APPROVAL SHEET

Title of Dissertation: BUILDING TRUST INTO UTILITY CLOUD COMPUTING

Name of Candidate: Frank John Krautheim
Doctor of Philosophy, 2010

Dissertation and Abstract Approved:

_____

Dhananjay S. Phatak
Associate Professor
Department of Computer Science and
Electrical Engineering

_____

Alan T. Sherman
Associate Professor
Department of Computer Science and
Electrical Engineering

Date Approved: _____

Curriculum Vitae

Name: Frank John Krautheim

Current Residence: Severn, MD

Degree and Date to be Conferred: Doctor of Philosophy, 2010

Date of Birth: September 25, 1967

Place of Birth: Madisonville, KY

Secondary Education:

Dawson Springs High School, Dawson Springs, KY.

Collegiate Institutions Attended:

University of Maryland, Baltimore County, Baltimore, MD.
Doctor of Philosophy, Computer Engineering, 2010.

National Defense University, Fort McNair, DC.
Advanced Management Program, 2006 *Distinguished Graduate*
Chief Information Officer Certificate, 2006.
Committee on National Security Systems 4011 Certificate, 2006.

University of Cincinnati, Cincinnati, OH.
Master of Science, Electrical and Computer Engineering, 1992.

University of Kentucky, Lexington, KY.
Bachelor of Science, Electrical Engineering, 1990.

Professional Publications:

F. John Krautheim, Dhananjay S. Phatak, and Alan T. Sherman, "*Trusted Virtual Environment Module: Managing Trust in Cloud Computing*," in 3rd International Conference on Trust and Trustworthy Computing, Berlin, June 2010.

F. John Krautheim, Dhananjay S. Phatak, and Alan T. Sherman, "Private Virtual Infrastructure: A Model for Trust in Cloud Computing," TR-CS-10-04, University of Maryland Baltimore County, Baltimore, MD, 2010;
`://www.cisa.umbc.edu/papers/krautheim_tr-cs-10-04.pdf..

F. John Krautheim, “Private Virtual Infrastructure for Cloud Computing,” in Workshop on Hot Topics in Cloud Computing 2009, San Diego, CA, June 15, 2009.

John Krautheim, “Summary of 2008 USENIX Annual Technical Conference Virtualization Session,” *;login: The USENIX Magazine*, October 2008, Vol. 33, No. 5, pp 95-97.

F. John Krautheim and Dhananjay S. Phatak, “Identifying Trusted Virtual Machines,” in Xen Summit 2008, Boston, MA, June 24, 2008.

F. John Krautheim and Dhananjay S. Phatak, “Controlled Reincarnation Attack to Subvert Digital Rights Management,” Work in Progress report, 16th USENIX Security Symposium, Boston, MA, August 10, 2007.

F. John Krautheim, “Utility of a General Purpose Digital Simulator for Fluid-Structure Interaction Problems,” Master’s Thesis, University of Cincinnati, 1992.

K.T. Moore., F.J. Krautheim, B.A. Naylor, B.K. Walker, A.H. Nayfeh, and P.K. Khosla, "Analytical Modeling and Modular Simulation of Feedline System Elements with Fluid/Structure Interaction," Proc. of 3rd Annual Health Monitoring Conf. for Space Propulsion Systems (Cincinnati), U. Cincinnati, November 1991.

Other Academic Activities and Achievements:

2009 USENIX Annual Technical Conference, San Diego, CA, June 16-19, 2009.
Hot Topics in Cloud Computing, Short Paper Presentation, “Private Virtual Infrastructure for Cloud Computing,” San Diego, CA, June 15, 2009.
Trusted Infrastructure Workshop, Scholarship, Carnegie Mellon University, Pittsburgh, PA, June 8-12, 2009.
UMBC CSEE Research Review 2009, Poster – 3rd Place, “Does the Empire Control Your Cloud?” May 1, 2009.
2008 USENIX Annual Technical Conference, Student Grant, Boston, MA, June 25-27, 2008.
Xen Summit Boston 2008, Invited Presentation, “Trusted Virtual Machine Identification,” June 23-24, 2008.
UMBC CSEE Research Review 2008, Poster, “Identifying Trusted Virtual Machines,” May 2, 2008
UMBC Graduate Research Conference 2008, Presentation, “Identifying Trusted Virtual Machines,” April 25, 2008.
IBM T.J. Watson Research Center, Invited Presentation, “Trusted Virtual Machine Identification,” Hawthorne, NY, November 30, 2007.
16th USENIX Security Symposium, Student Grant, Boston, MA, August 6-10, 2007.

Professional Positions Held:

National Security Agency, Fort George G. Meade, MD.

Engineering Researcher	April 2010 – Present
Computer Security Manager	July 2008 – March 2010
Information Assurance Scholar	September 2006 – July 2008
Project Director	July 2003 – September 2006
Technical Director	Aug 2002 – July 2003

IKOS Systems, Austin, TX.

Field Application Engineer	May 1998 – May 2001

Symbios Logic, Colorado Springs, CO.

Senior Principal IC Design Engineer	Feb 1997 – May 1998

Motorola, Libertyville, IL.

Senior Engineer	June 1994 – Feb 1997

Image Science, Inc., Mason, OH.

Staff Engineer	June 1992 – March 1994

Professional Certifications Held:

International Information Systems Security Certification Consortium (ISC)^2

Certified Information Systems Security Professional (CISSP)	2006

Defense Acquisition Workforce Improvement Act (DAWIA)

Program Manager – DAWIA Level II	2005
Systems Engineer – DAWIA Level II	2005

Project Management Institute (PMI)

Project Management Profession (PMP)	2004

ABSTRACT

Title of Dissertation: BUILDING TRUST INTO UTILITY CLOUD COMPUTING

Frank John Krautheim, PhD, 2010

Directed By: Dhananjay S. Phatak
Associate Professor
Department of Computer Science and Electrical Engineering

Alan T. Sherman
Associate Professor
Department of Computer Science and Electrical Engineering

We introduce three new mechanisms that allow trust to be built into cloud computing called the *Private Virtual Infrastructure (PVI)*, the *Locator Bot (LoBot)*, and the *Trusted Virtual Environment Module (TVEM)*. Cloud computing requires that organizations trust that a service provider's platforms are secured and provide a sufficient level of integrity for the client's data. Once a client's sensitive data are released into the cloud under the control of a third party, a significant level of risk is placed on the security and privacy of the data. PVI, LoBot, and TVEM provide a means for clients to establish trust in cloud platforms, thus reducing their risk exposure.

PVI is a new management and security model that shares the responsibility of security management in cloud computing between the service provider and client, decreasing the risk exposure to both. The PVI datacenter's security posture is set by the

client, while the cloud's configuration is under control of the service provider. Clients can then protect their information independently of the cloud configuration.

The LoBot pre-measures the cloud for security properties which can be used to determine the integrity and trustworthiness of the destination platform. LoBot then provides secure provisioning and live migration for the virtual datacenter. LoBot protects information by preventing data from being placed in malicious environments.

The TVEM helps solve the core security challenge of cloud computing by establishing trust in a virtualized cloud computing environment. The TVEM is a software appliance that merges trust from multiple sources, typically the information owner and service provider, to derive a root of trust for a virtual environment on a remote host. A unique *Trusted Environment Key (TEK)* combines trust from the information owner and the service provider to create a dual root of trust for the TVEM that is distinct for every virtual environment and separate from the host platform's trust.

PVI, Locator Bot, and TVEM can be used individually or combined to provide a foundation for trust in cloud computing. They enable organizations to maintain control of their information in the cloud and realize benefits of cloud computing.

BUILDING TRUST INTO UTILITY CLOUD COMPUTING

by

Frank John Krautheim

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, Baltimore County, in partial fulfillment
of the requirements for the degree of
Ph.D. in Computer Engineering
2010

Dedication

```
$DO || ! $DO ; try

try:  command not found
```

To Sean and Christopher: I have missed a lot of things while I have been working on this project. But hopefully you will see the benefit that hard work provides. I did this to give you guys a better life. I love you.

To Kimberly: Thanks you for your loving support, encouragement, and understanding. I love you.

To my mother, Lou Nell Hensley: Thank you for providing me with the will and courage to tackle the toughest tasks.

In loving memory of my uncle, Dr. Joseph Krautheim (1935-2005), who inspired me to reach for new academic challenges.

Acknowledgments

I could not have done this without the support of my wife Kimberly. Without her picking up the slack when I was unavailable for the family, there would have been no way I could have accomplished this. Sean and Christopher had to suffer because their dad couldn't be there since he was "working." I hope they see that it was all worth the effort.

My co-advisors Dr. Dhananjay Phatak and Dr. Alan Sherman have guided me through this journey. Their direction, support and encouragement were invaluable. Dr. Phatak's immense imagination and brilliance, which are sometimes overshadowed by his illness, provided an endless supply of ideas and research topics. Dr. Sherman provided not only financial support from the Information Assurance Scholarship Program grant, but his mentorship led my journey through the world of academia.

I wish to thank my committee members Dr. Chintan Patel, Dr. Brooke Stephens and COL Ronald Dodge (United State Military Academy). Several other faculty members at UMBC have been especially helpful including Dr. John Pinkston, Dr. Tim Finin, Dr. Charles Nicholas, and Dr. Jim Plusquelic.

I would be remiss if I left out Jane Gethmann and Dee Ann Drummey, whose administrative support and assistance have been invaluable. Without them, I would have been at the mercy of university bureaucracy. Also, Geoff Weiss' technical support and practical knowledge is greatly appreciated.

I wish to give a special thanks to Dr. David Challener for providing invaluable knowledge of the TPM, advice on the TVEM design, and history of the Trusted Computing Group.

I could only have achieved this with the grateful support of my employer, supervisors, and co-workers. There are too many to name, but Sandra Lehr, Mark Roberts, Anne Bishop, Tom Maskell, Jim Hession, and Stephen Campbell all provided support, friendship and advice when needed. Extra special thanks goes to Byron Wilson for reviewing this dissertation and the publications that preceded it.

Completing any graduate program would not be possible if it were a solo journey. The friendship and support of my labmates in the Cyber Defense Lab and classmates at UMBC were invaluable. Rick Carback and Russ Fink were always there to bounce ideas off of, discuss the latest security issues, or wallow in the misery of academia. Michael Oehler and I worked through many of the same issues juggling work, family, and school. Others friendships that have blossomed along the way are Patrick Trinkle, Kevin Fisher, Cory Jones, Patricia Ordóñez Rozo, Ryan Helinski and Bruce Howell.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AIK	Attestation Identity Key
API	Application Program Interface
AS	Application Server
BIOS	Basic Input Output System
CRTM	Core Root of Trust for Measurement
CVD	Corporate Virtual Desktop
CVF	Cloud Virtual Fabric
EK	Endorsement Key
EV	Extended Validation
HP	Host Platform
HVM	Hardware Virtual Machine
IaaS	Infrastructure as a Service
LoBot	Locator Bot
LSM	LoBot Secure Migration
LSP	LoBot Secure Provisioning
MAC	Mandatory Access Control
MLE	Measure Launch Environment
NVRAM	Non-Volatile Random Access Memory
OS	Operating System
PCR	Platform Configuration Register
PVI	Private Virtual Infrastructure
RNG	Random Number Generator
RTM	Root of Trust for Measurement
RTR	Root of Trust for Reporting
RTS	Root of Trust for Storage
SAN	Storage Area Network
SCR	State Certification Register
SRK	Storage Root Key
SSL	Secure Socket Layer
TCB	Trusted Computing Base
TCFP	Trusted Cloud Fabric Platform
TCP	Trusted Computing Platform
TEK	Trusted Environment Key
TF	TVEM Factory
TLS	Transport Layer Security
TPM	Trusted Platform Module
TSS	Trusted Software Stack
TTVM	Time Traveling Virtual Machine
TVEM	Trusted Virtual Environment Module
TXT	Trusted eXecution Technology

VECR	Virtual Environment Configuration Register
VM	Virtual Machine
VPN	Virtual Private Network
VS	Virtual Server
VTN	Virtual Trust Network
VTPM	Virtual Trusted Platform Module
XM	Xen Management

Technical Overview

Cloud computing is poised to revolutionize *Information Technology (IT)* acquisition and service models. Delivering massively scalable computing resources as a service with Internet technologies, resources are shared among a vast number of consumers allowing for a lower cost of IT ownership. Cloud computing provides on-demand computing resources dynamically, which allows companies to fundamentally change their information technology strategy.

As with any new technology, this new way of doing business brings with it new challenges, especially when considering the security and privacy of the information stored and processed within the cloud. This dissertation examines these challenges and proposes unique solutions to solve the core security problems of cloud computing.

Utility cloud computing allows users to rent *Virtual Machine (VMs)* from a service provider, placing an organization's sensitive data in the control of a third party. This situation places a significant level of risk on the privacy and security of the data processed by the VMs in the cloud. We propose a new management and security model for utility cloud computing called the *Private Virtual Infrastructure (PVI)* that shares the responsibility of security in cloud computing between the service provider and client, decreasing the risk exposure to both. The PVI datacenter is under control of the client or information owner while the cloud fabric is under control of the service provider. A cloud *Locator Bot* (LoBot) pre-measures the cloud for security properties and securely provisions the datacenter in the cloud. PVI and Locator Bot provide the tools that

organizations require to maintain control of their information in the cloud and realize the benefits of cloud computing.

To address the core security challenge of cloud computing where an information owner creates and runs a virtual environment on a platform owned by a separate service provider, we introduce a new mechanism for rooting trust in a cloud computing environment called the *Trusted Virtual Environment Module (TVEM)*. The TVEM is a software appliance that merges trust from multiple sources, typically the information owner and service provider, to derive a root of trust for a virtual environment on a remote host. The TVEM helps solve the core security challenge of cloud computing by enabling parties to establish trust relationships in a cloud computing environment.

Contributions provided by this dissertation are:

- An architecture for protecting privacy and confidentiality in a cloud datacenter that separates security relationships between the user and consumer of cloud services and combines trust from both for the virtual and physical environments.
- The concept of an agent or "bot" virtual appliance to protect provisioning and migrations of *Virtual Machines (VMs)* in a cloud datacenter through fabric pre-measurement.
- Protocols to ensure secure provisioning and secure live migration of virtual machines in the cloud datacenter.
- A new Trusted Virtual Environment Module for rooting trust in a cloud computing environment.

- A Trusted Environment Key that combines components of trust from the virtual environment owner and the platform to form a complex virtual environment trust.
- A Virtual Trust Network and TVEM Factory for managing and provisioning TVEMs from a central control facility.

Private Virtual Infrastructure is a security architecture for cloud computing that uses a new trust model to share the responsibility of security in cloud computing between the service provider and client, decreasing the risk exposure to both. PVI is under control of the information owner while the cloud fabric is under control of the service provider. The PVI architecture comprises a cluster of trusted computing fabric platforms that host virtual servers running an application server with a LoBot security service. The cloud LoBot pre-measures the cloud platform for security properties to determine the trustworthiness of the platform. LoBot uses Trusted Execution Technology and virtual Trusted Platform Modules to pre-measure the target environment and securely provision the Private Virtual Infrastructure in the cloud thus protecting information by preventing data from being placed in malicious or untrusted environments.

The *LoBot Secure Provisioning (LSP)* and *LoBot Secure Migration (LSM)* protocols ensures that when a VM is provisioned within the PVI it is not subverted in any manner and the VM that is provisioned is exactly the same as the VM intended to be provisioned. Secure provisioning is achieved through the combination of determining a target platform's trustworthiness, ensuring correct provisioning, and verifying the VM is created and launched on the host platform as intended. Live migration is an extension of the secure provisioning protocol where a live VM is relocated from one platform to another while remaining operational. LSM ensures that the VM's state is not

compromised by the migration and that the VM migrated to the new platform resumes operation at the point of suspension on the originating machine.

The TVEM provides enhanced features for cloud virtual environments over existing *Trusted Platform Module (TPM)* virtualization techniques including an improved application program interface, cryptographic algorithm flexibility, and a configurable modular architecture. TVEMs are managed via a *Virtual Trust Network (VTN)* with a central control facility called the TVEM Factory that manufactures and provisions TVEMs in the cloud. A unique *Trusted Environment Key (TEK)* is defined that combines trust from the information owner's VTN and the service provider's platform to create a dual root of trust for the TVEM that is distinct for every virtual environment and separate from the platform's trust.

Private Virtual Infrastructure, LoBot, and TVEM provide organizations with assurance that their information is protected. By providing a foundation for trust in cloud computing, they enable organizations to maintain control of their information in the cloud and realize benefits of cloud computing.

Chapter 1

The Future is Cloudy

Luke: I saw – I saw a city in the clouds.

Yoda: [nods] Friends you have there.

Luke: They were in pain.

Yoda: It is the future you see.

Luke: The future?

Star Wars Episode V: The Empire Strikes Back

A fundamental shift is occurring in the way *Information Technology (IT)* and computing services are delivered and purchased. Cloud computing is poised to revolutionize computing as a service where IT becomes a computing utility [1] delivered over the Internet. Cloud computing utilizes massively scalable computing resources delivered as a service using Internet technologies, which allows these computational resources to be shared among a vast number of consumers to allow a lower cost of ownership of information technology.

Companies can fundamentally change their information technology strategies as they turn to the cloud for datacenter and information technology services to improve scalability and global reach, and to lower overhead. There are many benefits to cloud

computing: lower overall cost of IT ownership, increased flexibility, fault tolerance, locality flexibility ability, and to respond to new business requirements quickly and efficiently. As with any new technology, this new way of operating brings new risks and challenges, especially when considering the security and privacy of information stored and processed within the cloud.

One of the risks of cloud computing is that the users, who are the information owners, lose control of their data when they release the information into the cloud for processing. Relinquishing physical control of the datacenter infrastructure and information increases the risk of data compromise considerably [2]; however, benefits (including lower operating costs and increased availability [3]) of moving to cloud computing for services may be significant enough to justify the risk. This dissertation provides solutions for information owners to increase their trust level in cloud computing providers and assure that their private and confidential information is kept secure.

1.1 Motivation

Ensuring the security and integrity of information in the cloud becomes an issue as the management and ownership of the hosting platforms is removed from the consolidated control of a single facility and a single owner. Many organizations such as financial institutions, health care providers, and government agencies are legally required to protect their data from compromise due to the sensitivity of their information. Generally, these organizations are required to manage and maintain their own datacenters with stringent physical and logical protection mechanisms ensuring that their data remains protected. These organizations simply cannot utilize cloud computing in a

generic manner due to the inherent risk of data compromise from systems they do not control.

To date, there has been minimal research published on cloud computing security. This dissertation introduces three new mechanisms for cloud management and security called the *Private Virtual Infrastructure (PVI)*, the *Locator Bot (LoBot)*, and the *Trusted Virtual Environment Module (TVEM)*. These mechanisms allow organizations to use cloud resources with the level of assurance that is required to meet their confidentiality concerns.

1.2 Thesis

Cloud computing requires a new trust model that shares the security responsibility between service provider and client. The core security challenge of cloud computing is that the information owner does not control the hardware that is operating on his data. Complicating the situation is that the hardware is multi-tenant with shared resources among many users.

We leverage trusted computing technology to build trust into utility cloud computing; however, existing techniques do not satisfactorily solve the problem. We define a new trust model for cloud computing and introduce three new mechanisms that provide for building trust into utility cloud computing environments and help solve the core security challenge of cloud computing.

We propose the following four tenets to cloud security to provide a secure framework and increase the security posture for utility cloud computing:

1. Provide a trusted fabric on which to build utility clouds.

2. Provide a secure management facility for utility clouds that also serves as a policy decision point and root authority for utility clouds.
3. Provide a measurement mechanism to validate the security of the fabric prior to provisioning of utility clouds.
4. Provide secure methods for provisioning, shutdown and destruction of virtual machines in utility clouds.

PVI, LoBot and TVEM implement the four tenets to allow organizations to build trustworthy datacenters, maintain control of their information in the cloud, and realize the benefits cloud computing provides.

1.3 Cloud Computing Models and Technologies

The National Institute of Standards and Technology (NIST) definition of cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (*e.g.,* networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model promotes availability and is composed of three service models: *Software as a Service (SaaS), Platform as a Service (PaaS),* and *Infrastructure as a Service (IaaS)* [4].

This dissertation only focuses on the IaaS model of cloud computing, which provides on-demand online computing infrastructure resource at a reduced overall cost of ownership [5]. The NIST definition for IaaS is a capability provided to the consumer to provision processing, storage, networking, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the

underlying cloud infrastructure but has control over operating systems, storage, deployed applications, and possibly limited control of select networking components (*e.g.*, host firewalls) [4]. The two major technologies that enable IaaS cloud computing are virtualization and elastic computing which are described in depth later in this section. In IaaS, all of the facilities required for a datacenter application are available over the Internet which clients purchase as an outsourced service and are shared among a large number of consumers to allow for lower costs.

This section also provides a brief introduction to trusted computing technologies. While trusted computing technologies are not specifically related to cloud, the merging of cloud and trusted computing technologies is the focus of this dissertation.

1.3.1 Virtualization

Virtualization is the process of decoupling hardware from the operating system on a physical machine. A *Virtual Machine (VM)* is the virtualized representation of a physical machine that is run and maintained on a host by a software virtual machine monitor or *hypervisor* [6]. The hypervisor implements the virtualization on the physical machine and can be one of two types. *Type 1* hypervisors are sometimes referred to as native hypervisors as they run on “bare metal,” or directly on the host's hardware to control the hardware and to monitor guest operating-systems. *Type 2* hypervisors are hosted hypervisors, meaning they are software applications running within a conventional operating-system environment. This dissertation focuses entirely on Type 1 hypervisors.

Xen [7] is an example of a Type 1 hypervisor. Xen provides full virtualization to partition the host machine into multiple VMs. Xen can also use a technique known as

paravirtualization, where the operating system is aware of virtualization and works with the hypervisor, to improve the efficiency of operation.

VMs provide many advantages to increase efficiency and decrease cost to datacenters; however, VMs have significant security implications. A primary concern is ensuring the proper virtual environment is operating within a VM and whether the VM is configured properly. Another concern is virtual machines specific vulnerabilities that can be exploited to mount attacks specialized to subvert the built in defenses of the guest operating systems.

As all virtual machines have a standard interface and appear identical to the software running within it, we consider all attacks that can take advantage of that interface. Examples of such attacks include an adversary copying a virtual machine to maliciously run multiple copies of licensed software on multiple computers since their environments are identical, thus breaking the licensing agreement. Another problem of abstracting the real hardware interface from the guest VM is that it has no binding to physical hardware, thus disallowing it to determine its physical location and being able to decide whether the host environment is safe. By not knowing what else is operating on the platform, the guest VM cannot determine the trust level of the host platform. If the authenticity of the VM is not known and the trust of host environment cannot be determined, the guest VM cannot be trusted.

Trusting Virtual Machines

The security and integrity of the VMs in the cloud becomes a major concern as more organizations turn to cloud virtualization solutions [8]. We wish to place a level of trust in the VM that it will perform its intended task without compromise or lose the data it is processing.

To trust the virtual execution environment, a method is needed to measure the environment to ensure that the VM meets all of the requirements and report that measurement to a *Policy Decision Point (PDP)* to authorize a VM for operation. The entire process of measuring a configuration and reliably reporting the measurement is *attestation* [9].

