components, including the AMD-SP API, the CPU microcode patch, and more. As these firmware components are considered trusted in the SEV-SNP threat model, they form the TCB of the architecture. As bugs are fixed or features are upgraded, new versions of these components may be released. If a security bug were to be found in one of these components, a guest owner may require a guarantee that their VM is running under patched firmware and not a vulnerable version.

Prior SEV and SEV-ES features have relied on a self-reported AMD-SP version number to implement TCB versioning. A guest owner could specify a minimum version of AMD-SP firmware and the VM could not be loaded on an older one. In SEV-SNP, this check has been enhanced to be cryptographically strong. In SEV-SNP, the version numbers of all TCB components are combined with a fused secret called the Chip Endorsement Key to create a Versioned Chip Endorsement Key (VCEK). The VCEK is a private ECDSA key which is unique to each AMD chip, running a specific TCB version. The construction of the VCEK uses cryptographic hash functions so that a given TCB version cannot fake being a newer TCB. The VCEK is used in several ways, including for signing attestation reports.

# VM Launch & Attestation

As with previous SEV and SEV-ES architectures, SEV-SNP VMs start from an initial unencrypted image. This initial image is expected to contain things like the guest VM boot code, but as it starts unencrypted, it should not contain any secrets. During the launch process, the hypervisor asks the AMD-SP to install this initial set of pages in the guest. The AMD-SP cryptographically measures the contents of these pages into a launch digest. In the SEV-SNP architecture, the AMD-SP also measures the metadata associated with these pages, namely the GPA where the pages are being placed and the type of page they are. This ensures that the launch digest captures the layout of initial guest memory, as well as its contents.

In SEV-SNP, at the end of the launch process, the guest owner can supply a signed Identity Block (IDB) to associate with the VM. The IDB contains fields that allow the guest owner to uniquely identify the VM and contains the expected launch digest. The IDB can only be associated with VMs which match the expected launch digest and is included as part of the attestation report.

While SEV and SEV-ES only support attestation during guest launch, SEV-SNP supports more flexible attestation. Attestation reports can be requested through a protected path from the AMD-SP by the guest VM at any time. As part of SEV-SNP VM launch, a set of private communication keys are created by the AMD-SP which the guest can use to communicate directly with the AMD-SP. The guest can use this path to request attestation reports, cryptographic keys, and more.

