Virtualized Trusted Platform Architecture Specification

Specification Version 1.0
Revision 0.26
September 27, 2011

Contact:
Paul Sangster – Paul_Sangster@symantec.com (Co-editor)
Lee Wilson – LeeWils@us.ibm.com (Co-editor, VPWG Co-Chair)
Dean Liberty– Dean.Liberty@amd.com (VPWG Co-Chair)

TCG PUBLISHED

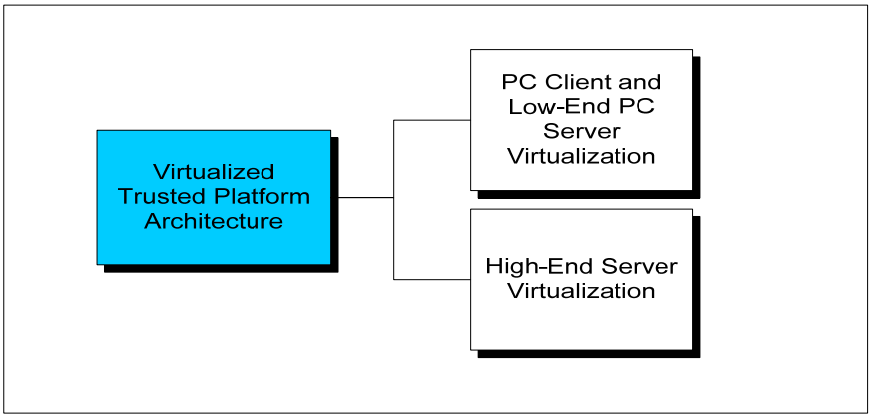
TCG

Disclaimers, Notices, and License Terms

Revision History

9/1/2009	Initial draft based upon Stephan Berger's document
11/11/2009	Updated based upon terminology feedback at TCG meeting in Austin
12/01/2009	Migrated to new template and reformatted spec plus some new text and integration of the use cases.
12/08/2009	Updated diagrams based on VPWG concall discussion
1/19/2010	Additional text discussing challenges to virtualization of trusted platforms
1/29/2010	Added new sections discussing architecture components
5/24/2010	Added new sections on attestation of VMs and initial migration text (v08)
5/27/2010	Updated section on attestation based upon VPWG discussions (v09)
6/4/2010	Added new sections discussing migration (v10)
7/12/2010	Included many of Dean's editorial suggestions (v11)
8/4/2010	Included many of proposed changes from Dean and Lee (v12 and v13)
8/17/2010	Additional updates based on comments from Dean to section 3.2 and isolating and indenting the requirements (normative) text.
8/31/2010	Added new sections and updated per comments at VPWG meeting on 8/19
9/8/2010	Complete the VMM component section
10/6/2010	Accept updates from previous revisions and address many of the remaining open issues plus complete the component discussions.
10/13/2010	Initial draft of security considerations and replace VPMA with Migration Controller
11/1/2010	Rewrite use case section and move to front of specification. The rewrite will uplevel the use cases. Also include support for TC proposed approach to restore of vTPM instances.
11/29/2010	Wrote security considerations section (first full draft)
12/8/2010	Updated based on comments from VPWG (David and Lee) final review before e-ballot
12/13/2010	Updated based on comments from Mark Scott-Nash, Francois Lucas and Chris James
6/2/2011	Approved by the TCG Board for Final Publication

VPWG Document Roadmap



Acknowledgements

The VPWG wishes to thank all those who contributed to this specification.

Asher Altman	Intel
Stefan Berger	IBM
Rene Bourquin	General Dynamics
Benoit Poulot-Cazajous	Bertin Technologies
David Challener	Johns Hopkins University
Conan Dailey	General Dynamics
David Gilbert	IBM
Carey Huscroft	Hewlett-Packard
Chris James	Johns Hopkins University
Dean Liberty (Co-Chair)	AMD
Francois Lucas	Thales Communications
Bob Malek	Unisys
Eugene Myers	US Government
Frank Molsberry	Dell
Florian Samson	BSI
Jose Sancho-Dominguez	Hewlett-Packard
Paul Sangster (Editor)	Symantec
Vinnie Scarlata	Intel
Mark Scott-Nash	Intel
Leslie Stolp	HP
Don Simard	US Government
Lee Terrell	IBM
Lee Wilson (Co-Chair)	IBM
Monty Wiseman	Intel

Table of Contents

1 Introduction

1.1 Scope and Audience

This document was created by the Virtualized Platform Working Group (VPWG) within the Trusted Computing Group. The VPWG is responsible for the definition of interfaces and services necessary to virtualize the trusted platform capabilities on a particular system. The intended audience of this specification is architects and developers of systems that provide virtualization platforms including standards and technologies defined within the TCG.

This specification defines a general architecture, terminology and envisioned set of deployment models for what capabilities virtualized trusted computing platforms are expected to offer. This document does not focus on how a particular design or implementation of a virtualized trusted platform should operate on specific hardware (e.g. what functions are done in hardware, hypervisor or VM protection model). Instead, platform or usage-specific (low level) specifications can be created that explain how the general architecture can be instantiated on particular hardware and what security properties it offers to the VM. The lower level platform oriented specifications will define their scope of operation to be clear what assumptions are being made about underlying operational environment. Not all platform level specifications will support every aspect of the general architecture, so implementers should carefully read the early sections to understand the scope of operation and set of security properties each platform level specification is providing.

Note that the general architecture discussed in this document is focused on virtualization of the underlying hardware platform and not on other forms of virtualization such as the Java Virtual Machine model. This document also focuses on virtualization of the TCG trusted platform components and is intended to be one aspect of a more general platform virtualization model.

1.2 Goals

The goals of this specification are to:

- Document a high level architectural view of a virtualized trusted platform identifying and describing each of the primary components the system
- Discuss potential issues and responsibilities for each of the identified components
- Identify general requirements for functionality of each component when appropriate. Detailed requirements and an approach would be document in the lower level specification.
- Minimize the number of changes to existing TCG technologies required to support virtualized trusted platforms when possible.

The goals of a virtualized trusted platform are to:

- Allow an OS/application that uses TCG trusted computing technology to run unmodified on either a physical (not virtualized) platform or in a VM on a virtualized trusted platform without loss of any security properties or features. However, it is understood that some reduction in assurance (or expansion in trust model) might be required to run virtualized when a less physically isolated (e.g. software) vTPM is used. Customers can determine the level of assurance offered by consulting any security compliance test results available. Similarly some OS/applications may benefit (e.g. performance) from the addition of trusted software to run in the VM.
- Enable a challenger to obtain information (if allowable by local policy) about the security state of the underlying virtualized trusted platform that is providing the isolation and resources to the VM.
- Support the dynamic migration of virtualized trusted platforms between multiple physical platforms, based on a policy. The target system of a migration could have different security properties than the source so relying parties should be able to learn the migration

policy for a system that intends to change the security properties offered to the VM over time.

1.3 Non-Goals

The following list several of the key non-goals for this specification. In some cases, these could be the topic of other or future VPWG specifications:

- System management details for virtual platforms.
- How to build a virtualized trusted platform targeting a particular platform or usage model (possible focus for lower-level specifications).
- Paravirtualization architectures and components. Paravirtualization might be addressed in a future version of this document.

1.4 Keywords

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in RFC 2119 [1]. This specification does not distinguish blocks of informative comments and normative requirements. Therefore, for the sake of clarity, note that lower case instances of must, should, etc. do not indicate normative requirements.

This document includes a number of requirements provided to ensure consistency across the virtualization architectures defined in more detail in the low level specifications. In order to improve the requirements visibility, each normative requirement is indented and placed in its own paragraph.

2 Background

Virtualized computing platforms such as hypervisors and virtual machines (VM) are rapidly becoming mainstream commercial products. Several classes of conventional commercial platforms, including PC clients, servers and mobile devices are expected to host virtualized trusted computing platforms. These platforms may already provide trusted platform [6] features (e.g. TPM [3] in a PC client) that need to be virtualized in order to provide what appear to each VM as a dedicated virtual platform similar to if the VM was running directly on the non-virtualized hardware.

This specification defines the general architectural components envisioned and terminology involved in virtualizing a physical trusted platform such that its resources may be shared across a set of guest virtual machines. In a non-virtualized environment, the Trusted Platform Module (TPM) specification assumes a one to one relationship between the operating system (OS) and the TPM. Evidence for this can be found in the fact that a TPM has a single owner, a single Storage Root Key and password, one Endorsement Key, and a single set of Platform Configuration Registers (PCRs) of which some registers have been reserved for a dedicated purpose. However, virtualization solutions allow multiple OSes to be running independently on a single platform. Therefore the assumed one to one relationship between an operating system and a physical trusted platform (including a TPM) does not hold anymore. The Virtualized Trusted Platform architecture extends this premise; it provides for a one to one relationship between a virtual platform (virtual machine) and a virtual TPM. In order to offer dedicated trusted platform (e.g. TPM) functionality to each virtual machine on the platform, the current TPM interface or feature set might need to be extended in order for the concepts to be offered in a virtualized environment.

This specification discusses the generic reference architecture and establishes a common set of terminology and components that can be used in the lower level case specifications envisioned by the Virtual Platform Working Group (VPWG). This document also discusses the responsibilities of each component including highlighted potential challenges and approaches. This specification does not try to establish an implementation level architecture, but rather describes a generic reference model that lower level specifications are to leverage in their specifications.

2.1 Virtualization

There exist many types of virtualization in the industry today; this specification focuses on hardware platform virtualization (as opposed to storage virtualization). For the purposes of this specification, virtualization architecturally involves layers of software that make the underlying (frequently hardware-based) resources appear as multiple, isolated instances that are allocated to higher layered software entities (e.g. virtual machines). This virtualization is done in such a way that each higher layered entity believes it has exclusive use of those capabilities and is unaware it's not running directly on top of the hardware. Therefore, virtualization offers the perception of exclusive use of potentially shared underlying physical resources. When virtualization offers the appearance of a complete hardware system to an operating system, this represents a virtualized (hardware) platform. This specification focuses on the components necessary to virtualize the specific trusted platform oriented capabilities and services (e.g. TPM) so that they can be included in multiple virtualized platform instances.

From a high level view, platform virtualization is an architecture where a software layer running on a hardware platform makes the resources of the hardware platform appears as multiple, isolated instances where each instance is associated with a particular Virtual Machine. Each Virtual Machine runs a unique OS instance, known as the Guest OS, and operates independently of the other virtual machines on the system. The software layer is known as the Virtual Machine Manager (VMM) or the Hypervisor. Each Guest OS requires an isolated virtual machine for execution. The VMM is generally a layer running beneath the VMs providing the execution environment. One possible implementation architecture for the VMM is to have some VMM functionality running in a special privileged VM (as shown in the diagram below). This helps isolate potentially risky software from the core VMM address space and could ease implementation.

To maintain performance, the VMM could allow the Guest OS to execute its instructions and functions as much as possible on a VM allocated subset of the native hardware, but still maintaining its isolation. From the Guest OS point of view, it is executing the same instructions and functions as if it were running directly on top of the native hardware platform. When the Guest OS executes instructions, accesses memory or registers that may affect the shared hardware state, the VMM will trap the instruction and may redirect or emulate a device to make it appear to the Guest OS that appropriate action for the instruction was taken. How this is done by the VMM is design dependent and may include hardware assists in the processors/chipsets and/or multiple copies of a resource created in hardware, software or a combination of both.

One example of a hardware assist is address translation of a Guest OS memory mapped register access to a global hardware address used in the platform. An example of virtual resource instances would be for the VMM to emulate a set of Real Time Clocks (RTC) where each RTC instance is dedicated to a particular VM.

On a virtualized platform, a virtual machine (VM) provides the execution environment for an Operating System through software emulation of the functionality of devices that are usually implemented in hardware and accessible through a hardware-specific interface. Through the software emulation of hardware functionality, a new layer of indirection between the OS and the underlying physical hardware is introduced and allowing new models of behavior. One example is the migration of an OS in a VM from one physical platform to another along with the virtual devices that are connected to it. Virtual machine migration, which has previously not been considered in the context of the trusted platform module, further extends the need to understand and define the trust relationships between the OS and the underlying physical platform.

Virtualization does not always imply a virtualized TPM. It is possible that the resources of a single physical TPM may be logically or physically partitioned among the VMM/hypervisor and several guest operating systems. In the context of this specification, the partitioning of a physical TPM is considered sharing of a TPM, which stands in contrast to virtualizing the TPM, where the appearance of a private TPM is provided to each VM. The sharing of a TPM by multiple virtual machines has consequences to the usage model of that TPM and how applications running inside of these virtual machines can use the TPM and its resources. A virtualized TPM, on the other hand, enables applications running in a VM to remain generally unaware of the fact that they are not communicating with a typical hardware implementation of a TPM.

2.1.1 Full Virtualization vs. Para-Virtualization

Two different types of virtualization are commonly used today. The first type allows the VM to be running all of its code without modification to be aware that its operating on a VMM. The VM would perform all its operations exactly as if it was running directly on top of the hardware. This type of virtualization is known simply as “virtualization” or more pedantically “full virtualization” because the Guest OS is completely unaware it is operating upon resources that may be virtualized (e.g. emulated). This model potentially has the drawback that the VM might try to perform privileged operations or direct hardware IO operations that it would typically be able to do when running directly on the hardware, but in a virtualized deployment this causes traps that need to be handled by the VMM. As a result, the VMM needs to be run more often which can result in slower Guest OS performance.

The second type of virtualization, known as “paravirtualization”, occurs when the Guest OS software has modified software that is designed to reduce the number of VMM traps and thus improves performance. One typical type of paravirtualization of a Guest OS would be the inclusion of special virtualization-aware drivers for the major IO devices. Another form of paravirtualization is for the VMM vendor to define its own IO device types and install Guest OS drivers for these devices. To the OS vendors these devices would appear as any other IO device found during discovery with signed drivers available. An example of this would be for the VMM to report a generic Fiber channel controller and a generic NIC controller when probed by the Guest OS regardless of what physical devices are present. The VMM vendor would release drivers for these generic devices. Depending on the features of the paravirtualized drivers, applications may become

aware that they are running on an OS offering paravirtualized drivers, though it is still desirable that applications work without being adapted. Paravirtualization might be addressed in a future version of this document.

2.2 Virtualization of a Trusted Platform

In a virtualized system, each VM is provided with a set of resources (e.g. devices) that it requires in order to execute. These devices are typically virtualized by the VMM into multiple (frequently logically) isolated instances of the physical device present on the system and allocated to each guest OS VM. By virtualizing the device, this allows multiple VMs to share a physical device while avoiding the operations of one VM on the device changing the state of the other VMs instance of the device.

The components of the trusted platform need to be virtualized so they can be shared by multiple VMs while maintaining the isolation of each VM's trusted platform state. The TCG has defined a number of key components that comprise the trusted platform including the Root of Trust for: Reporting (RTR), Storage (RTS), Measurement (RTM), and Verification (RTV) [6]. For today's physical trusted platforms, the RTS and RTR are housed in the Trusted Platform Module (TPM). The TPM offers an isolate execution environment enabling protected operations to be performed on data housed in isolated (shielded) storage locations. This means that the operating system can perform operations on secret data that is never visible in clear text outside of the TPM, so therefore is protected from theft while in use. The RTM and RTV exist outside the TPM and may be integrated into other aspects of the platform (e.g. BIOS), boot loader or parts of the operating system kernel.

One challenge with virtualizing the TPM is how to maintain these valuable security properties (isolated execution and storage environment) while still enabling the TPM's features to be shared yet isolated. This specification will discuss a number of possible techniques for virtualizing the TPM including some which allow the VMM to maintain and manage separate virtual TPM (vTPM) state tables associated with each VM's TPM instance. Other approaches will leverage more highly capable hardware for handling the TPM instance state. Each approach may offer different assurance against differing threat models. While the VMM managed state approach to virtualizing the TPM would resemble how many other devices are virtualized by the VMM, these state tables are exposed to threats inside the VMM so pose a higher risk then having the state kept inside a tamper protected hardware device. Therefore some implementations of the vTPM might offer lower assurance or protection from certain classes of threats. The assurance level of a particular architecture or implementation can be determined based upon its compliance with applicable TPM and virtualization platform protection profiles.

Another challenge facing the virtualization of the virtualized trusted platforms is the requirement that a VM be able to be moved to other physical systems. Migration of VMs is a common benefit on virtualized servers to allow the administrator to move a VM to another system while the VM is still in operation (e.g. allow for patching of the hypervisor or do hardware maintenance on the server). Traditionally, trusted platforms and their TPMs were physically bound to ensure the properties about the platform asserted in the credential accurately reflected the platform. Similarly TPMs could not be cloned and instantiated on other systems allowing the appearance of two systems with identical TPM state (e.g. SRK and EK private keys). In order for the deployment scenarios described in this document to support migration, they will need to address these restrictions.

There are many possible ways of creating virtual trusted platforms, including designs that use special privileged VM and VMM capabilities, shared multi-instance hardware or custom hardware assists. This specification will discuss a number of the primary models discussed within the VPWG and provide the basis for more detailed specifications to be written which focus on how to implement such a system.

2.3 Terminology

The following table lists the terminology used throughout this document. While some terms may have different meaning in other contexts, these definitions are how the terms are to be used within the Virtual Platform Working Group specifications.