Trusted hardware is one technology we can leverage to achieve our goals of building trusted virtual machines. Hardware protection mechanisms such as *Trusted Platform Modules (TPMs)*, secure virtual machine technology, and secure execution technology measure and securely execute software through hardware enforcement. Additionally, hardware protection is also provided to protect from rogue processors and to detect other malicious behaviors. We consider applications for this technology to build trust models for virtual execution environments.

1.3.2 Elastic Computing

Elastic computing provides scalable on-demand computing resources that are delivered as a service over Internet technologies [10]. Elastic compute clouds have automated management that handles the allocation and provisioning of VMs on cloud computing resources. This management layer provides up and down scalability of infrastructure resources. Amazon's *Elastic Compute Cloud (EC2)* [11] is one implementation of elastic computing. EC2 can have many different applications and servers, from many clients, simultaneously running within its cloud. Each client's view of the cloud will be different from another client's view.

Elastic computing biggest advantage is its payment and use model. In a traditional IT approach, organizations would have to purchase enough processing, storage, and network

hardware to support their peak computing requirements, as well as provide for space, power, and cooling for the hardware. In elastic computing, processing, storage and bandwidth incur charges of actual usage based on a normalized unit of measurement. The elastic computing payment model allows clients to reduce their IT cost by only paying for the resources when they are used.

1.3.3 Trusted Computing

Trusted computing technology provides a robust foundation on which to build a secure cloud. A *Trusted Computing Platform (TCP)* uses a trusted component to provide a foundation for trust for software processes [12]. The *Trusted Computing Group's (TCG's)* specification [13] states that TCPs have a two roots of trust, a *Root of Trust for Measurement (RTM)* and a *Root of Trust for Reporting (RTR)*. For x86 based trusted platforms the RTM is included in the BIOS boot block and the RTR is in the TPM [14]. The RTM provides secure measurement of the platform and the RTR allows a verified report of the measurement through the process of *attestation*.

Trusted Platform Module

The TPM [15] is the cryptographic component of the TCP serving as the RTR. The TPM also serves as the *Root of Trust for Storage (RTS)* providing a *Storage Root Key (SRK)* to protect encryption keys and other information. Other useful features of the TPM include a *Non-Volatile Random Access Memory (NVRAM)* for secure storage of keys and user data, and a true random number generator for key and nonce generation.

Measurement of the platform is accomplished by performing a *Secure Hash Algorithm (SHA-1)* hash on the code loaded on the platform and storing the result in the TPM's *Platform Configuration Registers (PCRs)*. The attestation process allows clients

to request a quote of the PCRs signed by the TPM and verify that the platform they are using meets their policy and configuration requirements. Clients can then determine whether they wish to use the services provided by the platform based on the attestation from the platform's TPM.

Each TPM has an *Endorsement Key (EK)* and EK certificate that provides a unique identity for the TPM and validates that the TPM is legitimate. An *Attestation Identity Key (AIK)*, signed by a privacy CA, can also be created that is used to verify that the TPM is legitimate; however, an AIK has no information that is uniquely identified to a single platform.

Virtual TPM

One problem associated with the TPM is that it works only for non-virtualized environments. Most TPM implementations closely follow the TPM specification for a one-to-one relationship between the OS and trusted platform; therefore, the TPM has a limitation in that it can be owned by only one entity at a time. If platform virtualization is used, the TPM must be virtualized to provide full TPM functionality for each guest entity. For this reason, specifications have been developed for a *Virtual TPM (VTPM)* [16] to provide the TPM functions for each virtual environment on the platform.

There are several implementations of the VTPM; each uses different methodologies to virtualize the TPM. For LoBot we chose to use the Generalized VTPM framework the TCG is proposing [17]. In the framework, each VTPM is implemented as a software based emulator that has its own emulated resources including EK, SRK, PCRs, and NVRAM allowing it to function identically to a hardware TPM.

Trusted Execution Technology

Trusted eXecution Technology (TXT) is a feature of Intel's microprocessors and chipsets commonly referred to as Intel vPro [14]. TXT provides a late launch capability for the secure creation and launch of virtual execution environments called *Measured Launch Environments (MLEs)*. MLEs can be launched anytime after a platform is booted. Hardware protections are provided by TXT against software based attacks on MLEs during launch and while the MLE is executing.

Late launch is initiated through function called SENTER, which provisions the MLE on the TCP. The SENTER process measures and verifies the MLE with a secure hash algorithm built into TXT and correctly configures the processor and chipset prior to passing control of the processor to the MLE. This capability ensures that the MLE launched is the proper MLE the user wishes to use and that the MLE has been provisioned on the platform as intended.

1.4 Contributions

This dissertation contributes the following novel concepts and specifications:

- An architecture for protecting privacy and confidentiality in a cloud datacenter that separates security relationships between the user and consumer of cloud services and combines trust from both for the virtual and physical environments.
- The concept of an agent or "bot" virtual appliance to protect provisioning and migrations of virtual machines in a cloud datacenter through fabric pre-measurement.

- Protocols to ensure secure provisioning and secure live migration of virtual machines in the cloud datacenter.
- A new Trusted Virtual Environment Module for rooting trust in a cloud computing environment.
- A Trusted Environment Key that combines components of trust from the virtual environment owner and the platform to form a compound virtual environment trust.
- A Virtual Trust Network and TVEM Factory for managing and provisioning TVEMs from a central control facility.

1.5 Dissertation Organization

This dissertation is organized into six chapters, including this introduction. The remainder of the dissertation is organized into four main chapters and a conclusion.

Chapter 2 discusses the trust and security assumptions used throughout this dissertation. The chapter then provides a catalog of known vulnerabilities and the threats of IaaS cloud computing.

Chapter 3 introduces the Private Virtual Infrastructure. PVI is our solution for managing trust and security that allows organizations with data confidentiality concerns to leverage the economies of scale of cloud computing technologies. PVI is based on the tenets of cloud security we propose. By sharing the responsibility for security between the service provider and the customer, PVI reduces the risk of using cloud computing services.

Chapter 4 presents the Locator Bot and *LoBot Secure Provisioning (LSP)* and *LoBot Secure Migration (LSM)* protocols in detail. LoBots are virtual appliances, or agents, for secure provisioning of virtual machines in IaaS clouds. LoBot pre-measures a host platform to determine whether the host is trustworthy and provides sufficient security controls for the intended application and information.

Chapter 5 describes the Trusted Virtual Environment Module, a new mechanism for rooting trust in a cloud computing environment. The TVEM helps solve the core security challenge of cloud computing by enabling parties to establish trust relationships in a cloud computing environment where an information owner creates and runs a virtual environment on a platform owned by a separate service provider.

Chapter 6 concludes the dissertation with a summary of the contributions, a brief discussion of areas of future work, and final thoughts.

Chapter 2

Assumptions, Vulnerabilities, and Threats

Cloud computing introduces new security threats and vulnerabilities that are not present in traditional IT environments. Current IaaS technologies lack adequate security mechanisms to handle these new threats and risks, potentially exposing information stored in the cloud to the service providers, attackers with Internet access, and all the other users of the cloud. This chapter describes the assumptions of trust used throughout this dissertation, the vulnerabilities associated with cloud computing, and provides the threat model.

2.1 Assumptions

This dissertation uses the following assumptions regarding trust: 1) we implicitly trust all TPMs, PVI and TVEM factories (see Sections 3.3.2 and 5.5.1 respectively for details on PVI and TVEM factories); 2) we trust standard cryptographic primitives and protocols; 3) we trust that users have an authenticated channel for receiving public keys need to authenticate TPMs; and 4) we trust that the cloud service provider runs certified hypervisor software on trustworthy physical machines. This section describes these assumptions in detail.

2.1.1 TPM and Factory Trust

For the information owner to have full trust in the PVI and individual components of the system, he must implicitly trust two components: TPMs and factories. Without trust in these components, he cannot have assurance in the security of his information in the cloud. It is reasonable to expect the information owner can trust these components as they are trustworthy devices.

A TPM is an explicitly trusted component of a computer system. A TPM is implemented as an integrated circuit that is physically attached to a platform's motherboard. The TPM has well-defined commands that allow software running on the platform to control it. Because the TPM is implemented in hardware and presents a carefully designed interface, it is resistant to software attacks [18]. An *Endorsement Key (EK)* certificate allows anyone to validate that a transaction signed by a TPM is genuine [19].

A factory is a trusted component of the architecture presented in this dissertation. Factories, whether they are PVI or TVEM factories, are the only components of the architecture that the information owner has complete control over. The factories generate and manage critical components of the architecture as well as the keys to secure the components. The information owner has assurance that the factory will perform as intended because he has complete control of the factories configuration and physical security.

2.1.2 Cryptographic Primitive Trust

We assume that standard cryptographic protocols and algorithms are in place and not compromised. Protocols such as SSL and TLS are required to provide secure links and

transmission of data between the service provider and client. Both symmetric and asymmetric encryption primitives are required to protect data confidentiality and must be able to decrypt ciphertext when required. We also assume cryptographic hash algorithms such as SHA-1 provide sufficient resistance to collisions that weaken the cryptographic strength of the hash.

2.1.3 Authenticated Public Key Channel

Our solution requires that TPM public key certificates are published in a location that is accessible by users of the cloud. The public key certificates allow users to validate the genuineness of any TPM with which they transact. The authenticated public key certificate channel prevents TPM spoofing by providers ensuring that all transactions with second party TPMs are confidential.

2.1.4 Service Provider Trust

A cloud service provider is trusted to provide a certified hypervisor and services that enable the client to verify the configuration of the provider's platform. The service provider should publish the configuration of their platforms and provide attestation signatures that clients can use to verify the configuration of the host platforms. We assume that the service provider does not lie about their configuration or there are other mechanisms to verify the configuration of the host platforms.

2.1.5 Other Assumptions

The control of the host machines at the service provider facility is done through a localized management network that is not accessible from the public Internet. Most servers have two network interfaces for this purpose. One network interface is used

exclusively for a management interface and services interface; the other interface is used for external access to the system from the Internet. These two interfaces are physically separated and the software on the host system isolates them. The means that an external attacker cannot gain access to the management interface from outside the physical premises; therefore, an external attacker cannot gain root privileges from an external network.

The host provider's datacenter has physical access control mechanisms in place preventing unauthorized personnel access to the servers and networking equipment. This is necessary to prevent attacks against the hardware and physical components of the system.

2.2 Known Vulnerabilities

The largest vulnerability in cloud computing is that data are processed on a machine that is owned by an entity different from the information owner. In the case of IaaS, the virtual machine is owned by the information owner and the host platform is owned by the service provider. Since the information is stored and processed on the service provider's machine, the information owner does not have full control of the data.

Utility cloud computing is, by definition, multi-tenancy with other users also using the same hardware resources, which introduces the risk of exposing sensitive information to unauthorized users. Information owners do not control the hardware resources used to operate on the information and must rely on the virtualization to provide the security needed to protect their information. The shared components that make up the underlying IaaS fabric (*e.g.*, CPU, caches, storage, etc.) were typically not designed to offer strong isolation for multi-tenancy. A hypervisor addresses this gap by mediating access between

guest operating systems and the physical computing resources; however, hypervisors have exhibited flaws that have enabled attackers to gain inappropriate levels of control of the platform [20].

Other vulnerabilities in the cloud computing environment are poor random number generation from low entropy of cloned machines [21]. Low entropy reduces the strength of the cryptographic operations performed in the cloud environment, increasing the risk of exposure by cryptanalysis.

2.3 Threat Model

The threat profile increases considerably for cloud computing compared to traditional IT environments. Attackers have much more access to attack exposed cloud computing services due to the open nature of cloud computing compared to the closed environment of traditional IT.

Potential adversaries in the cloud computing environment that we defend against are other tenants in the cloud and outside attackers. A cloud tenant can potentially eavesdrop on other tenants in the cloud or escalate his privilege by breaking virtualization containment and attacking other tenants. The outside attacker is generally the same as the outside attacker for a regular IT scenario with the exception that the attacker has a higher incentive to attack clouds since the opportunity for accessing valuable data is higher due to the concentration of users.

For this threat model, consider a platform running virtual machine machines under the Xen hypervisor. Figure 2.1 show the virtual execution environment (*i.e.* hypervisor and host OS) and the communication channel to a policy decision point that are in the

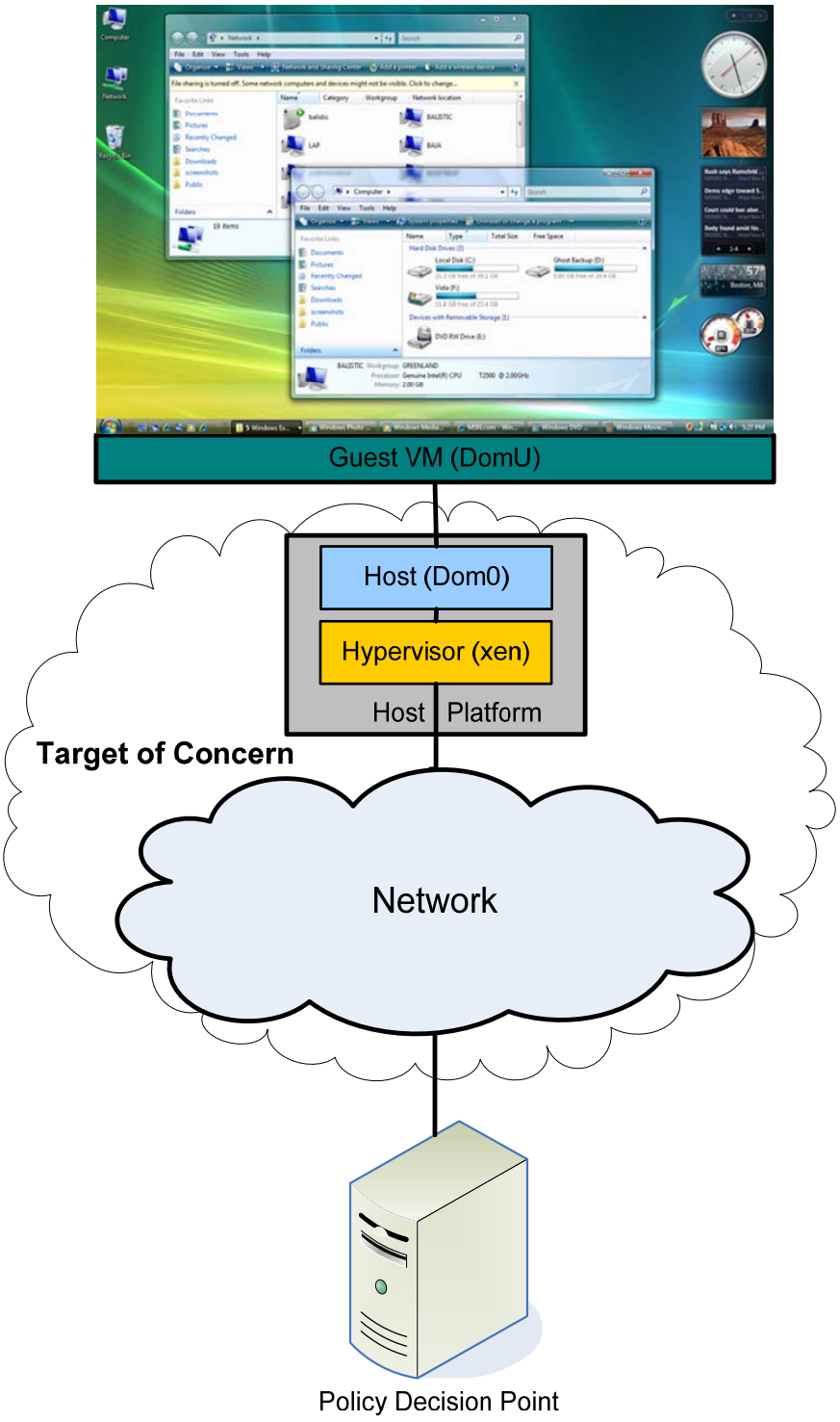

Figure 2.1 – The target of concern for threats to the guest virtual machine is the execution environment and its communication channel to a policy decision point.

target of concern for threats to the guest VM. We consider threats for the IaaS model in this section. We categorize the threats into two groups, those that are circumvented by our solutions and those that are not.

2.3.1 Circumvented Cloud Computing Threats

This section describes attacks which are prevented, handled, or detected by our solution.

Outside Attackers

An outside attacker is an adversary who does not have physical access to the hardware and attacks the system through the Internet. An outside attacker could place a malicious VM in cloud. This malicious VM, or *malvm*, in the cloud can purport to be a valid host machine of the service provider and receive information environments from unsuspecting users. This malvm could also snoop on the traffic within the datacenter. Once an attacker is able to implant a malvm in the cloud, the entire datacenter could potentially be compromised. An example attack is a malvm spoofing itself as a valid host machine on the service provider's network. When a new *Virtual Server (VS)* is requested in the cloud, the malvm receives the new VS from the customer, copies all confidential data, and transmits it back to the attacker. These actions are performed undetected without alerting the user that they are operating on a compromised machine. As far as the users know, their data assets are on a valid host server as it appears they sent the information to the service provider directly.

A similar attack we wish to defend against is the *malicious host* (*malhost*). In this scenario, an attacker gains control of a node by compromising the hypervisor or host OS. Once malicious control of the host is established, the malhost can accept new VS

requests. The malhost is legitimate and on the provider's network; therefore, it is implicitly trusted and treated as a non-malicious host. Since the malicious hypervisor and host have higher privilege than the guest machine, the malicious host has access to all operations and memory of the guest and ability to compromise any sensitive information to which the guest has access.

The LoBot Secure Provision and LoBot Secure Migration protocols ensure that PVI and TVEMs are provisioned only on authorized platforms of the service provider. The protocols prevent information from ever being sent to a malhost or malvm.

The Hypervisor

The hypervisor poses a significant threat to the virtual environment. The hypervisor has the highest privilege in the system; therefore, it has the ability to manipulate the guest operating system in multiple manners. The threat of a malicious or compromised hypervisor is that it can intentionally disrupt the guest's security mechanisms. A malicious hypervisor could also perform a man-in-the-middle attack on any communication between the guest VM and any outside entity, furthering the threat to the guest virtual environment.

This type of attack is demonstrated with *Time Traveling Virtual Machines (TTVMs)*. TTVMs are VMs whose state can be changed through the intervention of the hypervisor [22]. A TTVM has two capabilities. First, it can reconstruct the complete state of the virtual machine at any point. Second, it can start from any point in a run and from that point replay an instruction stream. This capability allows attackers to manipulate VMs to states which occurred in the past or force the VM into a state that it would not normally enter (*e.g.*, error states which force the VM into debug mode or disable security features). TTVMs were originally designed to help debug operating systems and complex

applications by replaying state and instruction information; however, a TTVM can be used for malicious purposes such as falsifying time stamps for transactions, and for replay and rollback attacks.

Our solution provides a mechanism to verify whether an approved hypervisor is in use. Assuring an approved hypervisor provides the information owner confidence that attacks against the virtual environment will not be performed by the hypervisor.

The Host Operating System

The host operating system is the controlling system on the platform, referred to as dom0 since it has higher privilege than other VMs, which are in unprivileged domains referred to as domU. A domU guest relies on the dom0 host for many services including VM control and management, device drivers, and in the case of *Hardware Virtual Machines (HVMs)* BIOS and system emulation. This forces the domU guest to place a large amount of trust in the host environment. If an attacker gains access to dom0, she could manipulate the control, drivers and emulators to force the domU to an untrusted state.

Device drivers are another threat associated with the host operating system. Since drivers typically operate at the highest privilege of the platform [9], equivalent to the host OS they can be compromised to obtain “root” access to the system through vulnerabilities such as buffer overflows and poorly designed interfaces. Drivers could also contain malicious code embedded within them. These vulnerabilities could allow an external entity to use a device driver to gain access to a domU.

Again, our solution provides a mechanism to verify whether an approved host OS is in use. Assuring an approved OS provides the information owner confidence that attacks against the virtual environment will not be performed by the OS.

Hypervisor and Rootkit Malware

A new class of attacks has evolved around building malicious hypervisors and operating system rootkits that subvert the built in security measures of many operating systems [23]. These malwares utilize a hypervisor or rootkit that allows them operate at a privilege level above that of the guest OS or maintain root access to the system. The malware at the higher privilege level can then intercept system calls from a victim OS and modify the calls in a manner that thwarts the security mechanisms of the victim VM. The malware can gain access to protected memory, intercept passwords or cryptographic keys, and perform a multitude of other malicious acts that the guest OS has no chance of defending against as it would be able to do on a physical machine.

An example of this type of attack is SubVirt [24] created by a University of Michigan research team, which is essentially a *Virtual-Machine Based Rootkit (VMBR).* SubVirt has been used to implement a phishing web server, a keystroke logger, a service that scans the target file systems system looking for sensitive files, and a defensive countermeasure that defeats a virtual-machine detector.

The Blue Pill attack is another example of this type of attack. The Blue Pill is an attack to virtualize a Windows operating system by installing a malicious hypervisor underneath the kernel that is theoretically undetectable even though the algorithm and code are publicly available. It avoids detection by trapping all attempts by the victim OS to determine it is in a virtualized environment and reporting fake information back to OS to make it believe it is operating normally. The Blue Pill attack can be performed on already virtualized machine, thus nesting itself between the real hypervisor and the victim machine [25].

By verifying the validity of the hypervisor and host OS, we can determine if any malware was present in hypervisor and OS at boot time; however, an infection after boot time may not be detected. For this reason, we use encryption of data to reduce the risk of data exposure.

Virtual Machine Migration and Duplication

VMs can be duplicated and migrated to help balance load. This process is performed by copying or moving a VM from one physical platform to another and can either be performed live or offline. The live migration poses a significant challenge to trusting a VM because the migration is done while a VM is operating.

The following scenario shows the critical flaw. A virtual machines launches, is attested and verified by the PDP, then the VM migrates to another platform. The virtual execution environment is changed, but since the VM has already attested to the PDP and keeps it authentications, it continues to operate as though it were on the previous platform. Migration is behavior we wish to know about and properly control, as changing the platform changes the security properties of the VM.

A man-in-the-middle attack against VM provisioning or live migration is another attack we defend against. In this scenario, the attacker wishes to gain access to encrypted information on the *Storage Area Network (SAN)* or database. The attacker listens for provisioning or migration requests by the target and intercepts the image and state information from the source machine. This interception can be accomplished through spoofing, ARP poisoning [26], DNS poisoning [27], or any other attacks that redirect traffic from the destination machine to the attacker. Since the state information is a copy of the VM's memory and processor state, the attacker can manipulate the image and machine state by implanting malicious software and then placing the compromised image

on destination server. The attacker now has access to the server's secure data stored on the SAN through the malicious software implant.

We ensure that these attacks are not possible by verifying the integrity of the image before and after provisioning. Providing keys for the LoBot and TVEM that are specific to a host platform reduces the risk of cloning and duplication of VMs and protects the information from being used on an unauthorized machine.

Data Loss and Leakage

The threat of data compromise is much greater in the cloud [28]. There are many ways data may be compromised in the cloud including deletion or alteration of records without a backup, loss of or changing an encryption key that results in the effective destruction of any data stored with the key, and unauthorized access by insiders or other cloud users. Again, encryption of sensitive data reduces the exposure of data loss and leakage.

