AMD SEV-TIO: Trusted I/O for Secure Encrypted Virtualization

March 2023

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.

AMD, the AMD Arrow logo, and combinations thereof are trademarks of Advanced Micro Devices, Inc. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

Introduction

Data confidentiality and integrity are two of the top security concerns in the modern datacenter. AMD is working to address this challenge with the Secure Encrypted Virtualization (SEV) [1] technologies which utilize hardware to isolate virtual machines from each other and from the hypervisor. These technologies provide guests a confidential computing environment which helps mitigate the risks that guests are exposed to from shared infrastructure environments like public cloud.

In 2016 AMD introduced the first generation of SEV and has since developed subsequent generations which introduced additional features and new SEV technologies that improved the confidentiality and integrity protection of guest VMs. SEV, SEV with Encrypted State (SEV-ES) [2], and Secure Nested Paging (SEV-SNP) [3] combine to deliver a layered and integrated confidential compute solution focused on hardware isolation primarily within the CPU.

The industry has quickly seized onto the value of confidential compute technologies and has sought new avenues to improve robustness, scalability, and performance in the evolving modern datacenter. One area of particular interest is the inclusion of trusted devices in the trust boundary of guests. Device I/O is central to important applications such as transaction processing, data analytics, artificial intelligence, and to applications that require large-scale data movement such as extract transform and load (ETL), snapshot, and backups, each of which underline the need for a secure yet performant device model that integrates with confidential guests.

As an example, cloud providers are increasing their use of SmartNICs for computational offload in guests. A SmartNIC is a guest programmable device that provides both network connectivity, storage, and programmable computational operations on the data it stores and transfers over the network. Financial institutions, health care providers, and research benefit greatly by this offload, but because their data has stringent confidentiality and integrity requirements, they have a responsibility to keep the data and the results of their workloads confidential. By expanding the trust boundary of a confidential guest to include a SmartNIC, the guest can better take advantage of their cloud provider's SmartNICs while also keeping their data and workloads protected.

Extending guests' trust boundaries to include devices requires adding new protection mechanisms to both host hardware and the device to help maintain guest confidentiality and integrity already provided by confidential compute technologies. On the host, the guest must be provided a means in which to establish trust with the device before the host includes the device in the guest's trust boundary. On the device, the workloads and the data of the guests must be sufficiently isolated from controls available to host software.

For the host-side support, AMD has developed SEV Trusted I/O (SEV-TIO), a new SEV technology. Through SEV-TIO, a guest can securely retrieve device identity and configuration information to establish trust in a device and its configuration. Once the guest establishes trust in the device, guests and devices can interact directly in private memory that can lead to I/O performance improvements.

To help advance the development of high-performance I/O in confidential guests, AMD has worked closely with industry partners over the past several years to develop and ratify the PCI TEE Device Interface Security Protocol (TDISP) specification [4]. The TDISP specification places requirements on devices to isolate guest data and workloads from host software, and it standardizes the way in which devices interact with confidential compute technologies in the host, such as the means to which guests

retrieve device identity and configuration information. Through SEV-TIO, any device that is TDISP capable can be securely bound to an SEV-SNP guest.

This whitepaper overviews the TDISP specification and describes how SEV-TIO integrates with the SEV-SNP technology and is designed to securely bind TDISP capable devices to guests.

Benefits of Trusted Devices

SEV-SNP allows a guest to separate its memory into shared and private memory. Shared memory is accessible to host software and is marked as hypervisor-owned in the Reverse Map Table (RMP), a data structure that stores the SEV-SNP security attributes of each page of memory in the system. Private memory is assigned to the guest in the RMP, writeable only by the guest, and encrypted with the guest's unique memory encryption keys. The guest uses private memory to store sensitive data and its executable code. The CPU and IOMMU both enforce the access control policy for guest private memory by checking the RMP as required during address translation to ensure that the software or device accessing memory has sufficient privileges.

The existing SEV-SNP architecture helps protect guests from maliciously programmed devices by treating all device accesses as if they originated from host software and are therefore untrusted. While direct device assignment to guests can be enabled by host software, this comes with limitations as the device can only operate within the shared memory space of the guest and cannot access the private memory of the guest.