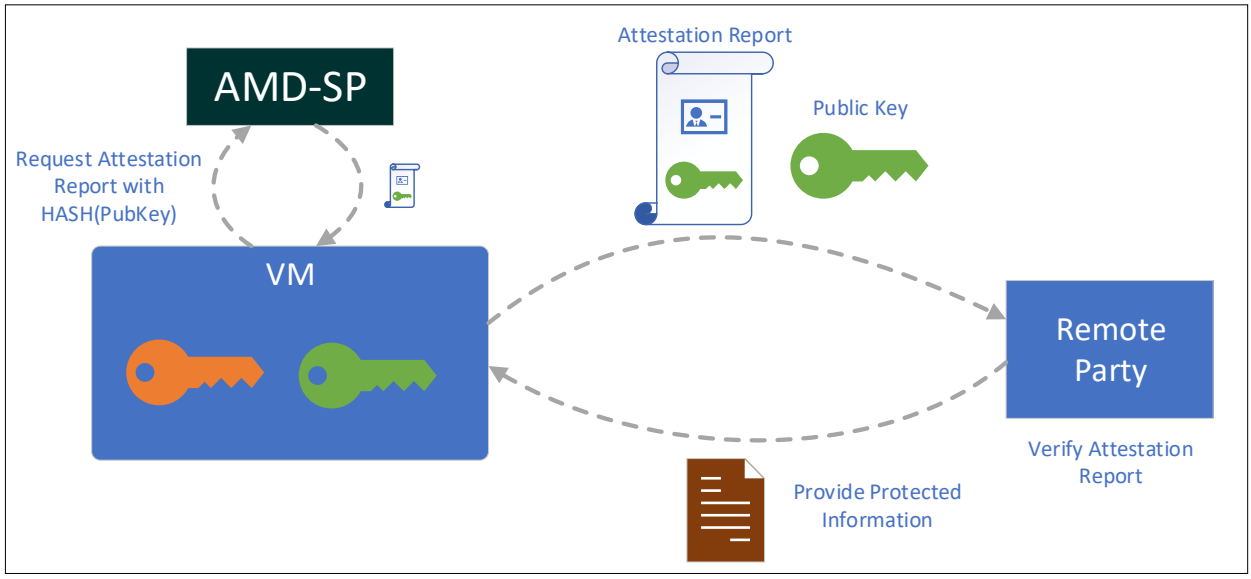


**FIGURE 10: SEV-SNP ATTESTATION**

Attestation reports contain the IDB information from launch, system information, and a block of arbitrary data supplied by the guest VM as part of the report request. The attestation report is signed by the AMD-SP firmware using the VCEK. Attestation reports enable a third party, such as the guest owner, to validate that certain data came from a certain VM.

For example, a VM can publish a public key and ask the AMD-SP for an attestation report containing the hash of this public key as shown in Figure 10. A third party can then verify that this public key is associated with this VM through the attestation report. The attestation report also proves that

the VM was running with the appropriate security features enabled, and that it started on an authentic AMD platform. Because the attestation report is signed by the VCEK, the verification of this report proves both the authenticity of the platform and the TCB version used (since the VCEK is derived from the TCB version). Upon successful attestation, a third party such as the guest owner can then choose to provide the VM with secrets, such as a disk decryption key, or other keys required for operation.

Besides remote attestation, SEV-SNP supports additional use models for generating guest key material. SEV-SNP VMs may request keys directly from the AMD-SP for various purposes such as data sealing. These keys may be derived from different sources, and the VM may select which sources are used as needed for their use case. For instance, local sealing keys can be requested which are specific to the current part at a certain TCB level and specific to the IDB signing key. Through these controls, the VM may request keys that are guaranteed to not be able to be derived by a malicious actor or another device.

## VM Migration

VM migration, and specifically live VM migration, are standard features of modern cloud architectures. Live VM migration allows for moving one VM to another physical system without interruption when needed for load balancing, host system maintenance, and other purposes. All SEV technologies support VM migration, but SEV-SNP enhances the flexibility associated with migration.

In SEV and SEV-ES, VM migration was dictated by a guest owner supplied policy. This policy indicated whether the VM was migratable, and if so to what type of systems. The AMD-SP was responsible for enforcing this migration policy and did so by authenticating the AMD-SP on the destination machine prior to starting migration.

In SEV-SNP, the role of migration policy enforcement has been offloaded to a new entity called a Migration Agent (MA). The MA is itself an SEV-SNP VM that runs on the same physical system as the primary VM. When a VM is launched, it can optionally be associated with an already running MA. Information about the MA binding of a VM is present in its attestation report as the MA is part of the TCB of the guest. Each VM can only be associated with a single MA, but a single MA can manage migration for an arbitrary number of VMs.

The MA is responsible for determining what systems to which the primary VM can migrate. While the details of the MA architecture are beyond the scope of this white paper, the MA can implement complex migration policies using whatever means it desires.

In a typical cloud scenario, the MA is not itself migratable. Instead, a separate instance of the MA runs on each physical machine. When a VM is about to migrate, the MA on the source machine authenticates the MA on the destination machine and establishes a protected network connection. If this is successful, the MA transfers the required guest information so the guest can be reconstituted on the new machine.

It is important to note that because of the flexibility of the MA, it is not required that both the source and destination machines be online at the same time. When a VM is paused, its state can be exported to its MA. The MA can choose to move that state over to another MA immediately for live

migration, or it may choose to hold onto it or put it into long term storage. Later, this VM can be reconstituted on this same machine or another one as desired by the MA.

## Side Channels

A lot of security research has recently focused on CPU side channel attacks, which are attacks that leverage the internal structures of the CPU to leak information[2]. Speculative side channel attacks such as Spectre[3] have been demonstrated leveraging standard techniques like hardware branch prediction to create data leakage in certain scenarios. AMD has added hardware capabilities[4] to help software defend itself against certain attacks, like Spectre Variant 2.

