Analyzing the Security of vTPMs with Intel TDX

Sebastian Gajek
sebastian@enclaive.io
enclaive.io
Berlin, Germany

Sören Langenberg
soeren.langenberg@stud.hs-flensburg.de
University of Applied Science Flensburg
Flensburg, Germany

Abstract

Addressing the security risks of running virtual machines in a cloud environment has led to the development of Trusted Execution Environments like Intel TDX, AMD-SEV or ARM Trustzone. However, these technologies are only able to guarantee that the hardware and firmware were not manipulated. In order to establish proper trust into a virtual machine that was launched it is important to guarantee that other parts of the software stack were not manipulated during configuration and boot time. For this an additional root of trust is needed which is typically fulfilled through a Trusted Platform Module. However, hardware based Trusted Platform Modules cannot be considered for a virtualized environment due to their limited space for measurements of different virtual machines. This work shows how Intel was able to solve this problem by introducing a virtualized Trusted Platform Module which utilizes the isolation properties that are provided through Intel TDX.

Keywords: TDX, Confidential Compute, TPM, vTPM

1 Introduction

The trend of moving software to the cloud or using Software as a Service (SaaS) and at the same time the need for stronger data and computation protection has led to the development of Trusted Execution Environments (TEE). These TEE provide a secure CPU bound execution environment which secures the data at runtime. Together with the encryption of data in transit through means like TLS and encryption of data at rest through full disk encryption or similar the TEE enables encryption of user data at every stage of computation. In order to achieve the confidentiality of data in use the processors are equipped with high quality cryptographic material from the manufacturer. The hardware bound cryptographic material is subsequently used to attest that the machine is indeed providing the promised security features and protects the data in use.

The attestation typically includes the trusted components of the underlying software and hardware stack to protect against adversaries.

This means that the trusted components shown in fig. 1 are the Platform (but only the CPU), and the Intel TDX Module as well as each Trust Domain in itself. Only these components are part of the attestation as shown in [23]. However, it is also a good idea to extend the measurements to include the kernel and other subsequent components in the boot chain as

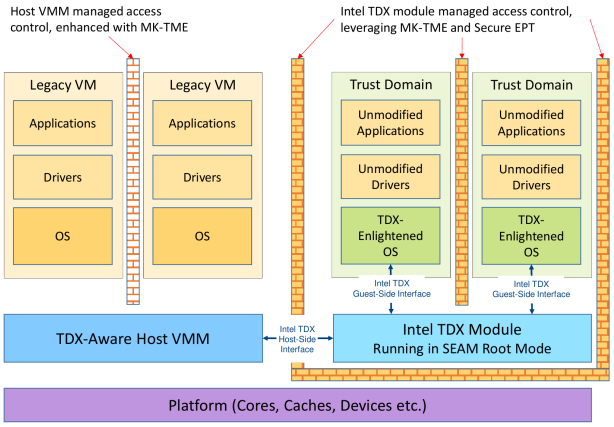

Figure 1. Intel TDX Components Overview [3]

this especially helps to protect in scenarios where the hoster of the VM is seen as untrusted but is providing a VM image to the user. To allow the trusted firmware to measure the content of the kernel and subsequent components Trusted Platform Modules (TPM) can be used. In order to efficiently scale the TPM is also virtualized. Therefore, the specifications and the functionality of the TPM are implemented in a software module that is typically managed by the Virtual Machine Manager. Creating these virtualized TPMs (vTPMs) allows to provide each VM with the functionality of a TPM without having to buy a dedicated chip for each VM.

1.1 Previous Work

CoCoTPM short for Confidential Computing TPM is a proposed vendor neutral design for the usage of TPMs in Confidential Computing VMs. The concept generally involves a separate CoCoTPM VM and a Kernel and UEFI Level CoCoTPM driver to allow communication between the CC-VM and the CoCoTPM VM [21, 3]. As the proposed concept is generally vendor neutral no specifications are made available on how the VMs should communicate the encrypted TPM commands to the Hypervisor or the other VM. However, the basic concept of spawning an additional VM which implements the TPM functionality and provides the TPM functionality to another VM via a secure channel and encrypted TPM commands is the same as the implementation for Intel TDX.

A non vendor neutral design has been proposed in [18] and is focused on AMD-SEV. This design uses the Secure

VM Service Module which is a guest communication interface that allows to offload workloads to another VM which runs with more privileges (Virtual Machine Privilege Level (VMPL) is a new concept introduced by AMD which blocks certain instructions for guests with lower privileges). Using the VMPL for isolation in order to securely pass the execution of a certain command or feature to a VM with more privileges allows eliminating the need for extending the TCB with additional protocol implementations like SPDM or TLS.