Name	Description
Remote Challenger	An entity (actor) wishing to obtain attestation information to determine the trustworthiness of a remote operating environment (e.g. VM or VMM). This attestation might involve communication with an Attestation Agent for the operating environment, an attestation services for any dependent sub-layers (e.g. VMM) and/or network infrastructure that could impact the trustworthiness decision (e.g. the Migration Controller if the operating environment can be migrated).
Deep Attestation	The procedure of attesting multiple layers of a virtualized system. A deep attestation could be necessary if a Remote Challenger needs to assess the trustworthiness of a VM, the underlying VMM(s) and potentially attestation evidence all the way down to the underlying hardware roots of trust (e.g. physical TPM).
Para-Virtualization	A virtualization environment where the Guest OS is augmented in some way to improve the performance or other properties of the Virtual Platform to understand aspects of the underlying VMM layer. For example, this may involve replacing the I/O drivers with ones that are more optimized or knowledgeable of the underlying VMM layer.
Virtual Machine Manager (VMM)	Software that oversees or virtualizes the underlying hardware platform in order to enable multiple Virtual Machines to run concurrently in isolated environments and share the platform's resources. The VMM controls isolation, creation/destruction, save/restore, and migration of each instance. VMM is synonymous with Hypervisor which is commonly used in other contexts.
Virtual Machine (VM)	A basic software execution environment typically including an operating system and applications, that is instantiated and run on top of a Virtual Machine Manager. The VMM provides the VM with the interfaces and resources needed to enable it to access the underlying physical platform. Multiple VMs can run concurrently in separate isolated environments (or partitions) from each other.
Trusted Platform Module (TPM)	Component consisting of hardware and/or software which implements the requirements described in the TPM specification. Different classes of TPMs can exist depending on the platform type the TPM is intended to be used within. For example the PC Client TPM is described in the interface specifications [7]. Similarly, a virtualized platform could define another class of TPM that is virtualized for use by a virtual machine.
Physical TPM (pTPM)	The class of TPM that is specifically referring to a physical, hardware-based TPM. pTPM and vTPM are subsets of the general case of TPM which spans both hardware and software based implementations. Generally a pTPM would exist at layer 1 of the virtualized trusted platform architecture.
Virtual TPM	The Trusted Platform Module (TPM) instance associated with a single

(vTPM)	Virtual Machine as part of a vPlatform instance. The vTPM is a component consisting of hardware and/or software that supports the functionality described in the TPM specifications. Different vTPM implementations can offer differing levels of assurance based upon its supported threat model but for the VPWG specifications these different implementations are still considered vTPM.
Root of Trust for Measurement (RTM)	Component consisting of hardware, firmware (e.g. BIOS, UEFI, microcode) and/or software that is responsible for measuring software that is executed on a platform such that the measurement can be extended into a PCR. Many RTM are able to immediately extend the measurement into a PCR while in some circumstances the CRTM may store the measurement for later extension.
Core Root of Trust for Measurement (CRTM)	The CRTM is a type of RTM that is specifically responsible for measuring the initial code executed on a platform upon initial reset such that the measurement can be included in the TPM. By measuring and extending into the TPM the initial piece of code run on a platform at reset, this allows the TPM to contain measurements of all code run on the system. The CRTM may not be responsible for measuring other code executed, so each piece of code that loads and executes other software may be responsible for measuring and extending the newly loaded codes measurements into the appropriate PCRs. These non-core measurers are considered (non-core) RTM. For example, a CRTM in a static root of trust platform could be logic in the BIOS that is able to detect the loading of the initial software to be executed on the processor and measure it prior to execution. The measurement might need to be stored until the pTPM is operational and able to store the measurement in a PCR. The dynamic CRTM (D-CRTM) is another example of a RTM.
Physical RTM (pRTM)	The class of RTM that is specifically referring to a physical, hardware-based root of trust for measurement. Traditionally, the pRTM is implemented in a combination of hardware and system firmware (e.g. BIOS, UEFI, microcode, etc.) in layer 1 of the virtualized trusted platform architecture.
Virtual RTM (vRTM)	The class of RTM that is logically associated with a particular VM instance. The vRTM is typically associated with each VM in order to measure and begin to populate its associated vTPM's PCRs starting at the initial execution of the VM (analogous to system reset with the pRTM).
Virtual Trusted Platform (vPlatform)	The Virtual Trusted Platform comprises all code and data that enables the proper functioning of trusted computing oriented software within a virtual machine and of the operating system running inside of it. This includes the state of the virtual TPM and the other virtual trust roots (e.g. vRTM). A vPlatform can be migrated from one physical platform to another. Note that while the term vPlatform might sound general it only refers to the virtual trusted platform.
Virtual Trusted Platform Manager (vPlatform Manager)	The software that controls the creation and destruction of the vPlatform associated with each VM including any vTPM instances and operation of the vPlatform including triggering the vRTM. It may be a stand-alone component or part of the VMM or hardware vTPM unit.

Migration Requester	Requester of the migration of a VM. There is no security policy enforcement done by this entity. It is strictly functional (load-balancing, device utilization, etc). This entity could be a human being or an automated system. Does not require trust by anyone beyond availability assurance. The protocol by which the Migration Requester makes a request is VMM specific and out of the scope of this document.
Migration Policy	The migration policy describes the concrete restrictions under which a VM may migrate between physical platforms. The Migration Policy is used by the Migration Controller (and potentially the Migration Engine) to base its migration decisions as it receives requests for VM migration. Similarly, Remote Challengers may wish to assess a VM's migration policy to understand the potential platforms that could host the VM should a migration event occur.
Virtual Trusted Platform (vPlatform) Migration Controller or Migration Controller	The Virtual Platform (vPlatform) Migration Controller evaluates its Migration Policy and approves or disapproves migration requests by the Migration Requester. The Migration Controller could exist in a number of places on the network including in the infrastructure (e.g. integrated within the virtualization management fabric) or even on the end system (e.g. integrated within the VM migration software). In addition to enforcing restrictions on migration requests, the Migration Controller is also capable of advertising its Migration Policy to Remote Challengers attesting a VM under the Migration Controller's control. For example, a Remote Challenger attesting a VM might wish to know whether the Migration Controller might migrate the VM to another system with different (generally lesser) security protections during its session.
Local Virtual Trusted Platform Migration Engine or Migration Engine	The Local Virtual Platform Migration Engine is the component within the Virtual Machine Manager that packages, serializes and protects (encrypts) vPlatform state data during transmission to a destination selected by a Migration Requester after the approval of the Migration Controller. This component is responsible for allowing only one instance of a unique vPlatform to be active at a time and the removal of a vPlatform instance once it was has been received by the destination system.
Layer 1	The physical platform components comprise the bottom layer of the architecture (layer 1).
Layer N	The top layer in the virtualization general architecture consisting of potentially multiple virtual machines each running independently.
Layer N-1	Highest layer in the virtualization general architecture offering a virtualized platform to virtual machines running above.
VMM Owner (Layer N-1 Owner)	The person who owns administrative rights to the VMM. This person may install new software or update and remove existing ones. The VMM owner generally controls what functions other users, such as VP owners, may perform. The VMM Owner also controls the management virtual machine in the helper suite that supports necessary functionality for enabling Trusted Computing on the Virtualized Platform.

Virtualized Trusted Platform Owner (Layer N Owner)	The VP Owner owns and controls a virtual platform (VP) associated with a particular VM. Multiple VPs on a single system may have different owners. The VP owners are authorized to perform operations on the VPs from the VMM owner.

3 Use Cases

This section discusses the use cases that provide a rationale for the features and requirements described in the remainder of the specification. The described use cases tend to be lower level than customer deployment level usages in order to provide the reader an easier understanding of how the usage impacts the architecture and requirements defined within this specification. In order to provide a linkage to higher level use cases (more familiar to deplorers), each sub-section provides an example deployment situation motivating the lower level use cases that follow. Finally, the last sub-section discusses some of the topics which are out of scope for this specification or merely are being held off for a future revision.

3.1 Creating a New Virtualized Trusted Platform

This usage covers the situation when a new virtual trusted platform needs to be created from scratch. This normally occurs when a new virtual machine requiring a trusted platform is being deployed for the first time. There are a many situations where a new virtualized trusted platform could be needed.

For example:

> *A cloud provider offering customers the ability to "push" a VM image into the cloud for execution on one of the cloud provider's virtualized systems. The customer's VM image expects virtualized trusted platform features to be available to perform its functions including allowing the customer to perform an attestation of the cloud providers VMM prior to deploying the VM to ensure adequate security is provided.*

There are two general scenarios when a new virtualized trusted platform needs to be instantiated to address the cloud and other similar usages:

- **Newly Booted Platform** - A newly configured physical platform with virtualization support is booted and its configuration indicates that a new VM is to be instantiated. The virtual trusted platform management software in the VMM is responsible for instantiating the vPlatform associated with the new VM. The instantiated vCRTM and vTPM are dedicated to the new VM and appear the same as their physical equivalents allowing an unmodified VM to execute unaware that the vPlatform isn't a physical trusted platform.
- **Newly Configured VM** – An administrator (possible via a management agent) notifies a virtualized platform that a new VM instance is desired and establishes the context. The change process advises the virtual trusted platform management components that a new vPlatform needs to be instantiated and bound to the new VM. The virtual trusted platform associated with the new VM is instantiated and started executing (as appropriate whether an SRTM or DRTM) to measure the initially executing VM software. The instantiated vCRTM and vTPM are dedicated to the new VM and appear the same as their physical equivalents allowing an unmodified VM to execute unaware that its TPM and CRTM aren't a physical components.

3.2 Instantiating a Previously Executed vPlatform

This usage covers the re-instantiation of a previously executed virtual machine. There are a many situations where a previously executed vPlatform that is no longer running might need to be started executing.

For example:

> *The IT administrator of a large web service provider operating multiple virtualized platforms decides that the load on one of the physical systems is getting too high and impacting the services responsiveness. The administrator decides to move one of the busier VMs to another physical platform that has a smaller work load. Using the virtualization platform's VM migration infrastructure, the administrator requests a VM on the busy system be*

quiesced and moved to another less loaded physical platform. This migration request also includes movement of the VM's underlying virtualized trusted platform.

This migration usage is one of many situations where a virtualized trusted platform that had been executing in the past now is needed to be re-instantiated (much like when the VM is re-instantiated on the less loaded target system in the example). The following include five other situations where a re-instantiation would be required:

There are several situations that might result in the vPlatform being re-instantiated:

- **Reboot of Platform** – An administrator causes the reboot of the physical platform running VMs. Each VM is notified that it must perform an orderly shutdown and the vPlatform is similarly notified. The appropriate components in the VMM save the persistent vPlatform state to persistent storage. When the physical platform power on event occurs, the VMM will re-instantiate VMs based upon its configuration. Each re-instantiated VM may require a re-instantiated vPlatform in order to continue operation, so each vPlatform is re-instantiated using the saved persistent state (the non-persistent state is reset as appropriate).
- **Backup/Restore** – Current non-virtualized trusted platforms enable the administrator to backup the migrateable portions of the TPM, therefore these backup solutions are still necessary when operating on a virtualized trusted platform (this usage overlaps with Local Security in section 3.3). However, vTPMs, by their software nature, offer the administrator the opportunity to back up the entire persistent state of a vTPM, since it may exist in memory at one point (e.g. inside the VMM). Later the administrator might decide to re-instantiate the backed up vPlatform using the saved persistent state.
- **Reboot of a vPlatform** – An event occurs (e.g. administrator reboot of a VM) notifying the VMM that a particular VM is shutting down. The persistent state of the virtual trusted platform instance needs to be saved to persistent storage in case the VM is later restarted. When the VM is later booted, the VMM is responsible for re-instantiating the VM and its vPlatform restoring its persistent state and initializing the volatile state as appropriate. This usage is very similar to the Backup/Restore use case above except that the state is generally stored in on-line storage (local disk) for more rapid resumption while a backup commonly could be moved to secondary storage (tape) for archival purposes.
- **Migration** – An event occurs (e.g. administrator action) notifying a virtualized platform that one of its VM needs to migrate to another system. The notification reaches the vPlatform management software which prepares the vPlatform associated with the migrated VM for migration. The full (persistent and volatile) vPlatform state is migrated to a different physical platform where the vPlatform management components re-instantiate the previously executing vPlatform just prior to the VM instantiation, so the VM may continue to execute without being impacted by the migration. After the migration has completed the source system deletes and sanitizes any state is still has about the migrated vTPM.
- **Suspend/Resume** – An administrative action causes a VM to be suspended to persistent storage for later execution. This event is communicated to the VMM's components responsible for the management of the executing vPlatforms. The VMM saves all of the operational state (persistent and volatile) to storage allowing the vPlatform instance to be fully restored at a future time. When the vPlatform is again later required, the virtual trusted platform components must verify that the saved vPlatform state was properly protected and is now authorized to be resumed on this platform. This usage resembles the migration usage because the entire vPlatform state (even non-persistent) is captured and re-instantiated on the target system. Note that this usage differs from the reboot usage where only persistent state is preserved.

3.3 vPlatform Operation

This usage covers the general operation of the VM and its interaction with the vPlatform. Some VMs contain software that directly leverages capabilities of the underlying physical trusted platform,

but weren't written to support virtualization. Such software should operate equally well when running on a non-virtualized platform as when inside a VM.

For example:

> *A test engineer has a non-virtualized physical platform containing new service software leveraging a trusted platform feature that is being tested. The engineer decides to deploy the new service in a VM on the company's large virtualized platform hosting other services. When the VM instance executes, the vPlatform support within the VMM allows the VM's trusted platform features to work just as they did on the test engineer's system.*

There are two general situations that might impact the portable operation of the VM:

- **Local Security** – The VM software might be written to leverage a TPM when present on the platform to provide local security. This includes a variety of security functions that are the topic of other specifications. The local security functions (ordinals) offered by a physical TPM need to be offered by the vPlatform in order for the VM to operate without virtualization specific modification.
- **Remote Attestation** – Remote systems may use the resources or services offered by a VM operating on a virtualized trusted platform. The VM may choose to allow itself to be attested for proof it's running in a trustworthy manner using cryptographic proof offered by the TPM. This is no different from when a Remote Challenger is trying to determine the trustworthiness of a non-virtualized system, however the details of what is included in the attestation exchange itself are likely to vary (no need to challenge lower layers on non-virtualized platform). The Remote Challenger may need a trustworthy way to attest the layers below the VM to ensure the entire stack is trustworthy. This usage also captures the Remote Challenger expectation that a consistent attestation model exists for virtualized and non-virtualized trusted platforms, so that it can assess trustworthiness and recognize special aspects of vPlatforms that impact the overall level of trust assigned to the system.

3.4 Hot Stand-by

This usage covers the situation where the two systems are performing identical functions so need to appear over the network as the same system.

For example:

> *A service provider wishes to offer a highly reliable solution by operating two (or more) physical platforms each running the same identical VM instance including its vPlatform. As service request come in over the network they are load balanced across these identical systems in such a way that remote requestors can't discern which system is actually providing the service including though each VM's attestation.*

More specifically, the following considerations exist for hot standby platforms:

- **Hot Stand-by** – In situation where a service providing platform is so critical that customers are unable to wait for a backup (replacement) system to be made operational in the case of a failure of the primary, many service providers look to hot standby systems. In some cases, it's necessary for scaling purposes for the hot standby system to actually be handling transactions in parallel while all appearing as the same system to remote parties. In a physical trusted platform, customers can not duplicate the pTPM state, but the situation changes for the vTPM. It's critical that the vPlatform on both physical systems remain identical whether one is a hot standby or is running in parallel since they both are logically the same vPlatform.

3.5 vPlatform Upgrade

This usage covers the situation where the virtualized trusted platform software (or firmware) on a platform needs to be upgraded.

For example:

> *An IT administrator becomes aware of an operational issue affecting the performance of a virtual trusted platform implementation and downloads the patch to address the problem. The administrator's virtualized system is running many VMs that would be impacted by the patch, so decides to install the patch after shutting down each VM during the maintenance window and rebooting the system.*

More specifically, the following considerations exist during vPlatform update:

- **vPlatform Update** – One or more virtualized trusted platforms on a system might be in need of a patch to upgrade the logic of the vPlatform. This could include any of the vPlatform components including the vCRTM or the vTPM. It's envisioned that the patch would be applied when the VM associated with the vPlatform is non-operational (either quiesced or shutdown) in order to ensure that no virtualized trusted platform operations were requested during the upgrade. After the application of the patch, the VM might need to be rebooted allowing the new vPlatform logic to take affect (e.g. vCRTM updates might only execute during the VM boot). Support for hot patching of a vTPM allowing the VM to continue execution after the patch is installed would be valuable if it does not introduce security or operational risks. Patches impacting the format (on persistent storage or in memory) of internal vTPM state could require a reboot after installation.

3.6 vPlatform Migration

This usage recognizes the unique aspect that VMs and their virtual trusted platform may operate under the supervision of a remote entity (Migration Controller) that can change the VMs trust properties at any moment. Such a change could be something a Remote Challenger would want to factor into its trust decision about the VM.

For example:

> *A cloud provider offers a group of systems able to execute and manage a collection of customer provided VMs. In order to optimize the operation of the system group, the cloud provider has software to detect hardware faults or performance issues associated with each particular system that is able to cause an automated migration of VMs from that system. A Remote Challenger using one of the VMs on the degraded system might wish to factor into their initial trust decision whether it's acceptable for the cloud provider's management software to move the VM during the transaction to another potentially less trustworthy system.*

There are a couple situations that might affect the Remote Challengers trust decision:

- **Attestation of VM Migration System**– Remote Challengers that wish to verify the trustworthiness of a remote VM may need to factor the migration policies that could impact when and where the VM could be moved. A Remote Challenger could establish that a VM and its vPlatform are trustworthy and start to perform sensitive operations with the VM. Moments later, the VM migration management system in response to events (administrator actions, network events, power events) could decide to initiate a migration of the VM to another platform with different (possibly reduced) security properties. Some Remote Challengers might wish to be able to proactively evaluate the situations where a migration might occur while accessing the VM and the potential types of target systems as part of the trustworthiness initial decision. Other types of challengers might just wish to be notified after a VM migration has occurred, possibly to re-perform the attestation, while others might rely on non-technical means (SLAs) for re-establishing trust.
- **VM Migration System Attestation of Candidate Platforms** – The VM migration system might be designed/configured to evaluate the integrity of a new platform being added to the group of systems within the migration domain of the controller. This attestation could impact whether the new system will be allowed to join the grouping. Similarly, the VM migration system might perform an attestation of the N-1 and lower layers immediately prior

to requesting the migration of a VM to that system. This would allow a comparison of the level of trustworthiness offered by the source and destination systems involved in a VM migration prior to authorizing the operation.

3.7 Out of Scope

The following list discusses major topics that are out of scope for this specification revision, but might be covered by a future revision or in another TCG specification:

- **VM Issues** – This specification is about the virtualized trusted platform associated with a VM. This document does not discuss other issues involving the use or management of a VM itself (e.g. creation, migration, deletion). Practically speaking it seems likely that implementations will wish to integrate the VM management software (and infrastructure) with the component features discussed in this specification, but such integration is outside the scope of this document.
- **VM Migration Interface** – As stated in the VM Issues item, it's envisioned that the VM management software is likely to incorporate or interface to the vPlatform management components described in this specification. The API or other interfaces between the VMM, VM management software and the vPlatform components described is outside the scope of this specification beyond the situation where then they interact.
- **vPlatform Migration Controller Protocol** – A protocol between the Migration Controller and the Migration Engine components will be required by vPlatform management products. This protocol is envisioned to be defined in a later version of this specification (or another specification). Initially, it's envisioned that the Migration Controller and the Migration Engine would be provided by the same vendor, so interoperability of the protocol isn't immediately critical.
- **vPlatform Migration Protocol** – A protocol between the Migration Engine on the source and destination platforms involved in a migration is outside the scope of this version of the specification. It's envisioned that the vPlatform migration functionality would most likely be integrated with the VM migration system. The VM migration system is likely to include a VM migration protocol addressing many of the same issues as is needed for migration of the vPlatform, so it is envisioned that the vPlatform state might be added to the VM state being moved. The additional security requirements for the vPlatform state could be met by components outside the VM migration system.
- **vPlatform State Details** - Certain more advanced topics regarding the management of the virtualized trusted platform has been put off for a future version of this specification. This includes the detailed structure and semantics of the vPlatform's state information that might be backed up/restored, migrated or deleted.
- **Migration Controller Event Protocol** – The Migration Controller might support an event notification interface allowing Remote Challengers to rapidly learn when vPlatforms are moved. This protocol is also outside the scope of version 1.0 of this specification and could be a topic for a future revision. However, it's worth noting that the IF-MAP [8] protocol from the TCG could be a good candidate for this functionality if appropriate schemas were defined.
- **VM or VMM Involvement in Migration** – The current version of this specification focuses on minimizing VM changes associated with virtualization. However a number of more advanced features could be provided by virtualization awareness including allowing the VM owner (potentially different from the system owner) to be involved in the VM migration decision and allowing it to proactively evaluate the proposed target platform before migration occurs.
- **Policy** - Several usages of policy are discussed in this specification and the detailed format and semantics of those policies are left for future specifications. This includes the migration controller's migration policy (that could be attested by a Remote Challenger) and

the backup/restore policy that restricts when/where a backed up vPlatform can be restored and re-instantiated.

- **Dynamic Physical Platform Changes** - Another potential topic for a future version of this specification is handling dynamic physical trusted platform changes gracefully in order to avoid impact on the VMs. For example, system motherboard containing the system pTPM may fail and need to be replaced in the field. If vTPM implementations are using pTPM internal state (so they are anchored to the pTPM), then they need to be re-anchored to the new pTPM before they can continue to be used on by the associated.