Another area of vulnerability of the VM is while the VM is at rest (*i.e.* inactive). A VM that uses a virtual file system – as opposed to a physical one – is susceptible to data modification while the system is at rest. It is possible for an attacker to modify the configuration of the VM by manipulating the virtual file system and alter the behavior, properties, and data stored on the VM. If an attacker gains access to a virtual file system, the data are vulnerable to theft as the attacker has full access to all data contained in the file system. Our solution detects data modifications through the attestation process during secure provisioning. Additionally, encryption of data in the VM image with keys locked to specific platforms reduces the exposure of data at rest attacks and data loss.

2.3.2 Non-circumvented Threats

This section describes additional attacks against cloud computing for which we do not provide solutions. These attacks are provided as a reference and to help frame the capabilities of our solution.

Malicious Insiders

Malicious insiders can have a huge impact on an organization using cloud computing technologies. A malicious insider is anyone in the service provider's organization with approved access or privilege to the cloud information systems that is motivated to compromise information confidentiality, integrity, and/or availability [29]. The insider threat is compounded when combined with lack of transparency into service provider processes and procedures. There is often little or no visibility into the hiring practices for cloud provider employees. For example, a provider may not reveal how it monitors employees or grants access to physical and virtual assets. Depending on the access granted, an insider could collect confidential data or even gain control of the cloud services with little or no risk of detection.

There are several attacks against the VMs that can be performed by malicious actors inside the *Cloud Virtual Fabric (CVF*). A malicious administrator can surreptitiously attack a VM in the cloud using her higher privileged access to inspect memory, monitor VM communications, and perform suspend and reboot attacks. This attack is very difficult to defend against as the insider needs to have these privileges to administer and maintain the host systems and it is difficult to determine legitimate access versus malicious access. While our solution does not prevent malicious insiders from accessing sensitive information, encryption of sensitive data reduces the exposure. Therefore, we

must be vigilant in determining the security settings and configuration of the host environment prior to provisioning.

Network-Based Attacks

A VM is vulnerable to network-based attacks during attestation and migration (especially live migration [30]). Network attacks that can be performed include eavesdropping, man-in-the-middle, data modification, spoofing, etc. It is imperative that the network communication be thoroughly understood and examined to understand all the possible attacks against it. While our solution does not provide any direct protection from network attacks, our protocols do use cryptographic protocols which limit the exposure to network attacks.

Hardware Attacks

An attacker that has access to a physical machine can mount a hardware attacks against the VM. The attacker can install monitoring hardware (via any number of buses including USB, Firewire, PCI, etc.), replace existing hardware, or even copy a VM configuration via hijacked memory via a cold boot attack [17]. Since our solution is entirely software based, we cannot defend against hardware attacks. We assume that even though the threat of hardware attacks is present, the likelihood of such attacks is rare and thwarted by non-technical means such as physical security, personnel background checks, and policy. We will not examine technical defenses to those attacks beyond the capabilities built into the existing hardware protection mechanisms in the Intel vPro architecture.

Chapter 3

Private Virtual Infrastructure

Cloud computing requires a new trust paradigm that shares responsibility for security between providers and consumers of cloud computing services. *Private Virtual Infrastructure (PVI)* first introduced in [31] is our approach to managing trust and security in cloud computing environments. We introduce *Locator Bot (LoBot)* to implement the PVI on cloud resources with a level of assurance that is required to meet data confidentiality and privacy concerns of sensitive information. LoBot is a *Virtual Machine (VM)* appliance that acts as an agent to locate trustworthy platforms, provides a root of trust for PVI *virtual servers (VSs)* via a *Virtual Trusted Platform Module (VTPM)*, and provisions PVI within the cloud.

Many organizations such as financial institutions, health care providers, and government agencies are legally required to protect their data from compromise due to its sensitivity. Consider a healthcare provider that wants to move a database of patient medical records to a utility cloud to allow patients and insurance companies easier access to the information. Medical records are considered private and must be protected under the Health Insurance Portability and Accountability Act (HIPAA) of 1996; therefore, medical records in the cloud must be protected from unauthorized disclosure. A utility cloud potentially exposes the information to the operators of the cloud, other users of the cloud, and anyone who has access to the Internet. Typically, organizations with privacy

requirements manage and maintain their own datacenters with stringent physical and logical protection mechanisms ensuring that their data remains protected. These organizations simply cannot use cloud computing in a generic manner due to the inherent risk of data compromise from systems they do not control. PVI provides these organizations a means to manage security in a utility cloud by facilitating sharing of security management roles and responsibilities between service provider and customer. PVI for cloud computing enables organizations to maintain control of their information in the cloud and realize benefits of cloud computing.

Private Virtual Infrastructure is an architecture for utility cloud computing that meets the information owner's requirements for data protection and privacy. The PVI architecture is a cluster of *Trusted Computing Fabric Platforms (TCFPs)* that are owned, operated and configured by cloud service providers. The TCFPs host VSs for clients, which are two tightly coupled virtual machines, an *Application Server (AS)*, and a LoBot that provides security services for the AS. Each PVI instance has a single trusted central control and policy decision point called the *PVI Factory (PVIF)*, which provides the root of trust for PVI. The PVIF maintains and manages keys for PVI and serves as the certificate authority for the LoBot's *Endorsement Keys (EKs)*.

Cloud security requires total situational awareness of the threats to the network, infrastructure and information. One of the biggest advantages to the cloud's utility is abstraction [10], but is also its biggest security weakness. Abstraction allows the cloud to be pervasive and removes knowledge of the underlying fabric of processors, storage, and networking; however, without knowledge of the underlying fabric, information owners' understanding how to secure their applications and information becomes very complex.

Many of the security principles used today to secure datacenters and networks rely on the information owners' ability to manage the underlying fabric of servers, routers, firewalls, and intrusion detection devices to understand when attacks are occurring and to responds to the threats by shutting down access to resources and isolating pieces of the fabric that are being attacked.

Outline: This chapter is organized as follows. Section 3.1 discusses related work. Section 3.2 presents our computing model introduces an illustrative example. Section 3.3 describes the PVI architecture in detail. Section 3.4 discusses the benefits, security and costs of PVI.

3.1 Related Work

Virtualization is a fundamental technology for implementing IaaS cloud services and PVI. Virtualization is the capability to share a single physical computer among many simultaneous guest VMs via a virtual machine monitor, or *hypervisor*. Xen [7] is an open source hypervisor that provides a the ability for VMs to run full operating systems as well as smaller "helper VMs" called stubs that provide services, such as hardware emulation and virtual TPMs.

There are many security issues that must be considered when using virtualization technology including separation of private data between VMs, virtualization containment attacks, and hypervisor attacks against the VM. IBM developed a secure hypervisor called sHype [32] to help solve some of these problems. SHype provides isolation between the VMs and controls resource sharing but does nothing to verify the integrity of the VMs launched on the host. Terra [33] is a trusted virtual machine monitor that can create isolated tamper resistant VMs on a host with the ability to verify the VMs;

however, it does not provide the ability to measure the environment before provisioning of the virtual machine and does not guarantee a secure launch of the virtual environment. There are also disaggregation techniques developed to improve Xen security [34] that reduce the trusted computing base of the hypervisor and VMs, which is useful for our needs; but we still need to validate that the environment is properly configured prior to provisioning our VMs.

A proposal for trusted cloud computing made by Santos [35] focuses on building trusted cloud computing platforms using the Trusted Computing Group's specifications. PVI does not focus on the host platform, rather the architecture for the virtual datacenter operating on the trusted cloud computing platforms. PVI also introduces our cloud trust model that accounts for trust beyond the trusted platform.

3.2 Our Cloud Computing Model

Consider a landlord of an apartment building and the tenants of each individual unit. Tenants expect the landlord to provide basic services such as facility maintenance, security for the complex, while the landlord expects the tenants maintain their personal property and personal security (keep their doors locked). As long as the tenants uphold all the requirements of their lease agreement, they expect the landlord will respect the privacy and not spy on them or interfere with their personal property. A trust relationship is established between the landlord and tenant that each will uphold their end of the agreement.

In the cloud computing realm, the service provider can be thought of as the landlord and the client can be thought of as the tenant. *Infrastructure as a Service (IaaS)* [36] is a model of cloud computing where all of the facilities required for datacenter applications

are available over the Internet, which clients purchase as an outsourced service. An IaaS datacenter is similar to the apartment building above in that it provides infrastructure hosting for multiple customers. A datacenter could have the capability to host thousands of VSs for clients who rent or lease the servers much as apartment tenants. By using cloud IaaS as a datacenter replacement, companies can manage traffic and loading agility. The clients can run web servers, applications, or databases on their VSs, dynamically increasing or decreasing their server capacity as needed. As long as customers abide by the service agreement, the provider should not interfere with the operation of the clients' VSs, monitor the clients' communications, or view or modify the clients' data. Since security is not an integral part of IaaS, and since the management and ownership of the hosting platforms is removed from the information owner, ensuring the security and integrity of information in IaaS is a major unresolved issue.

Let us consider a hypothetical example of a healthcare provider called CloudHealth. CloudHealth maintains a database of patients' records, insurance companies, and billing. CloudHealth needs to reduce overhead. Reasoning that being an IT tenant would be cheaper than being an owner, they choose Fabricorp as their service provider. Fabricorp is a reputable Internet company that has been in business since the early days of the Internet and recently moved into cloud datacenter services. Fabricorp built several massive datacenters, each with several thousand servers; thousands of individual clients may be using its cloud datacenter services at any given time. CloudHealth will move all of its corporate servers and data assets into Fabricorp's cloud, maintaining only thin clients, laptops or nettops, and mobile computing devices on site for doctors and administrators to access the cloud datacenter. This scenario allows CloudHealth to

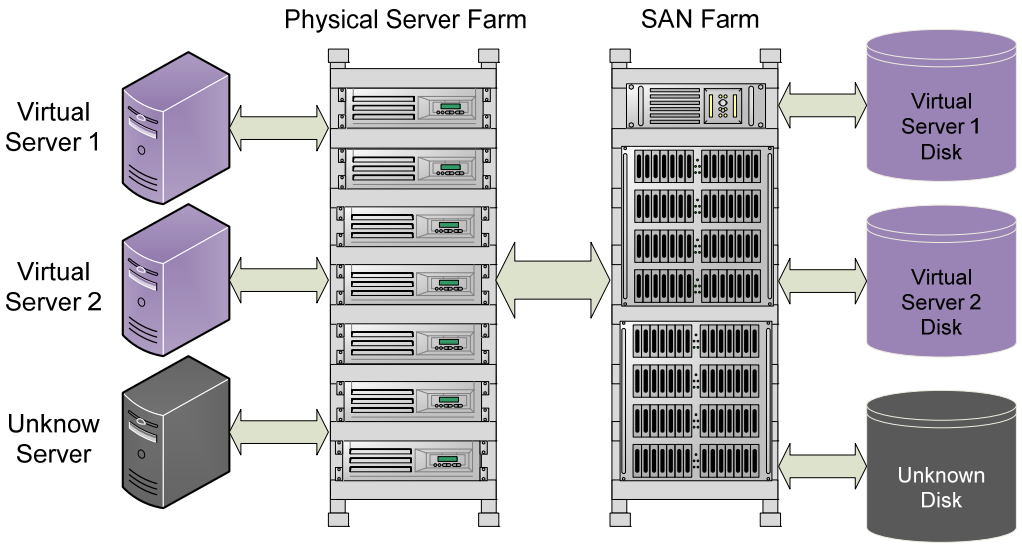

Figure 3.1 – Cloud Virtual Fabric topology consists of a server farm, SAN farm, virtual severs and virtual disk.

leverage large amounts of computing power at Fabricorp to run analytics on healthcare data and perform billing operations when needed without having to maintain those computing resources locally, thus considerably reducing IT costs. The cloud datacenter contains all the personal and private information about CloudHealth's patients, relationships with other companies, and financial information. To comply with HIPAA regulations, CloudHealth needs to have assurances that its data are protected from Fabricorp's other cloud users, Fabricorp's personnel at the cloud datacenter, and other Internet users who have access to Fabricorp's services. We will show how PVI is used to help CloudHealth leverage the benefits of cloud computing.

3.2.1 Cloud Virtual Fabric

Fabricorp uses an IaaS topology called the *Cloud Virtual Fabric (CVF)* for their cloud datacenter. CVF is a network of computing platforms that provides elastic cloud

computing services. No client should be able to see what any other client is doing within the cloud.

Figure 3.1 shows the topology of CVF which accommodates many VSs (known domains) and unknown domains (other users) on a fabric of computing platforms. These known and unknown domains need to be isolated from each other to provide privacy and security within the cloud. SANs provide virtual storage for the computing fabric and VSs. Virtual storage is accessed by VSs via a protocol like iSCSI, Fibre Channel over IP, or a similar protocol that physically separates storage from processing servers, providing the illusion of local storage for the VSs. This fabric configuration allows servers and storage to be spread across many physical facilities, while supporting dynamic provisioning and live migration capabilities for the VSs.

3.2.2 Cloud Trust Model

This cloud computing datacenter environment is laden with trust issues. The client must not only trust the datacenter operator, but each of the individual clients that are collocated in the facility and each third party service provider of the operator. Clients needs assurance that their sensitive information is protected from compromise and loss and that their application is available when demanded. If only one of the thousands of client VSs in the datacenter were malicious, it would have to capability to compromise and threaten all the remaining clients in the datacenter destroying the trust and integrity of all parties involved.

We define a new model for trust in the cloud that accounts for various sources of trust and multiple trust relationships. Figure 3.2 show trust relationships that are present in our CloudHealth example. The cloud by definition is owned and operated by a second party,

Fabricorp. A trust relationship is established that Fabricorp will provide trustworthy services to CloudHealth when they enter into a business relationship. This relationship is an *inferred trust* in that it is a social trust, not a logical trust. Other inferred trust relationships in the example are that patients trust CloudHealth with their medical records and Fabricorp trusts vendors to supply support services. CloudHealth's patients have no relationship with Fabricorp and want to be assured that their medical records are secure, which is CloudHealth's responsibility. A Fabricorp vendor, for example a billing processor, does not have a trust relationship with CloudHealth nor its patients, yet it is critical that the bills for Fabricorp's services are processed and the billing vendor does not interfere with CloudHealth's operations. Our trust model solves the problem of maintaining CloudHealth's trust with its ASs in FrabriCorp's CVF, and ultimately, its patients. This trust relationship must be maintained in the presence of parties who have no relationship with CloudHealth or its patients.

As can be seen from Figure 3.2, the trust relationships are not linear. An application server has two sources of trust, trust in the CVF (provider), and trust in the PVI (client). The PVIF first needs to establish a logical trust in the TCFP before it initiates any provisioning of PVI. This is done by inheriting the root of trust in the TCFP and combining it with the PVI root of trust with a LoBot. LoBot's compound trust relationship is then used as a root of trust for the VS. The VS's root of trust is distinct in that LoBot has combined two unique roots of trust into a third compound root of trust with attributes inherited from both sources. This LoBot root of trust is unique to the TCFP it is running on such that it cannot be cloned and used on another platform without unbinding the trust relationship to the TCFP. Additionally, the VS cannot be used by

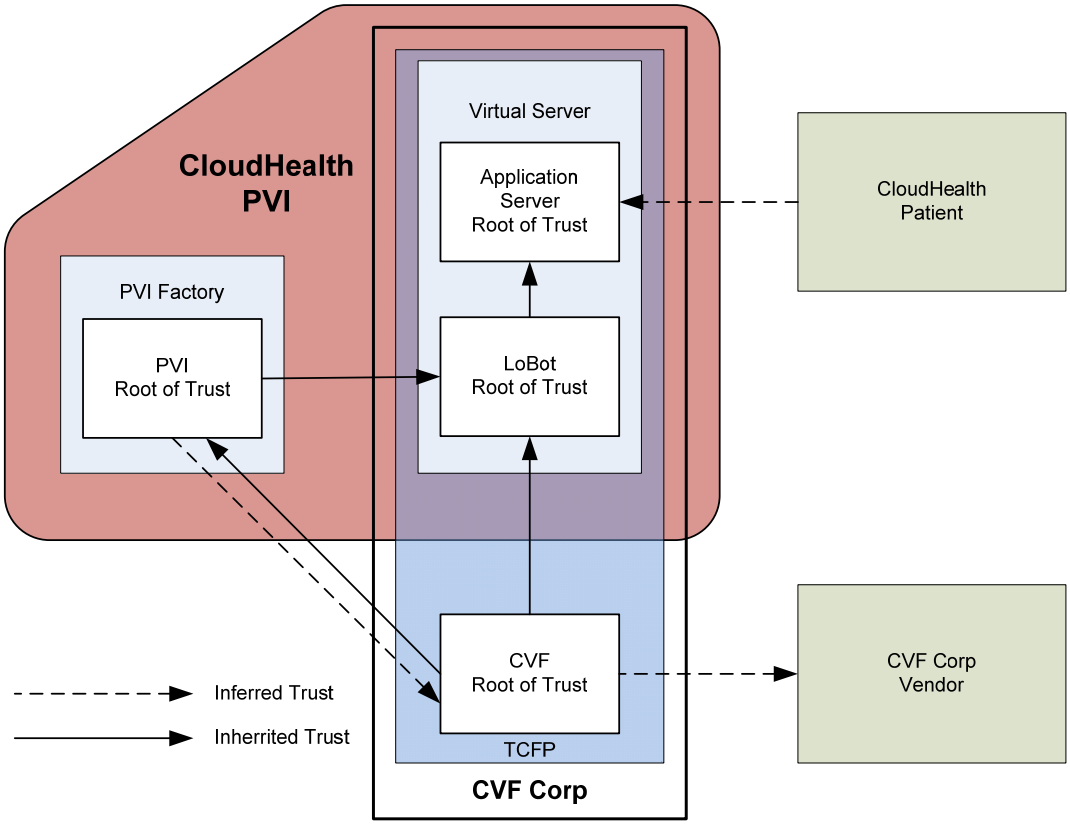

Figure 3.2 – Trust relationships in the cloud are complex with multiple relationships both inferred and inherited.

other entities in CVF, even on the same TCFP, as the trust is bound to PVI as well as the TCFP.

3.2.3 PVI Security Model

In a cloud, traditional security methodologies do not work as the service providers cannot allow information owners, or clients, to manipulate the security settings of the fabric. If this were allowed, it would be possible for one client to change security settings illicitly in their favor, or change security settings of other clients maliciously. This situation is unacceptable since the information owner cannot manage the security posture of their computing environment. Therefore, a security model is needed that allows

information owners to protect their data while not interfering with the privacy of others in the cloud.

We propose a model for security that is shared between operators and clients. Operators need to give clients visibility into the security posture of the fabric while maintaining control of the fabric. The clients need to have assurance that they can control the privacy and confidentiality of their information at all times and have assurances that if needed, they can remove, destroy, or lock down their data at any time. Adding security to any system inevitably leads to a compromise in some fashion. For PVI, the abstraction of the fabric is removed. It is impossible to have a completely obscure fabric for IaaS that provides the assurances of security properties required for the sensitive data contained in a PVI.

The PVI security model is a virtual datacenter over the existing cloud infrastructure. This virtual datacenter is under control of the information owner while the fabric is under control of the operator. Both parties must agree to share security information between themselves and possibly other parties in the cloud to achieve situational awareness of the security posture at all times.

The key benefit of the PVI security model is the ability to verify security settings of the underlying fabric. The service provider needs to supply security services which protect and monitor the fabric. Each service in the cloud needs to be able to report security properties present and the report must be verifiable. These properties must be cryptographically bound and signed such that anyone wishing to verify the properties, and has the proper authorizations, can do so. This ability means that clients need visibility into the security settings and configuration of the fabric. We have chosen to use trusted

computing techniques to provide the visibility into these settings and verifiably report the configuration of the fabric in PVI.

The service level agreement between the client and provider is critical to defining the roles and responsibilities of all parties involved in using and providing cloud services. The service level agreement should explicitly call out what security services the provider guarantees and what the client is responsible for providing. Clients should thoroughly examine and negotiate Service Level Agreements with their vendors to define and minimize their risk exposure before agreeing to use any cloud computing service.

Additional requirements for PVI are that communications to, from and within PVI should be done through a *Virtual Private Network (VPN)* and all links should be encrypted with IPsec or SSL tunnels. The encryption provides confidentiality on the network and prevents other users within the cloud from eavesdropping and modifying communications of PVI.

3.3 Private Virtual Infrastructure Cloud Security Architecture

Information owners need the ability to manage their cloud datacenters to respond to the threats and balance their computation and network loads. The PVI architecture provides this capability through two layers that separate the security and management responsibility between the service provider and the client. The CVF layer provides computation resources managed by the service provider, while the PVI layer provides a virtual datacenter managed by the client. The service provider assumes responsibility for providing the physical security and the logical security of the service platform required for the PVI layer. Clients are then able to configure their computing environment with the level of security required to meet their confidentiality and privacy requirements.

The primary requirement for the PVI architecture is protecting the AS where sensitive information is processing, which we call the *Application Domain (aDom)*. To do so, we need to ensure that the aDom is provisioned exactly as intended, provisioned where intended, and provisioned when intended. To achieve this requirement, each TCFP in the cloud needs to be able to report security properties and configuration information to the PVI. These properties must be cryptographically bound and signed such that any authorized person can verify the properties. We use trusted computing techniques to provide cryptographically verifiable reports of the security properties and configuration of the TCFP.

A TCFP is the basic computing platform of the CVF. On each TCFP, multiple cloud VSs can be provisioned, along with VMs from other clients in the cloud; therefore, strong isolation is required on the platform. A VS is the basic component of processing within PVI. The VS consists of two sub-components, the aDom VM running applications with sensitive data, and its companion LoBot virtual appliance. Provisioning and migration require a management layer, which is controlled via the PVIF. Trusted storage and networking are also required components of the PVI; however, trusted storage and network issues are out of the scope of this research.

3.3.1 Trusted Cloud Fabric Platform

A TCFP is one single physical machine in the CVF. Each TCFP implements Intel's vPro specification with TXT and a TPM version 1.2. The TPM implements a random number generator for nonces, encryption key generation, RSA 2048-bit encryption, and the SHA-1. Each TCFP may have multiple CPUs, but only has a single shared memory and a single TPM. A type 0 hypervisor is used to virtualize processing and most

resources are shared amongst all virtual environments; however, local configuration could provide unshared resources for certain applications or domains. The platform can support many unprivileged domains with strong hypervisor isolation and hardware support via TXT.

The TCFP must have a *Trusted Computing Base (TCB)* defined that can be measured via the TPM. We wish to have as small of a TCB as possible; however, there are certain components we wish to include in the TCB which make it rather large. At a minimum, the TCB should consist of the hardware components of the TCFP and the hypervisor. Additionally, we would like to include the kernel of the host OS in the TCB and possibly the management layer.

The TCFP node architecture shown in Figure 3.3 is developed around the Xen hypervisor [7]. The hypervisor runs at the highest privilege on the host platform; therefore, it must be a trusted and part of the TCB of the TCFP. The host operating system runs as a *privileged domain (dom0)*, which provides the management layer for PVI; therefore, it is desired that dom0 be a trusted component as well. The hypervisor and host OS can be verified through attestation of the host system by attestation from the host TPM's PCRs.