1.2 Contributions

This paper will explore the implementation of this technology in Intel TDX to see if vTPMs are a valid solution in order to establish a completely trusted chain.

2 Background

2.1 Intel TDX

Intel TDX short for Intel Trust Domain Extensions is an extension to the CPU architecture the Intel Xeon Scalable Processors since their 4th generation[9]. This technology implements as mentioned in the beginning a trusted execution environment where Virtual Machines are hardware isolated from the Virtual Machine Monitor (VMM), hypervisor and other software. In order to ensure the complete isolation of the protected VMs (Trust Domain or TD) the CPU architecture has been extended through a software based component called the TDX Module as well as through extensions to x86 instruction set and other architectural changes.

2.2 Security Model

In the case of Intel TDX—an evolution of Intel SGX— an already prevalent technology limited to single software applications instead of VMs, two different adversaries are considered.

The first adversary is called a System Software Adversary which could be a compromised VMM, SMM, BIOS or other VMs. This adversary basically covers every software based attack vector that that is not enabled through the TD itself or the firmware of the processor[5, p. 8]. A System Software Adversary is therefore able to read and write the complete system memory as well as access to the CPU state and the registers of the CPU. Therefore, the attacker has the ability to rewrite the page table to cause page table faults or remap/replay the memory pages of the TD.

The second adversary that is being considered is allowed to launch hardware attacks against Intel TDX. This could be the Cloud Provider trying to extract information through attacks on the memory bus or other interfaces. As this attacker has complete physical access to the hardware he is allowed to launch attacks like DRAM freeze or inject content into the DRAM through physical means[5, p. 8].

As already mentioned in the introduction considering an attacker which is allowed to use all these different hardware and software based attacks drastically reduces the amount of trusted components. In the Intel TDX architecture generally the CPU and its firmware/microcode are trusted components as executing programs without placing trust in these components is not possible. Additionally, the TDX Module is a trusted component due to its crucial role in relaying request from the TD to the hardware. TDs are generally considered a trusted component in the software stack from the perspective of a user, however other TDs are seen as untrusted as the user has no knowledge about the software that is running on these TDs.

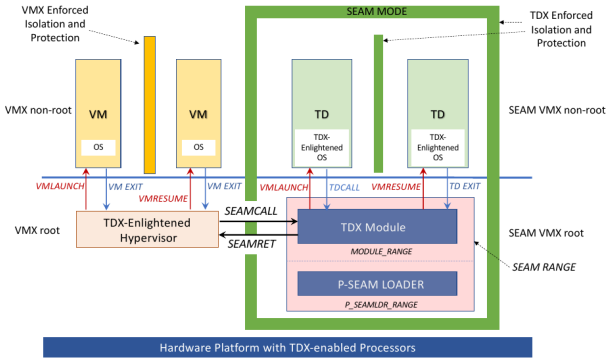

Figure 2. Intel TDX Architecture via [1]

2.3 TDX Module

The core of the new system architecture is the new TDX Module with its loader as it can be seen in fig. 2. The TDX Module itself is loaded as part of the boot chain through the P-SEAM Loader which is an Intel Authenticated Code Module (ACM). These special code modules are loaded and verified by the CPU early in the boot process to ensure that they have not been manipulated. Execution of these ACMs happens in on-chip memory which is called CRAM by Intel. In addition to the TDX the hardware virtualization technique Intel VT was extended to cope with Trusted Domains. In a classical scenario Intel VT operates in two modes the VMX root mode and the VMX non-root mode. The VMX root mode allows privileged components like the hypervisor to access specific calls of the virtualization technology in order to control its guests. The guest VMs typically run in the VMX non-root mode which introduces the Virtual Machine Control Structure (VMCS) which controls the CPU operations and can control how certain instruction behave. The VMCS allows to safe the state of VMs and therefore enables the pausing and switching between different VMs. The non-root mode also separates the address space between the hypervisor and the guest which prevents memory clashes[19]. For Intel TDX Intel VT got extended with a new mode called Secure Arbitration Mode (SEAM) which is responsible for the execution of TD. The Intel TDX Module and its loader are on

the same privilege level as the hypervisor and are therefore executed in the new SEAM VMX root mode, protected VMs are executed in the SEAM VMX non-root.

As it can be seen in fig. 2 the TDX Module has the same privilege level as the hypervisor and the communication with TDs to the hypervisor and reverse is routed through the TDX Module. In its role as the major trust anchor for the TDs the TDX Module ensures through this communication flow that request that come from the hypervisor do not pose a thread to the TDs or that TD request do not harm the TD. During the operation of VMs it is normal that the VMs are exiting in order to be resumed at a later stage or to request additional devices or information from the hypervisor. To request device emulation or additional information from the hypervisor and context switch from the SEAM mode to the normal operation mode is required. The TDX Module ensures in this scenario that the TD which just exited does not leak information to the untrusted hypervisor.

