CHAPTER 8

Trusted Virtual Machines: Ensuring the Integrity of Virtual Machines in the Cloud

In Chapters 3 and 4, we described how a service provider can ensure that the infrastructure on which the workloads and applications are instantiated has boot integrity, and how these workloads can be placed in trusted pools with compute assets exhibiting demonstrated trust that is rooted in hardware. This model provides an excellent framework for a trusted compute infrastructure, but it's not sufficient for the cloud. Cloud data centers today almost invariably run virtualized. Stopping the chain of trust at the bare hypervisor is clearly insufficient; that is but the proverbial tip of the iceberg. Protection needs to be extended to support the multi-tenancy and virtualized networks of the cloud. Extending the chain of trust described to encompass these virtualized resources, embodied in the concept of *trusted virtual machines,* is what this chapter is about.

Critical concerns for cloud users are for protecting workloads and data *in* the cloud and *from* the cloud, and for ensuring trust and integrity for virtual machine images launched on a service provider's cloud. For virtual machine and workload data protection, cloud-user organizations need a method to securely place and use their workloads and data in the cloud.

Current provisioning and deployment models include either storing the virtual machine and application images and data in the clear—in other words, unencrypted—or having these images and data encrypted by the keys controlled by the service provider, which are likely applied uniformly to all the tenants. Increasingly, however, virtual machine images—effectively containers for operating system and application images, configuration files, data, and other entities—need confidentiality protection in a multi-tenant cloud environment. These images need to be encrypted by keys that the tenant controls, and that can be decrypted for provisioning by the keys also under tenant control, all done in a manner that's transparent to the cloud service provider.

Additionally, tenants would like the chain of trust and the boot integrity of these virtual machines assessed by the attestation authority in the infrastructure or the cloud before they are launched and begin participating on the network to satisfy service requests. With the tenant-controlled encryption and decryption, and integrity attestation of virtual machines, there can be clear security statements made about the confidentiality and protection of these application workloads in the cloud.

With interest in these topics driving cloud security considerations, Intel has reached out to the independent software vendor and cloud service provider community to develop a set of technology components and associated solutions architecture built on top of the trusted compute pools foundation to build proof points for these usage models. These are reference and prototype demonstrations that, it is hoped, will light a path to deployable solutions that address the concerns voiced above.

Requirements for Trusted Virtual Machines

Organizations are clearly leveraging the cost-saving (from capX reduction) and flexibility benefits of virtualizing their data centers, as opposed to the traditional deployment model in dedicated, corporate-owned infrastructure resources. The ease with which virtual machines can be created, deployed, and moved brings a multitude of security issues not present in those traditional, physical data centers, however. Protecting data and applications in traditional data centers is fairly well understood. In addition to security appliances and measures like firewalls, intrusion detection, intrusion prevention, anti-malware controls, encryption, access controls, monitoring, and logging, there are physical security measures like locked cabinets and cages, and these all go a long way toward defending sensitive data.

Software-defined virtualized data centers are very different, though. The unit of deployment is a virtual machine, and the entire lifecycle, connectivity, and storage associated with these virtual machines is defined, managed, monitored, and secured using software. The ability of virtual machines holding sensitive and confidential data to be easily moved or replicated in a matter of seconds requires special solutions and protections designed for virtual and cloud computing environments. Most Infrastructure as a Service (IaaS) providers either store the virtual machines in the clear—meaning no encryption—or encrypt them with keys that they control. This may be acceptable for some tenants, but most public or hybrid cloud tenants are concerned about leakage of data and sensitive information stored in some of these virtual machines. For example, insider threats at the service provider constitute legitimate concerns for organizations. Organizations are also concerned about the privacy of data running and processed in the virtual machines, and they must demonstrate the ability to measure and control risk, owing to the significant implications for meeting legal and fiduciary responsibilities.

Another essential aspect of the virtual machine lifecycle is decommissioning. A cloud service provider replicates virtual machines to multiple locations and availability zones as a matter of policy, ensuring later availability and for disaster recovery. While this allows service providers to comply with demanding SLAs, it raises security risks. Geographically dispersed copies of virtual machines can also proliferate sensitive data, credentials, and information, leaving it floating in the cloud. Additionally, a benefit end users get from cloud use is the ability to switch providers. Former customers need to be assured that they can make a clean break when they switch providers. This includes the

ability to destroy any virtual machine and associated data left at the former provider, including backups. Most cloud service providers can't promise that, if only because in the current state of the art there are no standards for proving that disks and backup media are properly wiped before disposal or repair.

Standards organizations and compliance regulation bodies have started to acknowledge these needs for requirements. The Payment Card Industry (PCI) Standards Council recently released an information addendum to the latest data security standard (DSS) specification regarding vulnerabilities laid to virtualization, including exposure of personally identifiable information (PII) and credit card information residing in the virtual machines. The addendum also highlights vulnerabilities with regard to snapshot files and virtual machine backups. The dormant virtual machines in these backups can lie there for years, to be spun up anytime and anywhere, exposing the data and sensitive information.