3.3.2 Management Layer

The management layer is responsible for controlling PVI provisioning and migration of VSs to trustworthy platforms within CVF. The management layer is a function of several components including *Xen Management (XM)* and the PVIF. XM is used to manage local tasks on the TCFP while the PVIF manages task across multiple platforms throughout the entire PVI. Secure provisioning and live migrations are initiated through

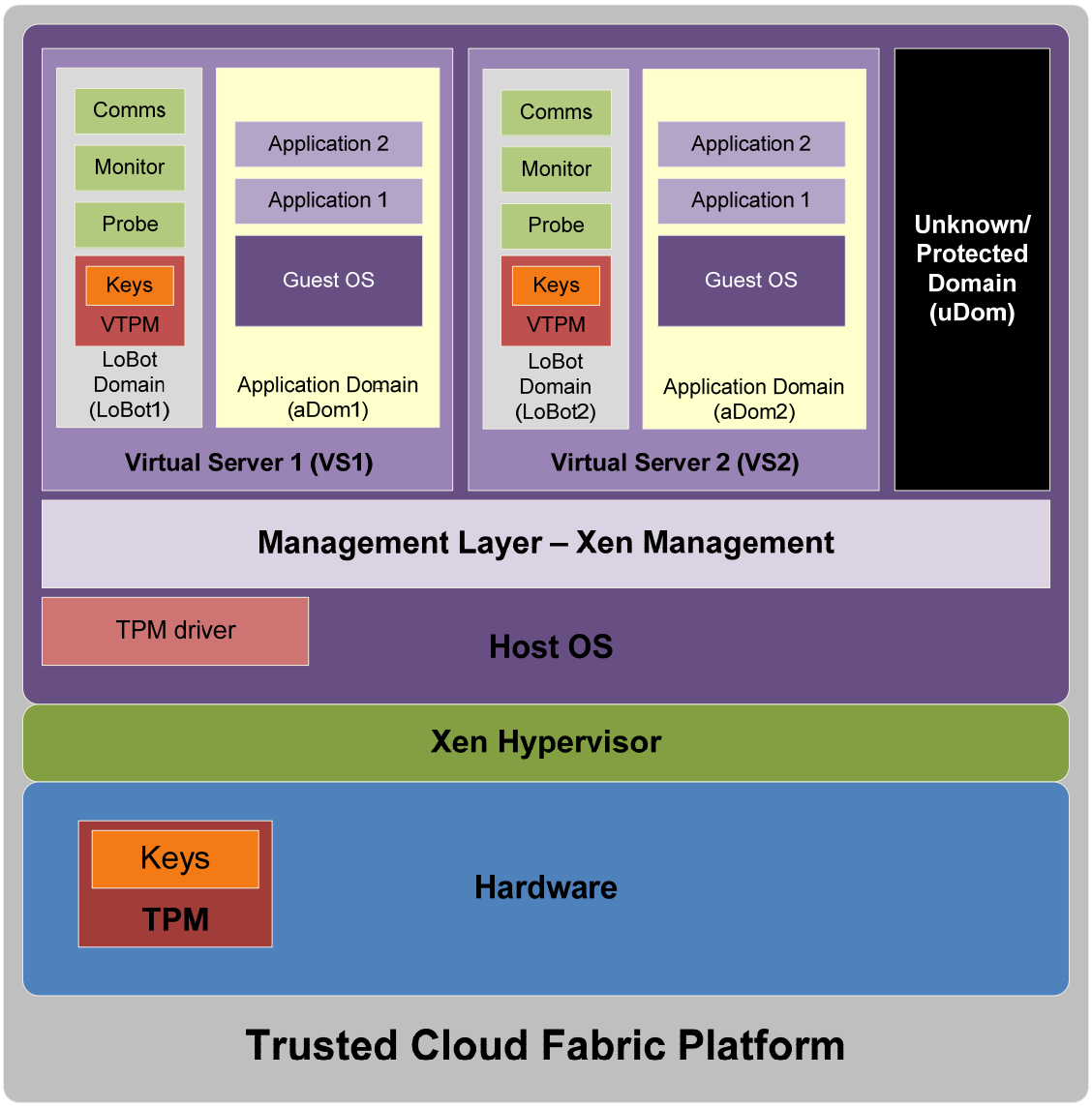

Figure 3.3 – Architecture of a Trusted Cloud Fabric Platform showing the Virtual Server Domain, Application Domain and LoBot Domain

the PVIF and performed through the XM instances on the TCFPs. Secure provisioning and live migration are monitored and controlled by the PVIF via the *LoBot Secure Provisioning (LSP)* and *LoBot Secure Migration (LSM)* protocols.

The management layer must have privileges to read and write information into the TCFP's TPM to obtain information about the configuration of the TCFP and bind the LoBot's VTPM to the TPM. This requirement makes the management layer a trusted

component of the architecture, which means that everything it is built upon must also be trusted.

PVI Factory

The PVI Factory serves as the controller, root authority, and policy decision point for the PVI. Figure 3.4 shows the overall PVI architecture with the PVIF as the management structure for PVI outside of the CVF. The PVIF is responsible for ensuring the integrity of the PVI and handling incidents in the event of a security breach. If any problems are detected, PVIF can shutdown the PVI, recall and inspect all images for tampering, and generate alarms and reports.

The PVIF is the most sensitive component of the PVI. The PVIF is the management controller and root of trust for PVI providing the following functions: virtual machine provisioning, VTPM key generation, EK certificate authority (VTPM Entity), and VTPM NVRAM storage. The PVIF is where all components of the PVI are provisioned and it is the root authority for provisioning, VTPM key generation, and certificate generation. Since VTPMs do not have valid EKs, an EK must be generated for each VTPM used within the PVI to trust the operations of the VTPM. The factory also maintains master images for ASs, LoBots, and handles data transfers within the PVI through the VPN configuration and management.

The PVIF is an application that runs on a dedicated trusted server, which could have built-in hardware to accelerate cryptographic operations and to provide true randomization, but a software-only implementation would suffice for most applications. Because the PVIF is the root authority, if it is compromised, then all existing PVI components are at risk of compromise and future provisioned components cannot be trusted. The PVIF should be a standalone component under full control of the information

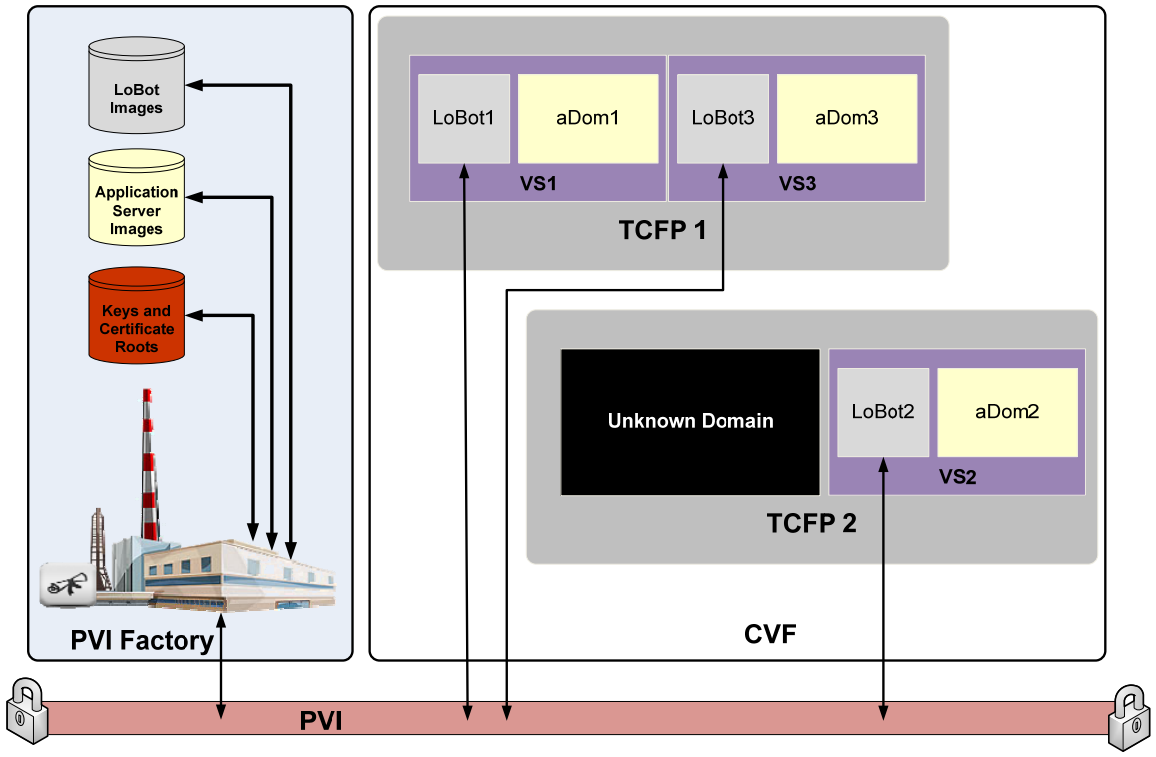

Figure 3.4 – PVI Architecture showing the PVI Factory and two TCFPs hosting three VSs and one unknown domain. The PVI Factory manages images and keys for the PVI and is the certificate authority.

owner and isolated to the greatest extent possible from other systems. If the PVIF is collocated within the service provider's datacenter facility, it should be physically isolated in an access controlled area.

3.3.3 Virtual Server Domain

Each *Virtual Server Domain (vsDom)* is a combination of an unprivileged aDom and a tightly coupled LoBot. The LoBot is implemented as an unprivileged stub driver domain to provide isolation and protection for the VTPM. VsDoms are provisioned, managed, and controlled by the client, not the operator, to ensure that the client maintains control of the information contained in the application server.

Application Domain

The aDom is the VM that the client runs application on in the cloud. This aDom is where all critical processing is performed and confidential data are processed; therefore, the domain needs to be protected at the highest level possible. ADom is the domain we wish to protect with PVI and LoBot. PVI's primary goal is protecting the aDom. Compromise of aDom would allow an attacker to gain access to or corrupt the sensitive data begin processed within the AS.

LoBot Domain

LoBot is a virtual appliance that is used to secure ASs in PVI. A LoBot is a self-contained VM image that has no external storage requirements, thus allowing it to move and replicate within a cloud environment efficiently. The LoBot domain's primary responsibility is to serve as the protector of the aDom. LoBot protects aDom with the following primary functions: 1) emulate a physical TPM for the aDom and provide a root of trust for aDom, 2) act as a probe to measure target platforms, and 3) provide secure provisioning and live migration services. The probing function ensures the destination platform provides sufficient security properties to protect information of the aDom.

The LoBot architecture is a driver stub domain bound to a single aDom creating a vsDom. The LoBot provides TPM services via a VTPM to the aDom. The VTPM is the core of the LoBot architecture providing the same level of trust to the virtual platform as the TPM does on the physical platform. The LoBot's VTPM is bound to the physical TPM of the TCFP tightly coupling the LoBot and aDom to the host. Note PVI is owner of VTPM and CVF is owner of TPM. Placing the VTPM in the LoBot has several advantages over operating the VTPM in the host domain including isolation of the VTPM process and reducing the burden of VTPM migration. The VTPM instance is isolated in

LoBot via Intel's *Virtualization Technology (VT-x)* and TXT such that no direct interactions with other domains are allowed. The only way to access the VTPM is through a rigidly controlled device interface between the LoBot and aDom ensuring that data stored by the VTPM is protected from other processes on the same platform. All information that a LoBot and VTPM are required to save, such as keys and non-volatile storage, is encrypted in a blob and sent to the PVIF for storage. The PVIF manages storing data for each LoBot eliminating the need for local storage.

Upon launch of the LoBot, the VTPM binds itself to the target's TPM then the probe application reads the platform configuration from the target TPM's PCRs and obtains identifying information about the platform. Identity information is provided in the form of cryptographic certificates. This information is then combined with the VTPM's PCR which is cryptographically sealed in a blob that is transferred to the PVIF. The PVIF decrypts the blob and examines the information received to make a trust decision. If the PVIF determines the target environment is trustworthy, it configures the aDom image and securely transfers it to the target environment in a blob encrypted such that only the target platform may execute source environment.

At the target environment, the LoBot probe application receives and unseals the aDom image. If the image was tampered with during transfer, it will be detected during the decryption phase. To make sure everything is safe, the probe measures the source environment one more time to validate its integrity to ensure the launch in the target environment was successful.

3.3.4 Trusted Storage and Networking

Each aDom in PVI needs a virtual storage device to serve as the system disk and to store the software and data. Storage must be in a location that is accessible to both source and destination nodes during migrations. It is assumed the underlying management layer of the CVF will handle availability of the drives during migrations. In CVF, storage is implemented as drive images files stored on the SAN which emulates a block storage device. For our Xen prototype system, we used NFS4 with simple files as drive images. Storage must be trusted to maintain trust in the vsDom. For this dissertation, we assume virtual storage is trusted such that the confidentiality, availability, and integrity are all uncompromised. Further discussions on trusted storage are beyond the scope of this research. Virtual storage issues are also out of the scope; however, storage migration may be necessary in the case a vsDom is migrated off of a network served by the SAN.

LoBots do not require block storage devices as they are non-persistent. This alleviates the need to have a trusted store for LoBots and ensures that the LoBot images are correct if the image is properly provisioned on the destination platform.

PVI operates on a high speed local area network inside CVF which is isolated from the Internet by firewalls, network address translation, and other security devices. To ensure private networking within PVI, encrypted VPNs and virtual LAN partitioning are used to isolate PVI from the other CVF network traffic. We assume the network boundary protection devices are robust enough to thwart attacks from the Internet including viruses, worms, and denial of service attacks.

3.3.5 Measurement and Secure Provisioning

Removing the abstraction of the fabric is a trade off that we must be willing to take to increase the security of the virtual datacenter. This means that service providers must allow clients transparent insight into their infrastructures. Most providers today do not want to provide details about their inner workings as they fear this will remove their competitive advantage; however, we feel that providing a synergistic relationship with their customer base can also be their competitive advantage.

Fabric pre-measurement is what allows PVI to share the responsibility of security management between the service provider and client. Pre-measurement is performed by a LoBot, which tests the fabric's security posture before provisioning occurs, allowing the information owner to determine the safeness of the fabric before deployment of a PVI. After LoBots probe target platforms for security properties they can securely provision VMs on those platforms. The probe application reads the platform configuration from the target TPM's PCR and obtains identifying information about the platform. This information is then combined with the VTPM's PCR which is cryptographically sealed in a blob that is transferred to the PVI factory.

The PVI factory decrypts the blob and examines the information received to determine whether the environment is safe. Once the target environment is determined to be safe, the PVI factory configures the VM and securely transfers it to the target environment, via the LoBot protocol, in a blob encrypted such that only the target platform may execute source environment.

At the target environment, the LoBot probe application receives and unseals the source environment. If the source environment was tampered with during transfer, it will

be detected during the decryption phase. To make sure everything is safe, the probe measures the source environment one more time to validate its integrity and to ensure the launch in the target environment was successful.

3.3.6 Secure Shutdown and Data Destruction

Since PVI runs on shared hardware platforms, secure shutdown and data destruction is required to ensure all sensitive data are removed before new processes are allowed to run on it. All memory used by virtual machines should be zeroized such that object reuse attacks [37] are thwarted.

Current virtual machine monitors do not provide secure shutdown or data destruction capabilities. A vulnerability occurs when a VM with sensitive information is shut down and a new VM is provisioned with the same memory space. The new VM could simply read its entire memory space looking for data left behind by the previous VM. The security and privacy implications of such a threat are very serious as many organizations process sensitive information that can be stolen and used for identity theft, fraud, blackmail, and other illicit activities. We recommend that secure shutdown and data destruction capabilities be built into future virtual machine monitors; however, we believe that through LoBot, we can provide the capability to wipe a virtual machine's memory space securely after shutdown thus eliminating any data that may have been left behind by the virtual machine.

3.4 Discussion

PVI is a new paradigm for securing and managing cloud computing services based on a synergistic relationship between the vendor and customer of cloud services. This

relationship provides an increased security posture while allowing both parties to set security controls required to protect the infrastructure and data within the cloud and virtual datacenter.

3.4.1 Security

Private Virtual Infrastructure meets the goals of a shared security posture where all resources necessary for the virtual datacenter are securely isolated from the greater cloud. LoBots provide secure provisioning of commodity internet resources within the PVI. Isolation of the client's virtual datacenter is accomplished through VM containment, encryption, and access control.

Improving the overall security of the cloud is the ultimate contribution of PVI and LoBot. The PVI architecture creates the shared security posture necessary to manage the virtual datacenter in the cloud and securely isolates the virtual datacenter from malicious actors, other applications, VMs, and malware (*e.g.*, viruses and worms), enabling the data owner to use commodity internet resources and reduce their overall IT overhead.

PVI reduces the risks of using cloud computing by allowing the information owner to set the security posture they require. PVI's security properties are in addition to any link encryption, secure tunneling, virtual private networking, virtual LANs, and other techniques used to protect the virtual datacenter. PVI is another layer of a good defense in depth strategy allowing us to achieve a very high security posture within the cloud and provide a high level of assurance that sensitive information is not being comprised.

Since PVI is an architecture and framework, many of the security features of PVI are implemented as sub-components of the architecture. The threats and vulnerabilities of

those components and their specific configuration will have will determine the security to the overall PVI.

3.4.2 Cost and Performance

Performance and cost of PVI is a major consideration for deployment. There are two phases where performance and cost are of concern: provisioning and operation.

During provisioning, PVI has a high overhead as LoBots and application servers have to be moved from the PVIF to the TCFP. An AS image could be as large as several gigabytes, which could consume a large amount of time and bandwidth. Bandwidth bills could be quite large as some service provider charge per gigabyte, megabyte or kilobyte used. For example, Amazon EC2 currently charges up to $0.15 per gigabyte, meaning a single machine could cost a dollar or more to provision. If thousands of machines are needed, then cost to provision a large datacenter could reach into thousands of dollars. The time of provisioning could be large as well, at modest Internet speeds, a single server could take over an hour to transfer. This would not be feasible for a large number of machines that are needed instantaneously. Therefore, we recommend an agreement with the service provider to establish PVI factories on site and used shared storage to reduce the overhead of provisioning.

Once a PVI is in the operating phase, performance impacts are not as severe. Cost becomes an hourly rental charge, on the order of pennies to dimes per hour. There is minimal performance impact from swapping between the AS and LoBot, and some communication between them, but this would be present in any virtualized environment. Bandwidth costs would only be required for any communication between the end users and the PVI Factory, which can vary depending upon the application.

Chapter 4

Locator Bot

LoBot answers the question of whether a cloud computing platform is trustworthy. While determining whether a particular cloud platform is secure is an undecidable problem, there are certain properties we can measure to help determine a level of trustworthiness of the platform. These properties include the EK and *Attestation Identity Keys (AIK)* of the cloud platform, an attestation of security properties from the TCFP, and an attestation of the configuration of the TCFP. Using these property measurements from the LoBot, the PVIF can decide whether to trust a particular cloud platform. Once the level of trustworthiness is determined, LoBot provides secure provisioning, migration, and monitoring services for the VSs in PVI.

The AS has trust rooted both in the TCFP and in the PVIF. The TCFP's physical *Trusted Platform Module (TPM)* binds the LoBot's VTPM to the TCFP's root of trust. The LoBot's VTPM trust is rooted with the PVIF that generates its EK and VTPM proof. This trust relationship is a new model that removes the root of trust for the AS solely from the physical server by adding a component from the virtual infrastructure. Typical trusted computing techniques have a single root of trust in the TPM, but due to the unique nature of the cloud computing environment, we propose this new model of combined trust as a method to verify the authenticity of a virtual server in a cloud environment.

Outline: This chapter is organized as follows. Section 4.1 reviews background of related security research in the virtualization and cloud computing. Section 4.2 presents the LoBot protocols. Section 4.3 analyzes how the components and protocols of PVI and LoBot work together to provide security and privacy in the cloud.

4.1 Related Work

This chapter extends our previously published work on Private Virtual Infrastructure [31] with more specific details on the protocols for secure provisioning and secure migration implemented by LoBot.

IBM's *Integrity Measurement Architecture (IMA)* [38] validates all executable content on a platform, from the hypervisor up to the application layer, by measuring the content before execution. IMA uses the TPM to store the measurements for attestation and validation at a later time. IMA provides guarantees that content integrity is maintained but provides no guarantees of confidentiality if the information owner decides not to trust the environment. IMA led to several proposals for endpoint integrity including [39, 40]. These techniques measure the endpoint, set up secure tunnels, and verify security properties of the target to determine the trustworthiness of the endpoint. LoBot uses similar techniques to pre-measure a target environment by combining the cloud trust model with endpoint trustworthiness verification and provides secure provisioning services.

Distributed *Mandatory Access Control (MAC)* research by McCune [41] defines a shared reference monitor that enforces MAC policies across a distributed set of machines, allowing the setting of a consistent security policies and access controls across them. PVI

uses a form of distributed MAC for the PVI layer, but MAC may or may not be in place on the CVF layer; therefore, we must still determine the security properties of the host.

Property based attestation [42] is one method to determine security properties of a host that allows a platform to attest to properties about itself rather than performing a binary measurement of the platform. Property based attestation provides more flexibility for attestation and is more resilient to configuration changes and patches than measurement based attestation. A protocol for property based attestation proposed in [43] provides a means to perform remote property based attestations but does not provide any provisioning capabilities.

IBM's *Trusted Virtual Datacenter (TVDc)* [8] incorporates trusted computing technologies into virtualization and systems management providing many features that can be used for securing a cloud datacenter including VM isolation via sHype, vTPM support, and system management software. TVDc uses Trusted Virtual Domains [44] to provide strong isolation and integrity guarantees that significantly enhance the security and management capabilities of virtualized environments. The capabilities of the PVI architecture and LoBot provisioning protocols enhance TVDc's features when used in a cloud environment.

An agent transfer protocol [45] for moving a trusted agent around the cloud between fixed and mobile platforms closely resembles LoBot; however, LoBot is designed specifically for building trust in PVI and securing provisioning and live migration of virtual machines. LoBot has an entirely different purpose than the trusted agent, but there are many similarities in the way trust is established on the remote platform. Both agents

use property based attestation to determine the security properties of the destination platform

4.2 The LoBot Protocols

The *LoBot Secure Provisioning (LSP)* and *LoBot Secure Migration (LSM)* protocols ensures that when a VM is provisioned within the PVI it is not subverted in any manner and the VM that is provisioned is exactly the same as the VM intended to be provisioned. LoBot ensures a safe provisioning by pre-measuring a target platform for trustworthiness. By sending a LoBot before provisioning an aDom, the target platform can be measured to determine whether the target provides a trustworthy operating environment. This allows us to make a decision to provision to the target based on the results of the probe. Secure provisioning is achieved through the combination of determining a target platform's trustworthiness, ensuring that provisioning is not subverted in any manner, and verifying the VM is created and launched on the host platform as intended.

Live migration uses the LSM protocol, which is an extension of the LSP protocol. The requirements of secure provisioning and secure live migration are the same: ensure a virtual machine is placed on the cloud platform as intended. Secure live migration is slightly more difficult than secure provisioning as there are three entities involved (PVI Factory, source and destination platforms) and the state of the source VM must be preserved to have a successful migration. In a live migration, LSM ensures that the VM's state is not compromised by the migration and that the VM migrated to the new platform resumes operation at the point of suspension on the originating machine.