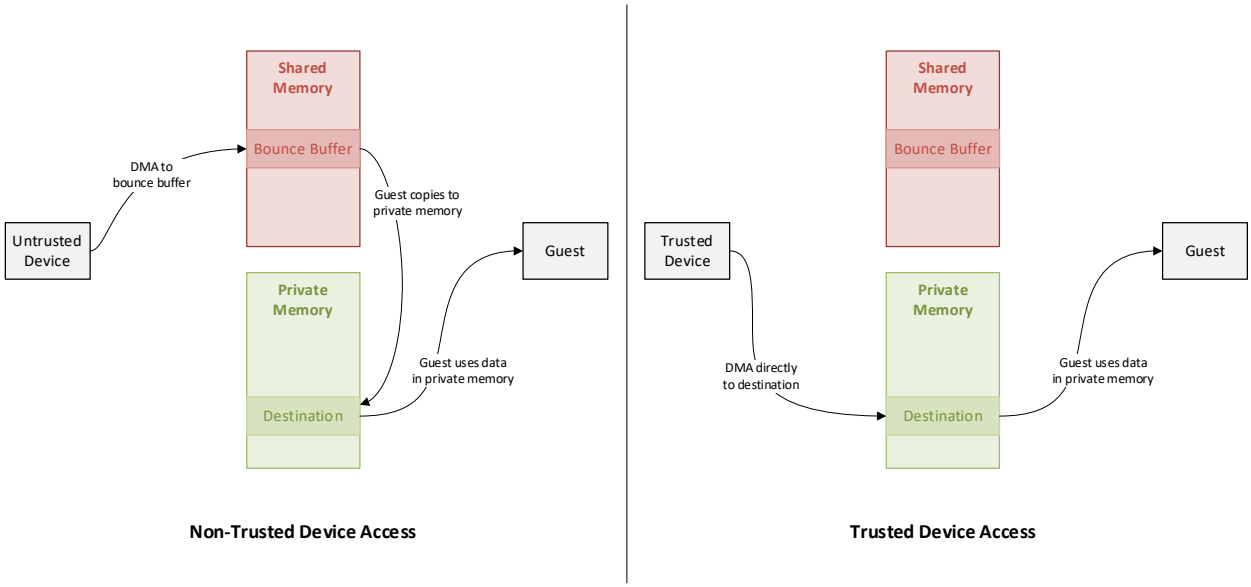

Figure 1: Diagram depicting bounce buffering required for a non-trusted device (left) and direct access to private memory by a trusted device (right).

If the guest needs data to flow between its private memory and the assigned device, the guest must copy the data in and out of a shared buffer that is accessible by both the guest and assigned devices. This method is called bounce buffering, depicted on the left portion of Figure 1, because the data bounces to and from the shared memory buffer. This may have performance impacts on I/O due to the extra memory movement required.

Further, because all communication between the guest and the device, including device register access, must occur through shared memory, all traffic is visible to host software. To protect the confidentiality and integrity of device communication with the guest, device specific protocols must be established between the guest and device. In some cases, this may leverage existing protocols such as software-encrypted encrypted storage, or network interface cards where all network traffic is encrypted between the software endpoints that are communicating and requires no additional support by the device. In other cases, protecting device communication is more complex such as graphic processing units (GPUs) or machine learning accelerator cards where the device must be involved in the protocol because it is processing plaintext data.

The new SEV-TIO technology can improve both performance and the security posture of device-guest communications. Fundamentally, SEV-TIO lets the guest choose whether it trusts a device enough to allow the device access to guest private memory. This has the potential to improve I/O performance by eliminating the need for bounce buffering device traffic within the guest, depicted in the right portion of Figure 1, and it limits the ability of host software to observe device-guest traffic.

Standardizing Trust in Devices

Over the last few years, AMD has worked with PCI SIG and industry partners to develop and ratify the TEE Device Interface Security Protocol (TDISP), a standard intended to address the need for trust in devices by guests in confidential compute environments. TDISP defines new protocols and functions of devices that enable them to authenticate themselves, prevent traffic interception or masquerading on the PCIe fabric, attest to their configuration, and isolate guest workloads from device controls available to host drivers.

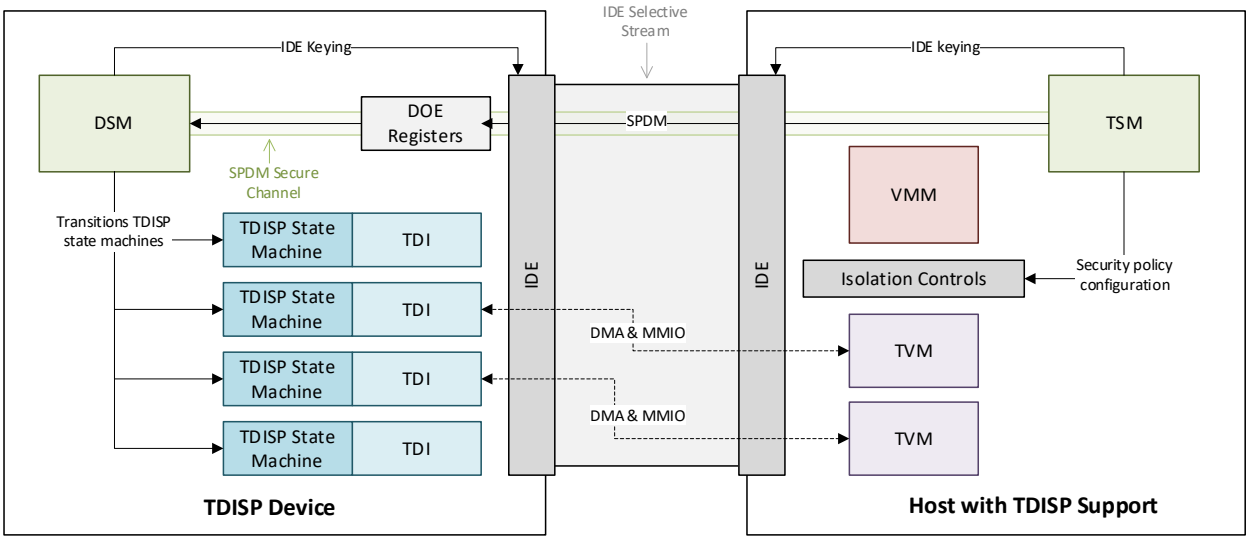

Figure 2: System diagram of a TDISP device connected to a host capable of supporting TDISP devices.