Consequently, in addition to the encryption of application data, new PCI guidelines recommend encrypting virtual machine images with an operating system and applications with keys managed securely to reduce the footprint of any sensitive data left behind. Figure 8-1 illustrates the concept of tenant-controlled virtual machine encryption and decryption.

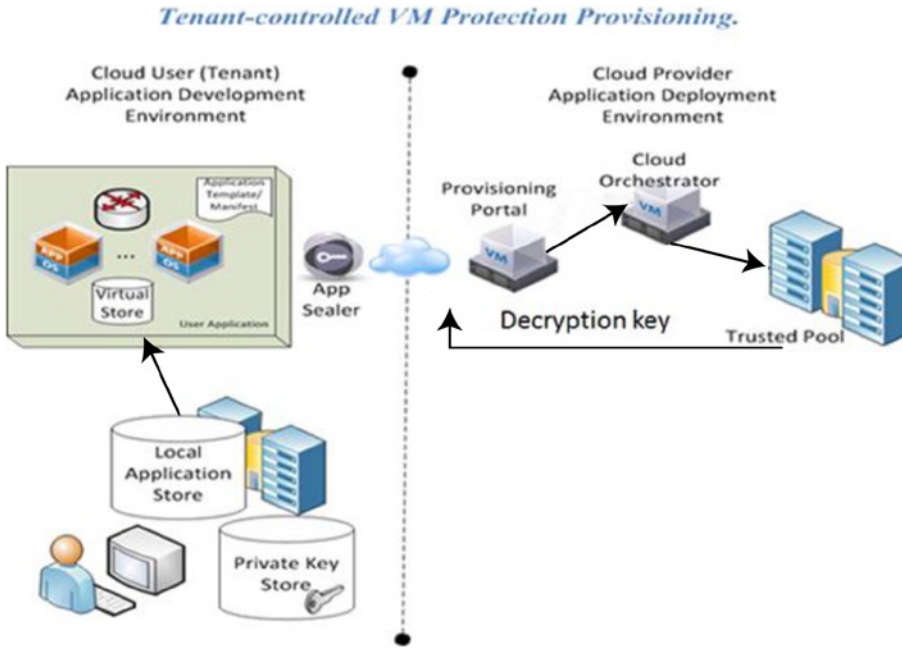

Figure 8-1. *Tenant-controlled virtual machine encrytion and decryption*

In addition to confidentiality protection, organizations would like to verify the integrity of virtual machines before launching them. For instance, if a hardened Linux virtual machine configuration is available, and a user wants it, the user will want a proof of this machine's being used. Doing so within the cloud model is harder than

within a corporate-owned infrastructure, since the tenant organization doesn't own the infrastructure. A chain of trust rooted in hardware, with continuous monitoring of the integrity of infrastructure, the workloads, and virtual machines, can provide the assurance the organization wants.

All of this can be distilled into three key requirements that need to be in place to ensure integrity and confidentiality of these virtual machines in a virtual and cloud environment:

- ***Virtual machines must be launched on servers with provable boot integrity.*** The trusted compute pools usages, platform/host attestation, and geo-fencing solutions described in Chapters 3, 4, and 5 address this requirement. This is foundational.
- ***Virtual machine images must be encrypted in transit, at rest, and during execution.*** This is essential to preserve confidentiality and secrets. The keys are under the control of the tenant, and they are released (key policy management) to the service provider only when they provide attestation that the virtual machine images are being launched on trusted servers.
- ***Launch and provision only qualified and attested virtual machine images***. Virtual machines about to launch must attest their launch integrity with the infrastructure.

In the rest of the chapter we'll cover usages, a conceptual architecture, and a reference implementation that addresses requirements for service providers to offer protection of tenant payloads. Before we jump into details, though, we have to look at the basics of a virtual machine image to understand what needs to be encrypted and what needs to be measured and attested. We will look at the various virtual image formats and also consider virtual machine templates, a standard operating procedure for cloud operators to instantiate virtual machines.

Virtual Machine Images

Virtual machine images come in two different formats: *disk* and *container*. Hypervisor management tools accept both formats, and so do most of the cloud management platforms. However, as of this writing, the OpenStack Image Service (Glance) and other projects do not support the container format. It is possible, however, to associate metadata with images using OpenStack Image Service properties (key/value pairs).

The *disk format* of a virtual machine image is the format of the underlying disk image. Virtual machines can have different formats for laying out the information contained in a virtual machine disk image, as outlined in Table 8-1.

Table 8-1. *Virtual Disk Image Formats*

Type	Description
raw	An unstructured disk image format
vhd	VHD, a common disk format used by virtual machine monitors from VMWare, Xen, Linux/KVM, Microsoft, VirtualBox, and others
vmdk	Common disk format supported by many common virtual machine monitors
vdi	Format supported by VirtualBox virtual machine monitor and the QEMU emulator
ISO	An archive format for the data contents of an optical disc, such as CD-ROM
qcow2	Format supported by the QEMU emulator that can expand dynamically and that supports Copy on Write
AKI	An Amazon kernel image (as used in Amazon Web Services EC2)
ARI	Amazon ramdisk image
AMI	Amazon machine image

A *container format* indicates whether the virtual machine image is in a file format that also contains metadata about the actual virtual machine. The container format for a virtual machine image is a self-contained package with two items:

- Metadata about the virtual machine.
- One or more virtual disks containing the operating system, applications and data. The virtual disk is on the formats as described above.