Symbol	Meaning
S or SP	Source Platform
D or DP	Destination Platform
L or LoBot	Locator Bot
P or PVIF	PVI Factory
aDom	Application Domain
vsDom	Virtual server domain (aDom + LoBot)

Table 4.1 – Abbreviations used in LoBot Protocols

We created a new management application on the PVIF called *pvif* that is used to initiate the provisioning and live migration LSP and LSM protocols. When *pvif* is initiated with a *create* subcommand, a secure provisioning is initiated. The *migrate* subcommand initiates the secure live migration. The protocol uses the symbols shown in Table 4.1.

4.2.1 LoBot Secure Provisioning Protocol

The following steps describe the secure provisioning of a VS (both aDom and LoBot) from the *PVI Factory (PVIF)* to a *Destination Platform (DP)* machine using the LSP protocol in detail. A diagram of the protocol can be seen in Figure 4.1.

1. PVIF initiates a provisioning of vsDom on the DP via:

   PVIF: pvif create vsDom DP

   The PVIF starts the process of provisioning vsDom, which consists of aDom and associated LoBot

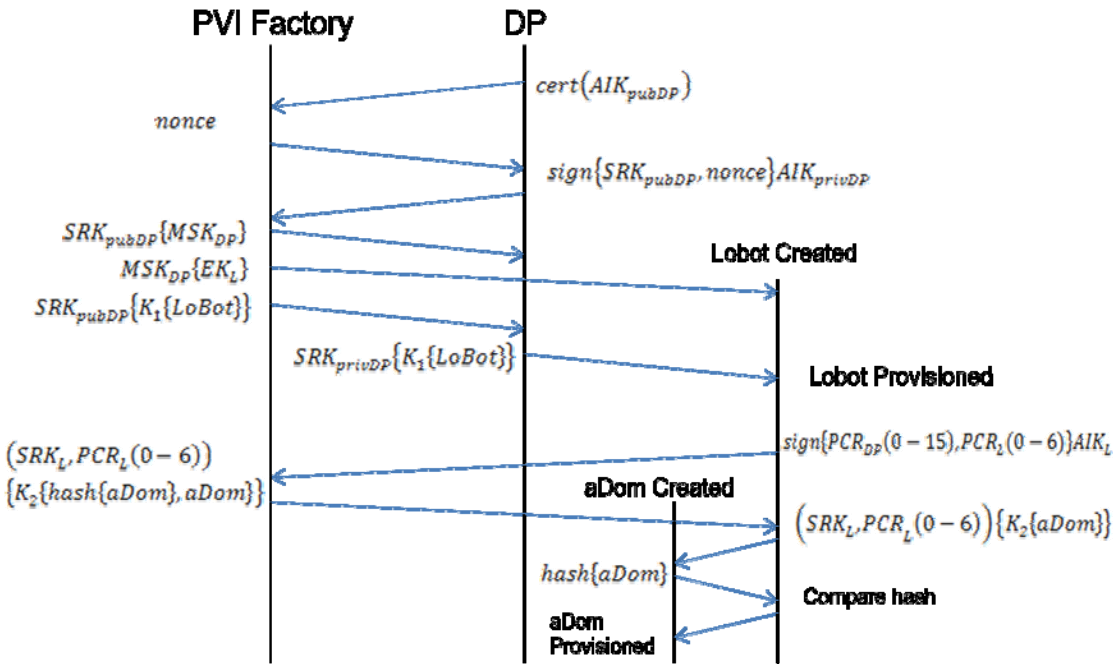

Figure 4.1 – Diagram of LoBot Secure Provisioning protocol.

2. PVIF requests an AIK certificate from DP.

$$PVIF \leftarrow DP\colon\ cert(AIK_{pubDP})$$

The AIK is used to verify that DP has a valid TPM and determine whether that TPM is a real or virtual. This step also verifies that the TPM has a feature set which is acceptable to the PVIF. Note that we chose to use the AIK in this step to preserve privacy of the CVF; however, an EK could be used if privacy is not a concern of the CVF operator or the AIK is not available.

3. PVIF creates a non-predictable 160-bit nonce and sends it to DP requesting a quote of the AIK.

$$PVIF \rightarrow DP\colon\ nonce$$

4. PVIF requests DP's SRK public key, which is sent along with the nonce signed with DP's private AIK.

$$PVIF \leftarrow DP\colon\ sign\{SRK_{pubDP}, nonce\}AIK_{privDP}$$

PVIF then verifies the nonce is the same as previously sent and that the signature is validated by the CA that created the AIK.

5. PVIF creates a *Migratable Storage Key (MSK)*, wraps it with DP's SRK, and sends to DP. PVIF then generates the EK for the LoBot's VTPM, encrypting it with the MSK so that is will be bound to the DP ensuring no other entity can use it. The encrypted EK_L is then placed in LoBot image.

$$PVIF \rightarrow DP\colon\ SRK_{pubDP}\{MSK_{DP}\}$$

$$PVIF \rightarrow LoBot\colon\ MSK_{DP}\{EK_L\}$$

6. PVIF then encrypts the entire LoBot image using the SRK public key and a symmetric *encryption key (K_1)*, and sends the entire package to DP.

$$PVIF \rightarrow DP\colon\ SRK_{pubDP}\{K_1\{LoBot\}\}$$

7. DP receives the encrypted LoBot package and decrypts it using its SRK private key and K_1.

$$DP \rightarrow LoBot\colon\ SRK_{privDP}\{K_1\{LoBot\}\}$$

If LoBot is successfully decrypted, then it is launched by DP creating a VM and verifying that the LoBot is launched correctly.

8. The newly launched VM measures LoBot, storing the results in the LoBot's vTPM's PCRs 0-5 and extends DP's PCR 6 with the measurements. The LoBot's PCR 6 is extended with DP's PCR(0-6).

$$LoBot\colon\ extend\{PCR_{DP}(6), hash(PCR_L(0-5))\}$$

$$LoBot\colon\ extend\{PCR_L(6), hash(PCR_{DP}(0-6))\}$$

9. LoBot quotes DP's PCRs and its own PCRs to PVIF

$$LoBot \rightarrow PVIF\colon\ sign\{PCR_{DP}(0-15), PCR_L(0-6)\}AIK_L$$

10. PVIF determines trustworthiness of DP based on measurement returned from LoBot.

    10a. If DP is deemed trustworthy, PVIF continues provisioning by initiating transfer of app domain.

    10b. If DP is deemed untrustworthy, PVIF notifies LoBot to destroy itself and terminates provisioning.

11. PVIF hashes aDom and encrypts aDom and hash value with K_2. PVIF then wraps K_2 with the LoBot's SRK and PCR(0-6) to ensure only the LoBot can unseal and the state of the DP has not changed. Note that PVIF has LoBot's keys and the ability to emulate a TPM enabling PVIF to seal images for LoBot.

$$PVIF \rightarrow LoBot\colon \ \big(SRK_L, PCR_L(0-6)\big)\big\{K_2\{hash\{aDom\}, aDom\}\big\}$$

12. LoBot unseals aDom and the hash measurement, hashes aDom and compares it to the sent hash to determine the transfer success.

$$LoBot \rightarrow aDom\colon \ \big(SRK_L, PCR_L(0-6)\big)\big\{K_2\{aDom\}\big\}$$

$$LoBot\colon hash\{aDom\} == \big(SRK_L, PCR_L(0-6)\big)\big\{K_2\big\{hash\{aDom\}\big\}\big\}$$

13. If the hashes compare, aDom is launched via SENTER. If SENTER succeeds, a safe provisioning occurred; otherwise, the LoBot reports unsuccessful migration to PVIF, destroys the aDom and then itself.

4.2.2 LoBot Secure Migration Protocol

A live migration can be initiated at the source platform or PVI factory, but to maintain trust in PVI, the PVI Factory manages the live migration. The following steps describe LSM protocol for live migration of a VS (both aDom and LoBot) from a *Source Platform (SP)* to DP.

1. Initiate a live migration from the source platform to the destination platform on PVIF via the following command:

$$PVIF\colon \ pvif\ migrate\ domA1\ DP$$

   In the event the source platform needs to initiate the migration, the command must be sent to the PVIF to start the migration.

2. PVIF requests an AIK certificate from DP to verify DP's TPM.

$$PVIF \leftarrow DP\colon \ cert\big(AIK_{pubDP}\big)$$

PVIF already knows the AIK of SP since it would have had to previously provision it.

3. PVIF creates a non-predictable nonce and sends it to both SP and DP.

$$PVIF \rightarrow SP, DP\colon\ nonce$$

4. PVIF requests DP's SRK public key be sent to SP, which is sent along with the nonce signed with DP's private AIK.

$$SP \leftarrow DP\colon\ sign\{SRK_{pubDP}, nonce\}AIK_{privDP}$$

SP then verifies the nonce is the same as previously sent and that the signature is validated by the CA that created the AIK. SP reports nonce back to PVIF.

5. PVIF requests P to halt aDom, capture its state, and begin migration process. SP then clones the LoBot and rewraps the MSK_{SP} for the VTPM with the public SRK from DP.

$$SP \rightarrow DP\colon\ SRK_{pubDP}\{MSK_{DP}\}$$

6. SP encrypts $LoBot_S$ image and sends to DP.

$$SP \rightarrow DP\colon\ SRK_{pubDP}\{K_1\{LoBot_S\}\}$$

7. DP receives encrypted LoBot blob and decrypts it using its SRK private key and symmetric key K_1.

$$DP\colon\ SRK_{privDP}\{K_1\{LoBot_S\}\}$$

If LoBot is successfully decrypted, then it is launched by DP creating an VM. $LoBot_S$ becomes $LoBot_D$.

8. The VM measures $LoBot_D$, storing the results in the LoBot's vTPM's PCRs 0-5 and extends DP's PCR 6 with the measurements. The LoBot's PCR 6 is then extended with DP's PCR(0-6).

$$LoBot_D\colon\ extend\{PCR_{DP}(6), hash(PCR_L(0-5))\}$$

$$LoBot_D\colon\ extend\{PCR_L(6), hash(PCR_{DP}(0-6))\}$$

9. $LoBot_D$ quotes DP's PCRs to PVIF and its own PCRs to $LoBot_S$.

$$LoBot_D \rightarrow PVIF\colon\ sign\{PCR_{DP}(0-15)\}AIK_L$$

$$LoBot_D \rightarrow LoBot_S\colon\ sign\{PCR_L(0-6)\}AIK_L$$

10. PVIF determines the trustworthiness of destination based on measurement returned from LoBot.

    10a. If DP is deemed trustworthy, PVIF notifies SP to continue transfer aDom.

    10b. If DP is deemed untrustworthy, PVIF notifies LoBot to destroy itself and terminates migration.

11. Source binds measurement with LoBot's PCRs and sends to LoBot.

$$LoBot_S \rightarrow LoBot_D\colon\ \big(SRK_L, PCR_L(0-6)\big)\{K_2\{hash\{aDom\}, aDom\}\}$$

12. $LoBot_D$ measures aDom, unseals measurement and compares its measurement to the sent measurement.

$$LoBot_D \rightarrow aDom\colon\ \big(SRK_L, PCR_L(0-6)\big)\{K_2\{aDom\}\}$$

$$LoBot_D\colon hash\{aDom\} == \big(SRK_L, PCR_L(0-6)\big)\big\{K_2\{hash\{aDom\}\}\big\}$$

13. Determine migration success.

13a. If the hashes compare, aDom is launched via SENTER. If SENTER succeeds, a safe migration occurred; $LoBot_D$ reports successful migration and $LoBot_S$ destroys aDom and itself on SP.

13b. If the hashes do not compare or SENTER fails, $LoBot_D$ reports unsuccessful migration to SP, SP unsuspends operation of LoBot and aDom, and reports failure. $LoBot_D$ destroys aDom and itself on DP.

4.3 Discussion

LoBot along with the LSP and LSM protocols provide high-assurance mechanisms to provision and migrate virtual machines securely in the cloud. LoBot uses VTPMs to provision trustworthy VMs in the cloud requiring individual computing platforms within the cloud to have a TPM accessible by LoBot. The LoBot's trust authority for the VTPM is the PVI Factory and linking the VTPM to the platform's physical TPM creates a dual rooted trust for the application domain. This dual rooted trust anchors the application domain to the host platform and PVI preventing tampering and cloning and ensuring the data are protected from adversaries in the cloud.

4.3.1 Security

The LSP protocol provides an assured mechanism to provision a VM securely on a destination platform. There are several advantages of this protocol over existing provisioning techniques. Existing provisioning protocols do nothing to assure the safety and integrity of the destination environment before the provision occurs and little to ensure that the provision occurred safely. The existing protocols have fallback

mechanisms in the event of a failed transfer, but once a VM is placed in a malicious or unsafe environment, the VMs confidential data are compromised and cannot be recovered.

The LSP protocol provides two assurances that were previously unavailable: first, the target platform we wish to provision on is a platform we trust (*e.g.*, it has a configuration and security properties that are acceptable) and second, we know if the VM provisioning or migration occurs successfully. The trust decision is made by the PVIF from the LoBot pre-measurement of the platform made by reading the PCRs of the target's TPM and measuring other properties of the target platform. If target platform is deemed trustworthy, we continue with the provisioning as planned; otherwise, we back out of provisioning preventing data in the VM image from being sent to an untrustworthy machine. The only risk of contacting a malicious server is an attacker would be able to obtain some information and keys from a LoBot, but since LoBot has a minimal set of functions, the risk is insignificant. If the LoBot is compromised, all keys and references to that individual LoBot must be destroyed to ensure that no compromises can be achieved via the LoBot's credentials. Determining if a VM provisioning occurred as intended and not subverted in any manner is accomplished by cryptographically binding the VM image to the target configuration ensuring that the VM cannot be provisioned anywhere other than the intended target and the configuration of the target is not be altered between measurement and provisioning.

From the above analysis, we can see that the LSP and LSM protocols defend against all three of the attacks discussed in Section 2.3.1. The malvm and malhost are detected by the LoBot during initial probe phase. LoBot detects if a machine is not registered, has an

invalid certificate, or has an alternate configuration from what is expected, and prevents the VM from provisioning on the machine. A man-in-the-middle attack is thwarted by the protocol through encryption of the VM image and tampering is detected by the hash comparisons.

A current limitation of the LSP protocol is the vulnerability of compromise to host machine after provisioning. Changes to the node's configuration are not detected after the initial trustworthiness of the node is determined. If the attacker gains control of the host machine after provisioning, the CVF architecture could be compromised. The PVI architecture and LoBot protocols are designed to protect the application server from being migrated to an environment that is predetermined to be malicious, but if the host becomes malicious after provisioning it may be possible to circumvent the security controls and compromise data.

4.3.2 Cost and Performance

This section examines the performance and cost of LoBots. There are three areas where performance and cost are of considered: provisioning, operation, and key management.

Provisioning has an impact as the entire VM must be transferred from the factory to the host. A LoBot is not a large device and should be an order of magnitude smaller that an operating system. Bandwidth cost and provisioning time are the largest concern.

LoBot has performance overhead while in operation that comes from context switching and communications. The impact to performance of context switching between the virtual environments is a slowdown of overall processing as the processor has to switch from one virtual environment to another. Communication with the PVI factory

and possibly other LoBots in the PVI are the largest component of overhead. If there is a large number of LoBots communicating at the same time, the communication overhead could impact the overall performance of the system.

The PVIF must create and maintain keys for each VTPM generated and used in the PVI. The PVIF is the certificate authority for keys it generates, therefore, it must be permanently keep a record of each key generated and used in order to prevent reuse attacks. This database could get quite large over time if large numbers of LoBots are created and destroyed.

Chapter 5

Trusted Virtual Environment Module

The *Trusted Virtual Environment Module (TVEM)* introduced in [46] is a new mechanism for rooting trust in a cloud computing environment. The TVEM helps solve the core security challenge of cloud computing by enabling parties to establish trust relationships in a cloud computing environment where an information owner creates and runs a virtual environment on a platform owned by a separate service provider. The TVEM is a software appliance that merges trust from multiple sources, typically the information owner and service provider, to derive a root of trust for a virtual environment on a remote host. The TVEM provides enhanced features for cloud virtual environments over existing Trusted Platform Module virtualization techniques including an improved application program interface, cryptographic algorithm flexibility, and a configurable modular architecture. TVEMs are managed via a *Virtual Trust Network (VTN)* with a central control facility called the TVEM Factory that manufactures and provisions TVEMs in the cloud. A unique *Trusted Environment Key (TEK)* is defined that combines trust from the information owner's VTN and the service provider's platform to create a dual root of trust for the TVEM that is distinct for every virtual environment and separate from the platform's trust. This chapter presents the requirements, design, and architecture

of the TVEM, VTN, and TEK in enough detail to support further analysis and implementation.

The *Private Virtual Infrastructure (PVI)* cloud trust model [47] describes the unique trust relationships that occur in *Infrastructure as a Service (IaaS)* [36] cloud computing environments. This chapter applies the PVI cloud trust model to IaaS clouds with our new *Trusted Virtual Environment Module (TVEM)* and *Virtual Trust Network (VTN)*.

In IaaS cloud computing, an *information owner*, or client, rents virtual computing resources in the form of a *Virtual Machine (VM)* on a host platform operated by a second party *service provider*. The information owner wishes to protect private and sensitive data that are processed in the virtual environment on the rented VM. The *virtual environment* is the entity that is controlled by the information owner and consists of *all* software components, from the *Operating System (OS)* to the applications, that execute on the VM. To assure the information is protected, the client needs to verify the trustworthiness of the host platform and virtual environment. The TVEM and VTN provide the mechanisms to verify the host platform and virtual environment within an IaaS cloud and report the results back to an information owner. No current capability exists to perform these functions.

A current means for establishing trust in computing platforms is the *Trusted Platform Module (TPM)*, a core component of the *root of trust* for the platform. A root of trust is a component of a computing platform that is implicitly trusted to provide a specified set of controlled functions to measure and pass control to other platform components [9]. TPMs are designed to support a single operating system on a single platform and typically do not scale well when virtualization is introduced to the platform [48]. Support for multiple

virtual environments that simultaneously access TPM resources is required. A *Virtual TPM (VTPM)* that replicates the physical resources of a TPM in software is one method of virtualizing the TPM functions for sharing among multiple virtual environments.

LoBot [31, 47] uses the VTPM to root trust for a virtual environment in a PVI; however, the VTPM implementation has several issues that make it problematic to use as a root of trust for cloud virtual environments. Three major shortcomings of the VTPM are: the VTPM's trust is rooted to the physical platform on which it is operating, which is typically not owned by the information owner; a VTPM must follow the TPM specification [15], which includes extraneous functionality that is not useful for virtual environments; and a VTPM has non-persistent storage, meaning that it loses all keys, settings, and non-volatile storage upon termination. The TVEM solves these problems through application of the PVI cloud trust model, providing a modular and extensible architecture that allows algorithm and function flexibility, and providing persistent storage for keys, non-volatile memory and settings.

The core challenge in cloud computing that TVEM solves is establishing trust that is distinct for the virtual environment and separate from the hosting platform. *Virtual environment trust* is defined as trust in the virtual environment that is a combination of trust in the service provider's platform and trust from the information owner's domain. Virtual environment trust is necessary to convey ownership and protect information in the cloud. To implement this virtual environment trust, a *Trusted Environment Key (TEK)* is defined and used as the *Endorsement Key (EK)* for the TVEM. The TEK, like the EK, is a unique value used as the *Root of Trust for Reporting (RTR)* to identify the TVEM and attest the virtual environment. The TEK is generated by the virtual environment owner

and secured with the service provider's platform storage key creating a *compound trust* distinct and separate from the platform.

The TVEM is a software appliance that is implemented as a helper, or stub, VM. The TVEM is protected by hardware enforced memory and process isolation via Intel's *Virtualization Technology for Directed I/O (VT-d)* [49] and *Trusted eXecution Technology (TXT)* [50]. The TVEM provides attestation support and trusted storage for the virtual environment similar to functionality provided by VTPM; however, the TVEM does not have to conform to the TPM specification enabling the TVEM to be extensible through functional and cryptographic algorithm flexibility in a configurable modular architecture. The TVEM has multiple interfaces, including an *Application Program Interface (API)*, which moves the *Trusted Software Stack (TSS)* into the TVEM eliminating the burden on the virtual environment to implement the TSS. The API provides for hardened and lightweight environments and reduces the opportunities implementation errors. These capabilities allow system designers to customize the TVEM and virtual environment to meet their information confidentiality and integrity requirements.

TVEMs are not stand alone devices; they are part of a system to implement trust in cloud computing. The system includes: the TVEM; a TVEM manager in the host hypervisor for host platform TPM access and TVEM provisioning; a VTN control plane that provides system management and support for persistent storage; and a *TVEM Factory (TF)* to manufacture TVEMs, manage keys, and provision TVEMs securely on host platforms.

Outline: This chapter is organized as follows. Section 5.1 motivates our work with examples. Section 5.2 covers trust in cloud computing. Section 5.3 provides background on trusted computing and discusses related work. Section 5.4 overviews the requirements and design considerations. Section 5.5 presents the TVEM system architecture. Section 5.6 is the detailed design of the TVEM. Section 5.7 discusses the advantages, cost, security, and requirements of TVEMs.

5.1 Motivation

Utility cloud computing can provide many benefits to companies wishing to reduce their IT expenses and overhead. Security of information in the cloud and the trustworthiness of the cloud environment is a major concern with IaaS clouds. We describe three example IaaS cloud computing applications: a cloud web server, a cloud datacenter, and a corporate virtual desktop. These applications benefit from the added security of using the TVEM and VTN to manage trust.

5.1.1 Cloud Web Server

A virtual web server in the cloud has many benefits over maintaining a web server locally, a significant advantage being increased availability. The cloud's always-on presence and location flexibility enhance the availability of a web server by providing scalability, migratability, and redundancy. If a server is overwhelmed with requests, it can be migrated to a platform that has increased capacity, or new instances of the server can be instantiated to handle the increased load. Migration and failure restart can be used if host hardware fails or Internet service becomes unavailable.

A server certificate is a critical piece of data on the web server that authenticates its owner. If a company wants to prove that it is the owner of a web server, it would obtain an *Extended Validation (EV)* certificate and a *Secure Socket Layer (SSL)* certificate from a certificate authority. The EV and SSL certificates have a public and private key portion that a guest may use to verify the server owner and establish an encrypted SSL session with a server. In a cloud environment, the identity of the web server owner and the service provider needs to be differentiated, which is accomplished via the certificates. The certificates should be accessible only by the web server owner and must be protected from the service provider and other users of the cloud service. If the private portions of the keys are disclosed, anyone who gains access to the private keys can purport to be a valid web server for the information owner. If the private key is stored on a public cloud service, anyone with access to the system could possibly access the key; therefore, the owner of the certificate needs to keep the private key protected from compromise by the service provider, other cloud users, and attackers on the Internet. TVEM protects SSL certificates on a cloud web server by encrypting the certificate such that is accessible only by the TVEM and decrypted inside the host platform's TPM ensuring the plaintext key cannot be observed.

5.1.2 Cloud Datacenter

A cloud datacenter is a network of virtual servers that allows a company to move all of its corporate data assets into the cloud. The only IT the company maintains on-site is data terminals, laptops, netbooks, or mobile computing devices for their employees to access the cloud datacenter. An example would be a company that maintains a database of client, suppliers, billing, and inventory. Moving the database to a cloud datacenter

would allow the company to leverage large amounts of computing power and throughput to run analytics and data mining operations as needed without having to maintain those computing resources locally. Moving to the cloud not only saves overhead for space, power, and cooling, but also provides employees at remote locations with easy access to the information.