As depicted in Figure 2, the TDISP architecture describes a device as comprising a Device Security Module (DSM) and one or more TEE Device Interfaces (TDIs) that can be securely assigned to Trusted Virtual Machines (TVMs), referred to as guests throughout this white paper. The DSM is responsible for managing the TDISP security configuration of the device overall and of the security state of each TDI. The DSM communicates with the host TEE Security Module (TSM) which is responsible for configuration

of the host isolation controls protecting guests from the Virtual Machine Monitor (VMM) and other host software. The TSM also drives the lifecycle of a TDISP enabled device through the configuration, binding, and unbinding of TDIs to guests.

A natural implementation of TDISP is with an SR-IOV capable device. The host driver controls the Physical Function (PF) of the device, and each of the Virtual Functions (VFs) assigned to guests are the TDIs of the device. In this configuration, the TSM manages the TDISP states of each of the VFs and is responsible for programming the host hardware on behalf of the guest to allow the device to access guest private data.

The TSM and the DSM communicate via the Secure Protocol and Data Model (SPDM) protocol defined by the DMTF [5]. SPDM is a request-response message protocol and is the control path through which the TSM queries and manage the TDISP feature on the device. The TSM and DSM protect the SPDM connection by negotiating keys and establishing a SPDM Secure Messages [6] session which encrypts and authenticates all SPDM messages.

While the TDISP control path between the DSM and TSM is protected by SPDM Secure Messages, the data path of a TDISP device to the guests it serves is protected with the PCI Integrity and Data Encryption (IDE) protocol [7]. IDE encrypts and authenticates all device traffic in an end-to-end stream where only the root port and the TDISP device possess the IDE stream keys which prevents intervening PCIe switches, physical attackers, and maliciously designed devices on the PCIe fabric from mounting man-in-the middle or masquerading attacks on fabric traffic. Any bad actors in the fabric will only see ciphertext and cannot alter the stream without detection by the device and root port. The TSM and the DSM negotiate the keys of the IDE stream through their shared SPDM channel.

TDISP defines isolation requirements that ensure devices protect guest data and workloads while within the devices themselves. The most prominent of these requirements is the TDISP state machine that each TDI implements. The TDISP state machine places restrictions on TDIs based on the state that they are in. For instance, the configuration of a TDI, such as its Base Address Registers (BARs), cannot be altered while the TDI is in the Locked state. This gives the guest an opportunity to examine the configuration of the TDI to establish trust without interference by host software. The TDISP state machine is discussed in greater detail in the following sections.

Finally, TDISP defines several data objects that may be consumed by the guest to help establish trust in the device. First, the guest can retrieve the certificate chain of the device which securely identifies the device and its manufacturer. Second, the device offers an attestation report that contains device specific measurements of its configuration. The guest can examine the attestation report to, for instance, determine whether the firmware loaded into the device is a sufficiently new version. Finally, each TDI has an interface report that describes the configuration of the TDI including information about where its memory mapped I/O (MMIO) registers are mapped into memory. The sections below discuss in further detail how each of these objects are used by the guest.

Overall, TDISP provides a standard interface between the host and devices to accomplish the security goals of confidential compute. With this standard interface, host implementations can support trusted I/O for any device that implements TDISP which AMD believes will help accelerate the adoption of secure I/O in confidential compute environments within the hardware and software ecosystems.

SEV-TIO leverages both the security controls of SEV-SNP and the standardized interfaces of TDISP to provide a comprehensive trusted I/O solution to confidential compute guests. In SEV-TIO, the AMD Secure Processor (ASP) serves as the TSM of the host and is responsible for managing the lifecycle of TDISP devices as they operate with SEV-SNP guests.

SEV-TIO Architectural Overview

Multiple hardware and firmware components are involved in SEV-TIO, including the ASP, IOMMU, and PCI root complex. Figure 3 illustrates the overall architecture of these components and their relationships with one another.