2.3.1 Memory Encryption and Integrity. In order to protect against a malicious hypervisor and other malicious VMs not only the communication between the hypervisor and the VM needs to be checked and secured, but also the resources used by the TD. Therefore, the memory which is used by a TD is encrypted using the Intel Multi Key Total Memory Encryption (MKTME) technology. During the creation of a TD a unique ephemeral encryption key is generated which is used to encrypt and decrypt the memory pages that belong to the TD. As the encryption key is generated by the CPU it is not visible to the software and can not be exchanged[2]. The protection of the memory is also extended through an integrity method as the encryption algorithm AES-XTS itself does not have any integrity measurements. Memory integrity protection is achieved through two different methods. The first method ensures that when a TD makes a request to retrieve content from its memory that the memory page that has been requested is indeed a private memory page. This method is called Logical Integrity (LI) and only provides limited security[3, p. 120]. The second method Cryptographic Integrity (CI) provides an additional MAC. This MAC is computed for each cache line together with the encryption tweak key and the TD owner bit with HMAC-SHA3-256 as the underlying algorithm[5, p. 3]. Afterwards the MAC is shortened to 28 bits and stored in the ECC region of the memory. This provides the necessary isolation and protection against a malicious hypervisor.

2.3.2 Remote Attestation. Another important concept in the Intel TDX architecture is the possibility of remote attestation. Attestation is a process where a server can prove to a challenger via a secure channel that certain configuration parameters are met and that the server is trustworthy. In the context of confidential cloud computing this is an essential feature as the trusted resources are outside the control of the user. To achieve remote attestation the user first needs to establish a secure connection (SSH, TLS etc.) to a TD. Afterwards a special CPU instruction can be invoked which creates a report for that specific TD. The report includes information about the Trusted Computing Base (TCB) and other measurements about the TD like the virtual firmware which were measured during the building process of the TD. The generated report is protected through the usage of a MAC which has been generated with a key that is only known to the CPU itself. In a second step the report is passed to the quoting enclave (QE), which is already present since the introduction of Intel SGX. A Quoting Enclave is able to verify that the report that has been sent to it was generated on the same platform through a secure manner and has not been manipulated. Afterwards the Quoting Enclave will transfer the report into a Quote which can be verified off platform by a different entity. The off platform verification can be achieved because the quote is through the private key of the quoting enclave which in turn is signed through the Provisioning Certification Enclave (PCE). Signing the key of the QE needs to be done because in the Data Center Attestation Primitives (DCAP) model a vendor can specify its own QE. However, in order to ensure the hardware root of trust the QE needs to be signed through an Intel controlled and signed enclave which is the PCE. The PCE is the only enclave allowed to retrieve a key for signing other enclaves which is derived from the hardware embedded keys[6]. Through this chain of signatures the challenger can contact an Intel service which has retained the hardware embedded key and ask it to verify that the signatures of the generated quote are correct. Verifying the signature chains gives the challenger the security that an authentic Intel TDX enabled platform was used and therefore that the collected measurements are correct. Now it is up to the challenger to interpret the measurements to conclude if the platform meets the expected security level or not.

The remote attestation however usually covers only the initial firmware of the VM and the platform information[23]. Intel TDX is also able to cover additional information through the usage of 4 Runtime Measurement Registers (RTMR)[23]. The RTMRs can be used to also cover the kernel as well as initrd and the whole userspace. Covering the userspace and all applications makes the measurement unflexible and depending on the size of the installation may also have additional performance impacts[23]. Therefore, one approach is to measure only the platform and firmware and afterwards use firmware based attestation together with Trusted Platform Modules, more on that in section 2.4 and section 3.

2.4 TPMs and vTPMs

A Trusted Platform Module short TPM is defined as a hardware root of trust which is recognized as more secure than a pure software implementation[17, p.1]. The specifications of the TPM are developed by the Trusted Computing Group and are today widely in use by different technologies like Intel

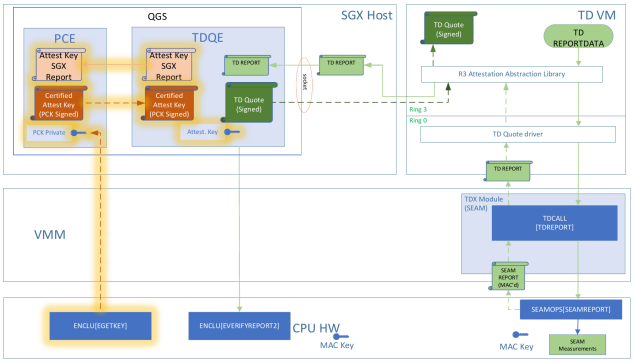

Figure 3. Intel TDX Attestation Architecture [6]