The cloud corporate datacenter contains all the sensitive and proprietary information about the company's relationships, processes, and accounting. The company wants to have assurances that its data are protected from other cloud users, the service provider, and personnel at the cloud datacenter. The TVEM protects the information in the datacenter and verifies the configuration of the host platform and virtual servers meet the requirements of the information owner.

5.1.3 Corporate Virtual Desktop

A *Corporate Virtual Desktop (CVD)* allows an employee of a company to use her own computer to run a virtual image of the company's standard desktop consisting of approved applications, data, and network connectivity [51]. The CVD is a virtual environment that connects the user's laptop or home computer to a private corporate network. The CVD contains the user's identity certificates, authentication credentials, and network encryption keys. These keys must be used only while in the CVD environment and not be accessible from outside of the environment. The TVEM encrypts the keys to ensure they are only used within the CVD.

Since the CVD is executed on a laptop or home computer where there may be limited bandwidth to transfer large amounts of data (*e.g.*, 3G wireless systems), the CVD image is stored on the user's machine. The corporate owner uses a TVEM to verify that the

configuration of the CVD has not been modified prior to allowing the CVD access the corporate network. The TVEM also verifies the user's computer configuration to ensure that it does not pose a threat to the secure information in the image or on the network.

5.2 Trust in the Cloud

Trust in cloud computing is more complex than in a traditional IT scenario where the information owner owns his own computers. Figure 5.1 shows the trust relationships in an IaaS cloud as defined by the PVI cloud trust model. The trust chain combines trust from the information owner's domain (or PVI) and trust from the service provider's platform into the virtual environment trust. The information owner has an inferred trust in the platform from a social trust relationship with the service providers. The information owner root of trust is established in the TF and is the core root of trust for the entire PVI. The TF needs to inherit trust from the host platform root of trust measurements to ensure that the PVI is being implemented on a trustworthy platform. The PVI combines the inherited platform trust and information owner trust in a TEK and places it in the TVEM. The TVEM trust is provisioned on the host platform such that it is bound to the platform root of trust, creating the dual rooted virtual environment trust. If either root of trust is revoked, the virtual environment trust is invalidated.

5.2.1 Social Trust

Social trust is a trust that arises between two entities based upon social relationships. Social trust is established based upon reputations, previous interactions, and contractual obligations. There are two critical social trust relationships that must be established in cloud computing from the perspective of the information owner: service provider trust

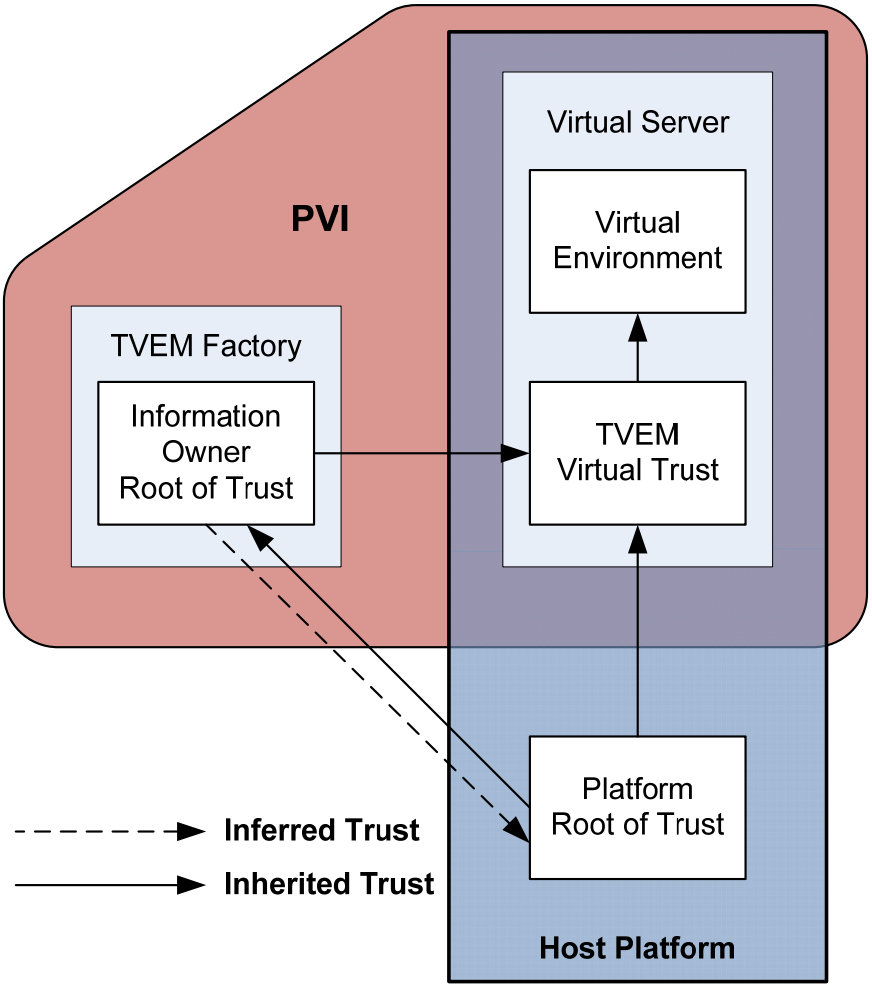

Figure 5.1 – Trust relationships in an IaaS cloud computing environment consist of inferred trust (social trust) and inherited trust (technical trust). The TVEM's virtual trust is a combination of the information owner's trust and host platform trust.

and cloud user trust. Social trust cannot be measured, but is important to build confidence that an entity is holding up its end of a contract.

Service provider trust lies in the relationship between customer and vendor. If the provider has a good reputation, then there is sufficient reason for customers to trust the provider. A vendor that has questionable service or ethics would not be as trustworthy as a vendor with excellent service and ethics.

Cloud user trust is the amount of trust the user places in the services delivered via the cloud. The user has to be confident that the system is going to protect their data, transactions, and privacy. The user's trust is a social trust in the information owner. The information owner must assure that the services being provided meet the user's expectations.

5.2.2 Technical Trust

In a cloud computing environment, multiple entities must trust the cloud services: the user of the cloud service or information owner, the provider of the cloud service, and third parties. PVI defined a new paradigm of cloud computing that separates the security responsibility between the service provider and information owner and accounted for third parties. A third party is an outside entity that is providing service to or receiving services from either the user or service provider.

The cloud trust model is based on *transitive trust*, which is the notion that if entity A trusts entity B and entity B trusts entity C, then entity A trusts entity C. This property allows a chain of trust to be built from a single root of trust.

There two basic sources of trustworthiness in a cloud: information owner trust and hosting platform trust. By combing these two sources of trust, virtual environment trust can be established.

Information owner trust is the foundation of trust that the information owner places in the PVI. Information owner trust is implemented by the TF. Since the information owner has physical control of the TF, the configuration of the TF is a known quantity and can be used as the root of trust for the PVI. As long as the information owner maintains trust in

the TF, trust can be established in the PVI and used to build trust chains with cloud host platforms.

Host platform trust lies in the hardware trust of the platform and is measurable. Trustworthiness starts with *Core Root of Trust for Measurement (CRTM)* of the platform. The CRTM is the core set of instructions run at boot or reset that are responsible for establishing trust in the system by measuring the BIOS, and then passing control to the measured BIOS and rest of the *Trusted Computing Base (TCB)* of the platform. The TCB consists of all measured components that provide the foundation of trust in the platform. Platform trustworthiness is determined by an outside entity via attestation from the TPM, which is the *Root of Trust for Reporting (RTR)* on the hosting platform. The TPM also serves as the *Root of Trust of Storage (RTS)* of the platform that is implicitly trusted to store information securely. The attestation from the TPM provides evidence of the state of the platform, from which other entities can decide whether to trust the platform.

Cloud virtual environment trust is the amount of trust placed in the virtual environments created in the cloud. Virtual environment trust is measureable, but there are complications in a cloud environment where the information owner's requirements are different than the platform owner's. LoBot measure trust within a PVI; however, LoBots use virtual TPMs as the basis for trust in the PVI, limiting the flexibility of the PVI and the virtual environment. We show how TVEMs can be used in PVI in place of VTPMs.

5.3 Background and Related Work in Trusted Computing

The *Trusted Computing Group (TCG)* is an international computer industry standards organization that specifies and encourages trusting computing techniques [52]. The TCG is responsible for maintaining and updating the TPM specification. The TPM

specification is currently at version TPM1.2. The TCG is currently working on the next version of the TPM specification, referred to at TPM.Next. TPM.Next will incorporate new hash algorithms and other features that will require TPM manufacturers to change the design of their TPM implementations.

The TPM specification was developed with the following high-level requirements [19]:

T1 Securely report the environment that booted.

T2 Securely store the data.

T3 Securely identify the use and system.

T4 Support standard security systems and protocols.

T5 Support multiple uses on the same system while preserving security among them.

T6 Be produced inexpensively.

These requirements led to a robust and secure design; however, design tradeoffs were made to lower costs through reduced functionality. Each TPM has limited resources but sells for less than one dollar in production quantities. Many of the TPM's limited resources cannot be shared with virtual environments; therefore, the TPM must be virtualized to support multiple virtual environments.

5.3.1 TPM Virtualization

There are many issues that must be solved to virtualize a TPM. The limited resources of the TPM must be either shared or replicated for each virtualized TPM. Specifically, resources that cannot be shared on the TPM are the EK, *Platform Configuration Registers (PCRs),* and non-volatile storage. These resources must be replicated by every VTPM implementation.

A common approach to virtualizing the TPM has been to emulate the TPM in software and provide an instance for each virtual environment. The VTPM can be bound to a physical TPM for additional security. Berger, *et al.*, [16] took this approach for their vTPM implementation along with an additional approach of using an IBM 4758 Cryptographic Coprocessor to implement the vTPMs. Scarlata [48] followed with a framework for TPM virtualization, which described a VTPM framework for emulating TPMs in software.

England [53] took a different approach to TPM virtualization with paravirtualized TPM sharing. Paravirtualization is a technique used by the Xen hypervisor [7] to present a software interface to the VM that is similar to the underlying hardware and requires the OS to be modified. This method uses the hypervisor to mediate access to a single hardware TPM. The hypervisor shadows the PCRs for each virtual environment thus overcoming the PCR limitation. This design reduces the ability for migration since the virtualization is done in the hypervisor and uses physical TPM resources that are not transferrable to other platforms.

Another unique approach is property-based TPM virtualization by Sadeghi [54]. This technique uses a different methodology to measure the platform's state and generate keys. Properties are measured for reporting the state of the platform, which are less susceptible to changes in software configuration updates and patches, and makes migration easier; however, this approach is incompatible with the existing TPM and renders existing software interfaces unusable.

The Berlios TPM emulator [55] is a form of TPM virtualization, providing a software emulation of a hardware TPM. The TPM emulator can provide TPM services to virtual

environments, but does not have any binding to the hardware, limiting its ability for operational use. Consequently, the Berlios TPM emulator is useful for development purposes only.

5.3.2 Trusted Execution Technology

Trusted eXecution Technology (TXT) is a feature of Intel's microprocessors and chipsets commonly referred to as Intel vPro [14]. TXT provides a *late launch* capability for the secure creation and launch of virtual execution environments called *Measured Launch Environments (MLEs)*. MLEs can be launched anytime after a platform is booted. Hardware protections are provided by TXT against software based attacks on MLEs during launch and while the MLE is executing.

Late launch is initiated through the SENTER function, which provisions the MLE on the host platform. The process measures and verifies the MLE with a secure hash algorithm built into TXT and correctly configures the processor and chipset prior to passing control of the processor to the MLE. This capability ensures that the MLE is the proper configuration the user wishes to use and that the MLE has been provisioned on the platform as intended.

5.4 Design Considerations

We explain our design considerations for the TVEM in this section. We considered multiple approaches to implementing a trust module for virtual environments and realized the best way to ensure that each virtual environment has a trust module is for the module to be implemented in software.

By implementing the TVEM in software, we do not have cost, physical, or resource restrictions. The TVEM design is bound by memory and computation restrictions, which are much less restrictive than physical restrictions. On a typical host platform, we can provide multiple fully functional, uncompromised TVEMs for many virtual machines at the same time.

There are several advantages to implementing a trust module in software versus hardware. A software platform can be changed to accommodate vulnerabilities that are discovered after release (*e.g.*, SHA1 collision attacks [56]). A software module can also support different algorithms for different applications and locations, which is important in cloud environments. Export controls on cryptographic algorithms may dictate that a certain algorithm may not be used in certain countries. With the worldwide presence of the cloud, algorithm flexibility is essential. A cloud environment in a restricted country will need an algorithm allowed to be exported, while a stronger algorithm may be used on a system in another country where more security is permitted. An advantage to a modular software design of the TVEM is flexibility to use algorithms that are compatible with the current TPM specification or use new algorithms for future applications and enhanced security. This flexibility allows the TVEM to be used in applications where features of TPM.Next are required, but the hardware does not support TPM.Next.

5.4.1 Threat Model

The TVEM must protect data and operations from attack. Since the TVEM will be implemented in software, there are multiple attack vectors. Potentially any code running on the host platform could attack the TVEM. We assume VT-d and TXT hardware isolation are in place, which protects the module from attack by entities with access to the

platform. The entities that have access to the platform include the following: the service provider, who has root access to the platform; other cloud users on the same system or within the same cloud environment; and outside attackers that access the system via the Internet.

A poorly implemented TVEM has numerous potential attacks. A malicious program may gain access to private data, including keys, inside of the TVEM. The malicious program can then modify and substitute data, to include replacing keys, modifying hashes, and state information. The code of the TVEM could be modified by replacing strong cryptographic algorithms with weaker ones. An attacker could also reduce the entropy of the *random number generator (RNG)* thus reducing the effectiveness of the keys. The attacker could also set the state to a previous state of the TVEM, known as a rollback attack. Rollback is used to weaken the cartographic output or replay the output of the device (see Section 5.6.6). The state could also be modified to a known or predetermined state to weaken cryptographic results.

The TVEM's main defense against attacks is robust isolation and state verification. Features of the trusted platform can be used to protect against many attacks including: software modification, malicious code, key exposure and modification, and VM isolation and containment attacks, without detection.

The TVEM cannot defend against hardware attacks since it is a software module. A sophisticated attacker with control over hardware would be able to circumvent the TVEMs security and gain access to protected information in the TVEM and modify state and/or instructions inside the TVEM to alter outcomes of the TVEM's operations.

5.4.2 Requirements

This section describes requirements that must be met in order to achieve a secure virtual environment. While this is not a complete requirement elicitation, these requirements are the key functional requirements for a TVEM.

The TVEM is to act as a TPM replacement, so the goal for developing the TVEM is the same as any trusted component – to move cryptographic computations into a locked virtual area, which is not under control of entities on the host platform [57]. The locked virtual area provides a secure location to process sensitive information and store cryptographic keys and other information that are required to be kept private. Therefore, the primary requirement for the TVEM is confidentiality of data. The purpose of the TVEM is to protect data and to do so the TVEM must be able to protect its keys and private data.

The following list enumerates and details the key functional requirements for the TVEM.

R1 Confidentiality of data. Internal TVEM data must be protected from being accessed by the host environment, hypervisor, and all other virtual environments on the platform.

R2 A TVEM should have persistent storage of keys and NVRAM from one session to the next and across migrations. This is a capability not provided by a VTPM.

R3 Flexible cryptographic algorithms are required due to export control restrictions and upgradability to newer, more secure algorithms.

R4 The TVEM must maintain the chain of trust and support transitive trust from the CRTM of the host platform, through the TPM's RTR and RTS, to the TVEM, and

on to the virtual environment. The transitive trust must provide the CRTM and RTR for the associated virtual environment.

R5 Robust random number generation is needed, especially since virtual environments are being used which are known to have low entropy [21].

R6 The TVEM must not be cloneable. If the TVEM is cloned, all trust in the environment is lost; therefore, protections must be put in place to prevent a TVEM from being copied to another platform.

R7 Rollback and state modification attack detection is required to ensure integrity of the cryptographic output. Detecting and reporting unauthorized modification to the state of the TVEM will detect these types of attacks.

R8 Migration is required to transfer the computing environment to a new platform when necessary.

Securely implementing these requirements will provide a trustworthy TVEM software appliance to serve as a root of trust for the virtual environment.

5.4.3 Deployment Model

The deployment model for the TVEM is to build and maintain TVEMs on the TF in a secure location that is under the information owner's full physical control. The TF should be isolated and physically separated from the service provider's facility to ensure that it cannot be compromised.

Fully self contained TVEMs images are configured and built in the TF. The TVEM image is provisioned on the service provider's platform through the secure provisioning protocol described in Chapter 2 that ensures the TVEM is loaded on the correct host without modification.

The service provider needs to accommodate the deployment of TVEMs by providing a TVEM Manager that is accessible by the customer for interfacing with the host TPM. The TPM access can be provided by delegating the customer certain privileged operations to configure the TVEM and interface with the TPM.

5.5 TVEM System Architecture

The TVEM is a software trust module for providing trust services to a VM or virtual environment in an IaaS cloud computing environment. The TVEM is the protection module and root of trust for a virtual environment that is in a remote location and the virtual environment has the ability to migrate to other platforms; therefore, it is not possible to implement a TVEM in hardware. Thus, a software implementation is the only solution for the TVEM.

Because the TVEM is a cryptographic module and data confidentiality is of utmost importance; we chose for the TVEM to be a self-contained virtual appliance that is implemented in a helper VM or stub domain.

Figure 5.2 shows the configuration of a *Host Platform (HP)* with multiple virtual environments that require a TVEM. The virtual environment may be an entire virtualized OS that supports many applications or a special purpose virtual environment that performs a single application. The TVEM lies between the hypervisor and its associated VM. The hypervisor must be aware of TVEMs and provide support via a TVEM manager. The TVEM manager provides mediation for TPM services from each TVEM and other processes that require TPM services. The host platform must provide the TVEM manager and allow access for TVEMs.

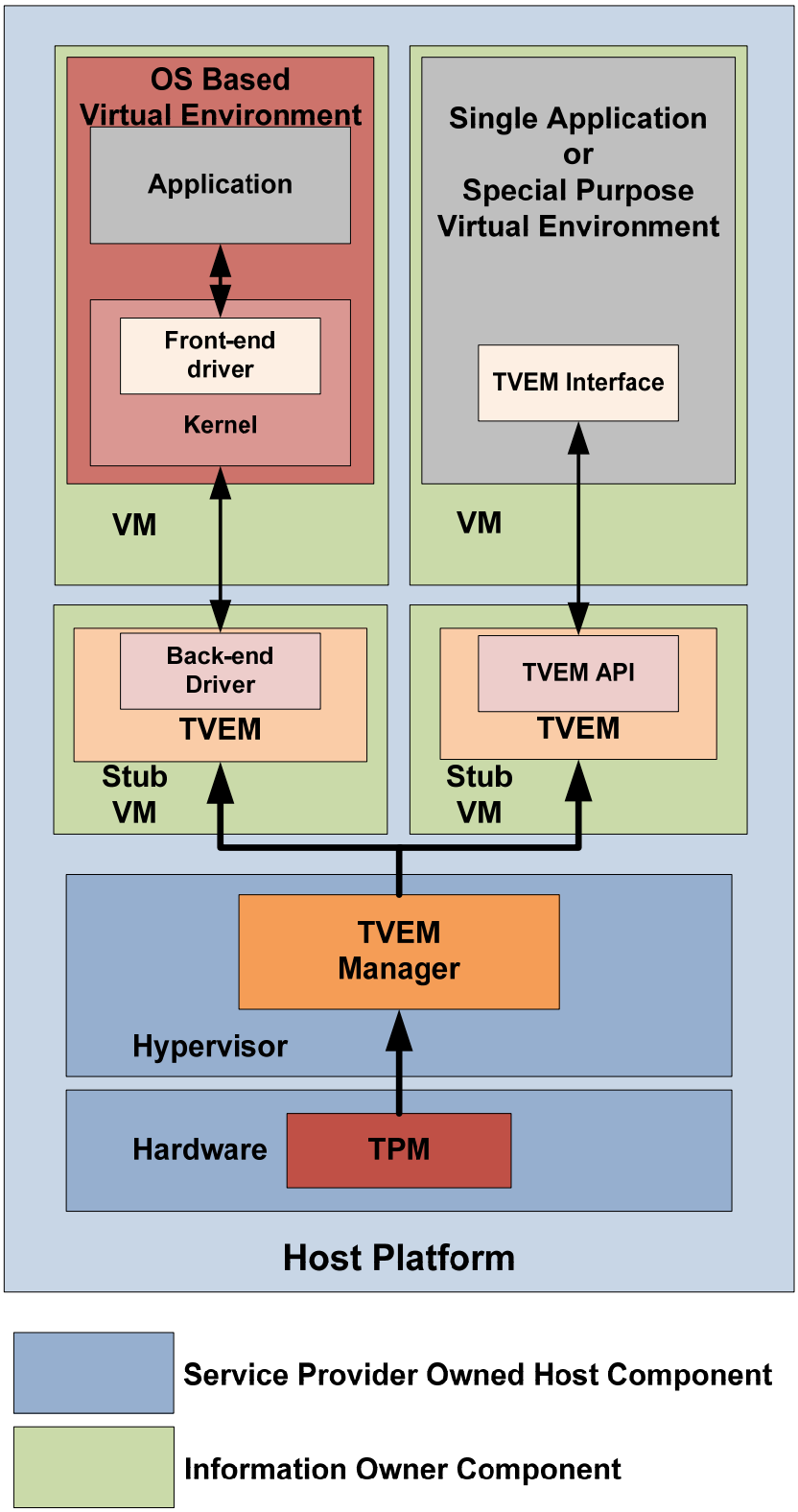

Figure 5.2 – The host platform has a single TPM, hypervisor, and TVEM manager supporting multiple virtual environments each in its own VM with a tightly coupled TVEM in a stub domain.

The host platform's TPM is used as the RTS to secure the TVEM's private information on the HP. A transitive trust chain is built from the TPM through the hypervisor and TVEM manager to the TVEM ensuring trust in the TVEM is rooted in the hardware trust of the platform.

To ensure that data within the TVEM remains private, several security mechanisms need to be in place on the host platform. An isolating hypervisor such as sHype [32] can be used along with Intel VT-d and TXT to provide additional protection through hardware isolation.

5.5.1 Virtual Trust Network

The VTN is the control plane that manages trust within a PVI. The VTN is a private virtual network on an encrypted link for communication between the TF, host platform, TVEM managers, and the TVEMs. The VTN is used for configuring and managing TVEMs in a PVI. VTN traffic is restricted to communications between TVEMs owned by the PVI, host TVEM managers, and the TF.

Figure 5.3 show the structure of the VTN. The TF is the controller for the VTN. The TF manufactures TVEM images based on requirements for the VTN, generates a TEK and inserts it into the TVEM images, and provisions the TVEM on a host platform. The TF is responsible for VTN management and monitors all TVEMs in the PVI. A single TF can manage multiple VTNs each with multiple TVEMs; therefore, key management is a critical function of the TF.