4 Virtualized Trusted Platform Architecture

This section discusses the high level conceptual architecture and its components in order to establish a common paradigm for the lower level specifications that focus on how to offer a virtualized trusted platform for a particular usage or platform type. Note that many different approaches can be used to offer a virtualized trusted platform. This section will focus on one generic model in order to establish the terminology, components, and relationships between the components. Implementations and lower-level specifications may choose to alter the model to fit their usage and requirement set, as long as they adhere to the normative requirements set forth in these specifications.

4.1 Reference Architecture

This section illustrates and discusses the conceptual reference architecture for virtualizing a trusted platform. Section 4.1.1 describes the model used as the basis for the majority of the specification, while the other sub-sections show some additional possible alterations highlighting their differences. This section does not show all or even most of the possible reference models but rather just provides a single base and shows how minor alterations to the components still result in consistent reference models.

The diagrams and terminology in this section are based on the concept of layering. Layering is the focus because platform virtualization is fundamentally about the creation of virtual hardware instances associated with each virtual machine (or another high layered virtual platform instance). Each following sub-section discusses the components and interactions between components to establish an introductory role for the architectural components that will be expanded upon in other sections.

The highest layer is considered Layer N, and underlying layers are considered N-1, N-2, etc. The layer naming is based upon the layer's position is relative to the top layer's virtual machine. For clarity, the bottom layer can also be referred to as Layer 1 as its always at the bottom of the layer stack since it's the physical (non-virtual) platform.

4.1.1 Non-Virtualized Platform

A fundamental premise in defining virtualized trusted platforms is to build on the architectures defined for non-virtualized trusted platforms. Architectural changes to support virtualization are minimized for purposes of simplicity and compatibility. In the non-virtualized example shown in the following diagram, the operating system and applications run directly on top of the platform hardware.

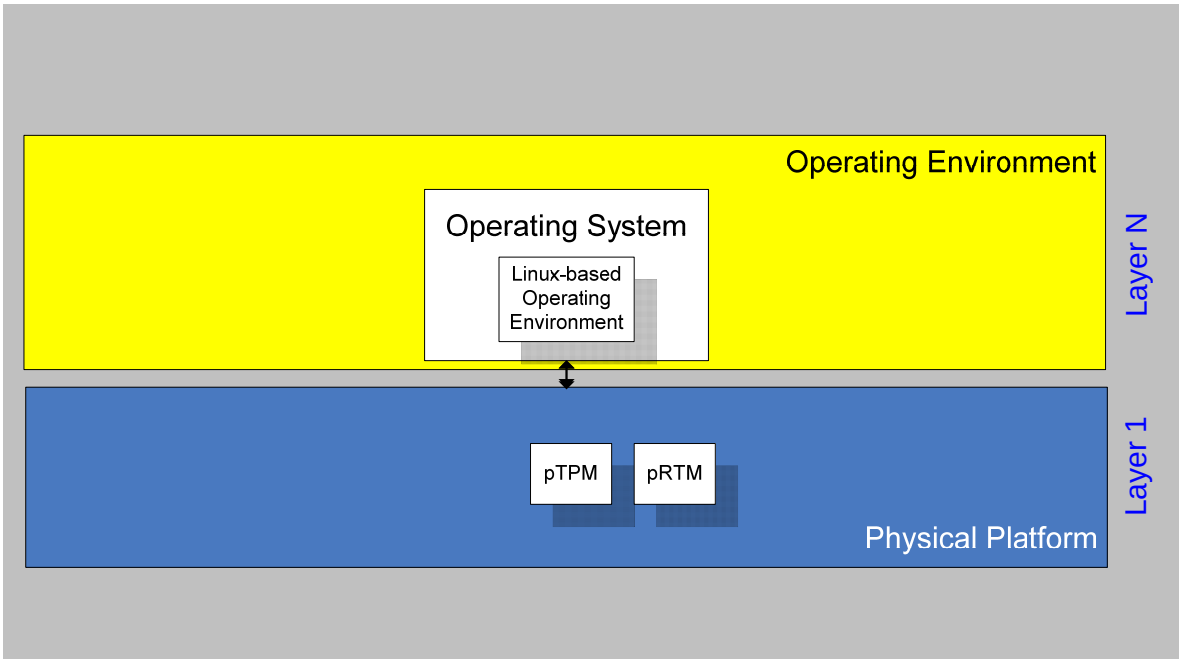


Diagram 1: Two layer (non-virtualized) architecture

Note that in this diagram, the physical TPM and CRTM are considered part of the physical platform and provide services to the Operating System.

The two layer model in diagram 1 represents a non-virtualized platform. This specification focuses on the components and interfaces required to virtualize the physical platform such that multiple independent Guest OS instances can operate in isolation. This virtualization is possible by the introduction of additional abstracting layers between the physical platform and the virtual machine. The upcoming sections will discuss several models for how this layered abstraction can be provided.

4.1.2 Three Layered With Privileged VM

This subsection shows the baseline model for use as the reference for the remainder of this document. This model includes three layers to the system. The top layer of the model is where the Virtual Machines reside. Each VM operates in a compartment, isolated for the other VMs leveraging the services of the Virtual Machine Manager (VMM).

The second layer of the model includes the VMM. The VMM layer provides services to each of the VMs which ensuring that they operate independently. The VMM vendor may choose to instantiate a special privileged VM that is capable of performing more complex or risky services on behalf of the VMM.

Finally the lowest layer of this example includes the physical hardware. This layer provides the physical roots of trust of the trusted platform. The physical roots of trust typically would offer trusted platform services for the VMM layer while the VMM would offer virtualized equivalents of those services to the individual VMs.

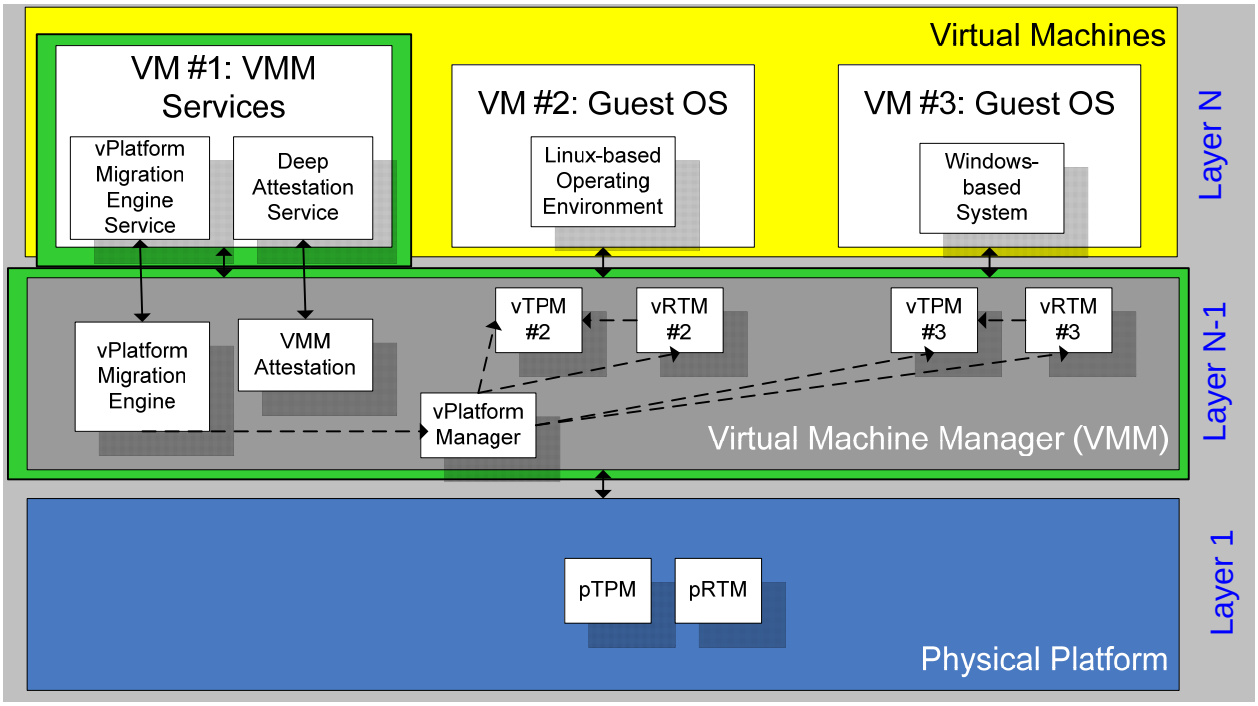


Diagram 2: Three layer architecture with privileged VM

At the top of diagram 2 are three virtual machines; two are running isolated operating system instances running user services. Traditionally, these instances might have been running on dedicated systems but the owner has elected to apply virtualization technologies so that both may run on the same hardware (e.g. this might be done to take advantage of unused CPU cycles). VM #1 is a special purpose VM that services requests from the VMM in layer N-1. Sometimes it is easier to implement more complex operations in a special privileged VM that help simplify the core VMM and security/fault isolate the core VMM from those services. The diagram shows dashed lines to indicate control-oriented flows (possibly with data) and solid lines for data-driven flows.

4.1.2.1 Layer N Components

For this example, the “VMM Services” VM #1 is providing the Deep Attestation Service that can be used to create attestation evidence about the state of the VMM and a Migration Controller for performing authorized management operations for the VMs such as a migration or shutdown request. The migration service might accept requests from a centralized set of management services that control the operation of a cluster of virtualized systems. The specific contents of any privileged VMs are implementation specific, but this reference architecture recognizes that special privileged VMs might exist.

Also shown at layer N of diagram 2 are a pair of Guest OSes each running independently. This example shows a Linux-based and Windows-based VM merely to point out that the VMs are independent and thus do not need to use the same operating system or applications. In this example, VM #2 and VM #3 are each potentially leveraging services offered by the privileged VM #1 but are unaware of its existence because their direct communications are to the VMM.

4.1.2.2 Layer N-1 Components

Moving down a layer, the VMM is providing the virtualization services that enable the VMs to execute in isolated environments and provides each VM with services such as vTPM and vRTM trusted platform services. In some models, the vRTM could be implemented within the Guest OS layer while this diagram shows them in the VMM. Within the VMM, a vPlatform Manager component could exist that is responsible for the creation, instantiation (if migrated or restarted), removal, cleanup and management of the virtual platforms associated with each VM. Similarly, the VMM could contain components that support the remote attestation of the VMM layer and components for servicing migration requests based upon requests received from the VMM Services

privileged VM. The migration oriented events likely would require the assistance of the vPlatform Manager to suspend and prepare the VM and its vPlatform for migration to another system.

4.1.2.3 Layer 1 Physical Components

At the bottom (N-2) layer of this simple example is the physical hardware layer so identified as layer 1. The physical platform layer would contain all the resources and devices typically found on current non-virtualized systems including a trusted platform with a physical TPM and RTM. In some usages, the VMM may leverage the physical TPM to protect its operation by sealing data to critical, relatively static measurements of portions of the VMM or potentially even a VM

4.1.3 Three Layer without Privileged VM

This subsection shows another three layered virtualization model except without the existence of a privileged VM to aid operations from the VMM. The choice of whether to allow for special privileged VMs lies with the product implementers. The use of isolated, privileged VMs seems to have benefits for isolating functionality such that it is unable to directly access the address space of the entire VMM. However, some implementations may have other ways to offer similar functionality or prefer a monolithic approach.

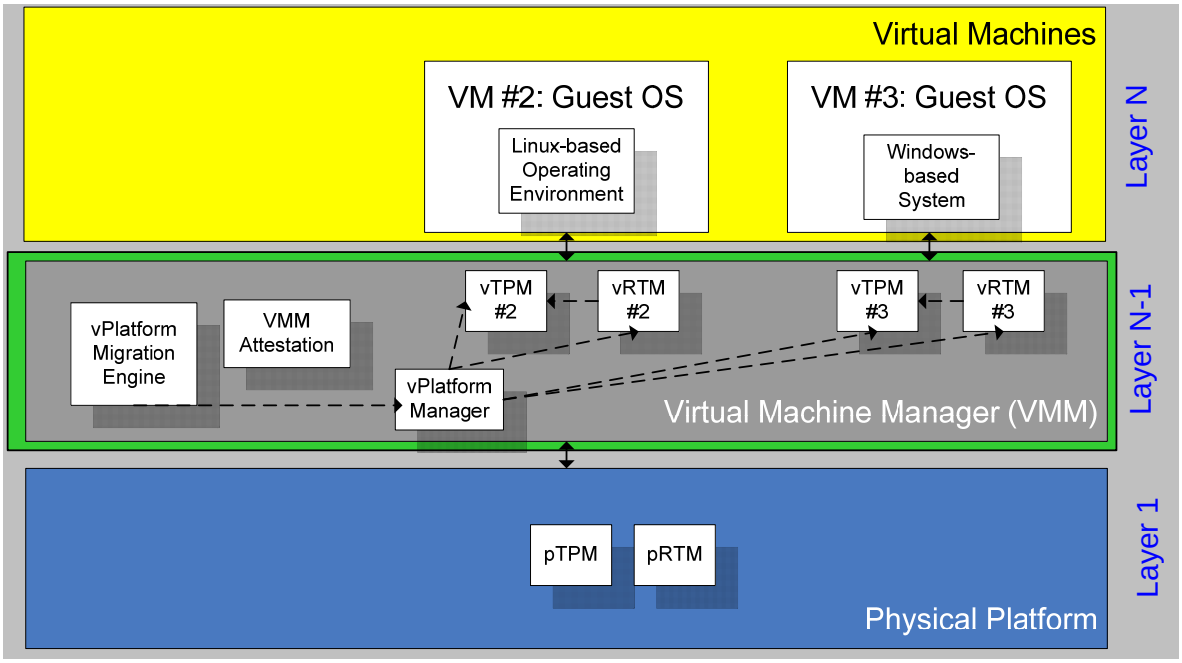


Diagram 3: Three layer architecture without privileged VM

As discussed, this potential architecture is consistent with the reference architecture shown earlier in that it contains the same core features just without a special “services” VM. In this diagram, the example services (Deep Attestation and Migration Controller) have merged into the VMM layer.

4.1.4 Four Layer (Nested Virtualization)

This sub-section discusses a four layered virtualization architecture that while more complex is consistent with the conceptual reference model described in this document. Instead of one virtualization oriented layer this model includes two between the hardware and VM #1 in Layer N. Because there is a virtualization layer running on top of (or within) another virtualization layer, this is also known as nested virtualization.

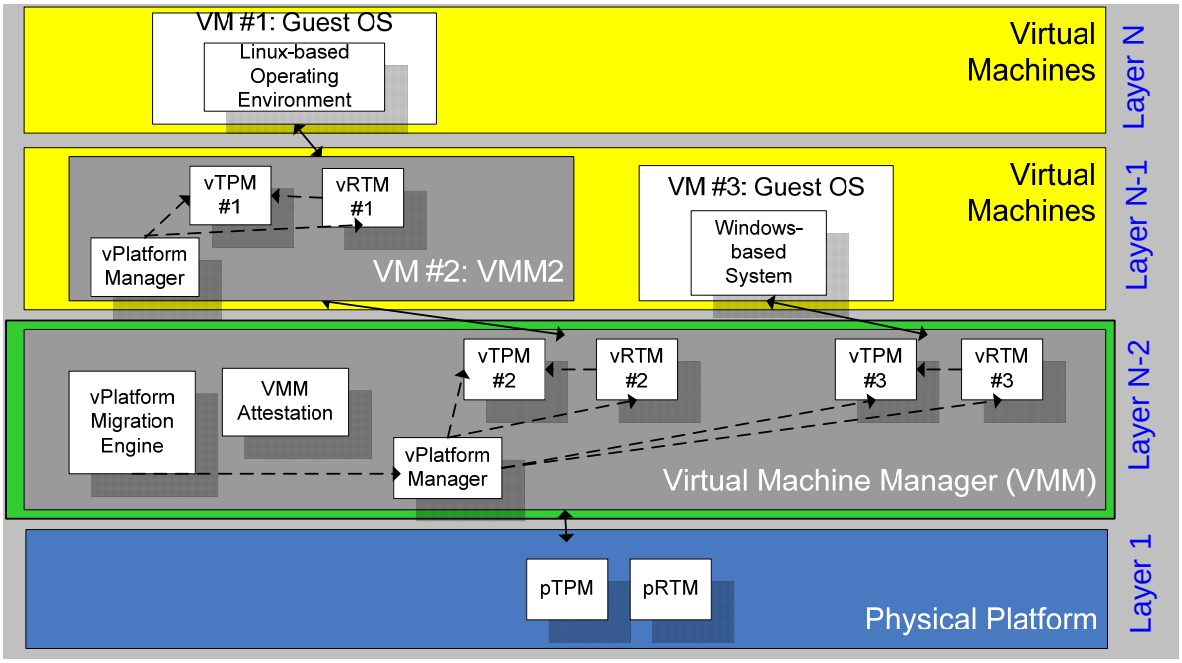


Diagram 4: Four layered architecture without privileged VM

This example shows a model including two virtualization layers between the VM #1 in layer 1 and the physical platform.

Note that this top layer of the model is referred to layer N, since it's the highest layer in the virtualization model. Other layers are named relative to their layering below layer. In a 4 layer model shown, there exist two virtualization layers between the VM and the underlying hardware platform. From the perspective of VM #1, it is at Layer N and the physical platform is at Layer N-3 (three layers down); however, from the perspective of VM #3, it is running in a three layered architecture where the physical platform is at N-2 (two layers down). The bottom layer can always be referred to as Layer 1.

4.2 Challenges for Virtualized Trusted Platforms

This section discusses some of the general challenges with virtualizing and sharing a trusted platform across several isolated VMs.

4.2.1 Protecting Virtual TPM Storage

In a non-virtualized trusted platform, the TPM provides protected storage and capabilities to perform operations on the protected storage such that attacks on the platform's software are unable to steal the protected secrets. This is possible because the TPM has a separate address space inside the TPM that is not directly accessible to platform software. However in a vTPM, there can be different threats to its protected storage that need to be considered.

For example, one model for vTPMs involve the N-1 layer (e.g. VMM) housing the vTPM state and protected storage and exposing the TPM ordinals to the VM. This approach is similar to the pTPM because malware in the VM is unable to directly address the protected storage in the vTPM. However, if malware were to invade the N-1 layer it might have direct access to the protected storage for all the vTPMs on the system unless the N-1 layer was using additional protections such as sealing to the trusted platform in the N-2 layer (which might also be virtualized). By having the N-1 layer (e.g. VMM) seal the vTPM state to the N-2 layer (e.g. pTPM) this would prevent theft or undetected alteration of the information while in VMM memory, however performance issues could arise with this approach if the sealed information is required often for vTPM operations.

4.2.2 Protecting vTPM Secrets across Reboots

As discussed in section 4.2.1, several expected models for vPlatform might store the vTPM state in memory, so this state could be subject to threats. Additional threats exist to the vTPM state when considering migration (discussed in section 4.2.7.1) and reboots. TPMs contain some state that is expected to persist through a reboot of the platform, while other state is expected to be reset. For example, a TPM might contain an Endorsement private key that needs to persist and be kept protected across power events of the system, while the same TPM contains PCRs that need to be reset to a prescribed initial value.

> The security of the vTPM requires that both the vTPM persistent state MUST survive a reboot, while non-persistent vTPM state MUST be reset to a value dictated by the TPM specification.