The image by itself can be a representation of one virtual machine image or could be a composition of multiple related virtual machine images pertaining to a multi-tier service or distributed application workload.

Let's look at the details of the image components. The metadata included in the virtual machine image includes specific information about the operation information about the virtual machines, including:

- Metadata describing server resources needed to run the image: number of CPUs, either dedicated or shared, and memory requirements for the workloads running in the virtual machine.
- Metadata articulating the goals and constraints. This comprehends performance and availability goals and any placement constraints, such as security isolation.

- Metadata describing virtual machine configuration variables like IP addresses and application configuration parameters.
- Metadata describing the package integrity, like SHA1/SHA2 hashes of the virtual disks and modules.

Virtualization management software uses metadata information to create the right type of virtual machine container with the required platform resources. Once the virtual disk image, built as a bootable image, is copied and made accessible to the physical host system, the virtual machine is started from the bootable disk image.

The Open Virtualization Format (OVF)

Virtual machine images can be assembled and delivered in many ways. To ensure interoperability of virtual machines and seamless deployment and management across virtualization platforms, standardization of the virtual machine distribution format is essential. That is exactly what Open Virtualization Format (OVF) is. OVF is a hypervisor agnostic virtual machine packaging and distribution format, standardized by the Distributed Management Task Force (DMTF). It provides a complete description of a single virtual machine or a complex multi-virtual machine software solution package. It is extensible so that DMTF or third parties can add features and extensions to it. OVF goes beyond just the description and virtual hardware attributes.

Open Virtualization Format allows a virtual appliance/ISV vendor to add items like a EULA, comments about the virtual machine, boot parameters, minimum requirements, and a host of other features. OVF specification calls for an OVF descriptor, typically called the OVF envelope, which is an XML file describing the software in the OVF package. The OVF descriptor has ten core sections for metadata, such as the virtual hardware, EULA, product information, and so on. In order to support the extensibility, the OVF specification provides *extension points*. These custom extension points may be used to specify items such as multiple networks, specific firewalls, firewall rules required for the virtual machines, and the setup of a load balancer for multiple instances of virtual machines. A virtual machine author can describe in "levels" the conformance to the OVF specification as part of the OVF envelope. Level 1 indicates that there are no custom extensions, Level 2 indicates that there are custom extensions but they are optional, and Level 3 indicates that the custom extensions are required. This information helps the deployer to figure out the appropriate set of virtualization environments to deploy the virtual machines.

Intel has been working with the industry to deliver a solution architecture and implementations that address security requirements in virtualized environments. The usage models are called *trusted virtual machines* and are code-named *Mystery Hill* (abbreviated MH). The usage models cover the tenant-controlled encryption of virtual machines with tenant-controlled key management , decryption of the virtual machine images and data on servers or hosts with attested integrity and hardware roots of trust, and the attestation of the virtual machines prior to their launch. In this section, we discuss a reference architecture, with details about the key components of the architecture and the workflow. Following this, we present a reference implementation in the OpenStack cloud environment.

A Conceptual Architecture for Trusted Virtual Machines

Figure 8-2 shows the conceptual architecture for trusted virtual machines. There are four key elements needed to enable the usages described in the previous sections:

1. Mystery Hill (MH) client
2. MH key management and policy server (KMS)
3. MH plug-in that runs on the host or server
4. Mt. Wilson trust attestation server technology