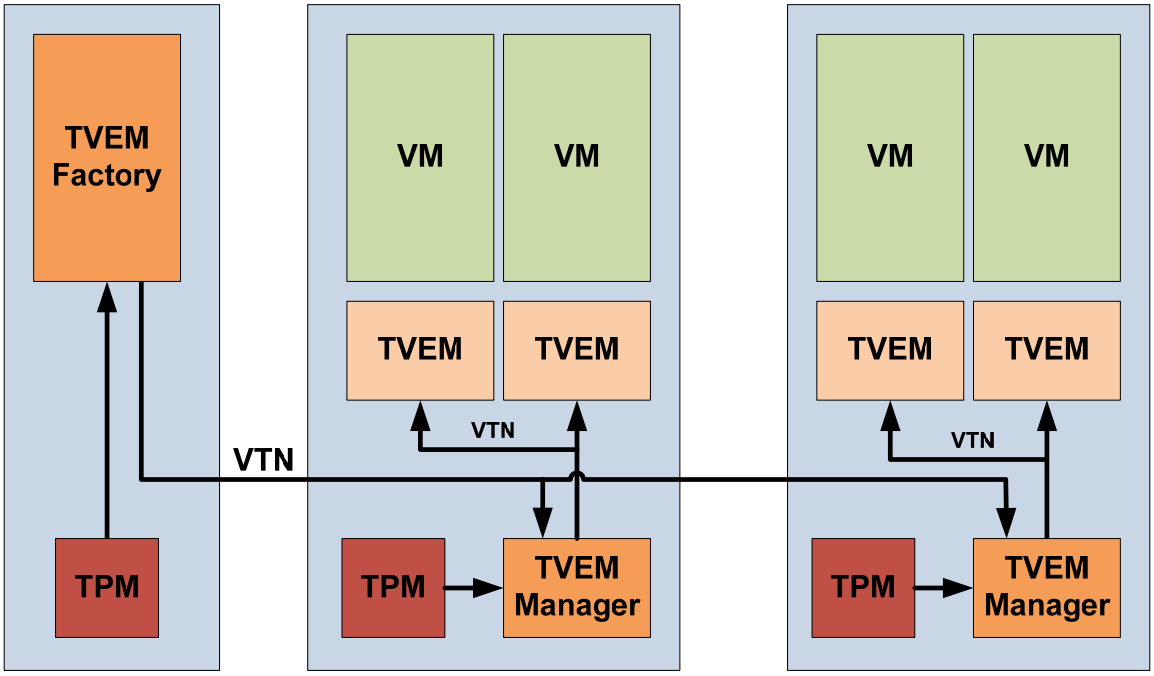

Figure 5.3 – TVEM system showing TVEMs on two host platforms with TPMs and TVEM Managers, the Virtual Trust Network, and TVEM Factory on a TF Platform with a TPM.

TVEM Manager

The TVEM manager is the hub for the VTN on the host platform. The TVEM manager is responsible for routing VTN communications between the TF and TVEMs and for arbitrating all access between TVEMs on the platform and the host TPM. The TF communicates with host platforms through the TVEM Managers.

Each host must have a TVEM manager that provides an interface to the host TPM. The TVEM manager must be placed in the hypervisor on the host platform so that it may have access to the host TPM and provide provisioning functions required to support TVEMs. Host TPM access is required for reading the platform PCRs and SRK so that TVEMs may be bound to the host TPM.

Importantly, the TVEM manager is part of the host platform, it is owned by the service provider and is not part of the information owner's domain (see Figure 5.2);

therefore, the TVEM manager must be a trusted component and part of the measured configuration on the host platform.

A TVEM manager on a host platform may support multiple VTNs from the same information owner or VTNs from other information owners simultaneously. The TVEM manager must be able to isolate communication from multiple VTNs and allow access only to TVEMs associated with the proper VTN.

5.5.2 Trusted Environment Key

The *Trusted Environment Key (TEK)* is critical in providing security and trust for the TVEM. It prevents cloning of the TVEM and protects the contents of the TVEM from the platform owner and other processes on the platform. The TEK is a unique key generated for a TVEM. The TEK is the TVEM's endorsement key and serves the same purpose as a TPM's EK. The TEK is generated from the VTN root certificate in the *TVEM Factory (TF)*.

Figure 5.4 shows a diagram of the TEK generation. The TF generates a VTN certificate, which is the parent for the all TVEM TEKs in a VTN. The VTN certificate is defined as:

$$VTN_{TF} = SRK_{TF}\{VTN\}$$

The TF's TPM generates a unique VTN certificate for each VTN protecting it with the TPM's SRK. The VTN key is a *Migratable Storage Key (MSK)* that can be transferred to other TPMs. The TEK is then generated as a child of the VTN key and is thus a MSK as well. Both keys are transferred to the *Host Platform's (HP)* through the key migration process. The TEK is defined as:

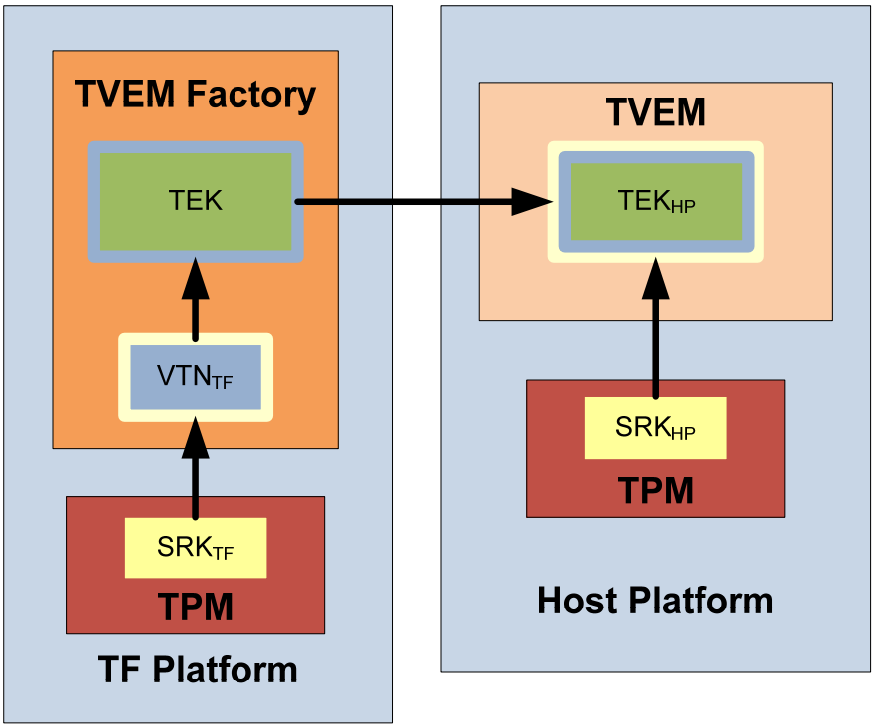

Figure 5.4 – A VTN certificate is generated and wrapped by the TVEM Factory (TF) Platform SRK. The TF TPM then generates the TEK as a child of the VTN key, places the TEK in the TVEM, and migrates the TEK to the Host Platform (HP).

$$TEK_{\mathrm{HP}} = MSK_{HP}\{VTN_{TF}\{TEK\}\}$$

The TF migrates the TEK and VTN key to the HP binding the TEK and VTN to the HP's SRK. The TEK is stored in the TVEM, which is a protected partition on the HP, thus it can be unencrypted only by the TVEM using the HP's TPM. The TEK will be protected in the TVEM and exposed on the HP only when requested by the TVEM for necessary operations.

The TEK is effectively "dual rooted" in both the host platform SRK and VTN root. This means that the TVEM cannot be cloned by copying its contents to another machine because the TEK is locked by the host's TPM. The TF maintains the VTN key and TEK

root certificates and can revoke the VTN key or TEK at any time effectively removing privileges from TVEMs on rogue hosts.

TVEM migration is achieved by performing a TEK key migration from the current host platform to the new target platform. The TVEM migration is not direct to the target platform; it must go through the TF and verify that the target environment has the same level or greater trustworthiness than does the current host. Once the trustworthiness is determined, the TEK can be migrated to the new host by rewrapping the TEK with the new host's SRK. The TVEM and associated virtual machine can then be migrated to the new host without losing any information sealed by the TVEM.

Key Hierarchy and Management

The highest level key in a VTN is the master VTN key. All TEKs in the VTN are rooted and secured with the master VTN key. A master VTN key and certificate is generated for each VTN that the factory is responsible for managing. The VTN key is protected and stored with the TF platform's physical TPM SRK. The VTN root certificate along with the host platform SRK are used to generate all TEKs in the VTN.

The TF becomes the root authority for all VTNs under it auspices. TVEMs can be verified by checking their TEK certificates with the TF VTN authority. Since the TF is the root authority, it must maintain a key list of valid and revoked keys for each VTN. Once the VTN is deactivated, the VTN key is destroyed and all keys for that VTN must be revoked. A record of the revoked TEK should be kept to ensure that it will never be used again.

5.5.3 The TVEM Lifecycle

The TVEM may go through six separate life cycle stages: creation, provision, operation, migration, destruction, and termination. A single TVEM is created and terminated once, but may be provisioned, operational, and destroyed multiple times. A TVEM may or may not go through the migration stage as migration is operationally dependent.

TVEMs are created as images on the TF that are built from the modular components with a configuration specific to the target host platform. Generally, all TVEMs for a single VTN will have the same configuration. The TF needs to generate a new and unique TEK for the TVEM and insert the TEK into the image.

After the TVEM image is built on the TF, it is provisioned on a host platform. Provisioning the TVEM on the host platform is not a trivial matter as there are several considerations that need to be understood. First, the target host platform must be inspected to determine whether the target is trustworthy for the application intended. The inspection includes determining whether the system is configured correctly, such as if the proper hypervisor is loaded with the TVEM manager. Once the configuration is determined, the TVEM must be provisioned securely on the platform. The TF provisions the TVEM on the target host machine using the LSP protocol, adapted to support the TVEM. After the TVEM is created and launched on the host platform, the virtual environment must then be provisioned on the platform and bound to the TVEM.

The stage where the TVEM will spend most of its lifetime is the operation stage. While in the operating stage, the TVEM provides security services for the virtual environment and performs routine communication with the TF. The biggest concern in

the operating phase is ensuring that the TVEM is not attacked and detecting the attack if it occurs.

There are situations where the virtual environment may need to be migrated to a new platform. In the event of a virtual environment migration, the TVEM must be migrated as well. The TVEM uses the LSM protocol for transferring the state and TEKs between two platforms. The TVEM must remain within the same VTN during a migration. Direct migration to another VTN is not supported, as the TEK would have to be changed to another master VTN key, which will invalidate all data encrypted with the TEK.

Just as important as secure provisioning is secure TVEM destruction. Local copies of TVEMs must be destroyed securely upon termination of the virtual environment or migration to protect the keys and data stored in the non-volatile memory. If a TVEM's memory is reused on the platform, the memory could be inspected to determine the TEK, SRK, and other sensitive information, which can be used to gain access to the virtual environment's information. It is especially important to ensure the keys are not compromised. If the virtual environment is reused or migrated and still operational, then the information protected by the TVEM's keys is still valid. Upon termination of the TVEM, all memory used by the TVEM must be securely wiped to ensure that no keys or sensitive information remains on the platform.

The TF terminates the TVEM instance only after it is no longer needed. If a virtual environment is suspended, and the image stored, then the associated TVEM instance must be kept by the TF to ensure that the encrypted information is able to be accessed. While the TVEM is suspended, the non-volatile memory of the TVEM must be maintained by the TF. Once the instance is no longer needed by the virtual environment,

its associated non-volatile memory is purged and its TEK is revoked by the TF ensuring that the TVEM and the information it protects can no longer be used.

5.6 TVEM Design

The TVEM design is a set of functions grouped into five categories: legacy TPM Functions, TVEM specific functions, storage functions, virtual environment interface, and user interface. Figure 5.5 shows a block diagram of the high-level TVEM functions. In addition to the functional features of the TVEM, security features are also required to ensure that the TVEM is not compromised.

5.6.1 Legacy TPM Functions

The TVEM implements the following TPM functions per the TPM specification [15]: PCRs (as shadow registers), AIK, SRK, public key engine, secure hash, monotonic counter, and RNG.

The PCRs from the TPM are shadowed so that the virtual environment has the ability to read the configuration of the host platform. The virtual environment cannot modify the PCRs because it does not own the TPM. This is an important distinction for cloud computing environments. PCRs are written and modified by the hypervisor and host operating system when virtual environments are launched. The configuration of the virtual environment is maintained separately in the *Virtual Environment Configuration Registers (VECRs)*.

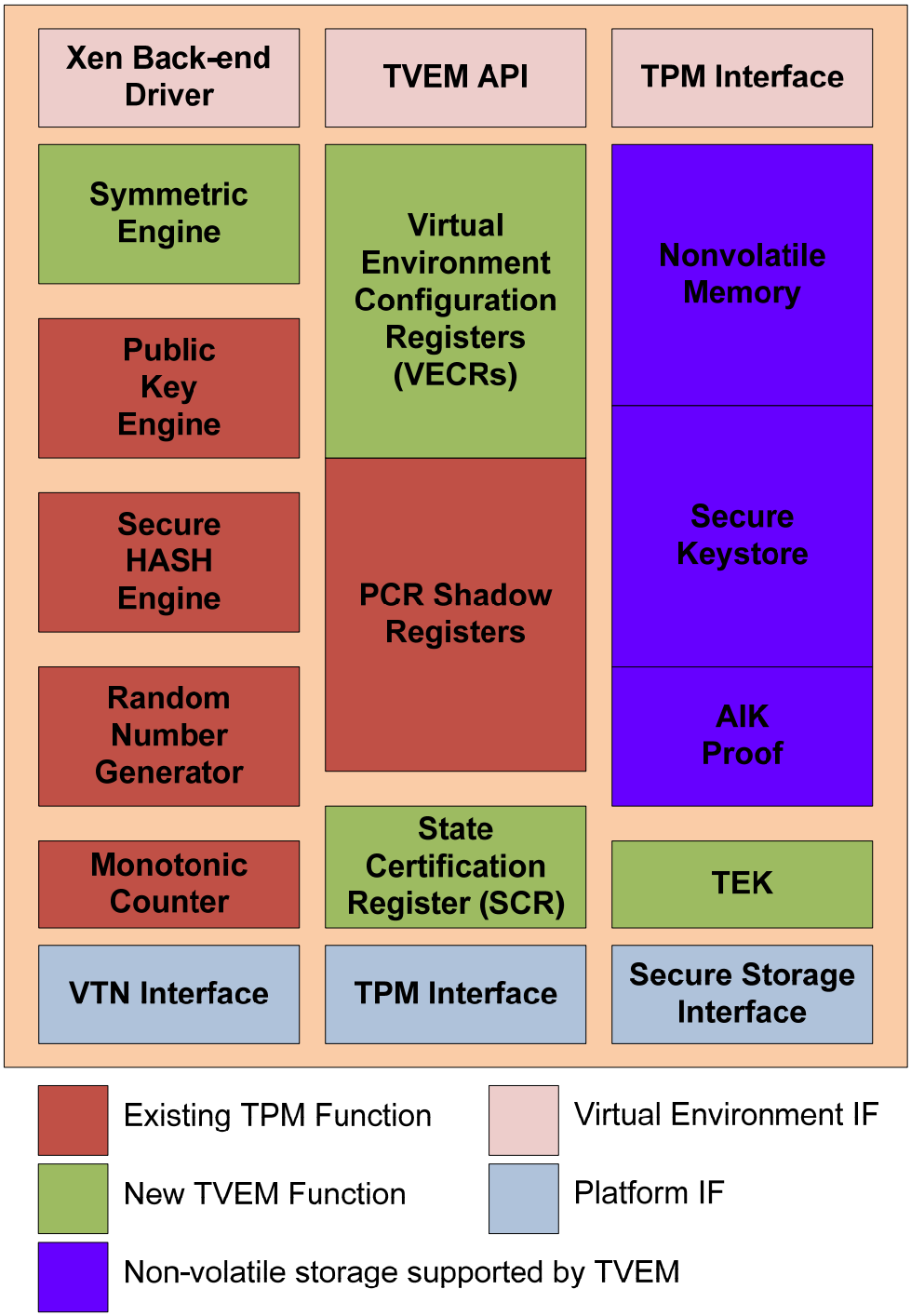

Figure 5.5 – The TVEM functional block diagram shows legacy TPM functions supported, the new TVEM functions, non-volatile storage, and host and platform interfaces.

The RNG is an important construct for the virtual environment. Since virtual environments have limited ability to generate entropy, an external source for entropy is required. However, the TVEM itself is a virtual environment; therefore, it must use also use an external source for entropy. The RNG for the TVEM needs to use the RNG on the host platform TPM to obtain the entropy required to generate encryption keys and nonces. The TPM's hardware based RNG generates sufficient entropy which the TVEM can use for the virtual environment. The latest Linux kernels (2.6.29 and above) support the TPM RNG, which can interface to a TVEM RNG in a VM.

The TPM hash function is implemented in the TVEM as a software module. Since the SHA-1 hash functions will be phased out in 2010 [58], another hash algorithm such as SHA-256 can be easily substituted.

The *Attestation Identity Key (AIK)* and proof is used to provide evidence that an entity is a valid TPM. The AIK is generated in conjunction with a trusted third party privacy authority in a manner that verifying the AIK established that the TPM is valid without revealing which specific TPM is validated. The AIK process can easily be converted to provide proof of a valid TVEM by simply adding the VTN factory's root certificate to the privacy authority's list.

There are also several TPM functions which may not be required at all. These include physical presence, physical maintenance commands (*e.g.*, field upgrade and set redirection) and other functions that are not needed for a software implementation.

5.6.2 New TVEM Functions

Several new TVEM functions have been created to enhance the capability of the TVEM for virtual environments. These new functions include VECRs, the TEK, the *State Certification Register (SCR),* and a symmetric encryption engine.

VECRs are equivalent to PCRs, and store configuration information of the virtual environment. There are 28 256 bit VECRs to support SHA-1; however, the VECRs can be configured up to 512 bits to support SHA-256. The VECRs are used to for the virtual environment exactly as the PCR's for the physical platform. When a virtual environment is configured, the VECRs store configuration information about the virtual environment. The PCRs from the TPM are used; however, they are shadowed and only used for the purpose of determining the configuration of the host platform. The two sets of registers provide the ability to obtain configuration information about the platform and maintain a fine-grain detail about the configuration of the virtual environment. This enhanced view of both environments gives the virtual environment owner the ability to understand the security posture the cloud.

The TEK is the endorsement key for the TVEM. The TEK functions exactly as the EK on a TPM, providing the master key for all TVEM functions and rooting all other keys. The TEK's dual trust root is essential to establishing trust in a virtual environment on a machine that is not owned by the information owner. Since the key is rooted both in the VTN and host platform, the key can be revoked by a change in configuration on either side thus invalidating the trust in the virtual environment.

A new register called the state certification register is used to verify that the state of the TVEM has not been modified or rolled back to a previous state. Section 5.6.6 describes the SCR and rollback protection in detail.

Another new feature of the TVEM compared to the VTPM is the addition of a symmetric data encryption engine. Cost is not a limiting factor and symmetric engines are very efficient in software; therefore, we can add an encryption engine and offload encryption tasks for small virtual environments. Providing an encryption engine allows smaller, hardened environments instead of bloated operating systems. Additionally, export controls are not a major concern with a TVEM. If a TVEM is to be exported outside of its originating country, the encryption engine can be easily removed or swapped with an engine without export controls. Finally, the encryption engine can provide enhanced security by ensuring that correct and verified implementations are used.

5.6.3 Non-volatile Storage

To support operation of the TVEM across multiple sessions and migration, the information placed in the non-volatile storage of the TVEM must be persistent. To make the non-volatile memory persistent, the contents of the memory are backed up to the TF where they are stored until the TF is terminated. Each TVEM's non-volatile memory image must be maintained until the TVEM is terminated.

To maintain the image of each TVEM's non-volatile memory, every time the non-volatile memory in the TVEM is updated the TF updates its backup image. The update is done by sending a message with the contents to the TF over the host platform secure storage interface. When the TF receives the message, it updates its backup image for the specified TVEM instance accordingly.

An additional benefit of keeping the backup image of the non-volatile memory is the ability to verify the local TVEM image. The TVEM may send the full contents of non-volatile memory to the TF at any point to request a verification of the content. The TF can compare the sent image with its backup image and verify that both are identical. If the content of the TVEM's memory were tampered, the comparison would be different.

5.6.4 Virtual Environment Interfaces

The biggest difference between a VTPM and a TVEM is the virtual environment interfaces. TVEM virtual environment interfaces on the VM side of the TVEM are designed to accommodate the many types of virtual environments that may need TVEM services. Not all environments will have a full TSS or cryptographic API; therefore, an API on the TVEM provides the cryptographic services for the virtual environment.

The TVEM has three unique interfaces to the virtual environment as shown in Figure 5.5: a Xen back-end driver, a TVEM API, and a standard TPM interface for compatibility with TPM 1.2. Each virtual environment that requires a TVEM service has the option to choose the interface it will use. A virtual environment may use multiple interfaces if desired. For example, the operating system may use the Xen driver in the kernel and an application may use the TVEM API.

TPM Interface

The TPM interface will work with any program written to support a TPM 1.2. This interface provides backward compatibility for the TVEM where applications are expecting a TPM or VTPM. The TVEM can appear to be a valid TPM and operate as a TPM replacement.

Note that when accessing a TVEM as a TPM, the VECRs are accessed instead of the PCRs. The VECRs represent the state of the virtual environment, which is the context that the guest is operating.

TVEM API

The TVEM API provides applications a direct interface to the TVEM through a set of function calls. This interface allows operating systems and applications without a full TSS or cryptographic library the ability to use the TVEM easily.

The API provides access to extended TVEM functions including the symmetric encryption engine and PCR shadow registers. The interface can also be extended to connect with additional cryptographic hardware such as smartcards and biometrics as well.

Xen Back-end Driver

The Xen back-end driver will interface directly to a Xen kernel front-end driver [59]. This capability enables any virtual environment running on a Xen hypervisor the ability to interface to TVEM with the simple addition of the front-end driver to the virtual environment. The Xen interface is an extension of the API and provides a seamless interface for the virtual environment.

5.6.5 Host Platform Interfaces

The host platform interfaces are to the host platform side of the TVEM. The TVEM interfaces to the host platform differently than a VTPM. The TVEM uses the TPM and host platform as a service to provide functions required for secure operation. The TVEM uses the host interfaces for host TPM services, communicating with the VTN, and for storing non-volatile information.

Host TPM Interface

The TVEM's host TPM interface communicates with the host TPM via the TVEM manager. The TVEM manager is responsible for arbitrating access to the TPM. All requests that require TPM access use this interface including the reading the TPM's PCRs, storing and retrieving keys to the TPM, and accessing the RNG.

VTN Interface

The VTN interface is used to manage keys and communicate with the TF over the encrypted VTN. One of the primary functions of the VTN interface is to implement the rollback protection and heartbeat security features (see Section 5.6.6). Section 5.5.1 discusses the VTN in depth.

Secure Storage Interface

The secure storage interface is a VTN interface used by the TVEM to store non-volatile memory securely. The secure storage interface uses the VTN to send all non-volatile writes to the TF for backup storage. On provisioning of a TVEM, the secure storage interface is used to populate the non-volatile storage areas on the TVEM from the backup image. The secure storage interface is also used to verify the contents of the non-volatile memory with the backup on the TF.