The vPlatform Manager is responsible for safely providing the following services:

- Creating a new vTPM instance (upon new VM creation)
- Instantiate a previous (saved) vTPM instance when needed
- Initialize portions of the vTPM during the boot
- Maintain the non-volatile vTPM state in a manner that the state will persist through power events and VM reboots
- Securing and storing vTPM non-volatile state on persistent storage when the VM is suspended or shutdown
- Delete vTPM instance state when VM is removed or migrated to another system

The primary challenge to safely offering these services is how to maintain the vTPM state on persistent storage that is protected from a variety of attacks including: theft, duplication, alteration or destruction. Similarly the vTPM state needs to be synchronized with the persistent storage state often enough to avoid loss of major state changes in the face of platform power loss as per the persistent state requirements in the appropriate platform specifications.

4.2.3 Attestation

Each VM is associated with a virtual trusted platform instance that provides security capabilities such as vTPM and vRTM. The virtual trusted platforms can leverage the security of one or more lower layers (e.g. VMM) that also have their own trusted platforms. Since each layer on a virtualized platform is bound to an independent trusted platform instance (including a pTPM or vTPM instance), remote attestation for establishing trust in an individual layer could require attesting multiple lower layers' trusted platforms.

In an initial attestation step, a Remote Challenger will typically request integrity evidence about the trustworthiness of software running inside of a VM. The VM is frequently the starting point for the attestation as the Remote Challenger likely already knows how to contact the VM (IP address), since it's about to access the VM's services or resources. While this is envisioned to be the common case, the model certainly allows the Remote Challenger to start the attestation by contacting another layer on the system (e.g. if it remembered how to attest the VMM from prior interactions with the system).

Initially, the Remote Challenger is unaware that the Attestation Agent it is speaking with is running in a VM (as opposed to directly on the physical layer). Therefore it's critical that the attestation protocol used by the Attestation Agent be the same protocol as when the Attestation Agent is running non-virtualized.

> The Remote Challenger MUST be able to use the same base attestation protocol when communicating with the Attestation Agent running in a VM as when it communicates with a non-virtualized Attestation Agent. However, the attestation protocol MAY offer additional features that are unique to virtualized platforms (e.g. component types only found on virtualized platforms like a VMM virtual switch.

During the attestation, the Remote Challenger MUST be able to detect that the virtual machine is operating on a virtual trusted platform (e.g. vTPM) and therefore has its security dependent on the correct operation of an N-1 layer (e.g. VMM).

This virtualization detection would be possible by having the VM's AIK credentials indicate that the VM's trusted platform is virtualized.

At this point, the Remote Challenger needs to decide if it is willing to trust the services of a virtualized system. If so, the Remote Challenger could complete the attestation of the VM and as part of the attestation learn how to attest the N-1 layer (VMM). Because a malicious VMM could tamper with the integrity evidence of the VM, it is envisioned that many challengers would attest the N-1 layer as well either via the VM or directly using the VMM contact information established during the VM's attestation. See section 4.2.3.2 for more information about attesting the lower layers.

4.2.3.1 Credentials

One important property of the VM attestation is that the VM needs to be able to assert in a protected manner that it is running on a virtualized trusted platform. The platform and AIK credentials include an optional certificate field called rtmType that indicates information about the trusted platform. One of the values for the rtmType field could indicate whether the underlying platform (e.g. TPM) is physical or virtual. This value could enable the VM to assert its operating on a virtual trusted platform when it presents the AIK certificate to a Remote Challenger.

A second piece of information that might be needed by the challenger is how to contact (directly or indirectly) the N-1 layer below this VM in order to pursue a deep attestation. This information could be encoded within the AIK certificate to provide the IP address (or a URI) for reaching the attestation service for the N-1 layer. If the N-1 attestation information is included, the certificate signature provides a cryptographic binding between the VM trust properties and N-1 layer's attestation services.

However, in an environment where VM migration occurs, a new certificate would need to be minted after the VM has moved to indicate how to attest the new platform's N-1 layer. Creation of a new certificate has potential logistical issues as it's difficult to prevent other parties from using the old (potentially cached) certificate, so the old certificate would need to be revoked. Similarly, injecting the new certificates into the running VM could be difficult since different parts of the system may have it in use in caches or its address space so coordinating a transition in real time would be an issue.

One solution to this problem might involve including a layer of indirection so the N-1 attestation service information doesn't need to change after a migration event. This is discussed below in section 4.2.4.

4.2.3.2 Deep Attestation

After an attestation of a VM has succeeded, a Remote Challenger might wish to attest the layers below the VM to determine if they are trustworthy enough to not modify the VM behavior and attestation reporting. The concept of iteratively attesting each individual lower virtualization layer in order to establish the trustworthiness of a VM (and its underlying trusted platforms) is known as a "deep attestation.

Because the N-1 layer might also be operating on top of a virtualized trusted platform (e.g. in a 4-layered system with nested virtualization), the Remote Challenger may need to repeatedly attest virtualization layers until it reaches the bottom-most layer (operating on top of a physical trusted platform). The Remote Challenger could decide to stop the attestation earlier if it chooses to trust the virtualized system based upon the attestations of a subset of the layers. Challengers should be careful when not attesting all of the virtualized system's layers because a lower layer could maliciously influence the higher layer's attestations.

4.2.3.3 Deep Attestation Layer Bindings

One form of attack against a virtualized trusted system involves deceiving the Challenger into thinking it's performing a deep attestation of the layers on a single physical platform when instead

redirecting the attestations to other systems. As discussed above, an individual attestation would cover the requested aspects of a single layer within a virtualized trusted platform. This attestation could provide information about how to contact the attestation services for the layer below to perform a deep attestation. If malware was able to cause this contact information to contain the address of another physical system that has a healthy N-1 layer, the deep attestation would attest this other system and might believe that the VM is trustworthy when running on the healthy N-1 layer that actually is on another platform.

Another similar attack could involve the N-1 layer acting as an active man-in-the-middle forwarding attestation evidence from another N-1 layer in response to attestation requests. In this attack, a proxy could exist on the N-1 layer's attestation software that would forward any attestation requests to another "clean" system's N-1 layer. When it received the response from the clean system, it could forward it to the Challenger as its own clean posture.

> Therefore, the attestation from a virtualized platform MUST include verifiable evidence that the VM's attestation is from the same physical platform as the N-1 (VMM) attestation.

4.2.4 Supporting Different vTPM Version

Another challenge potentially faced by virtualized trusted platform implementations is supporting vTPM that are functionally compliant with a different version of the TPM specification than the pTPM attached to the platform. Another similar challenge is supporting multiple VMs where each VM expects to have a vTPM associated with it that is compliant with a different specification then other VM's vTPM. The VPWG believes it is important to support these usage models to allow maximum flexibility for the VMs run on a platform. Such vTPM specification flexibility by the vPlatform increases the likelihood that the vPlatform could support a wide range of VMs (improving portability and migration opportunities). For example, a VM designed to operate on TPM 1.0 compliant systems might be desired to run on a virtualized platform with other VMs expecting TPM 1.2 compliant platforms, so if the VMM can support operating of multiple vTPM with different features and ordinals this is a big win for customers. TCG also benefits if a VMM can be upgraded to manufacture and operate vTPM compliant with the latest or evolving TPM specifications as this allow for earlier adoption by customers and easier testing by vendors. Finally an older platform that might contain a version 1.1 compliant pTPM might be deployed by a customer who wishes to leverage the system to support more recent VMs expecting semantics of the 1.2 specification. Since the vTPM can be manufactured and operated by the VMM, it might be possible for the VMM to map its functionality to an underlying pTPM supporting an older version.

4.2.5 Field Upgrade of vTPM

Because several of the discussed models for implementing a vTPM involve a predominantly software based vTPM with local state contained in the VMM, this eases the process of supporting field upgrades of the vTPM to different versions. While it's highly desirable for many customers that this upgrade be possible while the associated VM is running, it's envisioned that many of the upgrades would require a reboot or even that the VM would be quiesced or shutdown while the upgrade was occurring.

Another challenge to upgrading a vTPM is the impact to the vTPM's credentials. The EK credential contains information about the make, model and version of the TPM associated with the trusted platform. When a version of a TPM (physical or virtual) is changed, the EK credential and its associated AIK credentials now contain stale information that misrepresents the state of the trusted platform. Ideally the VMM and its CA would be capable of replacing the stale certificates, however it is non-trivial to know that the newer certificates are being used by the VM and other parties that might have previously received a copy (see section 4.2.7.3).

One interesting difference between upgrading a virtual TPM and a physical TPM is that the physical TPM upgrade typically is under the control of the TPM owner. In the case of the virtual TPM, the TPM owner or a privileged entity such as the VMM (TPM manufacturer) might authorize the upgrade of the vTPM (e.g. patching the vTPM code if it were a software module).

vTPM that provide a mechanism to allow a VM to apply a logic (e.g. firmware for pTPM) upgrade of a vTPM, this upgrade SHOULD NOT impact the other vTPM instances associated with different running VMs as this could cause instability and security issues.

4.2.6 vTPM Backup and Restore

Virtualization of a trusted platform introduces some unique opportunities and associated challenges. The vPlatform Manager is envisioned to be able to access most if not all of the state of a running vTPM, since it's frequently providing the TPM ordinal functionality in VMM software. This ability makes it possible for the vPlatform Manager to integrate with a backup software system to backup and later restore the full (including private state) vTPM. However, it's essential that the backup vPlatform image be strongly protected from attack (paralleling the discussion in section 4.2.2 and other threats for TPM state storage).

Any time the vTPM state is written to non-private, persistent storage the vTPM state MUST be provided analogous protections as to when it is stored inside a physical TPM (e.g. protected from unauthorized access or undetected tampering).

In order for backup/restore of vPlatform private state to be properly protected, it's critical that the state be protected from unauthorized access and alteration when outside of the VMM's vTPM instance (e.g. when on backup media). During a restore of the vTPM state, the authenticity of the backed up state needs to be verified (unmodified) and that restore policy limiting situations when/where the vPlatform state can be restored is enforced. The restore policy may need to be attested by a Remote Challenger in order for establishing the trustworthiness of a vTPM, so a service needs to be available allowing for the restore policy to be evaluated.

4.2.7 Migration

One of the major challenges faced by virtualized trusted platform implementations is how to support migration without sacrificing security. Traditionally, trusted platform roots (e.g. TPM) were bound to the hardware platform such that they could not be readily moved to other systems. In a virtualized platform, the VM's trusted roots might not have a strong (cryptographic) binding to the underlying hardware roots thus potentially allowing the vTPM to be duplicated, migrated or deleted. It's essential that the virtual trusted platform prevent unauthorized duplication, migration or deletion of the virtual trusted roots.

4.2.7.1 Duplicated vTPM Instances

Duplication of a vTPM is a significant concern as this would violate the single instance property required of a TPM's unique secrets. If a vTPM could be duplicated, this might allow two parties to be operating independently and appear as the same party on the network. Each vTPM's state could diverge and potentially lead to issues for attestation (one party might be able to re-use the other instance's attestation information).

Therefore, it is essential that any migration scheme MUST be able to ensure that after a vTPM instance has successfully been moved to the destination system and confirmed to be usable, that the source system permanently destroys the original vTPM instance and renders any backups to be unusable on this platform unless the vTPM instance is migrated back to the platform.

4.2.7.2 Destination vPlatform Less Trustworthy

Another major concern potentially introduced by support for migration of a vTPM instance arises when the destination virtualized platform offers differing security properties/assurance from the source platform. VM migration implementations could choose to allow movement of vPlatforms from more secure platforms to other systems that offer less assurance (the reverse is not a significant issue). For example, a vPlatform Migration Controller could support policies allowing a vTPM that is cryptographically protected by an underlying physical TPM to be moved to another platform that uses a less trustworthy VMM that does not cryptographically protect the TPM internal state using a physical TPM on the platform. Relying parties (Remote Challengers) might perform an attestation of the VM and consider it trustworthy based on the original platform's deep attestation

and then start using services of the VM. While the relying party is interacting with the VM, a migration event could occur causing the VM to move to a less trustworthy virtual platform on the destination system. The relying party needs to have a mechanism for determining whether the migration system for this VM might move the VM to a less trustworthy platform. If it does, the relying party might be willing to tolerate the migration if it can be notified when the migration occurs so that it can attest the destination vPlatform. One potential solution to this migration issue is to include contact information in the AIK credential for a trusted (but attestable) third party known as the vPlatform Migration Controller. The Migration Controller is a component that is integrated into the migration system (or VM management system) in such a way that it can enforce migration policy for VM and be attested by a relying party. The Migration controller contact information would remain the same regardless of whether a VM has been migrated or not, so could always be reached by challengers without the need to reissue the VM's certificates. The Migration Controller could be attested by the relying party to ensure trustworthy responses to migration requests and to learn the controller's migration policies regarding the VM in question. Specifically, a challenger wishing to perform a set of longer term transactions, might want to know if the Migration Controller might migrate the VM during the transaction to another virtualized platform that offers different and/or lesser security protections and factor this in to its trust decision about the VM.

4.2.7.3 Migration Impact on Credentials

If the vPlatform Migration Controller performs a migration of a VM and the destination platform has different security properties or manufacturer information this would result in the VM's credentials (e.g. AIK) containing incorrect information. For example, if a VM running on VendorA's Model 123 server that contains a VendorB TPM is migrated to a platform built by VendorZ Model 234 containing a VendorY TPM, then all the credentials used by the VM would contain old information (VendorA's Model123 with VendorB TPM) when running on the new platform.

Because the manufacturer, model and version information for the vTPM and vPlatform are contained in the AIK credential, having the information become incorrect could alter the trust decision made by the relying party. There are a number of possible approaches to addressing this problem including: issuing new credentials for the target system, having the Remote Challenger fetch new information from Migration Controller, or using more generalized virtual manufacturer information and having the challenger attest more information about the current platform to learn the correct information. Each of these options has issues that the implementer or deployer would need to address.

4.3 Architecture Components

This section discusses the role and responsibilities for the key components in the reference architecture described in section 4.1. This specification recognizes that numerous different implementation architectures are possible and appropriate to differing sets of requirements and platform capabilities, so this section will focus its discussion on the reference architecture described in section 4.1.2. This architecture includes the largest set of components, so allows for them to be described in more detail in this section. Note that many of the components described do not need to be standalone and could be combined with other trusted platform components while still resulting in a compliant implementation. Providing a more granular description allows for a more detailed understanding of each logical component's capabilities even if the lower level platform oriented specification or the implementer decides to combine the components.

This section includes a number of requirements provided to ensure consistency across the virtualization architectures defined in more detail in the low level specifications. In order to improve the requirements visibility, each normative requirement is indented and placed in its own paragraph.

4.3.1 Physical Platform

The physical platform refers to the set of base capabilities provided by the lowest level platform. These capabilities are normally implemented in hardware and firmware and are envisioned to be less subject to change than the higher layers in the platform. While the physical platform includes

the traditional CPU, memory, interface boards/chips, and disk, the focus of this section is on the trusted platform features used to enable TCG's trusted computing model.

In a virtualized system, the physical platform is shared amongst the various layers of the reference architecture including logically isolated VMs. In some cases, special hardware or firmware is present that is able to enforce the isolation of VM accesses; otherwise the VMM provides this capability. In some virtualization models such as para-virtualization, the VM might have direct access to a hardware device, to improve performance, particularly in situations where only one VM is given access to the device.

4.3.1.1 Physical Trust Roots

As discussed in section 2.2, the physical trusted platform typically offers a Root of Trust for Storage (RTS) and a Root of Trust for Reporting (RTR) embodied in a TPM chip on the motherboard (frequently on the LPC bus for PC Clients). The Core Root of Trust for Measurement (CRTM) might be housed in a chipset involved in booting the system like the BIOS. When present, the Root of Trust for Verification (RTV) and other higher level RTMs frequently exist in software leveraging the protections of the other trust roots.

> Physical platforms offering virtual trusted platforms as specified by this document SHOULD contain physical trust roots including the RTR, RTS as a pTPM and a CRTM (dynamic or static) to anchor the transitive trust chain measurements.

Prior to the initial measured boot of the platform, an authorized user needs to take ownership of the pTPM and enable the pTPM and pCRTM for measurement of the initially loaded software. How this operation occurs is outside the scope of this document, but is a prerequisite to using the pTPM to protect the higher layers in the virtualized platform. It is envisioned the steps performed would parallel those used if the platform was not virtualized.

Next the pCRTM (static or dynamic) would execute in order to measure the initial trusted code running on the platform and make the measurements available to the pTPM for future usage (e.g. attestation). This measurement starts the transitive trust chain for layer 2 of the virtualization architecture (frequently the VMM layer). These measurements might be used by future deep attestations of the VMs.

4.3.1.2 Credentials

As discuss in the TPM and platform specific specifications, each TPM has an associated set of credentials (X.509 certificates) that are used to assert properties of the TPM. These properties are cryptographically bound to knowledge of a secret private key stored in the TPM by the certificate issuer's signature.

> All pTPM that contain an Endorsement Key (EK) SHOULD have an associated Endorsement Credential for use during Attestation Identity Key (AIK) certificate enrollment.
>
> All trusted platforms SHOULD possess and make available a Platform Credential or equivalent (e.g. Unified Trust certificate [2] that asserts the security properties of the physical platform.
>
> The physical platform's Platform Credential MAY be expressed as an X.509 attribute certificate as defined in TCG Credentials version 1.0 or as an X.509 identity certificate as defined in TCG Credentials version 1.1.

The Platform Credential is also used during AIK certificate enrollment and frequently contains information that is copied into the minted AIK Certificates.

> Each AIK private key housed in a pTPM SHOULD have a corresponding AIK Credential (X.509 certificate) to assert properties of the pTPM and physical platform during deep attestation exchanges.
>
> If a physical platform possesses an EK, Platform and AIK certificates then each certificate MUST be compliant with the requirements in the TCG Credentials specification.

If a physical platform lacks an AIK certificate corresponding to a TPM-resident AIK private key, then the platform is unable to adequately attest its platform state in such a way that the challenger knows the attestation evidence is authentic (not spoofed by malware). The AIK certificate also potentially carries a number of security and assurance properties that the Remote Challenger may require to establish the trustworthiness of the platform as part of the attestation.

4.3.2 Virtual Machine Manager

This subsection focuses on the components that are envisioned to exist within the VMM or a privileged helper VM. Because the privileged VM architecture is largely an implementation decision (since VMM components could exist in the VMM or a helper VM), this discussion of the privileged VM will be minimized. Suffice it to say that any component that is discussed in this subsection might exist in the VMM or a privileged helper VM, but that will not impact the responsibilities of the component being described.

4.3.2.1 Initial VMM Boot

During the initial boot of the VMM or triggering of a DRTM initialization, the physical trusted platform uses the CRTM to measure and log information about the initial trusted code run on the system. Through a sequence of loaders being executed and taking measurements that are placed into the pTPM (if present); the platform loads enough functionality to start the initial trusted software layer in the platform. For an SRTM, this lowest layer is commonly the operating system kernel for non-virtualized platforms or a VMM for virtualized platforms. It is essential that the initial pieces of software that are executed on the system are measured and those measurements are extended into the pTPM. As the initial (measured) programs are run and start to load and execute additional pieces of code, it is important that every executed section of code also be measured and extended into the pTPM to continue the transitive trust chain. At this point in the boot sequence, there are no virtualization unique requirements for a platform that will eventually load and executive a VMM, so the platform needs to comply with the appropriate TCG platform specification.