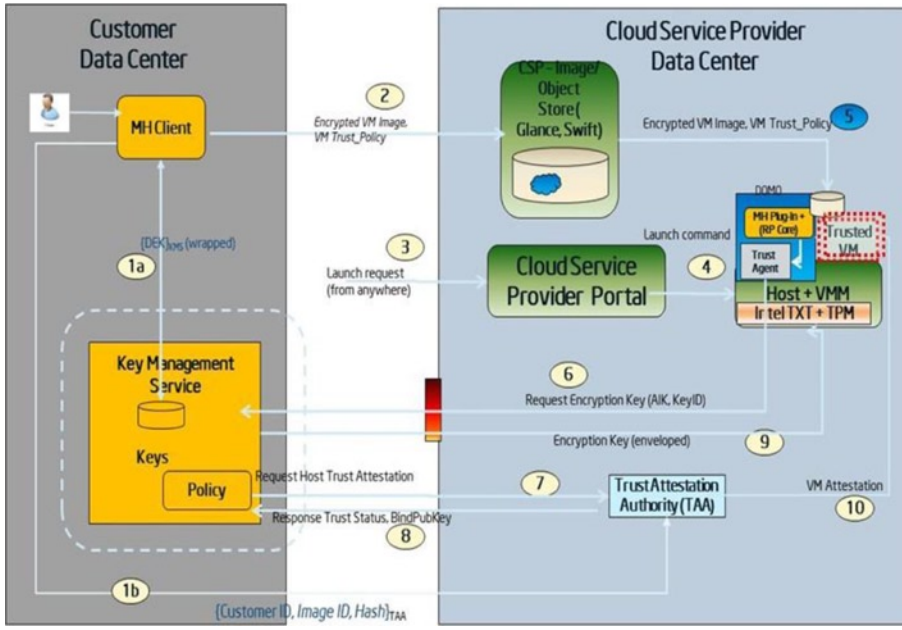

Figure 8-2. *Conceptual architecture for trusted virtual machines*

The conceptual architecture calls for loosely coupled components with well-defined APIs and interfaces. Let's look at each of these in detail.

Mystery Hill (MH) Client

The Mystery Hill (MH) client is an application that runs under tenant or organization control and that carries many functions. It is the primary mechanism by which the service owner (tenant) encrypts the virtual machine images, generates the module manifest for the VMs (list of all files that need to be measured and verified), generates SHA-1/SHA-2

hash values of the modules (whitelist), and specifies the VM trust policies. The MH client interfaces with the key management server (KMS), which is assumed to be a service under tenant or organization control. The MH client also interfaces with the cloud management software, such as Amazon AWS EC2 or OpenStack, to be able to upload the encrypted virtual machine images and the meta-data onto the image server for provisioning and launch.

The VM payload includes one or more encrypted virtual machine images and the metadata. The metadata contains:

- Module manifest and hash of the modules, signed by the trust authority (like Mt. Wilson)
- Key ID (data encryption key ID) to determine the decryption key in the KMS store
- URL for the KMS from where the decryption key can be obtained
- VM trust launch policies

We will cover the trust launch policies in a later section. The MH client generates a new symmetric encryption key for each VM that is being encrypted, wraps the symmetric key using an asymmetric key (provided by the KMS), and posts the wrapped key to the KMS. The metadata is stored in either the OVF envelope for the virtual machine (if the container format of virtual machine image is used) or as additional attributes or properties of the image server (for disk-based virtual machine image formats). Figure 8-3 shows the OVF envelope with the metadata extensions.

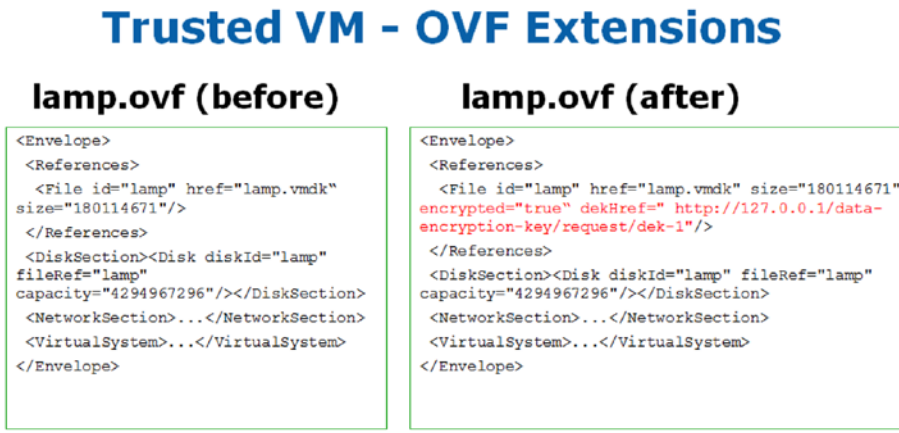

Figure 8-3. *OVF extensions for VM encryption*

Mystery Hill Key Management and Policy Server (KMS)

The Mystery Hill key management and policy server (KMS) is the core key generation and management service, designed to allow control of key generation and management functions by the tenant organization. It provides the standard web services interface for generating the keys and delivering those keys to the MH client after successful

authentication, to allow an MH client to encrypt virtual machine images. It also provides the decryption key URLs to the MH client to be included in the OVF envelopes. We recommend for the KMS to use a hardware security module (HSM) for the generation, storage, and processing of the keys. The second function of the key management and policy server is to evaluate trust policies before decryption keys are handed to the requesting entity. The trust policies can include user roles, license verification, application certification, server or host platform integrity and attestation, and geolocation or geo-fencing policies, as well as network attributes.

Mystery Hill Plug-in

The Mystery Hill plug-in is a set of two components that reside and run with the hypervisor on the host/server on which the virtual machines are being deployed. These components are:

- *MH Agent.* Part of the platform trust computing base (TCB), it performs the functions for decryption of the tenant virtual machines and integrates with the virtualization and cloud management environment. In OpenStack environments, the MH agent integrates with the Nova compute node service to intercept encrypted virtual machine launch requests to the hypervisor, obtains the decryption key from the MH key management service after the server or node attests to its integrity with a trust attestation authority, and decrypts the virtual machine images. It provides local (and transparent) encryption of the decrypted VM images on the compute node disk prior to the launch, so they are not in the clear, even for a short time prior to launch.
- *MH Measuring and Quoting Agent.* This component, also part of the platform TCB, measures the virtual machine images, verifies the measurements against the whitelist, and also provides a Quote-analogous to the TPMQuote, which is rooted to the physical TPM on the server. The measuring agent runs with the hypervisor on a host server and measures the VM images, per the manifest sent as part of the VM payload metadata. The MH quoting agent has the primitives to Quote, Seal, and Attest. Based on the VM trust policies, once the measurement and attestation of the VM image is completed, the decrypted image is passed on to the hypervisor to continue the normal launch sequence.