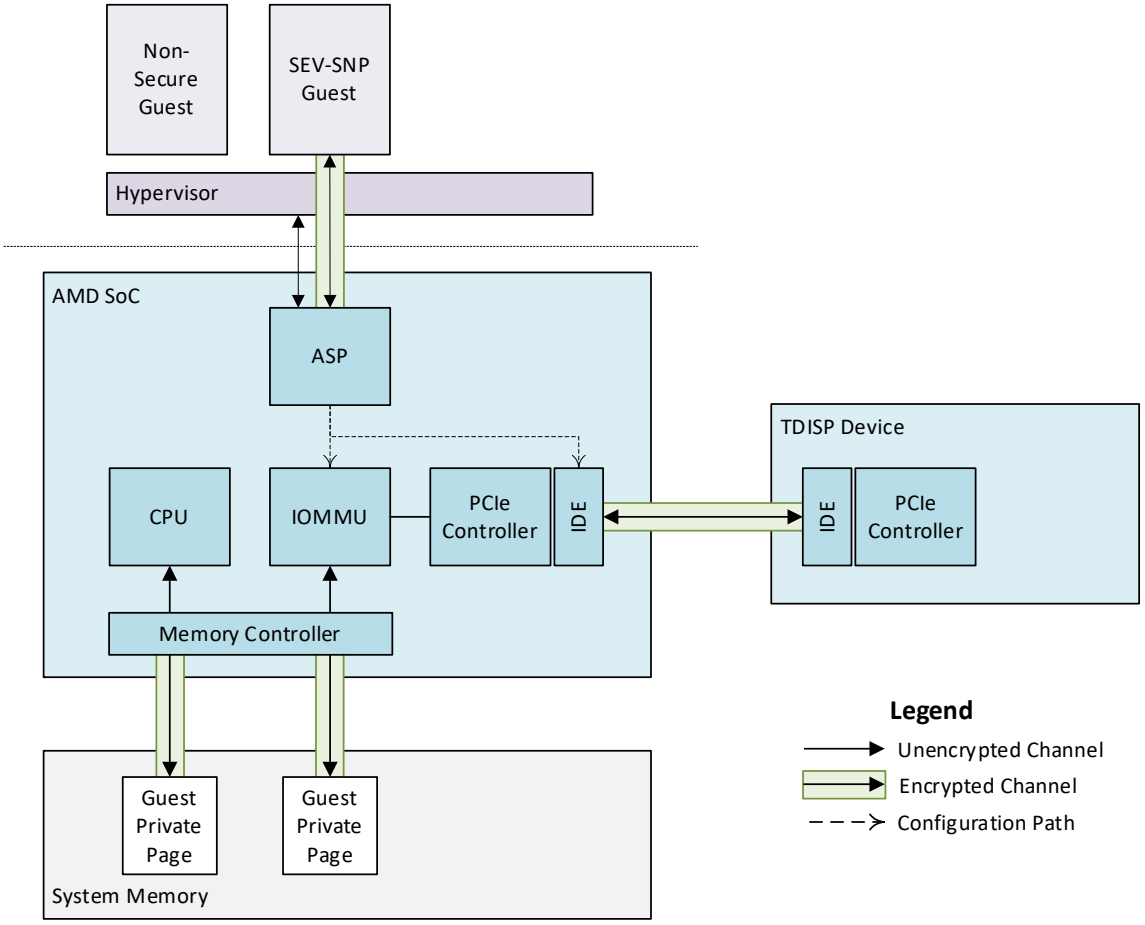

Figure 3: High level illustration of the system components that comprises SEV-TIO.

The AMD Secure Processor (ASP) hosts the SEV firmware which plays a central role in orchestrating the lifecycle of secure guests. SEV-TIO brings new commands and guest request messages to configure the IOMMU, the PCIe root complex, and the architectural data structures necessary to bring TDIs into the trust boundary of guests. Further, to serve the role of TSM, the ASP also implements an SPDM responder which communicates with the DSMs of TDISP devices.

As with SEV-SNP today, the IOMMU is responsible for address translation and performing RMP checks on DMA to protect the confidentiality and integrity of SEV-SNP guests. SEV-TIO enriches the RMP checks

performed by the IOMMU to allow devices to access guest private memory directly after the guest indicates that it trusts the device and its configuration.

Finally, SEV-TIO adds support to the PCIe controller to construct IDE streams for the purpose of protecting the confidentiality and integrity of guest data over the PCIe fabric between the root complex and the TDISP device. IDE streams also authenticate traffic to detect malicious agents on the PCIe fabric attempting to masquerade as a trusted device or as the root complex.

The following sections describe in greater detail how the ASP, IOMMU, and IDE streams in the PCIe controller work together with guests and TDISP devices to bring the device securely into the trust boundary of the guest.

Trusted Devices in Guest Private Memory

Guests interact with TDIs through two major data paths: Direct Memory Access (DMA) by the device to guest memory, and memory mapped I/O (MMIO) accesses by the guest to device registers. SEV-TIO provides a means to isolate DMA and MMIO which helps provide confidentiality and integrity protection of I/O traffic between the guest and its TDIs.

TDIs access memory via either guest virtual address (GVA) space or guest physical address (GPA) space. The I/O Memory Management Unit (IOMMU) in the host hardware is responsible for translating the provided GVAs or GPAs into system physical addresses (SPAs). Because SEV-SNP enforces access control at the time of translation, the IOMMU performs RMP entry lookups on translation.

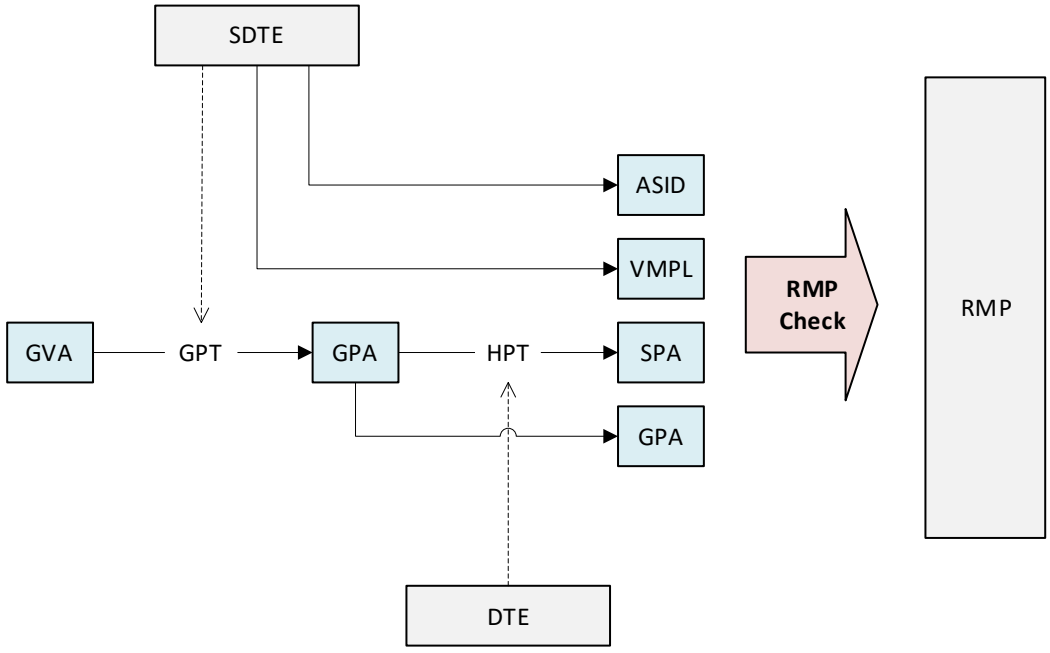

Figure 4: RMP check on DMA performed by the IOMMU during translation.