TXT[7, p.15] or Microsoft Bitlocker[11]. As part of its special position the TPM provides security capabilities like random number generators, key generation and storage as well as different cryptographic algorithms. In order to provide these functionalities the TPM is equipped with different crypto engines that can provide through a combination of hardware and software cryptographic engines that are responsible for asymmetric/symmetric operations or hash operations and other means[16, p. 69]. The TPM is also able to store its own secrets in Non-Volatile memory while storing measurements or other received information in Volatile memory[16, p. 69]. As the TPM is typically connected through a bus with the system the data in transit on this bus should be protected. For this the TPM supports the possibility to encrypt the communication on the bus through a shared session secret[14, p. 11]. In a first step the caller requests a handle for a TPM public key which can be the Endorsement Key of the TPM. After receiving the key handle from the TPM the caller can generate a random salt and encrypt it with the public key and send the encrypted secret back to the TPM[14, p. 13]. As the bus protection relies on the Endorsement Key (or any other signed TPM key) it is also possible to verify that the TPM is authentic as the manufacturer of a TPM issues a unique Endorsement Key for each TPM and signs the key before injecting it into the TPM.

These functionalities can be used in order to enable measured boot/platform attestation. For measured boot the TPM contains a chain of hash measurements from the BIOS/EFI up to the drivers. In a first step the TPM measures the BIOS/EFI and compares the measurement of the BIOS/EFI with its stored value. If the two values are not equal the BIOS/EFI has been manipulated and the boot process will not succeed. In subsequent steps measurements of each layer in the boot chain are taken and being combined with the measurement of the previous layer in order to only having the need to check one measurement. Computing these measurements allows for example for automatic provisioning of disk encryption keys when the measurement of the TPM is known good. It also allows proving to a 3rd party that expected configurations are met.

However not all TPMs are bound by the hardware root of trust. TPMs that are bound by the hardware root of trust are also often called discrete TPMs. These TPMs provide the highest level of security and is hardened against sophisticated attacks and are mostly used in high risk environments. Therefore, a discrete chip is designed and built which undergoes several evaluations to check if it is resistant to hardware tampering attacks[15].

So called integrated TPMs are hardware implementations of TPMs that are integrated into another chip which provides functions that are not focused on security. While still being a hardware implementation the TPM does not have the same level of tamper resistance[15].

Another class of TPMs are firmware TPMs. A firmware TPM is implemented in the protected software (firmware) of the CPU without a dedicated TPM chip being required. The firmware TPM typically resides within a TEE and is therefore separated from other programs that are being executed on the CPU. Implementing a firmware TPM is much more cost-efficient as dedicated hardware chips are no longer a requirement however the TPM now has a bigger attack surface through the TEE code and no longer being resistant against hardware tampering[15].

In the context of cloud computing TPMs that are fully implemented in software have been introduced. These TPMs are under the control of the hypervisor which can make the TPMs available to the different Virtual Machines. The hypervisor is able to create and destroy TPM instances as needed. In the classical virtualization model the hypervisor is a trusted component and is responsible for the separation of the different Virtual Machines. Therefore, the virtualized TPMs (vTPM) can be seen as trusted. However, similar to the firmware TPMs the attack surface has increased and no hardware resistant tampering is possible[22][13]. In the Confidential Computing model however the hypervisor is no longer a trusted component as it can access the memory freely. Due to their pure software implementation vTPMs therefore can not offer the same security guarantees as stored secrets can be freely accessed in the DRAM. Moving the vTPMs outside the scope of the hypervisor into a trusted entity like a Trust Domain helps to protect the secrets that are stored in the DRAM through the security guarantees of the Trust Domain.

2.5 Use Case: vTPM-based Attestation

With the introduction of the vTPM a TD can choose if it wants to use the standard RTMR based remote attestation flow or if it wants to use the remote attestation flow of the vTPM. Choosing the vTPM based attestation allows the verifier to verify additional properties of the user TD through the additional PCR banks of the vTPM which can contain measurements of the installed applications or anything else,

while the RTMR based remote attestation flow is limited to the measurement of the hardware and firmware. As the TDREPORT of the user TD is also stored in the PCR[0] bank which is transmitted to the verifier in the vTPM based attestation a verification of the user TD is still possible with this new attestation flow. However, a TD can not support both attestation flows as this could cause a mismatch in the attestation phase because a verifier may only be aware of the RTMR based attestation. If the verifier now tries to attest a vTPM flow this would not work even if the verifier retrieves the TDREPORT or TDQUOTE for the user TD the verification would not work as the vTPM based flow poisons the RTMR register 0 through 2.