Spectre Variant 2 demonstrated that the indirect branch predictor (BTB) can be exploited given certain software circumstances, and assuming an attacker is able to influence the branch predictions of another entity. In recent CPU designs, AMD has added support for the SPEC_CTRL MSR and PRED_CMD MSR which enables more software control of the BTB structure. In the SEV-SNP architecture, the SPEC_CTRL MSR is virtualized allowing the guest to choose its own speculation policy independent of the hypervisor. This allows the guest to use modes such as IBRS.

In traditional virtualization, the hypervisor takes steps to protect itself from guest-based attacks. This may include techniques like retpoline or running with IBRS set. When the hypervisor is not trusted, the guest also may be concerned with hypervisor-based attacks. For instance, a malicious hypervisor could attempt to poison the BTB entries the guest will use or may try to use another VM to poison the BTB before the SEV-SNP guest runs.

To protect against such attacks, SEV-SNP VMs may opt into additional protection whereby the CPU hardware will prevent the VM from speculatively using BTB entries installed by another entity. This feature tracks when BTB entries are installed by either the hypervisor or other software and will automatically perform a BTB flush when required so that the SEV-SNP VM does not speculatively use those BTB entries.

Simultaneous Multi-Threading (SMT) is another area of CPU hardware that has been a focus of side channel research. Due to the shared hardware resources in SMT designs, more channels of observation are possible. SEV-SNP VMs that believe themselves to be particularly sensitive to such observation may opt into a policy which restricts them to only being run on SMT-disabled systems.

While SEV-SNP offers guests several options when it comes to protection from speculative side channel attacks and SMT, it is not able to protect against all possible side channel attacks. For example, traditional side channel attacks on software such as PRIME+PROBE are not protected by SEV-SNP. These types of attacks require specifically targeting software algorithms that are vulnerable to these types of side channels, typically because they involve code paths which vary their cache or TLB access patterns based on a secret value. Modern cryptographic libraries take special care to avoid such behavior as

---

[2] For information about specific security vulnerabilities and their applicability to AMD, please see the AMD Product Security website at https://www.amd.com/en/corporate/product-security

[3] https://spectreattack.com/spectre.pdf

[4] https://developer.amd.com/wp-content/resources/Architecture_Guidelines_Update_Indirect_Branch_Control.pdf

these types of attacks can occur even on non-virtualized platforms. Because SEV-SNP hardware is not designed to explicitly protect against such attacks, it is the responsibility of VM owners to follow standard security practices and ensure their libraries and software are updated to not use algorithms which may be vulnerable to these attacks.

Another category of side channel attacks which are outside the scope of SEV-SNP include application fingerprinting attacks, such as performance or page fault monitoring. As mentioned earlier, SEV-SNP focuses primarily on protecting the data inside VMs and these types of attacks, which typically only attempt to determine what application is being run, do not directly break the confidentiality or integrity of guest VM data. Future versions of SEV may include additional protections against some of these attack vectors.

# Conclusion

SEV-SNP represents an enhanced level of security and isolation for virtual machines running in untrusted hosting environments. Building upon the SEV and SEV-ES features which provided data confidentiality protection against potentially buggy hypervisors, SEV-SNP adds integrity guarantees capable of protecting VMs from malicious hypervisors. Besides integrity protection, SEV-SNP also provides new architectural flexibility in the form of multiple VMPLs, new attestation and key derivation architectures, and an arbitrarily flexible migration policy.

SEV-SNP also raises the security bar by providing optional protection against malicious interrupt injection, certain speculative side channel attacks, and TCB rollback attacks. These features, like with previous SEV and SEV-ES features, are designed to be enabled at the guest operating system level, meaning that no changes to applications inside of VMs are required.

VM isolation is a challenging task for a modern cloud computing environment. SEV-SNP is the first x86 architecture designed to support both confidentiality and integrity protection for isolated VMs, enabling more secure cloud computing for a variety of workloads. AMD believes that secure cloud computing is a critical workload for tomorrow's datacenter and SEV-SNP is the next step toward that realization.