In SEV-TIO, The IOMMU tracks the security attributes of each TDI in a new data structure called the Secure Device Table (SDT) which is an array of SDT entries (SDTEs) indexed by the Requester ID (RID) of the TDI. When the IOMMU receives a request from a device, it looks up the ownership information in

the SDT based on the RID of the request. The SDT is not writeable by the hypervisor and is configured instead by the ASP.

The SDTE of a TDI contains security attributes necessary for the IOMMU to determine whether the TDI is allowed to access guest private memory. The security attributes include the guest and the Virtual Machine Privilege Level (VMPL) which is a privilege hierarchy within guests to support SEV-SNP functionality. On translation, the IOMMU performs the same RMP checks that CPU performs. It looks up the RMP entry of the SPA and performs an RMP check where bound TDIs are treated as guest accesses from the guest and VMPL as specified in the SDTE for the TDI.

The SDTE additionally contains guest-provided configuration for the IOMMU including the paging mode and location of the guest's page tables (GPTs) for the TDI. The guest programs the fields of the SDTE for its bound TDI through an SEV-SNP guest request message, which is a cryptographically protected interface between the guest and the ASP. Note that host software still uses the existing Device Table for host-controlled configuration of the TDI, such as the location of the host page tables (HPTs) for the TDI.

Guests can interact with TDIs via their memory mapped I/O (MMIO) registers, which in SEV-TIO are mapped into the guest as private memory. Like any guest private memory access, the CPU table walker consults the RMP to determine whether the access originated from the guest that is bound to the TDI. When an MMIO access to a guest private page passes the RMP checks, the access is routed to the TDI via the PCIe fabric.

An important security control of SEV-SNP is the Validated bit in the RMP. Each page tracked by the RMP has a Validated bit that can be set by the guest to indicate that it has started using the page. If the hypervisor alters the RMP entry for the page, the Validated bit is cleared and the hypervisor cannot set it back. Any guest access to a page with the Validated bit clear will cause an exception in the guest. This ensures that once the guest sets the Validated bit and begins using the page, the hypervisor cannot alter the RMP entry of the page.

Today, guests do not have a means for setting the Validated bit on MMIO ranges like they can with the PVALIDATE instruction for data pages. SEV-TIO introduces an interface to the ASP that requests the ASP to set the Validated bit on behalf of the guest. Once the MMIO pages have the Validated bit set, the CPU performs the RMP checks as usual.

Before the guest asks the ASP to set the Validated bit, the guest must be certain that the pages map to the registers of the TDI. This is accomplished as part of device attestation discussed below.

Hardware Accelerated Virtualized IOMMU

Conventionally, the hypervisor is responsible for emulating IOMMU behavior to a guest. When a guest needs to send a command to the IOMMU, the hypervisor intercepts that access and submits the request to the IOMMU on behalf of the guest. To improve performance of the guest IOMMU access path, AMD offers a Virtualized IOMMU (vIOMMU). A vIOMMU is a virtual interface to the IOMMU's command buffers, event log, and Peripheral Page Request (PPR) log. Through this interface, the guest interacts directly with the IOMMU instead of relying on hypervisor emulation.

In SEV-TIO, guests are required to use the vIOMMU because removing the hypervisor from the communication path between guest and IOMMU is necessary for the isolation of I/O device

management within a confidential compute environment. For instance, when a guest needs to invalidate the IOMMU address translation cache, which may be a sensitive security operation, the guest can directly use the vIOMMU command buffer to ensure the invalidation occurs.

Because the guest interacts with the vIOMMU via MMIO, the guest needs assurance that the MMIO ranges given to it by the hypervisor are in fact the vIOMMU that was assigned to the guest. To accomplish this, the hypervisor and guests use a protocol to bind a vIOMMU to a guest that is similar to how TDIs are bound to guests.

Device Attestation

Before guests configure the IOMMU to allow a TDI to access their private memory, they must establish trust in the identity and configuration of the device. SEV-TIO provides three data objects used by the guest for attestation: the certificate chain of the device, the attestation report of the device, and the interface report of the TDI.

The certificate chain of the device consists of one or more documents called certificates that contain a public key and metadata about the owner of the key. Each key represented in the chain signs the certificate of the next key in the chain which confers trust. The first certificate in the chain, the root certificate, represents the identity of a root Certificate Authority (CA). When a guest validates a certificate chain, it starts with the certificate of the root CA. If the guest trusts the root CA, then the guest verifies that the signatures on each of the chain certificates are valid.