A vTPM can also be used for measurements made by the Linux Integrity Measurement Architecture (IMA) which enables support for runtime measurements[10]. The IMA allows a user to detect if files which are part of the measurement have been altered either by accident or through another malicious program.

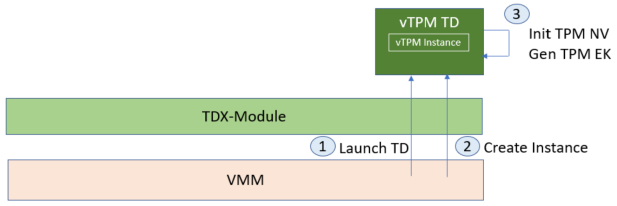

Figure 4. Intel TDX vTPM launch flow [4]

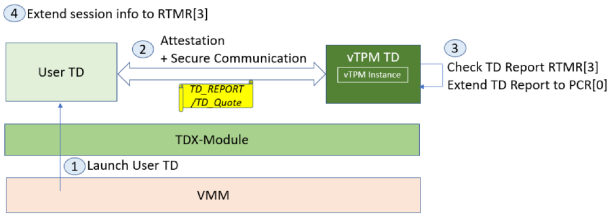

Figure 5. Intel TDX vTPM User TD launch flow [4]

3 vTPMs in Intel TDX

3.1 Implementation in Intel TDX

For the upcoming version of Intel TDX 1.5 a possibility was added to use vTPMs with Intel TDX. In contrast to the already known vTPMs that are in use today these vTPMs are not managed by the VMM as the VMM is an untrusted component in the Intel TDX security model. Therefore, a special implementation of vTPMs was developed which relies on the promised security guarantees from TDs.

In order to implement a vTPM within the Intel TDX architecture either a VM based approach can be used or a TDX Module based approach could be used. Implementing a vTPM within the TDX Module is challenging as the TDX Module itself is imposing a few limitations due to its major role in the software stack. Introducing a vTPM functionality to the TDX Module would mean that the TCB itself is increasing due to the additional complexity introduced which is unwanted due to the major security role of the TDX Module[4, p. 4]. Other reasons why this approach is not favored is the dependency on Non-Volatile storage for a TPM and a long latency in order to generate keys[4, p. 4].

To solve the challenges a TD-based solution has been implemented with two different designs where the vTPM is either in its own separate TD or in case of nested virtualization in the level 1 VM which hosts the VMM for the nested virtualization. As the vTPM shall work together with the existing TDX 1.0 architecture where nested virtualization is not supported the focus will be on the first solution where the vTPM is hosted in its own TD[4, p. 5]. In addition, the vTPM TD shall follow the TPM 2.0 specifications and shall not modify the TPM Software Stack as this ensures the compatibility with existing guidelines and implementations[4, p. 5]. However, the vTPM TD cannot fulfil the requirement of the persistent Non-Volatile storage due to its virtualization nature e.g. the hypervisor is able to delete the VM.

As the implementation relies on a separate TD which hosts the vTPM the VVM is responsible for launching the TD and afterwards creating the vTPM instance. After the vTPM instance has been created it initializes itself and therefore produces the Endorsement Key (EK) which is a unique asymmetric key for each TPM. The whole process is shown in fig. 4.

If a user TD with TPM support is launched the user TD establishes a secure connection to the vTPM TD and attests the vTPM TD, while the vTPM TD attests the user TD. As both parties can be sure that they are communicating over a secure channel with another TD the vTPM stores the TDREPORT within its PCR-Banks. The user TD on the other hand includes the session info in the RTMR[3] in order to have the evidence that a vTPM is connected and can be used. This simplified flow can be seen in fig. 5.

The communication flow between the user TD and the vTPM TD for the exchange of commands and the response is not as simple as shown in fig. 5 as the user TD and vTPM TD need to communicate through the VMM with each other which involves many calls like shown in fig. 6.

The vTPM TD and user TD are both extended with a command-response-buffer (CRG) in order to communicate with each other. The buffer therefore resides in the shared memory regions and is accessible by the VMM and due to that the TPM command and its response which are exchanged through this buffer are encrypted with an Authenticated Encryption with Associated Data (AEAD) algorithm in order to protect it against a malicious VMM. The encryption uses two separate keys the AEAD requester key and the AEAD responder key. Both keys are known to the user TD and the

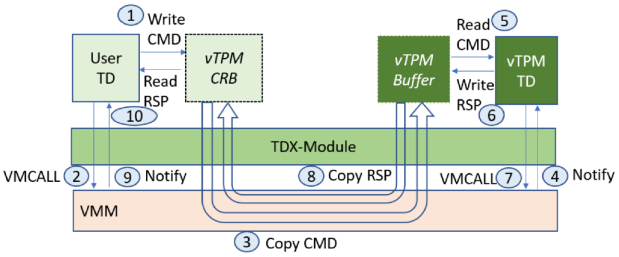

Figure 6. Intel TDX vTPM User TD runtime communication flow [4]

vTPM TD. These shared secrets are established during the initial session setup via the secure channel. The process of establishing these keys is explained in section 2.4.