Encrypted virtual machine requests are identified using attributes present in an OVF envelope, similar to what is shown in Figure 8-3. The attributes indicate that the VM image is encrypted and show a URL where the decryption key can be obtained. The URL points to the KMS. When the MH plug-in requests the decryption key from the KMS, it provides the compute node's attestation identity key, to request an attestation from Mt. Wilson. The section "Workflows for Trusted Virtual Machines" below describes the complete process.

■ **Note** The MH measuring agent could get complex, owing to the various virtual machine formats that would need to be comprehended. This would increase the TCB significantly. It is possible to migrate the measuring agent outside the TCB but still ensure its integrity. One way to accomplish this is to keep a hash of the measuring agent as part of the TCB, and to measure/attest the measuring agent before it is launched to perform the measurement of the VM image.

Trust Attestation Server

The trust attestation server is the attestation authority monitoring the compute nodes in the trusted data center, as described in detail in Chapter 4 and shown in Figure 8-4. The attestation server maintains the trust policy for every attested host and evaluates reports from the hardware roots of trust on each node to determine if each node is in compliance with its trust policy. The attestation server tracks the attestation identity key (AIK), public key, and an encryption public key for each compute node, among other information. When the KMS requests an attestation of a compute node that in turn requests a decryption key for an encrypted virtual machine, the MH key management server provides the AIK public key to the attestation server in order to uniquely identify the compute node to attest. The attestation server identifies the compute node to attest based on the provided AIK public key and returns the result to the KMS.

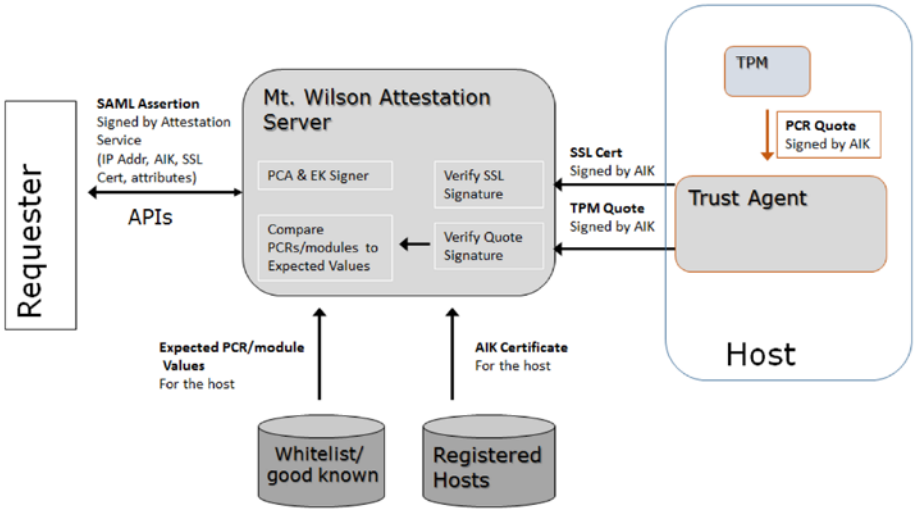

Figure 8-4. *Attestation architecture*

Workflows for Trusted Virtual Machines

Figure 8-5 shows the flow of encryption, decryption, and launch of the virtual machines according to the conceptual architecture. The workflow builds upon the foundation established by the trusted compute pools to support virtualized cloud environments, adding the concept of trusted virtual machines. This is how the security capabilities and benefits of trusted compute pools are extended and become usable in virtualized cloud environments. Note that the elements in the workflow can be implemented in a distributed and scalable fashion under the cloud paradigm, in which Mt. Wilson and the KMS can be delivered through a PaaS provider, the cloud service provider and the hosts with Intel TXT are IaaS instances, and the cloud storage is a SaaS instance. There are nine steps involved in this workflow, noted as follows:

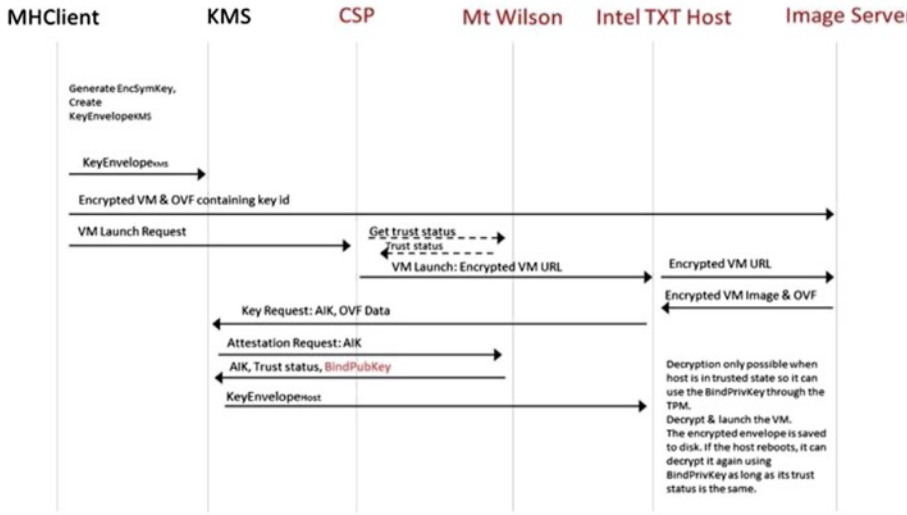

Figure 8-5. *Encryption and decryption flow diagram for trusted virtual machines*

***Step 1*:** The launch of an encrypted virtual machine begins with creating an encryption key. The MH client can automatically generate new encryption keys as needed. Users have the flexibility of using the same key multiple times to protect different virtual machine images, or to create a new key for every virtual machine image. Every key that the MH client generates is wrapped and posted to the KMS so that it can later be retrieved for decryption. The corresponding key URL, provided by the KMS when the encryption key was wrapped and posted, is also stored by the MH client.

***Step 2*:** This involves selection of an encryption key and a plaintext virtual machine image, and subsequent encryption with the key. The encrypted virtual machine image can then be uploaded to servers in the data center. See the upper arrow to "Cloud Storage" in the flow diagram. The MH client attaches metadata to the virtual machine image—in particular for this case, the URL identifying the encryption key.

***Step 3*:** Once the encrypted virtual machine image and associated metadata are posted in the cloud, the virtual machine can be launched using the cloud service provider's existing mechanism for launching virtual machine workloads. This is noted in the diagram as "VM Launch Request."

***Step 4*:** Cloud service providers featuring trusted compute pools can query the trust status of a compute node and ensure that it is trusted before sending a new workload to it. This check is performed by sending an attestation request to the attestation server and inspecting the result. After identifying an available trusted compute node, the cloud service provider sends a launch request to that compute node with the encrypted virtual machine image information. The launch request includes the image metadata, image download URL, and decryption key URL. At this point, if the compute node does not have the MH plug-in, or has not been enabled with Intel TXT and registered with the attestation server, it will not be able to continue with the launch. The system prevents the encrypted image from even reaching the hypervisor.

***Step 5*:** An Intel TXT-enabled trusted compute node with the MH plug-in can detect that the VM image is encrypted by examining the metadata or the OVF plug-in. The MH plug-in connects to the decryption key URL and sends the compute node's AIK public key to that URL to identify the compute node that is requesting the decryption key. The URL points to the KMS, which already has the decryption key, wrapped by MH client.

***Step 6*:** The attestation server looks up the compute node details using the AIK public key, obtains a TPM quote from the compute node, and verifies the compute node's state against its trust policy. The result is reported back to the KMS. In addition, if the attestation server determines the compute node complies with its trust policy, the attestation server will then provide an asymmetric encryption public key for the compute node to the KMS, noted in the diagram as "BindPubKey."

***Step 7*:** The KMS, upon receiving a favorable trust report on the compute node from the attestation server, wraps the requested decryption key and sends it back to the compute node. The key wrapping is done using the public key of the compute node, as provided by the attestation server. The compute node has the corresponding private key. It's possible to bind the private key to the TPM on the compute node such that it can be used only if the host complies with its trust policy.

***Steps 8, 9, and 10*:** The compute node unwraps the decryption key with the TPM, decrypts the encrypted VM, and as an additional step, can measure the virtual machine image and attest with the trust attestation authority before handing control to the compute node to proceed with the launch as usual.

As seen from this flow, the virtual machine image is protected at rest, in motion, and up until execution with protection under tenant control. The decryption keys are released only when the cloud service provider can demonstrate the integrity of the server on which the virtual machine is being provisioned.

Deploying Trusted Virtual Machines with OpenStack

Here, we document a reference implementation constructed to demonstrate trusted virtual machine usages with OpenStack. This proof point provides an opportunity to highlight the finer points of an actual implementation, allowing a more accurate evaluation of the value proposition, its applicability, and its usability. The reference implementation has been demonstrated at a number of industry-wide security conferences, including the OpenStack Summit, with excellent reviews. We expect the independent software vendor and cloud service provider community to scale production implementations of this usage to their own infrastructures—and ultimately to their service offerings—so as to offer a soup-to-nuts chain of trust, running from cloud service offerings all the way down the layers to the underlying host hardware. This implementation is an instantiation of the workflow described in the previous section.

Figure 8-6 shows the reference implementation in OpenStack. It works with OpenStack environments using either KVM or Xen hypervisors. A set of screenshots of the implementation follows, illustrating the flow and the process of integration into OpenStack.