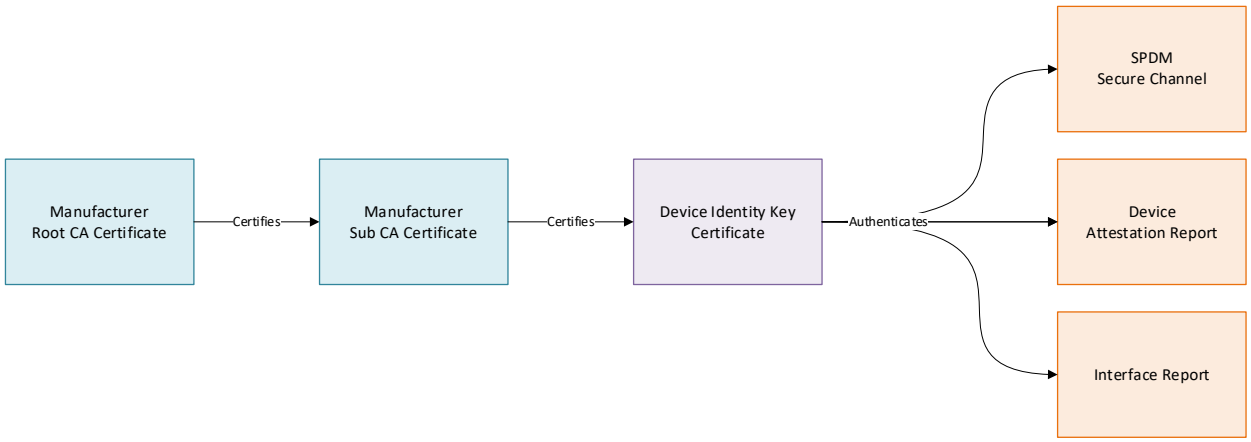

Figure 5: An example of a certificate chain where the manufacturer maintains a two-level certificate authority that certifies the device identity keys of each of its devices.

TDISP uses certificate chains to authenticate device identity. The ASP and DSM establish the secure SPDM connection by using the identity key in the device's certificate chain. The ASP gives the guest the certificate chain who then validates the chain to ensure it trusts the device. It is up to the guest to determine how to validate the certificate chain. For instance, a device manufacturer can maintain its own CA that signs device keys, and then guests could trust the manufacturer's CA to authenticate devices by placing the root CA certificate in the guest's trust store. Alternatively, the guest can delegate the validation of certificates to a remote party who has the authority to approve which devices the guest trusts.

Once the guest has authenticated the device, it must determine if the device is securely configured. Through SPDM, TDISP provides a means for a device to present to the guest an attestation report which is a collection of measurements of device configuration. The attestation report format is defined by SPDM and by the device vendor. Measurements contained within the attestation report may include cryptographic digests of currently loaded firmware and manifests of configuration data that affect the security of guest workloads and private data on the device. In SEV-TIO, the ASP provides an interface for the guest to securely retrieve the attestation report of the device.

Finally, the guest needs to determine whether the hypervisor has configured the TDI correctly. TDISP defines a data object called the interface report that contains MMIO mapping information, security attributes of MMIO ranges, and optionally device specific information. In SEV-TIO, the guest may use the interface report to determine whether the hypervisor mapped the MMIO ranges into guest private memory correctly.

For instance, the interface report could help the guest detect that the MMIO ranges were mapped out of order, mapped with gaps, or mapped to completely different devices. Once the guest has concluded that the MMIO ranges are configured as expected, the guest sets the Validated bit on each page of the TDI's MMIO ranges via the ASP as described above.

Critically, the above attestation steps are not useful to a guest if the configuration of the TDI can be altered during the attestation process. To address this, TDISP enforces restrictions on what configuration can change according to the current state of the TDI. This state machine and its related security controls are described in the next section.

TDISP State Machine

One of the fundamental aspects of the TDISP design is the TDISP state machine. Each TDI has its own state machine that governs its behavior. The TDISP state machine comprises the following states: Unlocked, Locked, Run, and Error. The TDISP state machine is depicted in Figure 6 below.

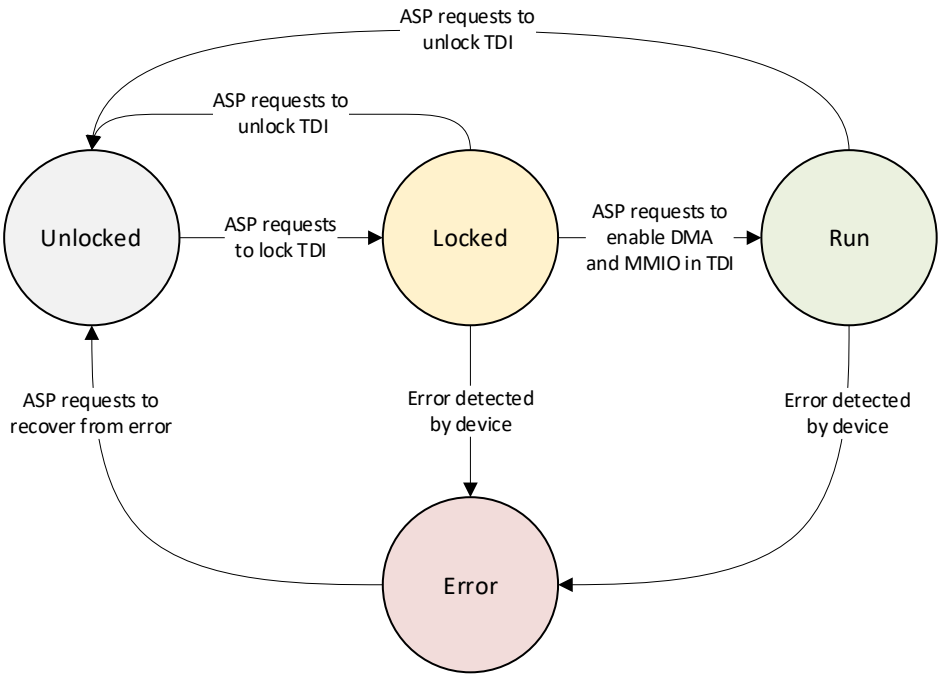

Figure 6: Diagram of the TDISP state machine that controls the behavior of each TDI.