The communication flow for exchanging TPM commands and responses between the vTPM and the TD contains the following steps:

1. The user TD selects the necessary TPM command and supplies the necessary information and afterwards encrypts the message with the AEAD requester key which is subsequently copied into the CRB for the TPM.
2. The user TD can use the Guest-Host-Communication-Interface (GHCI) TDG.VP.VMCALL<**Service.vTPM**> call to notify the VMM that an action for the specified service is pending.
3. The VMM copies the encrypted TPM command from the shared memory of the user TD into the shared memory of the vTPM TD.
4. The VMM notifies the vTPM TD that the vTPM has been written to its CRB and can now be processed.
5. After the vTPM TD has received the notification it copies the command from the CRB into its private memory and decrypts the TPM command with the AEAD requester key which enables the vTPM to process the command.
6. After successfully processing the TPM command the response is getting encrypted with the AEAD responder key and written back to the CRB.
7. The vTPM can now also use the GHCI with TDG.VP.VMCALL<**Service.vTPM**> to notify the VMM.
8. The VMM copies the encrypted TPM command from the CRB of the vTPM TD to the CRB of the TD.
9. The VMM notifies the user TD that the response has been written to the CRB and can now be processed.
10. After the user TD has received the notification it copies the response from the CRB into its private memory and decrypts it using the AEAD responder key.

3.1.1 Trust Relationship. As the client of the vTPM the user TD generally places a lot of trust in the vTPM while the vTPM as the root of trust does not trust the user TD[4, p. 9]. The user TD itself will only trust the vTPM TD after the virtual firmware of the TD has verified that the report or quote which has been sent from the vTPM TD is valid[4, p. 9]. To communicate with the vTPM TD a secure session via SPDM or TLS is established and after successful verification of the Quote or TDREPORT extended to the user TD and stored in RTMR[3]. If the connection however should fail a random value is written to RTMR[3] in order to indicate that the connection failed. Due to the limitations of the user TD it can generally not verify if the vTPM TD is acting malicious however once the verifier request a complete verification process which includes the measurements of the vTPM this can be detected through the Endorsement Key of the vTPM. In order to trust the Endorsement Key of the vTPM, which due to the nature of the vTPM is ephemeral, a Certificate Authority is needed. The CA itself should be an external service that communicates with the vTPM TD and requests a TD Quote of the vTPM TD which contains a hash of Endorsement Key and as a separate message also the Public Key of the Endorsement Key. As the vTPM TD itself is implemented through an Unikernel approach the TDX hardware is able to measure the complete binary that was used to create the vTPM TD. Having a measurement of the complete vTPM TD which was launched allows the CA to verify this measurement against their own measurement of the binary that was used to create the vTPM TD. Matching measurements indicate to the CA that the vTPM TD is trustworthy and that the Endorsement Key can be signed and be sent back to the vTPM TD.

In contrast to the user TD a vTPM TD only trusts the TDX Module. Therefore, the vTPM records all TDREPORTs that have been received over the secure session in order to prevent the TDVF from forging any measurements. As the TDREPORT is generated through a syscall which the TDX Module passes to the hardware, therefore the TDREPORT can be trusted after a successful verification. In order to protect fully against forged measurements from the TDVF the vTPM TD needs to ensure that the TDVF itself is not malicious. A detection is possible because the TDX Module itself measures the TDVF and includes it in the Measurement of Trust Domain (MRTD) which subsequently can be checked by the verifier. An additional protection against forged measurements from the guest OS is that the vTPM verifies that the secure session is initiated by the TDVF. This is achieved through checking the RTMR[3] in the TDREPORT to be zero and rejecting the session if the RTMR[3] is non-zero.

3.2 Security Considerations

The threat model for vTPMs within Intel TDX considers only the Intel TDX enabled hardware alongside the firmware as trusted. Within a TD the virtual firmware and the software inside the TD are seen as trusted, when the TD is under the control of the user. As the vTPM TD is a special TD which is not under the control of the user this TD is generally

considered untrusted until it has been successfully attested. However, we assume that the TDs as well as the hardware and firmware do not suffer from software based implementation flaws and are free from classic software vulnerabilities. In line with the TDX specifications and the threat model mentioned in section 2.2 software based attacks from the VMM and hardware based attacks are allowed.