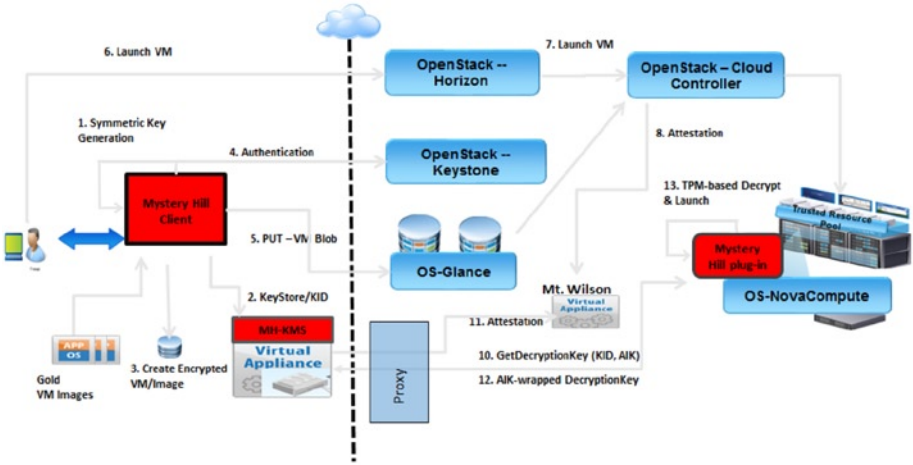

Figure 8-6. *Tenant-controlled virtual machine encryption and decryption with trusted launch in OpenStack*

As shown in Figure 8-6, the process begins with creating the encryption keys to be applied to the virtual machine images. The goal is to implement a secure process under tenant control. The tenant or the service consumer, perhaps DevOps staff, uses the MH client to create keys, store them in the KMS, and encrypt the virtual machine images. In OpenStack, a virtual machine image is assumed to be in disk image format, ISO, vmdk, xva, or vhd. Although it is possible to upload encrypted virtual machine images manually to OpenStack Glance image service, and to use the Glance client to add the encryption metadata, it is far more convenient to let the MH client upload the encrypted virtual machine image and add the encryption flag and decryption key URL to the Glance image metadata using the Glance APIs.

As can be seen in Figure 8-7, there are two steps before an encrypted virtual machine can be uploaded to OpenStack Glance. First, either an existing key is selected or a new key is generated to use for encryption. The new key interfaces with the KMS. (For the reference implementation, no HSM was used with the KMS.) The KMS returns the URL and a decryption key ID (DKID) to be associated with the encrypted target virtual machine. Second, a virtual macine image to encrypt is selected and encrypted using the key just selected or generated. Third, the encrypted VM image is uploaded along with its encryption metadata to Glance, using the Glance client to upload the image and set its metadata in Glance.

Figure 8-7. *Selecting and binding a virtual machine image to an encryption key*

The Launch tab, as shown in Figure 8-8, simulates a launch of a specific virtual machine. In practice this will be done either through use of OpenStack virtual machine launch APIs or from a portal such as OpenStack Horizon. The Launch tab features a single launch step and a progress monitor. To launch an encrypted virtual machine, the encrypted virtual machine image is selected from the list, a trusted flavor is chosen to use for the virtual machine instance, and the Launch button is clicked. The MH client reference implementation conveniently preselects the last virtual machine image encrypted in the Upload tab. The flavor list is downloaded from Nova Controller and reflects the same list as is available in the Horizon dashboard.