> Once the VMM is operating, the platform TPM's PCRs MUST contain the extended (aggregate) set of measurements reflecting the operational state of the VMM in order to allow a Remote Challenger to attest the entire state of the platform.

The VMM is responsible for continually updating the pTPM's PCR to reflect its current state as significant changes occur including initiation of new virtual machine above it.

4.3.2.2 vPlatform Manager

After the VMM has booted and the transitive trust chain established, the VMM's trusted platform capabilities are ready for service. This subsection discusses the role and responsibilities of the vPlatform Manager. As discussed above, the vPlatform Manager fulfills the role of full lifecycle management of any vPlatforms instantiated on the platform to service VMs. The full lifecycle includes vPlatform: initial creation (fresh or after migration), association with VM, quiesce state changes, through deletion.

When the VMM is notified (possibly via configuration) that a VM needs to be instantiated, the VMM core contacts the vPlatform Manager component and indicates that a newly instantiated trusted platform instance for the VM is required. There are several usages where this could occur, so the vPlatform Manager is responsible for serving each of these types of requests:

- New VM is being instantiated, so the vPlatform Manager needs to be capable of creating a fresh (unused) vPlatform instance associated with the VM prior to the start of the VM boot, so the vCRTM can measure the initial piece of VM code. The details of what a fresh vTPM's state looks like are defined by the appropriate TPM and platform specifications (e.g. vTPM may be partially unusable waiting for owner to enable).
- Previously shutdown and saved VM is being instantiated during a VM boot, so the vPlatform Manager needs to be capable of obtaining the saved vPlatform state and decrypting and validating the state is authentic, then creating/updating the vTPM state structures in the VMM and possibly leveraging the pTPM for protections. Again the vCRTM

also needs to be prepared to measure the initial code run within the VM. For this usage, the vPlatform Manager resets the known volatile state (e.g. PCRs within the vTPM) and loads the non-volatile state into the vTPM instance.

- Previously quiesced (e.g. suspended) and saved VM is being re-instantiated. This usage carries the same responsibilities as the previously shutdown VM except that the VM was only quiesced, so the vPlatform Manager re-instantiates the vTPM with the same state including volatile aspects like the PCRs). This usage also requires the vPlatform Manager to decrypt the vTPM state off persistent storage and verify its integrity prior to re-instantiating it in the vPlatform.
- Local vPlatform Migration Engine component indicates that a live migration of an incoming VM is occurring and that a vPlatform needs to be instantiated using the state provided as part of the live migration vPlatform package. This event might include obtaining or installing keys in the pTPM that are used to decrypt and verify the integrity of the vPlatform state. If authentic, the vPlatform Manager would use this state to re-instantiate the exact vPlatform that was operating on the other platform including all internal vTPM state. This re-instantiation might resemble how a quiesced vPlatform is re-instantiated since it included volatile vTPM state.

Similar to the startup of a VM, the VMM might contact the vPlatform Manager indicating that a VM is being shutdown or quiesced. Again there are several usages where this event could occur, so the vPlatform Manager is responsible for supporting:

- VM instance being shutdown and needs its vPlatform non-volatile state preserved for future re-use.
- VM instance is being quiesced and needs to have all of its vPlatform state preserved for when a resume is requested. vPlatform Manager needs to secure the content of the vTPM prior to storage on persistent storage where adversaries could attempt to steal, modify or destroy the secrets. It's envisioned this would involve encrypting the vTPM state possibly using a pTPM resident key or a key sealed to some VMM state. The vTPM state also needs to be integrity protected.
- Local vPlatform Migration Engine indicates that a VM instance is being migrated to another platform, so the vPlatform Manager is responsible for using the private keys of the target platform to encrypt all of the state of the vPlatform for transmission over the network. In this usage, the VMM must notify the vPlatform Manager when the state has successfully been transported, so that the vPlatform Manager can erase and sanitize the vPlatform state off the system.
- Local physical platform is being shutdown or experiencing a power state change that will prevent the VMs from running. This usage parallels the quiesce or shutdown usage but applies to all vPlatforms on the system.

4.3.2.3 vTPM

The vTPM is the component responsible with offering a particular VM a dedicated TPM interface and feature set compliant with the TPM specifications so that VM can operate the same whether its virtualized or running directly on a physical TPM.

> The vTPM MUST be architected in such a way that the VM is unable to directly access the internal state of the vTPM without going through the exposed ordinals.

This restriction is required regardless of what malicious software is running inside the VM. It's envisioned that the vTPM state would be located inside the VMM (or a lower level including physical platform) to ensure that nothing in the VM is able to access the vTPM state directly. When a VM makes a vTPM hypercall into the VMM, the VMM is responsible for ensuring that the state associated with the vTPM for the calling VM is used for fulfill the request.

> The vTPM implementation MUST ensure that only the authorized VM associated with the vTPM has access to the vTPM's ordinals (and thus internal state). vTPM implementations

MAY allow access from the VMM for management or attestation related operations (e.g. linkage between an VM and vTPM instance).

Note that there are different strategies for implementing a vTPM with potentially different assurance levels for different threat models. Some products may focus only on VM originated threats, while others may choose a broader threat landscape including VMM-based attacks. The level of protection and assurance offered by a vPlatform are identified and certified as part of the common criteria process and in the virtualized platform protection profile. These security certifications can be represented in the vTPM's certificates for Remote Challengers.

The vTPM non-volatile state MUST be periodically synchronized to persistent storage to avoid potential loss during an abrupt power loss. The vTPM is also responsible for ensuring that its non-volatile state is properly mirrored on persistent storage, so that an unexpected power event doesn't cause a loss of current state. vTPM implementers need to weigh the performance impact of repeated state synchronizations to persistent storage versus the potential for lost state in the case of an abrupt power outage.

The vTPM state MUST be secured against unauthorized access, duplication, alteration and destruction when placed on persistent storage.

The vTPM state is envisioned to be encrypted and cryptographically signed when placed on persistent storage in order to avoid off-line attacks or other attacks where the VMM security protections are unavailable. It's essential that the private vTPM persistent state cannot be stolen by an attacker as that could lead to the creation of a duplicate TPM instance. Similarly it's critical that an attacker not be able to rollback the vTPM state with an older or modified copy of the vTPM and fool the VMM into loading it as a vTPM. The vTPM state could be protected using the TPM capabilities of the next lower layer (factoring in the synchronization issues discussed above).

4.3.2.4 vTPM Certificates

The vPlatform Manager is responsible for the creation of a new vTPM when a brand new VM is instantiated. Because the vTPM has not previously existed, it lacks an endorsement key (EK) credential (X.509) certificate and a platform credential asserting the qualities and public key information associated with the vTPM and vPlatform respectively. There are several possible models that a vPlatform Manager could choose to implement in order to create these certificates.

One option involves the vPlatform Manager using a secure enrollment protocol to an Attestation Certificate Authority (ACA, also known as a Privacy CA) present on the network to obtain the needed EK and Platform certificates. It's essential that a solution including an infrastructure ACA be able to rely upon that ACA to always be available and to return certificates very quickly. This is technically feasible as long as the ACA doesn't require human interaction during the registration process (prior to certificate issuance). The advantage of this approach is that it allows for the stronger protection of the ACA's signing keys, consistent registration and certificate content policies, and enables the ACA to include fewer EK and Platform certificate issuers in its trust anchor database used during AIK enrollment (EK and Platform certificates are only used by the ACA for privacy reasons).

A variation on the first option is for each virtualized platform to host its own ACA. The ACA might exist in a privileged VM (isolated from the Guest OS VMs) where is can provide some isolation of its operation and private state (private signing keys) from the rest of the system. In this case, the ACA would interact with the vPlatform Manager via hypercalls or directly if the ACA existed in the VMM. This approach has the advantage of being potentially higher availability (it's on the same platform as the newly created VM) avoiding network and power outage issues. However this approach might offer less security particularly if the customer threat model considers VMM originated attacks.

Finally, another approach is for the vPlatform Manager to have a built-in ACA feature to quickly issue the needed certificates. In this example, the vPlatform Manager might contain a template X.509 EK and Platform Certificate that the ACA can quickly fill out the few fields that different for each VM (e.g. EK public key and subject alt name) then includes a signature. This approach has the advantage of excellent speed and dependability while potentially offering lower security with no

isolation of the ACA functionality and reduced registration checking processing. This also might require the ACA used during AIK enrollment to trust EK and Platform certificates form a large number of issuers.

After the EK and Platform certificates have been issued, the vPlatform Manager needs to inject them into the new vTPM or find a suitable way to communicate them to the new VM. Ideally this would follow a process that is consistent for the TCG platform type that is expected by the new VM. Because the vTPM is likely a software entity, the vTPM is capable of having a huge amount of non-volatile storage associated with it (unlike today's TPM chips), so one option is to place the certificates in vTPM NVRAM. The TPM specifications provide a slot in NVRAM to store the EK and Platform Certificates (index TPM_NV_INDEX_EKCert and TPM_NV_INDEX_PlatformCert) where they can be found by the operating system software.

> The vPlatform Manager MUST store the EK and Platform (if applicable) certificates in the vNVRAM of the vTPM at the TPM_NV_INDEX_EKCert and TPM_NV_INDEX_PlatformCert indexes using the encoding defined for the platform type (e.g. struct tdTCG_FULL_CERT for PC Client) and including a TCG standard certificate.

Later after the boot of the new VM, an authorized entity may take ownership of the new vTPM and wish to create AIK keys and obtain an AIK certificate. The enrollment process for obtaining an AIK certificate takes the EK and Platform certificates and interacts with the ACA to create the certificate [reference to new ACA protocol spec]. Again the ACA might exist in the infrastructure, privileged local VM or within the VMM components. Note that a different approach for obtaining EK and Platform certificates could be taken then used for the AIK certificate. Generally the enrollment protocol for obtaining an AIK certificate can be slower (not holding up a VM boot) so using network infrastructure might be more reasonable. Having fewer AIK certificate issuers might allow Remote Challengers to have fewer trust anchor certificates in its database.

4.3.2.5 vCRTM

The vPlatform Manager is responsible for instantiating a vCRTM to measure the initial code run within the VM and populate the appropriate PCRs in the vTPM. It's envisioned (but not required) that most VM's vPlatform would leverage a static CRTM but implementation could attempt to provide a dynamic CRTM if required. There are many challenges posed by try to implement a VMM offering a virtual dynamic CRTM to VMs that are outside the scope of the 1.0 version of this specification. However, because the VMM is intimately involved with the initial load and starting of a VM, it's able to measure the initial code into the VM's vTPM. The vCRTM completes its role in the vPlatform after control over the boot is passed to another process loader/RTM that continues the transitive trust chain.

4.3.2.6 VMM Attestation

This component is responsible for obtaining and reporting upon the current operational state of the VMM. As shown in section 4.1.2, the VMM might be implemented in such a way that a portion of this component resides in a privileged, helper VM. When this component is sub-divided into a VM and VMM piece, the portion in the privileged VM is known as the "Deep Attestation Service". Note that this division is an implementation decision, so for this subsection we will only discuss the VMM Attestation component as if no Deep Attestation Service existed. Similarly, the Attestation Agent that executes within the VM could include functionality allowing the Remote Challenger to request information about the N-1 layer though the Attestation Agent without connecting to a new Deep Attestation Service component. Whether a Deep Attestation Service component is used or proxying the requests through the Attestation Agent, either way the VMM Attestation component is present to provide integrity information about the VMM. The VMM Attestation component will use the same attestation protocol stack and general feature set that the VM Attestation Agent uses.

However, while the attestation models and protocols are very similar, the types of components that VMM Attestation is able to report on can be very different. For example, the VMM's internal virtual router component that controls which VMs and intercommunicate could only be attested via the VMM Attestation component, but not the VM's Attestation Agent. Similarly, the VMM Attestation component might include awareness of the underlying physical trusted platform, since the VMM is

allowed to know the platform is virtualized (unlike the VM). Finally the VMM Attestation Service needs to take care not to report on the private internals of any vTPM on the platform. This isn't a concern for a VM since it's unable to directly access the internal state of its TPM.

- The VMM Attestation component (possibly via the Deep Attestation Service or the VM's Attestation Agent) listens on a network interface for requests from Remote Challengers to use the attestation protocol to obtain measurements about the VMM and evidence of trustworthiness using the VMM's TPM (pTPM in a 3 layer model). Authenticating the Remote Challenger and making an authorization decision about what information the challenger is allowed to retrieve
- Communicating using the attestation protocols to understand the Remote Challenger's request
- Determining what posture and component measurements to return (factoring in privacy and security policies)
- Creating the necessary attestation reports and returning them to the Remote Challenger
- Potentially initiating a software upgrade if requested to perform remediation by an authorized Remote Challenger (this is expected to be rare but could be used by the VMM controller to push out patches)
- Continue a deep attestation chain by facilitating contact with the layer below the VMM (in an architecture where this VMM isn't running on the physical platform)

See section 4.4.2 for more details on VMM attestation and deep attestation.

4.3.2.7 Migration Engine

The role of the Migration Engine is to transport vPlatform state packages securely across the network and provide them to the appropriate Local vPlatform Manager. Similar to the VMM Attestation component, the Migration Engine functionality could be implemented with a portion running in a special, privileged VM. When the Migration Engine has a portion running in a VM, that portion is shown in section 4.1.2 as the Migration Engine Service. For the purposes of this subsection, the Migration Engine will be discussed as if it's a solitary component.

The Migration Engine is responsible for:

- Listening for a vPlatform migration notification from the migration system
- Contacting the vPlatform Manager indicating the particular VM that is being migrated so the vPlatform Manager can package up the vPlatform state
- Handing off to the vPlatform state package to the VM migration software for transporting of over the network to the destination platform for re-instantiation.
- Notifying the vPlatform Manager once the vPlatform state package has been successfully received by the destination platform
- Listening for inbound vPlatform migration events
- Receive the vPlatform state package from the local VM migration software
- Passing the received vPlatform state package to the vPlatform Manager for instantiation
- Returning the completion status to the sending Migration Engine (so the vPlatform state package on the other side can be deleted)

For more information on migration and the Migration Engine, see section4.5.

4.3.3 Virtual Machine

This section discusses the aspects of the VM expected to be present in order for the virtual trust platform architecture to properly function. In the architecture figures above, the VM layer (not

including the privileged VMs) only contains operating system oriented components that we need to support without virtualization specific modification. Note that the privileged VMs are effectively treated as being part of the VMM layer, so are covered in section 4.3.2.

While it is possible for the VM to contain virtualization-aware code (as discussed in section 2.1.1), the virtualized trusted platform architecture discussed in this specification focuses on supporting operating system images lacking such specific awareness. However the VM will still be interacting with the VMM likely without specific virtualization modifications, so these interactions are the topic for the remainder of this sub-section.

> The vPlatform (particularly the vTPM) MUST offer ordinals and functionality consistent with the interfaces described in the TPM specification in such a way that a non-virtualization aware VM can operate equally well on a physical or virtual trusted platform.
>
> The vPlatform MAY offer additional ordinals or features beyond those defined in non-virtualized trusted platform specification, however the use of these extensions MUST NOT be required to be used by a VM in order for it to function equivalent to how it operated on a non-virtualized trusted platform.

The remainder of this sub-section will assume a non-paravirtualized VM.

4.3.3.1 Transitive Trust Chain

The code running within the VM will expect an underlying physical platform including the physical trusted platform roots (pCRTM and pTPM). As discussed in section 4.3.2.5, when the VM is running on a vPlatform, the vCRTM is responsible for measuring initial VM code and placing the result in the vTPM. This measurement provides the start of the transitive trust chain for the VM and because the vCRTM is measured by the VMM allows a linkage down to the VMM's transitive trust chain. In order for a Remote Challenger to retrieve the full chain down to a physical trust root, it needs to perform a deep attestation and connect the chains. Note that the initial measurement by the vCRTM, is done prior to the start of the VM's operating system; therefore is not consider a virtualization-aware portion of the operating system.

As the operating system boots, the trusted platform measurement roots within the operating system continue to be responsible for measuring code that is subsequently loaded and executed, thus continuing the transitive trust chain. Again, this is not a virtualization specific feature of the operating system and is subject to requirements from the appropriate TPM, trusted platform and attestation specification. When the trusted loaders within the operating system store the measurements into the vPlatform, the VM performs the same operations as it would if running on a physical platform but those operations are transformed by the VMM into vPlatform ordinals. At the end of the VM's operating system and application boot, the vTPM associated with the VM now contains a full set of measurements that can be used for later attestation.

4.3.3.2 Attestation Agent

In order for an operating system to support a remote attestation capability, the OS needs to include a service that speaks an attestation protocol known by the Remote Challenger. Therefore, it is not a virtualization specific service (paravirtualization) that needs to be present to support attestation when the operating system is running in a VM. However, the attestation protocol itself may include virtualization specific features that the Attestation Agent might choose to support. The ideal attestation protocol would not require support of such features in order for deep attestation to operate.

As discussed above, the Remote Challenger learns that an entity is virtualized by looking at properties asserted within the VM's AIK certificate from the attestation. The VM itself should have no reason to look within the certificates to determine whether these properties exist or to change it behavior or else it would be considered paravirtualized. Therefore, the Remote Challenger is able to determine an entity is virtualized and learn how to attest lower layers below the VM without virtualization awareness within the VM.

One layer-binding model described in section 4.4, would leverage virtualization specific features of an attestation protocol that could optionally be provided by the VM's Attestation Agent to ease the attestation effort potentially required by a Remote Challenger. This discussion will be provided below more in context, so it's possible the Attestation Agent could optionally support virtualization oriented features present in an attestation protocol but this is not required.

4.4 Attestation Components

This section discusses the expected capabilities of the components that comprise the attestation system for a virtualized trusted platform. The description leverages an example architecture to ease understanding of the role and relationships of each component involved in an attestation. As usual, implementations may merge one or more of the described components with existing components providing other features as part of a product solution.

As discussed in section 4.2.3, the attestation system provides a service to authorized Remote Challengers to request integrity evidence of the virtualized system (typically one layer). This integrity evidence is cryptographically linked to the platform's trusted roots (e.g. vTPM) and can be verified by the remote party. In order to better understand the steps involved in the attestation, the subsequent sections will provide more depth on each step in the process as illustrated in the diagram 5 below.