A TDI begins in the Unlocked state. While a TDI in the Unlocked state can be arbitrarily configured and reconfigured, the TDI cannot be bound to a guest or access guest private memory. However, TDIs in the Unlocked state can be assigned to guests that cannot or do not want to support TDISP, including legacy guests that do not support SEV-SNP at all.

To prepare a TDI for guest binding, host software can ask the ASP to request the DSM to transition the TDI to the Locked state. In this state, any configuration that could impact the security of guest private data is either made read-only or will cause a transition to the Error state if changed. This allows the guest to review that configuration through the device attestation process described above and make the determination of whether the device is known to the guest and is trusted. While in the Locked state, the TDI cannot be used and cannot perform DMA requests.

Once the guest establishes trust in the device, the ASP asks the DSM to transition the TDI to the Run state. Like the Locked state, the security sensitive configuration of the TDI cannot be altered. However, now that the TDI is in the Run state, the guest can interact with it by programming its registers and performing reads and writes via DMA.

Finally, the TDI may transition to the Error state if an error condition is detected that impacts the TDIs ability to protect the guest's private data. TDISP also defines several error conditions, such as reprogramming the Base Address Registers (BARs) of the TDIs that affect the location of device registers in system physical address space. TDISP also allows the device vendor to define other error conditions that would send the TDI into the Error state.

To unbind a TDI after a guest is finished with it, or to recover from a TDI in the error state, host software can ask the ASP to unbind the TDI from the guest which performs the necessary cleanup of system resources and ask the DSM to return the TDI safely to the Unlocked state. If the TDI contains any guest

private data on this transition, the TDI is required by TDISP to purge it from the device before transitioning to the Unlocked state. Once in the Unlocked state, the TDI can be reassigned and bound once more.

Device Communication with AMD Secure Processor

The TDISP specification requires that the TSM of the host communicate with the DSM of the device using the SPDM protocol over the PCI Data Object Exchange (DOE) mailbox protocol. The DOE mailbox is part of the PCI configuration space of function 0 of the device which is controlled by host software.

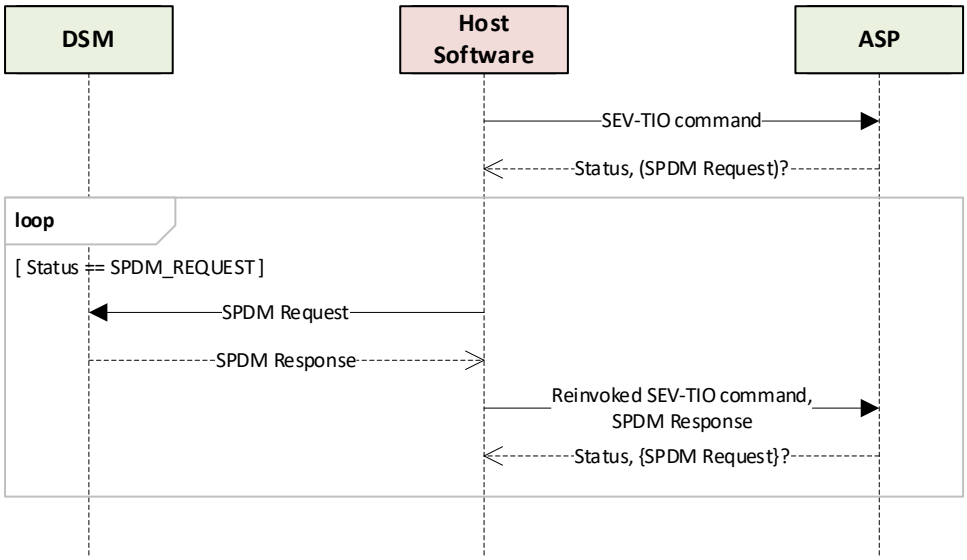

Figure 7: Host software initiating SEV-TIO commands and transporting subsequent SPDM messages to the DSM until the command completes.

In SEV-TIO, the ASP shares the DOE mailbox registers with host software by requiring host software to transport SPDM packets on its behalf as depicted in Figure 7. The ASP constructs SPDM request packets and provides them to host software. Host software writes the packets into the DOE registers and returns the device's SPDM response back to the ASP.

The security of the SPDM connection between the ASP and DSM is assured by the SPDM Secure Messages protocol. Before performing any actions on the device, the ASP negotiates SPDM Secure Messages keys with the device.

The new ASP commands introduced in SEV-TIO may need to generate one or more SPDM requests to complete the command. Host software does not need to know anything about which commands might generate SPDM requests and instead transports request packets until the ASP informs software that the command has completed.

Lifecycle of a Trusted Device

TDISP devices proceed through a common SEV-TIO lifecycle as they are enumerated and initialized, assigned and unassigned, and reset. Host software and guests communicate with the ASP to drive a TDISP device and its TDIs through the SEV-TIO lifecycle. Figure 8 depicts this lifecycle flow.