SC1. Replacing the vTPM binary Due to the nature that the VMM needs to launch the vTPM TD, which is supplied through a binary, the VMM could easily launch a different vTPM TD than expected[8, p. 84][20]. Under normal circumstances were the vTPM software stack is part of a VM with a complete Operating System detecting a replacement of the vTPM would be difficult to detect due to size of the TCB that would need to be measured. However, the design of Intel vTPM TDs relies on a lightweight bare-metal approach which only includes the absolutely necessary components[4, p. 18]. This approach allows to measure the complete vTPM as part of the firmware during the boot process together with the TdShim layer[4, p. 19]. Through the attestation flow a verifier is now able to verify the measurement of the complete binary that is used to launch the vTPM TD with the measurement of the binary that should have been used to launch the vTPM TD. If the measurements are matching the verifier can be sure that the vTPM binary was not replaced and that the vTPM TD generally can be trusted.

SC2. Initial secure session between TD and vTPM As mentioned in [4, p. 7] the virtual firmware of the TD creates a secure session with the vTPM and exchanges the TDREPORT or TDQUOTE with the peer. Due to limited information available regarding the specific implementation details of how this secure session is established we make a few assumptions which allow us to define the security of the communication. In section 3.1.1 and [4, p. 9] it has already been explained that the user TD has no way of knowing if the vTPM TD is malicious, therefore we can assume that the user TD has no knowledge of any ephemeral vTPM keys or certificates at this point. To achieve a secure session between the user TD and the vTPM TD generally a session with authenticated endpoints is needed in order to prevent Man-In-The-Middle-Attacks. We assume that the vTPM TD is following the Service CA signed EK model shown in [4, p. 15] and using SPDM with public key cryptography based on certificate for which the endorsement key is used[12, p. 70]. Another assumption that is made is that we trust the Service CA to only sign Endorsement Keys that are authentic and have been generated through an authentic vTPM TD image as described in section 3.1.1. Therefore, we indirectly also assume that we can trust the Service CA that signs the vTPM TDs. With these assumptions we can establish a secure session between the vTPM TD and the user TD which at this moment could still be hijacked and replaced without our knowledge. Therefore, as mentioned in [4, p. 7], we need to extend the session information to RTMR[3] which in our case would include the public key or a fingerprint of the public key used for the session setup. Extending the RTMR with this information allows the verifier to receive the information via a TDREPORT or TDQUOTE and match the public key or the fingerprint against a public database of signed keys from the Service CA for the vTPM.

4 Conclusion

Implementing secure vTPMs in Intel TDX through the usage of a separate VM is a feasible approach and allows to extend the measurement from the firmware level to the complete software stack. While the current architecture of Intel TDX has the need for increasing the TCB through the usage of standard protocols like SPDM or TLS instead of being able to use CPU embedded features to securely exchange information between VMs this is only a limited risk when well audited codebases are used that do not possess more capabilities than required. In addition, the support for vTPMs allows to use the standardized attestation interface of vTPMs instead of the vendor specific communication interfaces. In order to completely verify the attestation provided by the vTPM vendor specific checks are still required.

References

[1] Pau-Chen Cheng, Wojciech Ozga, Enriquillo Valdez, Salman Ahmed, Zhongshu Gu, Hani Jamjoom, Hubertus Franke, and James Bottomley. 2023. Intel TDX Demystified: A Top-Down Approach. arXiv:2303.15540 [cs.CR]

[2] Intel Corporation. 2022. *Intel Architecture Memory Encryption Technologiess.* Retrieved 13 June, 2024 from https://cdrdv2-public.intel.com/679154/multi-key-total-memory-encryption-spec-1.4.pdf

[3] Intel Corporation. 2023. *Architecture Specification: Intel Trust DomainExtensions (Intel TDX) Module.* Retrieved 10 June, 2024 from https://cdrdv2.intel.com/v1/dl/getContent/733568

[4] Intel Corporation. 2023. *Intel Trust Domain based Virtual TPM Design Guide.* Retrieved 17 June, 2024 from https://github.com/intel/vtpm-td/blob/main/doc/Intel%20TD%20based%20virtual%20TPM%20Design%20Guide%20Rev%200.7.8.pdf

[5] Intel Corporation. 2023. *Intel Trust Domain Extensions.* Retrieved 10 June, 2024 from https://cdrdv2.intel.com/v1/dl/getContent/690419

[6] Intel Corporation. 2023. *Intel Trust Domain Extensions Data Center Attestation Primitives (Intel TDX DCAP): Quote Generation Library and Quote Verification Library.* Retrieved 13 June, 2024 from https://download.01.org/intel-sgx/latest/dcap-latest/linux/docs/TDX_Quoting_Library_API.pdf

[7] Intel Corporation. 2023. *Intel Trusted ExecutionTechnology (Intel TXT) Software Development Guide Measured Launch Environment Developer's Guide.* Retrieved 10 June, 2024 from https://cdrdv2-public.intel.com/315168/315168_TXT_MLE_DG_rev_017_4.pdf