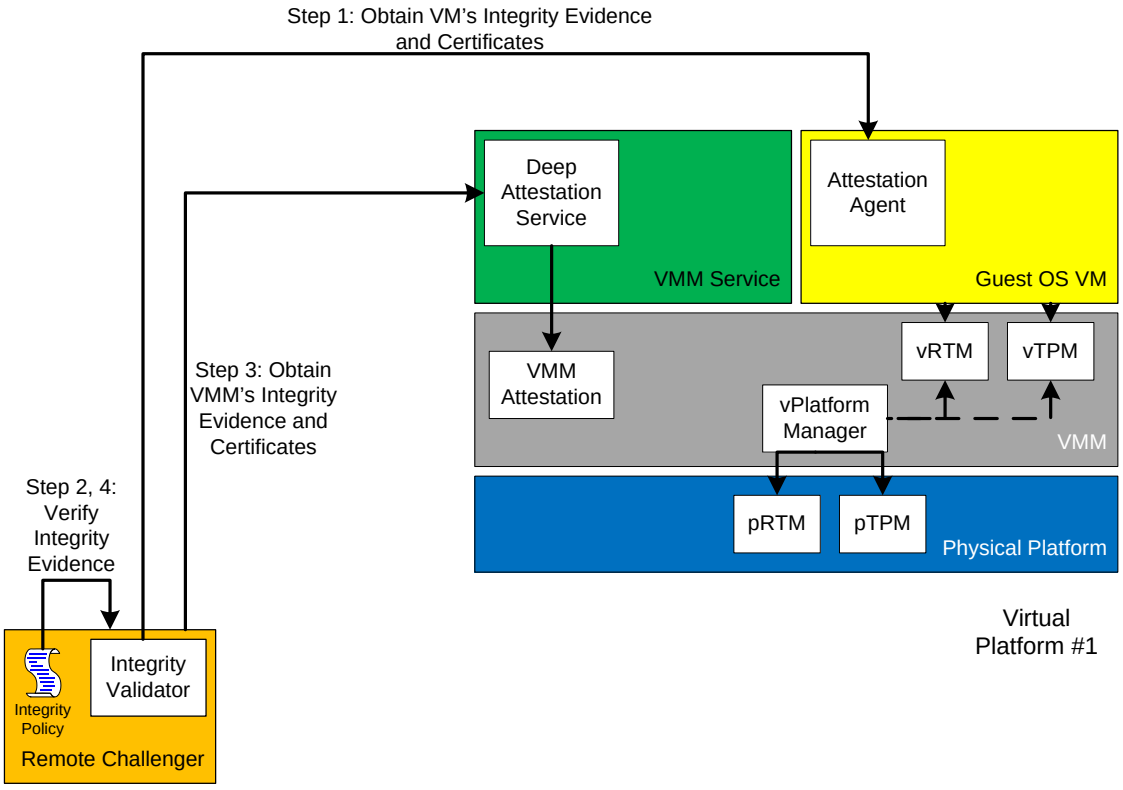


Diagram 5: Component Interactions during an Attestation of a Virtual Machine

Diagram 5 shows a Remote Challenger (lower left) attesting a VM in step 1 and following up with a deep attestation of the VMM leveraging a Deep Attestation Service in a separate, privileged virtual machine. This architectural approach allows the VMM to not communicate directly on the network but rather to leverage a privileged, contained environment for offering services to the network. Note that implementations may choose to have the Attestation Agent also support this functionality, but

this approach could be considered para-virtualization because the Attestation Agent would now need a special interface to the VMM to request attributes about the VMM's integrity.

4.4.1 Virtual Machine Attestation

Initially, the Remote Challenger decides that an attestation is necessary to determine the level of trustworthiness of the remote platform (Virtual Platform #1) that it wishes to use for some purpose. Because the challenger plans to leverage some aspect of the remote platform, the challenger typically would already know the IP address of the Guest OS VM. (Note that in other circumstance a different process for attesting the remote system might be beneficial, but this is envisioned to be much less common.)

Initially the Remote Challenger would establish a security-protected session with the Attestation Agent on the Guest OS VM.

> The security protections MUST be capable of providing strong cryptographic platform or user authentication in conjunction with cryptographic integrity and confidentiality protection for the exchanged attestation information.

At this point the Remote Challenger is not expected to know the remote platform is virtualized so would act the same as if it were attesting a non-virtualized system. The Attestation Agent running in the VM does not constitute making the VM paravirtualized as the agent would be required even if the system wasn't virtualized (but supported TCG attestation).

The Attestation Agent is listening on the network for attestation requests. It is expected that several attestation protocols will be defined over time. As of version 1.0 of this specification, no complete specification of an attestation protocol for virtualized systems have been published, but it's expected that the PTS Protocol Binding to IF-M will provide a good solution leveraging the Trusted Network Connect (TNC) suite of protocols [5]. Therefore, a future revision of this specification will require a particular standard protocol when it is published. In the meantime, this specification will establish some required behaviors of the attestation protocol to ensure security, privacy and reasonable operation of the attestation.

> After an initial handshake to establish and authorize the identities of the parties involved in the session, the Attestation Agent SHOULD support checking a privacy policy to determine whether the challenging party is authorized to obtain information about this platform.
>
> The authorization checks MUST leverage a strongly authenticated platform or user identify of the challenger in addition to other information about the session already known (e.g. IP address).
>
> The privacy policy SHOULD express granular enough policy to identify precisely what information about the platform can be disclosed to the specific authenticated Remote Challenger.
>
> When a privacy policy is implemented, the policy MUST be consulted prior to responding to any attestation requests from the Remote Challenger.
>
> After both parties have agreed to perform the attestation, the session SHOULD include an exchange to ensure the freshness of the attestation as an anti-replay and man-in-the-middle (MITM) resilience measure.
>
> Next, the Remote Challenger SHOULD obtain the AIK (or equivalent) credential associated with the signed vTPM quotes it expects to receive.

In this case, the AIK credential MUST assert that the remote system is virtualized and if applicable (no pass-thru support exists) how to contact the layer below (e.g. VMM) for a deeper attestation

After the session is fully established, the Remote Challenger will send requests for integrity evidence about the remote virtualized platform (Virtualized Platform #1).

The Attestation Agent MUST determine whether the requested information is authorized for disclosure to the Remote Challenger using its security and privacy policies.

It's envisioned that a multiple roundtrip exchange would ensue until the Remote Challenger has enough information to make a decision based upon its Integrity Policy as to the trustworthiness of the remote platform for the particular desired service. The early integrity evidence retrieved is expected to consist of a set of PCRs that establish an initial level of trust on the base functionality of the VM's trusted platform and ideally would lead (via the transitive trust chain) to trusting a higher-level measurement entity (e.g. Platform Trust Services or PTS [4][3]) that can provide additional evidence.

Once the Remote Challenger has decided that it trusts the contents of the VM to perform the desired service, the Remote Challenger needs to decide whether an attestation of the VMM (or N-1 layer) is necessary.

If a deeper attestation is necessary, the Remote Challenger MAY attempt to contact the VMM (or N-1 layer) via the Attestation Agent if the utilized attestation protocol and Attestation Agent support this capability.

Otherwise, the Remote Challenger would leverage the AIK (or equivalent) credential's N-1 contact assertion to directly attest the VMM.

4.4.2 VMM Attestation

Now that the Remote Challenger has decided that the N-1 layer below the VM needs to be included in the attestation of the Virtualized Platform #1 (see section 4.2.3 for why this might be important), the challenger needs to establish (or leverage an existing) secure session to the VMM. Two attestation models are supported by the architecture and offer different security properties.

The first model leverages the existing attestation session between the Remote Challenger and the Attestation Agent running in the VM. The Remote Challenger can attempt to request information about components executing in the N-1 layer (VMM in a 3 layer model) using the attestation protocol semantics. If the Attestation Agents doesn't support this operation an appropriate error is returned, so the Remote Challenger learns that the second model needs to be pursued.

If the Attestation Agent does support obtaining information about the N-1 layer, the Remote Challenger can request what is required to assess the trustworthiness of the N-1 layer. For example in a 3-layer model including a processor-constrained pTPM chip, the VMM might proactively create and maintain a pTPM quote blob indicating the state of the entire VMM. This quote blob could be available to the Attestation Agent either via a VMM interface or potentially via vNVRAM in the VM's vTPM. The Attestation Agent could return this cached quote blob when a request is received for TPM based attestation information about the VMM avoiding the per attestation use of the potentially expensive tpm_quote. On a platform with a more capable pTPM, this approach may not be required and the pTPM tpm_quote operation could occur at attest-time allowing for just the requested PCRs to be included. Note that by pre-creating an ephemeral AIK public key (unique per VM boot) for the VM as its booted and used to sign the quote, this ephemeral AIK could be extended into a resettable PCR and included in the pre-computed tpm_quote operation to effectively bind the AIK public key to the state of the local VMM (see the layer binding problem).

If the first model is unsuccessful or the Remote Challenger prefers a second isolated connection to a Deep Attestation service indicated by the VM's AIK credential, the challenger can use the second attestation model. The second model has the Remote Challenger establishing another attestation connection to the VMM Attestation component (possibly via a surrogate) to start the attestation of the N-1 layer. For the example shown in diagram 4, a surrogate "Deep Attestation Service" is used to handle the requests, so the challenger connects to the listening service's port and starts the attestation.

The attestation proceeds in much the same manner as the attestation of the VM including ensuring the security requirements are met for the session.

> Similar to the VM attestation, the VMM (or its surrogate) SHOULD use privacy and security policies to control the release of information to the Remote Challenger.
>
> For example, if the vTPMs are implemented in software within the VMM layer, exposing the private state of the vTPM (e.g. TPM-resident private keys) MUST NOT be allowed to be measured as part of the attestation; however, measurements of the vTPM logic MAY be part of the attestation since it's important to trusting the operation of the associated VM.

The primary difference between the VM and VMM attestations are the particulars of what types of components are requested and in this case how the Deep Attestation Service communicates with the VMM to obtain the necessary information. The base attestation protocol remains the same whether the attestation is of the VM or the VMM layer. Note that the VMM Service VM is considered part of the VMM for the purposes of this attestation, so its integrity is also reported.

The VMM interface between the Deep Attestation Service and the VMM is not specified in this document, so is considered to be virtualization platform implementation specific. This is deemed acceptable because the VMM Service VM isn't migratable, so is tightly bound to the specific VMM implementation being used. Part of the interface between the VMM and the privileged VM supporting the Deep Attestation Service includes an indication of the identity of the particular VM that is involved in the attestation in case the Remote Challengers requests information about the vPlatform associated with a particular VM. The attestation protocol enables the Remote Challenger to specify the VM's identity (included in its AIK credential), so the correct measurements are taken.

4.4.2.1 Attesting Four Layered Platform

After the internals of the VMM are deemed to be trustworthy, the Remote Challenger needs to decide whether this is sufficient to trust the Virtualized Platform #1 system's VM and VMM.

> In order to support more layers in the virtualized platform (see section 4.1.4) such as when the layer N-1 is another VM and N-2 is the VMM, the Remote Challenger SHOULD check the N-1 layer's AIK (or equivalent) certificate to determine if it is virtualized (meaning not running directly on the hardware) and has contact information to the N-2 layer's attestation service or attempt to attest N-2 via the currently open attestation connection.

If there is a virtualized N-2 layer, the process repeats as described in this section until a non-virtualized layer is considered trustworthy or the Remote Challenger decides to consider the system trustworthy. Note that implementations may choose to have another Deep Attestation Service (possibly in a separate VM) that is privileged and capable of providing integrity evidence for the N-2 layer instead of having such functionality exist within the N-1 or N-2 layer. Such an approach has the benefit of simplifying the N-2 layer and potentially improving its security by avoiding the threats associated with offering a direct network communication channel. Similarly, the VM's Attestation Agent or the N-1 layer's Deep Attestation Service might offer a way to proxy attestation requests for lower layers avoiding the need for the Remote Challenger to establish additional attestation connections.

4.4.3 Credentials Extension for Attestation

This section discusses some of the proposed certificate extensions that are needed in the TCG Credentials [2] specification to support virtualization. Because the TCG Credentials specification is a work product of another working group, this section summarizes the requirements only.

Typically an integrity challenger is only able to obtain a copy of the AIK (or equivalent) certificate because the Endorsement and Platform certificates can be considered privacy sensitive. Therefore, in order for the Remote Challenger to properly support detection of a virtualized platform and how to attest the next lower virtualization layer in the platform, the following extensions to the TCG Credentials 1.1 specification are required:

- Platform binding type of "virtual" in platform certificate (or equivalent) so can be copied into the AIK certificate
- New optional extension including the URI or an IP address to the layer below the certificate holder's Deep Attestation Service. For example, in a three layer architecture the VM's certificate layer below extension would point to the contact location of the VMM's attestation service.

4.4.4 Deep Attestation Layer Bindings

One of the largest challenges with performing a deep attestation of a virtualized system is how to know that the integrity evidence being presented is actually coming from the same physical system. There are several possible attestation approaches that a challenger may employ.

The following represent several possible approaches that were considered for performing layer binding:

1. Include a reference to the attestation service of the layer that runs on physical trusted roots and do the attestation from the bottom up. This ensures that no redirection code exists in the lower layers. Doing a bottom up mean each layer would need an "up" attestation service URI (instead of down as discussed above).
2. Include a reference to the attestation service of the layer that run on the physical trusted roots. Do the normal top down attestation. Then contact the physical trusted root attestation service to verify the evidence matches the top down assessment. This approach allows just the "down" URI to be in the certificates except the VM layer which would include the "bottom" URI.
3. Leave open the attestation session with 2 adjoining layers (say N and N-1) and ask both to perform an operation (e.g. signing) using a private key held only in the VM's vTPM (which both can access since its normally implemented in the N-1 layer). If both can access the private key its some assurance they coexist. For example, the attestation unique identifier could be extended into a PCR in a vTPM and both the VM and VMM attestation could include the PCR value and integrity log to prove the VM and VMM know the value.
4. Leverage the existing connection to the Attestation Agent in the VM to redirect attestation requests for lower layer integrity evidence including some unique property of the VM (e.g. request a quote of the VMM layer's state including the VM's AIK public key).

Using the described attestation architecture, the use of approaches 3 and 4 above offers implementations the most flexibility and ease of implementation, so are recommended to be supported to ensure the two layers co-exists on the same platform.

> Approach #4 SHOULD be supported if a VM's Attestation Agent supports proxying of attestation requests for components present in layers below it; otherwise approach #3 SHOULD be supported for implementations not including proxying of attestation requests.

4.5 Migration

This section discusses the components responsible for controlling the migration of a VM's vPlatform from one physical platform to another. Note that migration of a VM between systems includes the suspending, relocation and resuming of a large amount of VM state. One portion of the VM state could involves the vPlatform state that is the focus of this section. Other aspects of migration (e.g. migration control for VM application state) are outside the scope of this specification.

Diagram 5 below shows the components that could make up the attestation and migration capabilities on the virtualized platform. The shown components could be integrated into other aspects of a vendor's implementation and are called out just to define their role in a migration. The attestation components are also identified to augment the attestation discussion in section 4.4 allowing the Remote Challenger to detect migrations and potentially re-assess the VM's new platform. This is important because a Remote Challenger wishing to attest a VM would need to understand what migrations might happen to the VM during the lifetime of its transaction. Similarly, the challenger may wish to register with a notification service so as to be notified if a migration has occurred allowing it can decide whether it still trusts the VM based on its new platform.

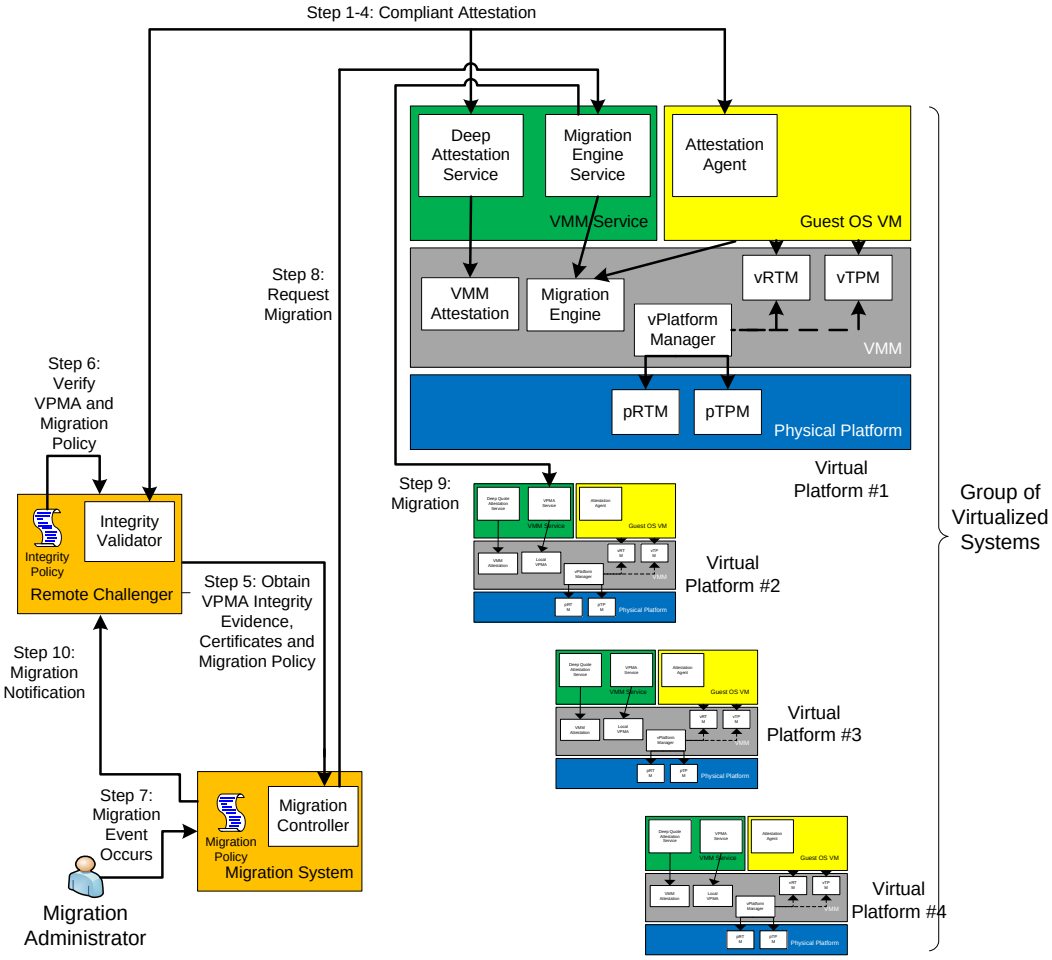


Diagram 6: Component Interactions between an Attestation and Migration of a Virtual Machine

Diagram 6 builds upon the attestation architecture diagram shown above by including the components responsible for VM migration and control. Diagram 6 also includes multiple virtualized platforms (shown as smaller instances of the main diagram) that collectively form a grouping of systems that the migration controller software can use to reallocate VM workloads. While the

diagrams of these systems appear the same, this section will discuss some of the issues that arise if they are different.

4.5.1 Migration Components

First, let's examine the new components introduced by diagram 6. On the lower left, the Migration Controller exists. Most virtualization solutions that support migrations include some privileged software that is capable of keeping track of each VM across the set of controlled systems and triggering a VM migration when necessary. Therefore, it's not envisioned that the Migration Controller concept is new but rather we are augmenting it to also support migration of the vPlatform.

This specification will not focus on the triggers that might cause a VM to be moved between platforms but a few possibilities include: administrator request (as shown), reliability concerns due to faults detected on a particular platform, power loss, or detection of degraded performance. When the Migration Controller decides that a VM needs to be moved, it communicates with the Migration Engine that resides on the virtual platform which performs the local operations necessary to perform the migration. These operations include suspending the VM's vPlatform (if it's running), packaging up the vPlatform's state, establishing a migration session with the selected target system and transmitting the state over the network. At the completion of the transmission of the state, the Migration Engine needs to receive a confirmation that the vPlatform state was successfully received so it can delete the vPlatform (including all the vTPM state). For this example of the virtualization architecture, the Migration Engine exists within the VMM and leverages a Migration Engine Service for its network communications so the VMM doesn't speak directly on the network. Therefore, the Migration Controller would work with the Migration Engine (possibly via a Migration Engine Service) on the source and destination platforms to trigger and verify completion of the migration.