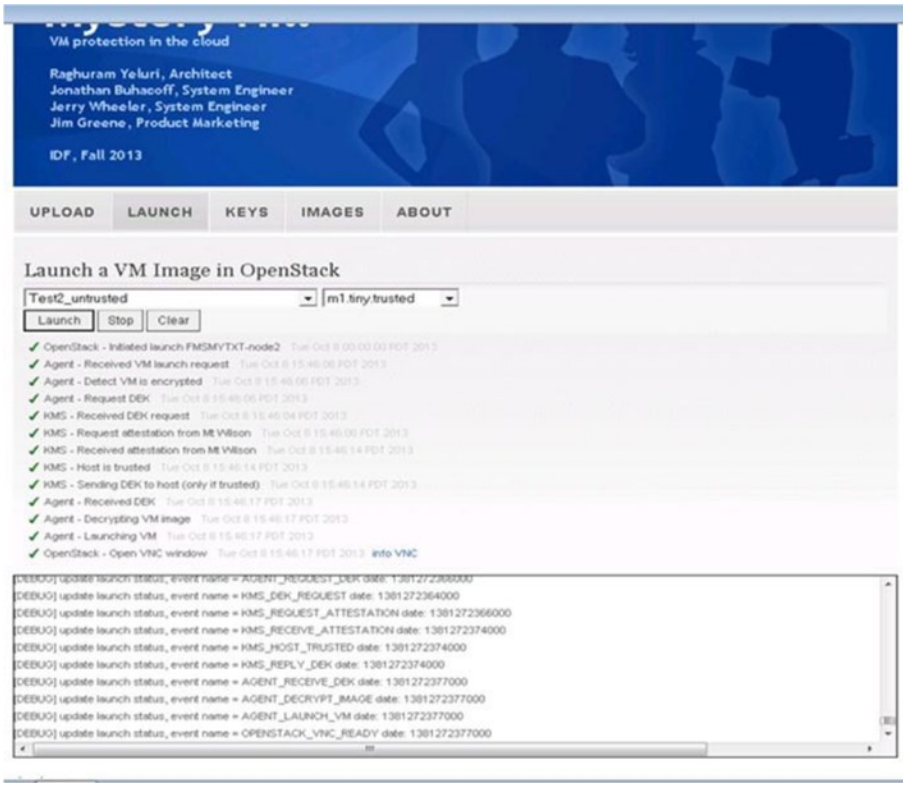

Figure 8-8. *Launching a virtual machine image*

■ **Note** Trusted compute pools (TCP) function, as described in chapter 3, has been implemented in OpenStack as scheduler filters. These extensions, and the Horizon dashboard and API extensions to tag Flavors with trustpolicies, have been part of OpenStack since the Folsom release.

In the screenshot of the Launch tab, the selected flavor is `m1.tiny.trusted`, which in the reference implementation refers to a single virtual CPU, with 512 MB of memory, no extra disks, and a trust requirement from the attestation server. When a virtual image is launched with a trusted flavor, the trust scheduler in OpenStack Nova queries the attestation server for the trust status of available compute nodes and only selects trusted compute nodes for launching the virtual machine instance. When the Launch button in the Launch tab is clicked, the MH client uses the Nova client to request a launch of the selected encrypted virtual machine image using the selected flavor. The trust scheduler selects a trusted compute node and initiates a launch of the virtual machine image instance on that compute node.

The reference implementation integrates MH plug-in with the Nova compute node for KVM and Xen hypervisors. On KVM, the MH plug-in integrates with the *libvirt* driver (`driver.py`) on the compute node to intercept the launch request of an encrypted image, request the decryption key, and decrypt the image. On Xen, the MH plug-in integrates with the `xapi.d` plug-in for Glance (in *dom0*) to perform the same actions, but integrates the decryption into the image stream downloaded from Citrix into the Xen hypervisor's work area.

When the MH plug-in requests the decryption key from the KMS, it sends an HTTP POST request to the decryption key URL noted in the image metadata. The body of the POST request is simply the AIK public key of the compute node. On KVM nodes, the AIK public key is managed by the trust agent, an additional component required for the attestation server. On Xen nodes, the AIK public key is managed by the Xen API performing the same functions as the trust agent.

The KMS forwards the AIK public key to the attestation server to obtain a report on that compute node. If the compute node is trusted, the report includes a public key that can be used for wrapping keys to be sent to the compute node. The corresponding private key is bound to the TPM on the compute node. The KMS wraps the decryption key using the compute node's public key and sends it to the compute node. This mechanism ensures that the key can be unwrapped only by the compute node that was reported as trusted by the attestation server. This enables some flexibility on the part of the cloud service provider to anonymize, proxy, aggregate, or otherwise manage the decryption key requests without the risk of leaking the decryption key to any intermediate network node.

The MH plug-in receives and unwraps the decryption key, uses it to decrypt the encrypted VM image, measures the virtual machine image with additional primitives in DOM0, and attests to it with the attestation server. The attestation report from the attestation server indicates whether it can be launched. If so, the virtual machine launch continues as usual. In this reference implementation, the sequence of steps is reflected in the checkboxes on the Launch tab.

Summary

Building on the foundation of trusted compute pools, the concept of trusted virtual machines extends the chain of trust in the cloud computing environment to cover guest virtual machines and associated workloads. In this chapter we covered what trusted virtual machines mean, how a tenant can control the encryption and decryption keys, and how to protect the confidentiality and integrity of the virtual machines in transit, at rest, and up to execution, using encryption and decryption and other policy implementation techniques. We presented a reference architecture for realizing the vision of trusted virtual machines, and also reviewed a reference implementation of that architecture as it appears in OpenStack. Clearly, this implementation is very early in the industry's process for realizing the full vision of trusted virtual machines. The need is there: users are demanding cloud providers to offer security for their virtual machines, while permitting cloud customers to retain control over encryption keys. They would also like to see decryption keys released only when the service provider can

demonstrate the integrity of the compute nodes on which the virtual machines are going to be deployed and launched. The model for trusted virtual machines showed how it is possible to bind decryption of the keys to the TPM on a server that has demonstrated integrity. This ensures that the virtual machine is decrypted only inside the trusted server and not anywhere else.

Intel has started to work with the community of independent software vendors and cloud service providers to develop the solutions that bring trusted virtual machine usages and associated technical architecture to scalable and production-ready offerings that can be used with private, public, and hybrid cloud deployments. Chapter 9 brings together all the concepts, usages, and technologies that we have reviewed in the first eight chapters, via a compelling usage model called "Secure Cloud Bursting."