[8] Intel Corporation. 2023. *Linux* Stacks for Intel Trust Domain Extension 1.5.* Retrieved 17 June, 2024 from https://cdrdv2-public.intel.com/790888/White%20Paper%20-%20Linux%20Stack%20for%20Intel%C2%AE%20TDX%201.5-v0.1.pdf

[9] Intel Corporation. 2023. *What Intel Xeon Processors Support for Intel Trust Domain Extensions (Intel TDX)?* Retrieved 11 June, 2024 from https://www.intel.com/content/www/us/en/support/articles/000091103/processors/intel-xeon-processors.html

[10] Intel Corporation. 2024. *Runtime Integrity Measurement and Attestation in a Trust Domain.* Retrieved 18 June, 2024 from https://www.intel.

com/content/www/us/en/developer/articles/community/runtime-integrity-measure-and-attest-trust-domain.html

[11] Microsoft Corporation. 2023. *BitLocker overview.* Retrieved 10 June, 2024 from https://learn.microsoft.com/en-us/windows/security/operating-system-security/data-protection/bitlocker/

[12] DMTF. 2022. *Security Protocol and Data Model (SPDM) Specification.* Retrieved 06 July, 2024 from https://www.dmtf.org/sites/default/files/standards/documents/DSP0274_1.2.1.pdf

[13] Google. 2024. *TPM 2.0 A Brief Introduction.* Retrieved 11 June, 2024 from https://cloud.google.com/vmware-engine/docs/vmware-ecosystem/howto-vtpm

[14] Trusted Computing Group. 2023. *CPU to TPM Bus Protection Guidance - Passive Attack Mitigations.* Retrieved 26 June, 2024 from https://trustedcomputinggroup.org/wp-content/uploads/TCG_CPU_TPM_Bus_Protection_Guidance_Passive_Attack_Mitigation_8May23-3.pdf

[15] Trusted Computing Group. 2024. *TPM 2.0 A Brief Introduction.* Retrieved 11 June, 2024 from https://trustedcomputinggroup.org/wp-content/uploads/2019_TCG_TPM2_BriefOverview_DR02web.pdf

[16] Trusted Computing Group. 2024. *Trusted Platform Module Library Part 1: Architecture.* Retrieved 26 June, 2024 from https://trustedcomputinggroup.org/wp-content/uploads/TPM-2.0-1.83-Part-1-Architecture.pdf

[17] Trusted Computing Group. 2024. *Trusted Platform Module (TPM) Summary.* Retrieved 10 June, 2024 from https://trustedcomputinggroup.org/resource/trusted-platform-module-tpm-summary/

[18] Vikram Narayanan, Claudio Carvalho, Angelo Ruocco, Gheorghe Almasi, James Bottomley, Mengmei Ye, Tobin Feldman-Fitzthum, Daniele Buono, Hubertus Franke, and Anton Burtsev. 2023. Remote attestation of confidential VMs using ephemeral vTPMs. In *Proceedings of the 39th Annual Computer Security Applications Conference* (<conf-loc>, <city>Austin</city>, <state>TX</state>, <country>USA</country>, </conf-loc>) *(ACSAC '23)*. Association for Computing Machinery, New York, NY, USA, 732–743. https://doi.org/10.1145/3627106.3627112

[19] Oracle. 2020. *Oracle VM VirtualBox Administrator's Guide for Release 6.0.* Retrieved 12 June, 2024 from https://docs.oracle.com/en/virtualization/virtualbox/6.0/admin/hwvirt-details.html

[20] Libvirt Organization. 2024. *Domain XML format.* Retrieved 17 June, 2024 from https://libvirt.org/formatdomain.html

[21] Joana Pecholt and Sascha Wessel. 2022. CoCoTPM: Trusted Platform Modules for Virtual Machines in Confidential Computing Environments. In *Proceedings of the 38th Annual Computer Security Applications Conference* (<conf-loc>, <city>Austin</city>, <state>TX</state>, <country>USA</country>, </conf-loc>) *(ACSAC '22)*. Association for Computing Machinery, New York, NY, USA, 989–998. https://doi.org/10.1145/3564625.3564648

[22] Ronald Perez, Reiner Sailer, Leendert van Doorn, et al. 2006. vTPM: virtualizing the trusted platform module. In *Proc. 15th Conf. on USENIX Security Symposium*. 305–320.

[23] Dov Murik IBM Tobin Feldman-Fitzthum. 2022. *Unifying Confidential Attestation.* Retrieved 10 June, 2024 from https://static.sched.com/hosted_files/kvmforum2022/02/Unifying-Confidential-Attestation-KVM-Forum-2022.pdf