The 1.0 version of this specification recognizes that in order to have an interoperable migration of a VM's vPlatform between different implementations of the Migration Engine components, a standardized representation of the vPlatform's state and a standard protocol for sending the state over the network is required. However, the definition of the vTPM state format and the migration protocol are left for topics in a future revision of this specification. However, version 1.0 of this specification will identify key requirements for the vPlatform state format and migration protocol to be addressed later. Similarly, the system management protocol used between the Migration Controller and the Migration Engine Service (or Migration Engine) is not specified in this document so could be a future topic in this or another document.

4.5.2 Migration Controller

The Migration Controller is a logical component within a deployment of a group of virtualized platforms that is integrated with the VM migration portion of the system management backplane. It's not unusual for the system management communications to be physically or logically isolated on a separate network from the general data network used by the Remote Challenger. Note that because this architecture is logical, the Migration Controller could exist on a centralized management-oriented control station (as shown) or one (or more) of the group of virtualized platforms. The main factor is that the Migration Controller is to be identified in the AIK credential and should be highly available, so that migrations can occur as needed.

The Migration Controller becomes active when an event triggers the need for a virtual trusted platform to be move along with its associated VM. The Migration Controller also needs to have access to information about the trustworthiness of each system in the grouping so it can optionally enforce restrictions on where the migration is allowed to occur. These restrictions are expected to be identified in a vPlatform Migration Policy.

The vPlatform Migration Policy defines what types of platforms the Migration Controller is allowed to migrate a VM to relative to its current platform. The vPlatform Migration Policy could include policies indicating whether a VM is allowed to move to another platform that has different:

- Physical trusted platform components (e.g. different TPM manufacturer)

- Firmware version of the physical trusted platform components
- VMM vendor, version or assurance level
- vTPM vendor, version or assurance level

For example, a grouping of systems could exists where each system has the same physical trust roots (e.g. identical TPMs) but half of the system are running a different version of the VMM software. In this case, the vPlatform Migration Policy could state that it will allow migration to different systems that have identical physical vPlatforms but different version of the same VMM manufacturers' software. Some Remote Challengers might not choose to trust such migrations (to different VMM versions) so would want to be able to discover the policy at the start of the transaction. However, over time newly purchased systems might come with newer (or different) physical trust roots, so the deployer would need to decide whether they would like to create a separate grouping or allow VMs to be migrated across all the types of systems while recognizing this might limit what challengers will just the grouping.

Another possible vPlatform Migration Policy could indicate that a VM would never be migrated to a platform offering a lower overall security assurance. This might allow confidence by a Remote Challenger that while a migration might happen after an attestation but before the Remote Challenger has completed its interactions with the VM that the VM would never move to a less trustworthy environment that could introduce vulnerabilities.

4.5.3 Migration Engine

The Migration Engine is privileged software on the virtualized platform that is capable of controlling the operation of each VM's virtual platform on the system. In this example, the Migration Engine is housed in the VMM and is only reachable via the Migration Engine Service running in a privileged VM. The Migration Engine needs to be integrated with the vPlatform Manager on the platform responsible for suspending, and snapshoting the state of the vPlatform.

> Specifically, the Migration Engine MUST notified when a migration is requested so it has an opportunity to securely package up the vPlatform state for transmission over the network and to receive an indication when the migration has completed.
>
> After the completion of the migration, the Migration Engine MUST remove and sanitize memory used for any local instance of the migrated vPlatform, so a duplicate can't exist.

The Migration Engine component might get contacted to migrate a VM that is installed on its system but isn't currently running. In this case, the Migration Engine will need to contact the vPlatform Manager in order to determine where the VM state is stored on persistent storage. The vPlatform Manager should be able to help decrypt any protected portions of the VM allowing the Migration Engine to focus on creating the migration package for the vPlatform. This packaging effort parallels the other migration agent functionality that exists on the system to package the remainder of the VM's state.

Because the vPlatform state be strongly protected (e.g. to avoid disclosure of the vTPM's private endorsement key), it is possible that these protections exceed any requirements for protecting the other VM state so is discussed in this specification in isolation. Clearly, if a migration system supports strong protection of all of the VM's state, then the vPlatform's state could just be included in the larger VM state package.

4.5.3.1 vTPM State

One of the key portions of the vPlatform's state package includes the state of the vTPM. The following list a few of the key state items that need to be included:

- Endorsement private key (EK)
- Storage root private key (SRK)
- Current tick counter

- Currently loaded keys in use on TPM
- Platform Configuration Registers (PCRs)
- Non-volatile RAM

4.5.3.2 vPlatform Package Security

It is critical that the many secrets housed within a vTPM be well protected during a migration. These secrets must only be visible to authorized entities involved in creating the migration package or instantiating the vTPM on the destination system.

Therefore, the vPlatform package security encoding MUST meet the following requirements:

- Package encodings MUST be unambiguous and canonical to avoid misinterpretation
- Package MUST be encrypted using a strong cipher (e.g. AES 512) capable of keeping the information safe for at least 5 years
- Package decryption key MUST be strongly protected such that only the Migration Engine on the destination system may decrypt the package (e.g. symmetric key encrypted with destination's asymmetric public key associated with a TPM resident private key)
- Package MUST be cryptographically signed so destination system can verify its produced by an authorized Migration Engine
- Package MUST be integrity protected to ensure any changes to the package are detected prior to vTPM instantiation (or removal on the source system)

4.5.4 Duplication of a vPlatform

Under very controlled situations, it might be permissible to allow two (or more) VMs running on different platforms to be running using the same vPlatform state. This situation is allowable in order to allow for hot standby usage or even when a collection of VMs need to appear as the same platform to remote parties over the network.

For example, a service provider might choose to run a hot standby VM instance on another machine and have a rapid switchover process to the standby should the primary instance become non-operational. This minimizes downtime for customers of the service and ideally the switchover would be invisible so as to not cause errors. Similarly, a service provide might choose to run the standby as a parallel service with transactions being split across both VM instances using some type of load balancer. Again, the remote service customers would not be able to distinguish which of the VMs is providing the services so as to be unaffected when the load balancer changes the VM in use.

Both of these situations raise some potential security concerns since they expect the same vTPM state (including private state) to exist on both platforms. In a physical trusted platform world, this situation doesn't arise since the pTPM state isn't available to be cloned on to both systems. However in a virtual environment, the Migration Controller and Migration Engine could be authorized to establish these duplicate vPlatforms. Normally duplication is not allowed because it could potentially open the door to an attestation of one "clean" instance to be used by the other compromised instance. There are numerous potential countermeasures to such attacks but these are the topic of attestation protocols.

> Virtualized trusted platform implementations supporting vPlatform duplication MUST include a trustworthy mechanism for ensuring that both instances of the vPlatform have identical state that accurately reflects the state of both instances.

However, in order to allow for duplicate vPlatforms to exist the products need to offer a synchronization protocol (similar to a clustering protocol) to ensure the vPlatform state is identical and both reflect the current state of both VMs. This synchronization protocol would ensure that an attestation of either VM would effectively be identical and reflect the true state of both platforms. Note that a deep attestation of the layers below the VM on each system would likely be different,

since each physical platform (or other N-2 layer) would likely use different AIKs and have varying state. The details of the synchronization protocol are outside the scope of this version of the specification.

4.5.5 Attestation Including the Migration System

This section discusses the role of the migration components discussed above in an in-depth, deep attestation. As mentioned, the Remote Challenger may have an Integrity Policy that requires a deep attestation (similar to the example is Diagram 4) and also wishes to evaluate the potential for the VM to migrate during the transaction and possible new host platforms. In this example, the Remote Challenger wants to know any possible change in the VMs location and security posture throughout its potentially long period of time where it's using the VM's services. Therefore the Remote Challenger needs to not only attest the VM and VMM but also its migration components and the central Migration Controller.

Rather than replicate the text in section 4.4.1, steps 1 through 4 remain the same. During the attestation of the VM, the VM's AIK certificate indicates that the VM is under the control of the Migration Controller and includes the URI with how to interact with the Migration Controller. In some deployments, the Migration Controller might not be reachable from the regular data network (e.g. if it only exists on an isolated management network) so the Remote Challenger will need to factor in the lack of information about the Migration Controller.

In this case, the Migration Controller is able to respond to the Remote Challenger's network connection so can participate in the attestation. The Remote Challenger establishes a network connection to the Migration Controller and starts an attestation session similar to steps 1-4. After deciding that the Migration Controller is running a trustworthy configuration, the Remote Challenger fetches the migration policies involving the VM. At that time, the Remote Challenger needs to decide whether it is willing to trust the deployment enough to perform the entire transaction or whether more information is required. A more concerned Remote Challenger might request to be notified by the Migration Controller when a migration event occurs, so it can decide whether to re-attest the VM post-migration. If a new attestation is required, the sequence cycles back to Step 1 in the diagram but this time the new hosting platform's VMM is evaluated.

4.5.6 Migration Policy

The Migration Controller has a migration policy file that dictates how to select a new target system to host the VM. The migration policy is something that is commonly present in virtualization implementations, so this specification merely seeks to extend the policy to cover the topics discussed in section 4.5.2. The details of how the migration policy is expressed within a product is out of scope for this specification; however, the details of how the policy appears when shared with a Remote Challenger such as in flow 5 above does need to be standardized at some point to achieve interoperability between challengers and virtualization systems. Version 1.0 of this specification does not define how this policy will be expressed but this might be a future topic for this document or in the PTS Protocols specification. Note that the expression of the policy that is needed by the Remote Challenger for attestation purposes is likely to be a small subset of the full migration policy, so the representation of the policy is likely to appear very different from the implementation's internal representation.

The migration policy needs to express at least whether Migration Controller is willing to migrate VM to:

- System with a different physical trusted platform (e.g. TPM)
    - Which can be different: TPM vendor, TPM model, TPM version, CRTM type
- System with a different virtualization platform
    - Which can be different: VMM vendor, VMM version, VMM architecture
- System with different architecture

    - Which can be different: number of layers, different attestation model
- System with lower (or unknown) assurance
- System that was not attested by Migration Controller prior to migration
- System owned and/or operated by the same entity
- System is in a different location with different physical security properties

4.5.7 Migration Events

Flow 10 in the diagram 6 above shows the Migration Controller sending a notification event to the registered Remote Challenger that a migration event has occurred involving a VM that it is using. In order for this event to be triggered, the challenger registers with the Migration Controller for migration events specific to that VM. This requires the Remote Challenger to use a registration protocol with the Migration Controller that allows for subscriptions to events and notifications. As of 1.0 of this specification, this protocol is not yet defined. One interesting protocol from TCG that might suit the needs of this protocol is IF-MAP from the Trusted Network Connect (TNC) working group. However, the use of IF-MAP requires an IF-MAP server to be present, so additional infrastructure is needed. The IF-MAP will be evaluated post 1.0 of this specification for suitability and necessity of schema extensions.

The migration protocol requirements include:

- Ability for Remote Challenger to register for events involving a single VM (likely by IP address for many use cases)
- Capable of Migration Controller (or infrastructure) to asynchronously send events to one or more Remote Challengers when migration event occurs

4.5.8 Migration Protocol

Flow 9 in the diagram 6 depicts the migration of the vPlatform state package (and other VM state) between the source and destination system. This information is carried over the migration protocol which could be used to transport the entire VM's state or be a specific session for moving the vPlatform state. Version 1.0 of this specification does not define a specific protocol, but future versions of this or other VPWG specification may define an interoperable protocol.

Because the migration protocol is carrying information essential to properly instantiating the correct VM and vPlatform on the target machine, it is essential that the secrets held within the vPlatform (vTPM) be adequately protected. Section 4.5.3.2 contains a list of security requirements that the vPlatform state package needs to meet for adequate protection.

4.5.9 Credentials Extension for Migration

This section discusses some of the proposed certificate extensions that are needed in the TCG Credentials [2] specification to support migration. Because the TCG Credentials specification is a work product of another working group, this section summarizes the requirements only.

As discussed earlier, an authorized Remote Challenger is only able to obtain a copy of the AIK (or equivalent) certificate because the Endorsement and Platform certificates can be considered privacy sensitive. Therefore, in order for the Remote Challenger to properly support attestation of VMs that could migrate, the following extensions to the TCG Credentials 1.1 specification are required:

- Optional extension including the URI to the Migration Controller's migration event registration service
- Optional extension including the URI to the Migration Controller's Attestation Agent so Migration Controller can be attested by Remote Challenger

4.6 vTPM Backup and Restore

The virtualized platform architecture allows for a vTPM's internal state to be backed up for later restore on the same vPlatform or potentially to another vPlatform should the original fail. The backup of vTPM software state isn't unique to the virtualized trusted platform architecture except that private TPM state is not normally available for inclusion in a backup, so offering a backup service comes with additional requirements to protect the sensitive private data.

> The vTPM state included in a backup MUST meet the same requirements as described in the vPlatform Package Security (see section 4.5.3.2), since the vTPM state could become exposed while on backup media.

When a vTPM state needs to be restored, the VMM component of the backup/restore software is responsible for ensuring that restore policy is strongly enforced. The restore policy needs to define the situations when the vTPM state is allowed to be restored and whether the state may be installed on other platforms. The trusted restore software is expected to prevent vTPM duplicates from being created resulting in two different vTPM existing on differing vPlatforms (thus potentially introducing a security threat of spoofing attestations). Similarly, the VMM component of the restore software needs to address whether the restore is resulting in a rollback of the vTPM state (e.g. monotonic counters would now be lower than the previous running vTPM instance).

> Trusted vTPM restore software MUST maintain a protected, persistent counter indicating the number of times a particular vPlatform backup has been restored on to a system and potentially caused a rollback from the state previous executing. This counter will be known as the "vTPM Generation".
>
> The vTPM Generation counter MUST be set to zero for all new vTPMs.
>
> The vTPM Generation counter MUST be incremented by one each time the vTPM is restored potentially resulting in a rollback.
>
> After the re-instantiation of the restored vTPM, trusted software in the VMM MUST extend the current vTPM Generation counter into vPCR 7 (reserved for the OEM) of the vTPM.

4.6.1 Credentials Extension for Backup

This section discusses some of the proposed certificate extensions that are needed in the TCG Credentials specification to support backup/restore. Because the TCG Credentials specification is a work product of another working group, this section summarizes the requirements only.

As discussed above, an authorized Remote Challenger may be concerned that a vPlatform may be a duplicate (created via backup and restore to another system), so would like to determine the restore policy enforced by the backup authority. Therefore, in order to provide a Remote Challenger with the ability to evaluate the restore policy, the following extension to the TCG Credentials 1.1 specification is desired:

- Optional extension including the URI to the Backup Authority, so the Backup Authority can be attested by the Remote Challenger
- Indication in EK and AIK credential that the vTPM can be rolled back via a restore to an older version

5 Security Considerations

5.1 Trust Model

This section discusses the trust relationships that exist between the different layers and in some cases components of the architecture described in this specification (see section 4.1.2). Products conforming to VPWG specifications may assume a more conservative (less trusting) set of trust relationships in order to expand their potential field of use. However, this section describes a good maximum set of assumed trust relationships between components, so it's recommended that products not assume additional trust is present.

Generally speaking, each layer in the virtualized trusted platform architecture needs to trust the proper operation of the layers below it, thus creating a transitive trust chain down to the physical platform's trust roots through each layer. A malicious lower layer can cause damage to higher layers through a variety of attacks (e.g. direct memory modification) that a higher layer would not be able to defend against. This is analogous to how malware in an operating system kernel can compromise application layer processes. However, the attestation system discussed in this specification enables a Remote Challenger to detect a malicious lower layer if the challenger performs a deep attestation as discussed above. The worst case scenario is that the malware in the lower layers will block or not provide the verifiable attestation, so the Remote Challenger should consider this lack of a trustworthy attestation.

5.1.1 VM Layer

Starting at the top layer of the architecture, the VM layer is comprised of a number of components, many of which are unaware of the (virtual) trusted platform that exists below it. The internal trust relationships (not involving trusted computing) within the VM's operating system and applications are outside the scope of this document. Some portion of the VM layer may leverage a trusted platform, provided by the VMM, in order to provide security services such as: measuring platform boot and running software, performing attestation reporting and (un)sealing data to TPM keys to VM layer software. The VMM does not trust the operation of VM software for its own security, while the VM must trust the proper operation of the VMM, since it cannot detect or defend against all forms of attack the VMM could mount.

The VM might contain trusted platform maintenance or other vTPM-aware software that might possess authorization data or have privileged sessions open to the vTPM, so needs protection from VM malware. This protection could prove difficult in the face of certain types of compromises. Similarly, the Attestation Agent in the VM plus its dependencies (e.g. PTS and TNC software) needs to perform their duties even when faced with malware. The measurement agents inside the operating system (both during boot and operation) need to properly measure and extend into the PCR all software that could affect the trustworthiness of the VM. However, the threats listed in this paragraph are also possible for non-virtualized systems.

5.1.2 VMM Layer

The VMM Layer consists of components that are responsible for providing the virtualized environment to each of the VMs. The VM needs to trust the underlying VMM as it provides the VM with its resources and therefore could attack the VM in ways the VM couldn't defend against. However, the VMM does not trust operations performed by the VM, so requires proper authorizations to be presented in order to access privileged TPM ordinals or to access protected data. With that said, the TPM does offer use of a number of features to the VM that the VMM doesn't attempt to safeguard from improper usage once access is granted. For instance, the content of the PCRs is largely controlled by measurement agents running in the VM. If a malicious VM decides to corrupt the PCRs by extending irrelevant data, the VM is merely hurting itself by eliminating the use of the TPM to prove that the VM is running in a trusted manner (since the resulting PCR values won't match expected verifier policy). The use of these ordinals will not result in the compromise of the VMM or its instantiated vTPM.

The VMM supports the isolated operation of potentially multiple VMs and therefore is responsible for offering fault and security compromise isolation from attacks mounted against the vPlatform. Each VM trusts the VMM to prevent the spread of malware between VMs through VMM provided services (e.g. vTPM operations). Similarly, the VM trusts the VMM to provide a unique virtual trusted platform to each VM in such a way that vPlatform operations perform by one VM can not affect the state of other VM's trusted platforms.

Depending on the implementation architecture of the VMM, a compromise of a single component could lead to a broader breakdown in the security of the VMM. For example, if the VMM is constructed as a monolithic layer all within the same memory protection domain, malware that is present in the VMM could eavesdrop or manipulate the vTPM instances or vPlatform Manager to cause damage to the VMs. The use of privileged VMs (as shown in section 4.1.2) to support potentially more risky services (e.g. involving network communication) could constrain damage from such attacks should they only be able to compromise the privileged VM. Since the VMM is protected from attack by VMs, this should allow some trustworthy services to continue to operate.

5.1.3 Physical Trusted Platform

At the bottom layer of the virtualized trusted platform architecture is the physical trusted platform. Precisely like when the physical platform is running on a non-virtualized platform, the physical platform is physically isolated from the software running above it, allowing the physical platform's trust roots to remain protected even when malware is present in the software. Similar to the VMM's trust model, the physical trusted platform does not rely on the VMM for its own isolation and protection. However, the VMM layer might be written to use the physical trusted platform to safeguard its own secrets (e.g. using the seal capabilities). The physical platform must be trusted by the VMM since it is providing the resources used by the VMM (e.g. CPU and memory).

5.1.4 Remote Challenger

Section 4.4 discusses the process of attestation where a Remote Challenger is able to establish the level of trustworthiness of a VM and its underlying layers. The Remote Challenger and the target platform do not inherently trust each other until they have a reason to do so. The Remote Challenger typically wishes to make use of the services or resources of a remote virtualized platform, so chooses not to blindly trust the platform. Therefore, the Remote Challenger uses an attestation protocol to obtain specific attestation evidence that the system is trustworthy. During the attestation, the target platform authenticates the Remote Challenger and must make authorization and privacy oriented decisions about what information it is willing to share with the Remote Challenger, since it too does not trust the Remote Challenger.