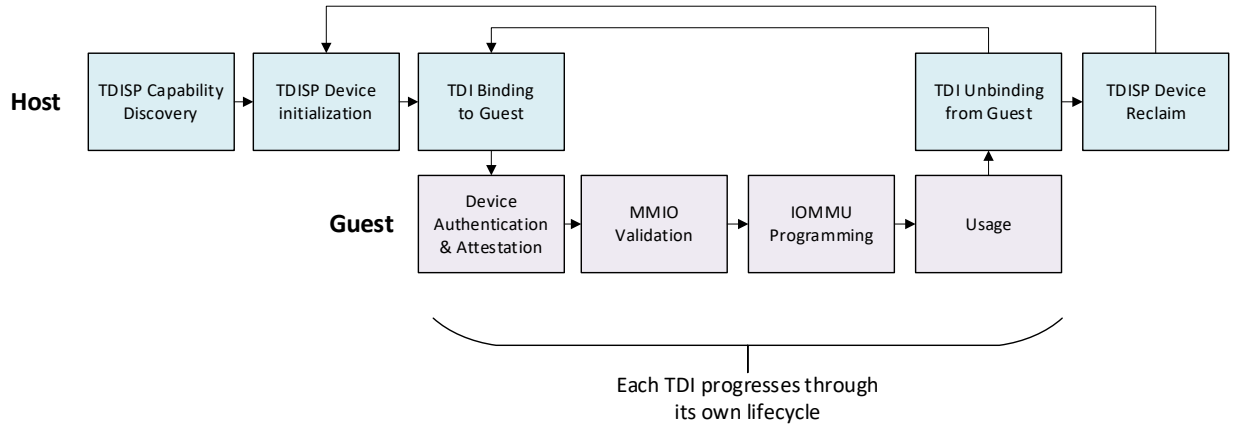

Figure 8: SEV-TIO lifecycle of a TDISP device and each of its TDIs.

During host device enumeration, host software detects that a device supports TDISP through the device's capability registers. At this point, the host software can ask the ASP to enable TDISP on the device by establishing an SPDM connection with the device and construct the IDE streams. The host then initializes the device to the desired configuration, such as by using an SR-IOV device's PF to construct VFs and assigning them device resources.

At TDI assignment, SEV-TIO requires host software to bind the TDI to the guest through the ASP which prepares the TDI for device attestation by transitioning the TDI to the TDISP Locked state. In the Locked state, any configuration that alters the security posture of the TDI is locked and cannot be altered without causing the TDI to transition to the Error state. As discussed in previous sections, when the TDI is in the Locked state, the guest can securely retrieve device attestation information from the ASP and establish trust in the TDI. Once trust is established, the guest can send guest requests to the ASP to grant the TDI access to private memory by programming the SDTE of the TDI and setting the Validated bit in the RMP entries of the MMIO ranges of the TDI.

Next, the ASP asks the DSM to transition the TDI to the TDISP Run state. In this state, the TDI is now allowed to accept MMIO requests and send DMA requests. Note that while in the Run state, the TDI still prevents configuration changes by host software that would alter the security posture of the device.

Finally, the guest sends a guest request to the ASP to program the SDTE of the TDI with its configuration such as the VMPL that the TDI has access to, IOMMU paging modes, and pointers to the guest's page tables. After this completes, the TDI has access to the guest private memory granted to it by the guest.

The hypervisor may choose to unbind a TDI from a guest at any time for several reasons. The guest may have finished using the TDI and requested that the TDI be unbound, or the guest may be shutting down and the hypervisor is reclaiming the guest's resources. The hypervisor may also choose to reclaim the TDI because the guest or device is hung or misbehaving. In these cases, the hypervisor may ask the ASP to unbind the TDI from the guest. To do this, the ASP unwinds the binding configuration within the

hardware and sends a TDISP command to the DSM to transition the TDI back to the Unlocked state. Once the TDI is in the Unlocked state, the host can assign and bind the TDI to other guests in the same manner as above.

The unbinding operation is designed to robustly address several error conditions. For instance, it requires no interaction with the guest to unbind a TDI in case the guest is unresponsive. Also, the unbind operation will still free system resources if the TDI is not responsive or even if the device is no longer physically present. Finally, if the TDI is in the Error state due to an irrecoverable error condition, the unbind operation transitions the TDI to the TDISP Unlocked state.

Finally, all system resources associated with the TDISP device itself, such as the SPDM session context and tracking of all its TDI state, can be reclaimed with a request to the ASP. After reclamation of the TDISP resources, the IDE stream and SPDM session will be closed. To use the device with SEV-TIO again, the ASP must reconnect to the device and reestablish SPDM and IDE.

Conclusion

SEV-TIO enables support for trusted TDISP capable devices with AMD SEV-SNP guests, allowing directly assigned devices to access guest private memory. Through SEV-TIO, a guest can establish trust in the identity and configuration of a device before bringing the device into the trust boundary of the guest. As a result, SEV-TIO creates a direct path between a guest and device which has the potential to improve I/O performance for a variety of workloads. AMD looks forward to working with our ecosystem partners to enable this new technology and expand the potential of confidential computing with AMD SEV.

References

[1] Advanced Micro Devices, "AMD Memory Encryption," 2021.

[2] Advanced Micro Devices, "Protecting VM Register State with SEV-ES," 2017.

[3] Advanced Micro Devices, "AMD SEV-SNP: Strengthening VM Isolation with Integrity Protection and Mode," 2020.

[4] PCI-SIG, "TEE Device Interface Security Protocol (TDISP)," 2022.

[5] DMTF, "Security Protocol and Data Model (SPDM) Specification," 2022.

[6] DMTF, "Secured Messages using SPDM Specification," 2022.

[7] PCI-SIG, "Integrity and Data Encryption (IDE)," 2022.