After the Remote Challenger has attested the VM, it's likely to perform a deep attestation to determine whether the virtualization layers below the VM have been compromised. At the conclusion of the attestation, the Remote Challenger decides whether the platform is trustworthy to perform the desired operation.

5.1.5 Migration System

Section 4.5.2 covers the components that enable migration of vPlatforms between physical systems. The migration system consists of a virtualized trusted platform component known as the (vPlatform) Migration Engine while typically a more centralized component known as the Migration Controller exists in the infrastructure to authorize and oversee the migration.

The Migration Engine (and optionally the Migration Engine Service in a privileged VM) on the source and destination machine are involved with the packaging and transmission of the vPlatform securely over the network. The Migration Engine on source and destination systems does not inherently trust each other until they've strongly authenticated and received a migration authorization from the Migration Controller. During the vPlatform transmission, the vPlatform package should be protected such that only the destination's Migration Engine is able to re-instantiate the vPlatform. However, the source Migration Engine does trust the authenticated and authorized migration destination Migration Engine and/or Migration Controller to notify it once the vPlatform has been received, so that the source can delete and sanitize its copy.

The Migration Controller doesn't inherently trust the virtualized platforms that it controls. The controller authenticates each platform prior to authorizing a migration event and may attest the target systems VMM and physical platform to ensure that it meets its migration policies limiting the movement of VMs between different platforms.

5.2 Threats and Countermeasures

This section discusses some of the primary threats against the layers and individual components of the virtualized trusted platform architecture and highlights some recommended countermeasures. Note that this section focuses on new threats introduced by the virtualized trusted platform architecture so therefore doesn't repeat existing threats to the non-virtualized trusted platform.

5.2.1 VM Layer

The VM layer execution on a VMM is very similar to when an operating system runs directly on a physical platform, so the threats to the VM are generally not unique to virtualization. As discussed above, the VM layer trusts the VMM to operate appropriately and to not act maliciously. If the VMM is compromised, the VM has very little it can effectively do to defend itself recognizing that the VMM can directly read/write the VM's address space and can alter the VM's vTPM state without detection. One potential new class of attack involves the Attestation Agent running in the VM, but this issue is discussed below in section 5.2.4.

5.2.2 VMM Layer

The VMM layer has control over the VMs operating above it, so therefore needs to be strongly protected from potential compromise. Depending on the implementation architecture employed, the VMM layer could face a variety of threats, so it's recommended that the VMM layer be kept as simple as possible and to apply the principal of least privilege when possible. For example, segregating less privileged code into restricted helper VMs that use a private interface to interact with the core VMM layer might contain a breach in the privileged VM.

5.2.2.1 VMM Integrity

It's critical that the VMM's integrity be maintained during its operation even when the platform is connected to potentially hostile networks. The damage that could be caused by malware being able to execute within the VMM is huge including the ability to compromise the higher layered VMs and steal their secrets.

> The VMM SHOULD consist of the minimum functionality necessary to manage and run the VMs and associated vPlatform services to limit the attack surface.
>
> The VMM MUST be architecturally isolated/protected from any access by the VMs except via a well defined set of VMM hypercalls.
>
> The VMM hypercalls MUST be protected from buffer overflow and other conventional API calling attacks.

The VMM instantiates and offers a dedicated vPlatform to each VM running on top of it. It's critical that there be no way for one unprivileged VM to directly or indirectly access the vPlatform associated with another VM via the VMM as this could introduce threats against the target VM's integrity.

> The VMM MUST isolate the operations performed on one VM's vPlatform from affecting the state of any other VMs' vPlatforms.

The VMM is responsible for facilitating the creation of a newly instantiated VM's vPlatform credentials (EK, Platform and AIK). In cases where the VMM directly (or indirectly via a privileged VM) mints the certificates for a VM, it's important that the information be accurate and properly reflect the security properties of the VM's vPlatform. In some cases, the credentials contain asserted information that Remote Challengers are unable to verify with measurements, so the VMM claims on these properties must be correct.

In order to minimize the complexity and attack surface presented by the VMM, many of the more complex features of the VMM should be partitioned into privileged, but restricted VMs. This is particularly important for network-facing features that could potentially be remotely exploited. For example, the attestation service provided by a VMM could be complex, thus containing more vulnerabilities, so it should be run in an isolated and less privileged environment then the VMM layer. However, this VMM attestation service requires special access to the VMM in order to perform the VMM attestation, so the VMM would need to include a privilege model and ideally a private hypercall API isolation feature, so unprivileged VMs can't attempt to use the private interfaces.

5.2.2.2 VMM Internal State Confidentiality

The internal state of the VMM could contain private secrets associated with each of the VMs, so theft of these secrets could compromise the trustworthiness of the VMs. As discussed above, unprivileged VMs (not VMM "helper" VMs) are never allowed to observe the internal state of the VMM. It's possible that some privileged VMs working on behalf of the VMM (not the user) might have access to the VMM internal state.

> The VMM SHOULD NOT provide access (even read) to its address space to any VM including privileged VMs to avoid a VM compromise leading to compromises of the other VMs operating on the platform.

Instead privileged VMs should request services from the VMM when access to its address space is required (e.g. taking measurements of VMM features instead of giving access to memory locations).

Another path for accessing state within the VMM is via attestation. This topic will be covered in section 5.2.4 below. Finally, there are a few situations where VMM internal state needs to be stored in persistent storage. The following sub-sections will discuss attacks on the vPlatform state in different circumstances.

5.2.2.3 vTPM State Protection

As discussed in section 4.5.3.2, the vTPM state may need to be written to persistent storage, therefore requires strong protections to address off-line attacks. During the operation of VMM, it will instantiate and potentially execute many vPlatform instances. Each vPlatform instance contains a vTPM whose state is likely to reside in the VMM address space at some point in order to perform vTPM operations for the VM. When the vTPM state is resident in VMM memory, it is afforded the same isolation protection as the remainder of the VMM (see section 5.2.2.2) so this subsection will not repeat the threats against the private information when stored on persistent storage.

Occasionally, the vTPM state for each running VM needs to be written to persistent storage. These occasions include: backup, migration, suspend and for reliability in case of a crash. Regardless of the reason for the state is being written to storage, it's important that the vTPM state be protected from off-line theft, alteration, replacement or other attacks which could lead to the exposure or modification of the vTPM state. As discussed in section 5.2.2.2, the vTPM state needs to be encrypted appropriately when written to storage to avoid potential off-line attacks outside the purview of the VMM. In this case, the off-line attack could be the attacker stealing the drive and attaching to another system running a compromised OS that mounts and tries to access the vTPM state on the storage device. In this off-line attack scenario, the VMM isn't running on the system, so its protections aren't available and the only thing that prevents the attacker from stealing or modifying the secrets is the cryptographic protections.

Another form of attack against the vTPM image is rollback. Rollback occurs when an unauthorized party replaces a vTPM image with an older (potentially compromised) version in order to trick the vPlatform Manager into executing it instead of a newer instance. Rollback attacks are difficult to defend against, since an off-line attacker might be able to replace the encrypted image with an older one. The vPlatform Manager needs to have facilities to detect replaced vTPM images on persistent storage.

Another situation that resembles replacing the vTPM state is vPlatform restores. When the trusted restore software is run to restore a vTPM instance from a backup media, the result can be the same as the off-line replacement, since an older version of the vTPM instance now is available to the vPlatform Manager for execution. Restore policy needs to restrict the circumstances when a restore can occur and this policy should be reflected in the vTPM's EK and AIK credentials. Remote challengers also need to be able to detect when it is communicating with a restored vTPM instance (see section 4.6 for details) in case this impacts its trust decisions.

Finally, the VMM is responsible for the lifecycle of each vTPM operating on the platform. Since a virtualized platform may have multiple VMs running, this means multiple vTPM instances would also be operating. It's essential that the VMM ensure that each VM can make use of one and only one vTPM. Similarly, each vTPM is exclusively for the use by the one VM, so other VMs are not allowed to share the single vTPM instance. This results in a one-to-one mapping between vTPM and VMs.

5.2.3 Physical Platform

The physical platform layer at the bottom of the virtualized trusted platform architecture greatly resembles a standard trusted platform; therefore the vast majority of the threats are not unique to virtualization. The pCRTM and pTPM perform the same role as on a non-virtualized platform, so the threat model and countermeasures apply equally well to a virtualization layer running above it. The primary difference isn't in the mechanism, but rather what the role of the code being measured into the TPM is playing (operating system boot loaders and kernel vs. VMM boot loaders and VMM).

5.2.4 Attestation

While the attestation components allow Remote Challengers to evaluate the operational status of the VM's vPlatform and potentially its underlying layers, they also expose the platform to some additional new threats. These threats fall into several general categories: threats to the Attestation Agents, attacks on the attestation protocol and misuse of disclosed of platform information.

5.2.4.1 Attestation Agent

The attestation system of a trusted platform (virtualized or not) introduces the need for an Attestation Agent to be available for Remote Challengers to use to obtain evidence of the trustworthiness of the platform. While the Attestation Agent is not unique to virtualized trusted platforms, this specification potentially expands its functionality to allow the challenger to detect the platform is virtualized and offer information about how to attest the layer below. This sub-section will focus on those aspects that are unique to attestation of a virtualized platform.

The Attestation Agent is allowed to operate in two slightly different ways to support attestation of the layer below: redirection and pass-thru. The first approach (redirection) has the Attestation Agent returning credentials indicating that it's running on a virtualized platform and how to redirect the challenger in order to contact the next lower level's Attestation Agent for a deep quote. In this redirection model, the Attestation Agent running in the VM doesn't require any new functionality, since the VMM handles creating the credentials containing the virtualization oriented information. Therefore, for the redirection model there are no special new threats to the VM's Attestation Agent but the Deep Attestation Agent is now required so faces threats.

The Deep Attestation Service is the agent that operates within the VMM or a designated special purpose VM in order to allow for attestation of the VMM's trusted platform. As discussed in section 4.4.2, this service can be located within the VMM layer or a privileged VM in order to provide isolation of mechanism from the VMM itself. Due to the potential complexity and fact that the Deep Attestation Service will be communicating with an untrusted network party, like the Attestation Agent in the VM it's potentially subject to vulnerabilities. By isolating the service in its own restricted VM, a breach can be contained outside of the VMM's address space.

In the 2nd attestation model (pass-thru), the Attestation Agent is able to pass attestation requests from the Remote Challenger down to the VMM for processing and response. This model avoids the

challenger needing to establish additional attestation sessions to perform a deep quote, but does mean the Attestation Agent in the VM must be virtualization-aware in that it has virtualization unique features that could be subject to threats. In the pass-thru model, some of the attestation evidence requests may indicate components existing within the VMM, so request the VMM (or a surrogate help VM) to respond. This model is enabled by a special set of privileged attestation hypercalls being available to the VM's Attestation Agent in order to make these requests. The interface could be used by other software (including malware) in the VM, so VMM implementations need to be prepared for unauthorized and malicious requests (see section 5.2.2.1). Similarly, the Remote Challenger needs to be able to verify that the VM is going to pass through the requests to the VMM and not manufacture false responses. This can be achieved by an attestation of the VM first to ensure no malware is involved in the deep attestation process and by verifying the VMM oriented responses by leveraging the VMM's TPM quote operations.

5.2.4.2 Attestation Protocol

The attestation protocol is mentioned in this section for completeness, but the actual attestation protocol is the subject of other TCG specifications. Therefore the attestation protocol specification itself will focus on the threats to and countermeasures offered by the protocol stack. Clearly it's unacceptable for network based attackers to be able to observe, modify, delete or otherwise impact the attestation messages while traversing the network without detection by the recipient.

> The attestation protocol stack MUST be capable of strong authentication, integrity, replay and confidentiality protections for the attestation evidence exchanged during an assessment.

5.2.4.3 Attestation Information

Inside the attestation protocol is a set of attestation evidence attributes. These attributes describe the platform being attested so could be subject to falsification, replay or other content based attacks. At the start of the attestation protocol handshake, a strong authentication occurs to establish the identities of the parties and leads to the authorization decisions on what information can be requested. The Remote Challenger needs to evaluate the underlying trusted platform and Attestation Agent against trust policies prior to believing the evidence provided.

Another content-based threat is that a VM or VMM containing malware could be proxying the attestation evidence from another "clean" system. In this attack, the VM or VMM responding to an attestation request could forward the request to another clean system and then would respond with this information as it own. The attestation protocol needs to offer support for replay detection for example using a fresh secret negotiated by the Remote Challenger and attested platform where both parties contribute to the secret. This secret and the attested information could be mixed into a TPM quote operation to ensure the quote came from the correct system. This attestation secret approach when used with the countermeasures discussed in section 4.4.4 enables the Remote Challenger to detect replayed responses.

5.2.5 Migration

This subsection focuses on the VM migration components present in the virtualized trusted platform architecture. The two main components of the migration system include the Migration Engine which is present on the virtualized platform's VMM layer (or in a privileged helper VM) and the Migration Controller that could be located on a virtualized platform or as a centralized infrastructure component.

Both of the migration components offer network facing services, so they are potentially subject to attack from malicious parties on the network. Similar to the attestation system, the migration system might face attacks directly upon: the migration components, the migration protocol or the vPlatform state package being carried by the migration protocol.

5.2.5.1 Migration Components

The Migration Controller is the component that makes migration decisions and instructs the Migration Engine on what operations to perform. The Migration Controller needs to be able to

speak over the network at least to the Migration Engines for each virtualized platform in its group. The Migration Controller operates much like the other virtualization management software being used in the group and is likely to be integrated into the management software. The Migration Controller may also provide for migration event notifications to registered Remote Challengers, thus the controller needs to be accessible on the untrusted network.

The Migration Controller's network facing responsibilities place it at risk for directed attack over the network. It's essential that the Migration Controller be adequately protected from network visible vulnerabilities and be run in a contained environment that is remotely verifiable. This contained environment could be a VM on a virtualized platform (using a vPlatform) or just run on a separate management (non-User based) platform. The Migration Engine running on an end platform may wish to attest the Migration Controller when it receives a migration request in order to verify that the Migration Controller has not been compromised.

The Migration Engine needs to listen for network connections from the Migration Controller and other Migration Engines, but does not need to communicate over the network with non-management oriented components. Therefore, deployments offering a parallel isolated management network should move the Migration Engines networking functions off of the network interfaces used by users. The Migration Engine is part of the VMM, so needs to be protected from compromise just like the rest of the VMM (see section 5.2.2.1). The Migration Engine needs to communicate with other authorized Migration Engines when they are facilitating the movement of a vPlatform over the network. It's essential that Migration Engines strongly authenticate each other to ensure only authorized entities are able to initiate and receive vPlatform state. The vPlatform state contains many private secrets, so a migration should only occur between authorized Migration Engines (possibly leveraging VM migration protocols) as instructed by the Migration Controller.

5.2.5.2 Migration Protocols

The details of the migration protocols (Migration Controller to Migration Engine and Migration Engine to Migration Engine) are left to a future version of this or another specification. However, some products today do support migration of VMs between platforms and it's envisioned that in the near term that the existing VM migration protocols could be expanded to support migration of the vPlatform as well (instead of a separate inter-Migration Engine vPlatform migration protocol).

Similarly, VM management software exists that supports controlling the operation of a group of virtualized platforms, so this management software could be expanded to perform the Migration Controller role. Therefore, this section will highlight some expected security properties that the migration protocols would need to posses to allow for adequate protection for operation in hostile environments.

Both the migration control protocol used between the Migration Controller and Migration Engine to request the movement of a VM, and the vPlatform migration protocol used between the source and destination Migration Engine may be subject to attack by network-based adversaries. In some environments, both of these protocols might operate on a separate "management only" trusted network to help isolate the attacks. This is recommended particularly if the VMs are unable to directly communicate on this network (e.g. controlled by the VMM). In cases where a separate physical management network isn't pragmatic, a virtual (overlay) network could be present for the migration protocols leveraging VPN technology.

> In order for the Migration Controller and Migration Engine to safely communicate over an untrustworthy network, the migration protocol MUST be capable of offering strong authentication, integrity, replay and confidentiality protection for network dialogs between the Migration Controller and Migration Engines all over a reliable channel.

It's important that the Migration Controller and Migration Engines can strongly authenticate each other and apply policy to determine what types of operations (e.g. instructions from the Migration Controller as well as inter-Migration Engine operations) are authorized to be performed. In some cases, Migration Engines might not accept migrations from other Migration Engines as dictated by policy or restrict what operations they allow certain Migration Controllers to request.

5.2.5.3 Migration Information

Within the protected communication protocol channels discussed in the previous section, the Migration Controller sends migration instructions to the Migration Engine on the source and destination virtualized platforms. These instructions indicate what migration actions the pair of system should perform to move a VM's vPlatform between systems. This authorization should include specific details of the authenticated identities of both parties involved in the transfer. The Migration Engines should consult policy to verify that the Migration Controller is authorized to make the request and to verify local policy doesn't prevent migrations to/from the other Migration Engine. The Migration Engines may choose to attest the Migration Controller to ensure its not acting maliciously. Similarly, the Migration Controller may choose to attest the target platform to ensure that it offers at least as strong security protections as the source platform.

The details of the Migration Controller instructions to the Migration Engine and the policies used by each entity are the topic for a future specification (or extension of this specification). Similarly, the details of the protocol used between Migration Engines to transfer vPlatform state packages are also a future specification effort. However, it's important to note that any authorization decision made by the parties' factor in the strongly authenticated identity of the entity and considering local policy. Migration components should not merely make security decisions based upon the contents of some message received unless it's been strongly bound to a session with an authenticated party. By not doing this binding, malicious parties could replay prior messages to repeat operations that aren't currently required. For example, a malicious party could eavesdrop and replay a message sent by the Migration Controller to the Migration Engines to move a VM.

6 References

6.1 Normative References

[1] Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels”, Internet Engineering Task Force RFC 2119, March 1997.

[2] Trusted Computing Group, TCG Credentials Profiles Specification Version 1.1, http://www.trustedcomputinggroup.org/resources/infrastructure_work_group_tcg_credential_profiles_specification, May 2007.

6.2 Informative References

[3] Trusted Computing Group, TPM Main Specifications, http://www.trustedcomputinggroup.org/resources/tpm_main_specification, March 2006.

[4] Trusted Computing Group, Platform Trust Services Interface Specification (IF-PTS), http://www.trustedcomputinggroup.org/resources/infrastructure_work_group_platform_trust_services_interface_specification_ifpts_version_10, November 2006.

[5] Trusted Computing Group, TNC Architecture for Interoperability, http://www.trustedcomputinggroup.org/resources/tnc_architecture_for_interoperability_specification, May 2009.

[6] Trusted Computing Group, TCG Architecture Overview, http://www.trustedcomputinggroup.org/files/resource_files/AC652DE1-1D09-3519-ADA026A0C05CFAC2/TCG_1_4_Architecture_Overview.pdf, August 2007.

[7] Trusted Computing Group, PC Specific Implementation Specification, http://www.trustedcomputinggroup.org/resources/pc_client_work_group_pc_specific_implementation_specification_version_11, August 2003.

[8] Trusted Computing Group, TNC IF-MAP Binding for SOAP Specification, http://www.trustedcomputinggroup.org/resources/tnc_ifmap_binding_for_soap_specification, July 2010.