5.6.6 Security Features

Protecting the TVEM's content and state is critical to maintaining trust in the TVEM. There are three primary features to protect the private information and enhance the security of TVEM: virtual environment isolation, rollback detection, and TVEM heartbeat.

Isolation

To ensure that no other process on the platform can read or modify memory of the TVEM, the TVEM must be isolated from the other processes on the platform. Three mechanisms are available to ensure that a TVEM is isolated from other processes on the platform: secure hypervisors, hardware virtualization, and trusted execution technology. These mechanisms are layered and work to prevent virtualization containment attacks and protect the VM in which the TVEM is operating.

The secure hypervisor is the first layer of defense which sets the VM boundaries and access privileges for all VMs on the system. The second layer is Intel VT-d hardware support for VM isolation and containment detection. The hardware features of VT-d detect and prevent any unauthorized read or write to not only the memory of the TVEM, but also I/O space owned by the VM. The final layer is Intel TXT, which can further restrict processors in a multi-processor system from accessing the TVEM preventing a rogue processor from hijacking the execution space of the TVEM. With these layered security technologies based in software and hardware, software-based attacks against the TVEM isolation become very difficult.

Rollback Detection

One of the most important security features to maintaining trust of the TVEM is rollback and state modification prevention. Rollback is when the state of the TVEM is manipulated backwards to a previous state to break encryption or weaken keys. Protecting rollback in software only is difficult; however, there are features of the system architecture that can be used for rollback and state modification detection, namely the VTN.

To protect against rollback, the monotonic counter function of the TVEM is used to determine state progression. Every time the monotonic counter is increased, the counter and all TVEMs are hashed to create an HMAC and sent along with the HMAC to the TVEM factory. The TVEM factory maintains the current count for the TVEM and verifies its states with the previous messages. If the monotonic counter is incremented forward by one, the factory signs the HMAC and sends it back to the TVEM. The TVEM stores the signed message from the factory in the SCR. If the state is ever rolled back, the counter detection will detect that the count is incorrect and flag an error. Anyone wishing to verify the state can simply request a state verification from the TVEM and verify it with the factory.

Heartbeat

A periodic heartbeat message sent from the TVEM ensures that the TVEM is operating as intended. The heartbeat is a simple message sent at a regular interval indicating that the TVEM is functioning. The heartbeat should be authenticated and include a message that proves that the TVEM is operating.

The unexpected absence of a heartbeat is cause for alarm as this is an indication of several undesirable events. The absence of the heartbeat could indicate benign system problems such as a power or communication outage, to more serious issues such as attacks against the TVEM. Attacks that can be detected by the heartbeat absence include denial of service attacks or malicious suspension, which could indicate more serious attacks against the TVEM such as memory inspection and modification attacks.

5.7 Discussion

The TVEM provides many advantages over a VTPM in a cloud computing environment. The management of TVEM from the TVEM factory provides the ability to control and monitor TVEMs in a VTN and provides enhanced situational awareness to the information owner. The TVEM also provides system designers and information owners support for everything from simple single purpose applications to full operating systems. The virtual environment specific functions enable ease of use, and the modular design enables flexibility for deployment. The TVEM's dual rooted keys provide cloud environments security and trust that is separated from the host platform.

TVEM configurability is another advantage over VTPMs. By allowing information owners to customize their protection requirements, they have flexibility to use cloud computing services that were previously unavailable.

5.7.1 Security

TVEMs provide strong cryptographic support for securing a virtual environment on a cloud host platform. The unique dual rooted key structure provides flexibility to maintain trust in the virtual environment and allows information owners to control the confidentiality of their data on the host platforms.

A TVEM can be trusted to report host and virtual environment configurations securely as long as it is operating on a platform that is trusted by the information owner to provide the secure hypervisor, VT-d and TXT mechanisms that ensure isolation and protection for the TVEM. Once the TVEM is launched, it will report to the TF with the heartbeat and rollback protection mechanisms. These messages are verified by the TF

ensuring proper operation of the TVEM. The TVEM is constantly monitored by the TF through the messages ensuring that any corruption of the TVEM is detected.

TVEM improves security in our three example applications by ensuring that the environments are executing on a trustworthy platform, ensuring the environments are correctly configured, providing trusted storage for keys and other sensitive information, and providing a high entropy source for random number generation.

For the virtual web server, the TVEM provides secure storage of server certificates. As the RTS, the TVEM protects the server SSL and EV certificates by encrypting the keys with a unique SRK and storing them in persistent non-volatile memory. For stronger protection, the TVEM can bind the keys to the configuration of the host platform and/or virtual environment.

The cloud datacenter uses the TVEM to verify configuration information about the virtual servers of the datacenter. Through the PCR shadow registers, the configuration of the host platform can be determined. The TVEM provides security in the cloud datacenter by protecting private information with the SRK. The dual rooted TEK allows the information owner to control access to the information protect by the TVEM and revoke the TEK if necessary to protect the information.

The CVD uses the TVEM to verify its configuration on a remote machine through the VECR. The shadow PCRs allows the CVD owner to query the configuration of the user's computer and decide on the trustworthiness of the machine. Through the TVEM secure storage, the CVD can encrypt and store the network access keys, user identification keys, and other information the desktop owner wishes to be protected.

It is important to remember that TVEMs are not designed to defend against hardware based attack. TVEMs are software devices and any attacker with access to certain ports (*e.g.*, IEEE 1394 FireWire), hardware monitoring devices, emulation and debug equipment, or memory inspection equipment can circumvent the TVEM's security. Since hardware attacks cannot be detected or defended against, physical security of cloud datacenters is of utmost importance.

Another type of attack that TVEM cannot defend against is a dishonest host or service provider. The information owner is at the mercy of the service provider to provide the services agreed upon in a service agreement. If the host platform lies and falsely reports its attestation values to the TF, the TF has no basis for challenging the integrity of the platform. To prevent the dishonest host, social trust must be used as it is likely that once it is detected that the hosts is falsely reporting, word of the dishonesty will be spread through the community and the service provider's reputation will diminish.

5.7.2 TVEM Requirements Examination

The TVEM design meets the stated functional requirements of Section 5.4.2 as discussed below:

R1 Confidentiality of data is achieved through hypervisor, VT-d and TXT isolation.

R2 Key and NVRAM data are stored persistently on the TVEM factory via the VTN.

R3 Flexible cryptographic algorithms are provided via the modular design of the TVEM.

R4 The TVEM maintains the chain of trust from the CRTM of the host platform, through the TPM's RTR and RTS, to the TVEM. The TVEM then becomes the RTM and RTS for the virtual environment.

R5 The host TPM is used as a RNG providing sufficient entropy to the virtual environment.

R6 TVEM cloning is prevented through the TEK generation process. By using the dual rooted TEK, any attempt to clone the TVEM will not be able to duplicate the TEK without detection.

R7 Rollback protection accomplished via the SCR and state verification with the TVEM factory.

R8 Migration of the TVEM to another host platform while maintaining information sealed by the TVEM is enabled through the TEK migration

By meeting the above stated requirements, TVEM provides security and trustworthiness for the virtual environment.

5.7.3 Challenges to Deployment

One of the issues that must be considered is the openness of the vendor. Without the service provider's support for TVEM protocols, allowing visibility into the host configuration settings, and providing information about the host operating environment, implementing a TVEM on the provider's system would be impossible.

Cloud providers that use proprietary or closed systems will need to find ways to support openness and transparency to enable TVEMs to be effective in increasing the security of cloud computing. Providing their customers the ability to control the security and privacy of information in the cloud is a win-win for both the provider and customer.

5.7.4 Cost and Performance

This section examines the performance and cost of TVEM. There are three main performance areas to consider: provisioning, operation, and key management.

The largest impact during provisioning is transferring the VM image from the factory to the host. A TVEM is not a large device and should be an order of magnitude smaller that an operating system; but the cost could still be significant if large numbers of TVEMs are routinely provisioned, migrated, and destroyed.

In the operation phase, TVEM context switching and communication have the largest impact on performance. Overhead from context switching between the virtual environments slows down overall processing as the processor has to switch from one virtual environment to another; however, this impact is not as drastic as the communication overhead. If a TVEM is being heavily used, communication overhead between the TVEM and TF could be very high. While the individual communications are not large, a large number of TVEMs could overwhelm a TF and lead to high bandwidth costs. Load balancing the TFs could solve this problem.

Key management is a major cost of using TVEM. Unlike TPM's, the operator of TVEMs must create and maintain certificates for each VTN and TVEM in use. Additionally, each key must be permanently kept in order to prevent reuse attacks, so a database of all used keys is required to ensure that a key is never duplicated. This database could get quite large over time.

Chapter 6

Conclusion

No more training do you require.

Already know you, that which you need.

Yoda, *Star Wars Episode VI: Return of the Jedi*

This work represents a new paradigm of information protection and security in cloud computing. We examined and defined a new trust model for cloud computing and addressed the core security challenge of utility cloud computing with multi-tenancy. We have shown that using trusted computing technologies in the cloud computing environment can benefit both operators and clients.

Our solution implements the four tenets of cloud security providing a secure framework and increasing the security posture for IaaS computing:

1. **Tenet:** Provide a trusted fabric on which to build utility clouds.

   Solution: A trusted foundation is provided with the TCFP, which can be further enhanced for cloud computing by adding the TVEM to the TCFP.

2. **Tenet:** Provide a secure management facility for utility clouds that also serves as a policy decision point and root authority for utility clouds.

   Solution: The PVI and TVEM factories are the management facilities.

3. **Tenet:** Provide a measurement mechanism to validate the security of the fabric prior to provisioning of utility clouds.

   Solution: LoBot provides secure measurement of the fabric via the fabric premeasurement stage of the LSP protocol.

4. **Tenet:** Provide secure methods for provisioning, migration, shutdown and destruction of virtual machines in utility clouds.

   Solution: LoBot provides for secure provisioning and migration via the LSP and LSM protocol. Secure shutdown and destruction of virtual machines still require additional research.

Outline: This chapter is organized as follows. Section 6.1 overviews the benefit and need of PVI. Section 6.2 discusses the need for LoBot and the LSP and LSM protocols. Section 6.3 presents the contributions of TVEM and the benefit provided. Section 6.4 discusses future areas of research in cloud computing security. Section 6.5 provides final thoughts on the research.

6.1 PVI

PVI is a new model for securing and managing cloud computing services based on a synergistic relationship between the vendor and customer of cloud services. This relationship provides an increased security posture while allowing both parties to set security controls required to protect the infrastructure and data within the cloud and virtual datacenter. Private Virtual Infrastructure increases security and privacy in the cloud by isolating the virtual datacenter from the greater cloud.

Cloud computing service providers need to enable a transparent view of their infrastructure so their customers can understand the security posture and threats to the

system. This capability will give the vendor a competitive advantage as a secure system provider over vendors who choose to obscure their infrastructure inner workings to protect proprietary technology. Cooperation between vendor and customer will result in increased security while lowering the overall cost of ownership for IT infrastructure.

Security is the responsibility of all parties involved in IaaS cloud computing. Vendors are responsible to provide a secure fabric. Information owners are responsible to protect their data. PVI provides information owners the flexibility to manage their own data while realizing the cost benefits of cloud computing.

6.2 LoBot

We have shown how to provision and migrate virtual machines securely in a cloud computing environment via the LoBot Secure Provisioning and LoBot Secure Migration protocols. The architecture and protocols implement our new security concepts for securing virtual datacenters in the presence of threats. Information owners can verify configuration and security settings of target platforms by probing the fabric with a LoBot to obtain the destination environment's properties increasing the assurance that their VMs will not be compromised. The TPM and VTPM provide the root of trust and security needed by the protocol and Intel TXT provides the late launch capability needed to securely launch a VM.

These are the first protocols that we are aware of for provisioning and migrating live virtual machines that first tests the target environment to ensure it meets policy and configuration requirements before the actual environment is relocated. Additionally, the protocols provide assurance that the transfer is not tampered with during the process. The

protocols are a preemptive measure to ensure that critical data are not exposed to malicious actors without a means for recovery.

6.3 TVEM

TVEM is a new and unique concept for rooting trust in the cloud. The TVEM solves the problem of rooting trust in IaaS cloud computing where a service provider owns the platform on which an information owner's virtual environment is operating. TVEM enhances security by allowing for trust in the virtual environment that is distinct and separate from the hosting platform. The TVEM protects information and conveys ownership in the cloud through the TEK generation process, which creates a dual rooted trust for the virtual environment. This dual rooted trust is necessary to accommodate the unique relationships that occur in cloud computing.

The TVEM gives information owners control of their sensitive and private data in the cloud by providing assurance that their environments are correctly configured and data are kept confidential. The TVEM provides management control of trust through the centralized TVEM factory control facility, key hierarchy, and modular configurable architecture.

This dissertation introduces the high level system architecture and design concepts of a necessarily somewhat complex TVEM system. The definitions of the TVEM, TVN, and TEK provided here are strong building blocks to continue developing the details of the sub-modules and components. To ensure the TVEM meets the needs of the cloud computing users, the TVEM system should go through a formal specification development cycle with representatives from many stakeholders, including providers, customers, trusted computing experts, and cloud computing researchers. With proper

vetting and industry support, the TVEM can be a valuable security component for IaaS cloud computing, enabling a higher adoption rate and a more secure cloud.

Although an industry standard VTPM implementation upon which we can further build and refine the LSP and LSM protocols will soon be a reality, the additional extensions added by TVEM would improve trust in a cloud computing environment and provide for easier integration of different types of virtual environments.

6.4 Further Work

We have shown how to build trust into the cloud computing infrastructure, but much work remains to have a viable solution that can be implemented on the cloud scale. The LSP and LSM protocol needs to be integrated into the Xen management infrastructure and the PVI factory needs to be fully designed and implemented. Beyond the basic features of LoBot, additional functionality such as continuous monitoring and sensor net like capabilities for detecting malicious activities would be beneficial to add.

A number of details still remain to be worked out regarding premeasurement. Examining and understanding which properties required for premeasurement and which properties are effective in determining the configuration of a platform are necessary.

Further development of the TVEM to include addition of new algorithms and interfaces would help enhance the adoption of TVEM as a standard trusted computing component.

Other areas of research that would be interesting include how to apply the techniques and research described in this dissertation to other models of cloud computing including the *Platform as a Service* and *Software as a Service* models. Another area of exploration

is determining if the model is applicable to large scale computation clusters such as Hadoop [60] and Google BigTable [61] clouds.

6.5 Final Thoughts

Utility cloud computing has the potential to revolutionize the way companies purchase and use IT. With this new technology come new risks and threats that need to be thoroughly understood and analyzed. The opportunity exists to understand these risks and threats now and build in the security to circumvent the risks and threats early in development and adoption phase.

We have provided three solutions that will enable trust to be built into utility cloud computing. PVI, LoBot, and TVEM are a very powerful set of technologies that allow trusted virtual infrastructures in the cloud. They allow us to establish trust relationships, understand the surroundings, and protect information in the cloud in ways that were previously unavailable. Their usefulness extends beyond cloud computing to any environment where remote virtual machine provisioning and live migrations are required.

Cloudy, the future is.

Bibliography

[1] N. G. Carr, *The Big Switch: Rewiring the World, from Edison to Google,* New York: W.W. Norton & Company, 2008.

[2] J. Heiser, and M. Nicolett, *Accessing the Security Risks of Cloud Computing,* G00157782, Gartner, Inc., Stamford, CT, 2008.

[3] M. Armbrust, A. Fox, R. Griffith, *et al.*, *Above the Clouds: A Berkeley View of Cloud Computing*, University of California Berkeley, Berkeley, CA, 2009; http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf.

[4] P. Mell, and T. Grance, *The NIST Definition of Cloud Computing version 15*, National Institute of Standards and Technology, Gaithersburg, MD, 2009; http://csrc.nist.gov/groups/SNS/cloud-computing/.

[5] J. Leach. "The Rise of Service Oriented IT and the Birth of Infrastructure as a Service," March 20, 2008; http://advice.cio.com/jim_leach/ the_rise_of_service_oriented_it_and_the_birth_of_infrastructure_as_a_service.

[6] S. Campbell, and M. Jeronimo, *Applied Virtualization Technology*, Hillsboro, OR: Intel Press, 2006.

[7] P. Barham, B. Dragovic, K. Fraser, *et al.*, "Xen and the Art of Virtualization," *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, pp. 164-177, 2003.

[8] S. Berger, R. Cáceres, D. Pendarakis, *et al.*, "TVDc: Managing Security in the Trusted Virtual Datacenter," *ACM SIGOPS Operating Systems Review*, vol. 42, no. 1, pp. 40-47, January, 2008.

[9] D. Grawrock, *The Intel Safer Computing Initiative*, Hillsboro, OR: Intel Press, 2006.

[10] D. Nurmi, R. Wolski, C. Grzegorczyk, *et al.*, *Eucalyptus: A Technical Report on an Elastic Utility Computing Architecture Linking Your Programs to Useful Systems*, Technical Report 2008-10, University of California Santa Barbara Computer Science, Santa Barbara, CA, 2008.

[11] "Amazon Elastic Compute Cloud (Amazon EC2)," http://aws.amazon.com/ec2/.

[12] S. Pearson, *Trusted Computing Platforms: TCPA Technology in Context*, Upper Saddle River, NJ: Prentice Hall, 2003.

[13] *TCG Specification Architecture Overview*, vol. 1.4, Trusted Computing Group, 2007; http://www.trustedcomputinggroup.org/ resources/tcg_architecture_overview_version_14.

[14] D. Grawrock, *Dynamics of a Trusted Platform*, Hillsboro, OR: Intel Press, 2009.

[15] *TPM Specification Version 1.2 Revision 103*, Trusted Computing Group, 2007; http://www.trustedcomputinggroup.org/resources/tpm_main_specification.

[16] S. Berger, R. Cáceres, K. A. Goldman, *et al.*, "vTPM: Virtualizing the Trusted Platform Module," in 15th USENIX Security Symposium, Vancouver, B.C., 2006.

[17] J. A. Halderman, S. D. Schoen, N. Heninger, *et al.*, "Lest We Remember: Cold Boot Attacks on Encryption Keys," in 17th USENIX Security Symposium, San Jose, CA, 2008.

[18] *Trusted Computing Group (TCG) Personal Computer (PC) Specific Trusted Building Block (TBB) Protection Profile and TCG PC Specific TBB With Maintenance Protection Profile*, Common Criteria, 2004.

[19] D. Challener, K. Yoder, R. Catherman, *et al.*, *A Practical Guide to Trusted Computing*, Upper Saddle River, NJ: IBM Press, 2008.

[20] A. Cargile. "Hypervisor Security Concerns," http://thecoffeedesk.com/news/index.php/2009/12/01/hypervisor-security-concerns/.

[21] A. Stamos, A. Becherer, and N. Wilcox, “Cloud Computing Models and Vulnerabilities: Raining on the Trendy New Parade,” in BlackHat USA 2009, Las Vegas, NV, 2009.

[22] S. T. King, G. W. Dunlap, and P. M. Chen, “Debugging Operating Systems with Time-Traveling Virtual Machines,” in 2005 USENIX Annual Technical Conference Anaheim, CA, 2005.

[23] G. Hoglund, and J. Butler, *Rootkits*, Upper Saddle River, NJ: Addison-Wesley, 2006.

[24] S. T. King, P. M. Chen, Y.-M. Wang, *et al.*, “SubVirt: Implementing Malware with Virtual Machines,” in 2006 IEEE Symposium on Security and Privacy, Oakland, CA, 2006.

[25] J. Rutkowska, “Subverting Vista Kernel for Fun and Profit,” in BlackHat USA 2006, Las Vegas, NV, 2006.

[26] S. Gibson. "ARP Cache Poisoning," March 20, 2009; http://www.grc.com/nat/arp.htm.

[27] S. Gibson, and L. Laporte. "BailiWicked Domain Attack," *Security Now*, No. 155; http://www.grc.com/securitynow.htm.

[28] *Top Threats to Cloud Computing V1.0*, Cloud Security Alliance, 2010; http://cloudsecurityalliance.org/topthreats.html.

[29] M. Maybury, P. Chase, B. Chiekes, *et al.*, “Analysis and Detection of Malicious Insiders,” in 2005 International Conference on Intelligence Analysis, McClean, VA, 2005.

[30] J. Oberheide, E. Cooke, and F. Jahanian, “Empirical Exploitation of Live Virtual Machine Migration,” in BlackHat DC 2008, Washington, DC, 2008.

[31] F. J. Krautheim, “Private Virtual Infrastructure for Cloud Computing,” in Workshop on Hot Topics in Cloud Computing, San Diego, CA, 2009.

[32] R. Sailer, E. Valdez, T. Jaeger, *et al.*, *sHype: Secure Hypervisor Approach to Trusted Virtualized Systems*, RC23511, IBM, Yorktown Heights, NY, 2005; http://www.research.ibm.com/secure_systems_department/projects/hypervisor/

[33] T. Garfinkel, B. Pfaff, J. Chow, *et al.*, “Terra: A Virtual Machine-Based Platform for Trusted Computing,” in 19th ACM Symposium on Operating Systems Principles, Bolton Landing, NY, 2003.

[34] D. G. Murray, G. Milos, and S. Hand, “Improving Xen Security through Disaggregation,” in Fourth ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, Seattle, WA, 2008.

[35] N. Santos, K. P. Gummandi, and R. Rodrigues, “Towards Trusted Cloud Computing,” in Workshop on Hot Topics in Cloud Computing, San Diego, CA, 2009.

[36] L. M. Vaquero, L. Rodero-Merino, J. Caceres, *et al.*, “A Break in the Clouds: Towards a Cloud Definition,” *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, pp. 50-55, 2009.

[37] H. Tipton, and K. Henry, *Official (ISC)^2 Guide to the CISSP CBK*, Boca Raton, FL: Auberbach, 2007.

[38] R. Sailer, X. Zhang, T. Jaeger, *et al.*, "Design and Implementation of a TCG-based Integrity Measurement Architecture," in 13th USENIX Security Symposium, San Diego, CA, 2004.

[39] Y. Gasmi, A.-R. Sadeghi, P. Stewin, *et al.*, "Beyond Secure Channels," in 2007 ACM Workshop on Scalable Trusted Computing, Alexandria, VA, 2007.

[40] K. Goldman, R. Perez, and R. Sailer, "Linking Remote Attestation to Secure Tunnel Endpoints," in First ACM Workshop on Scalable Trusted Computing, Alexandria, VA, 2006.

[41] J. M. McCune, T. Jaeger, S. Berger, *et al.*, "Shamon: A System for Distributed Mandatory Access Control," in Annual Computer Security Application Conference, Miami Beach, FL, 2006.

[42] A.-R. Sadeghi, and C. Stüble, "Property-based Attestation for Computing Platforms: Caring about properties, not mechanisms," in 2004 Workshop on New Security Paradigms, Nova Scotia, Canada, 2004.

[43] L. Chen, R. Landfermann, H. Löhr, *et al.*, "A Protocol for Property-Based Attestation," in First ACM Workshop on Scalable Trusted Computing, Fairfax, VA, 2006.

[44] A. Bussani, J. L. Griffin, B. Jansen, *et al.*, *Trusted Virtual Domains: Secure Foundations for Business and IT Services*, RC23792, IBM, Yorktown Heights, NY, 2005.

[45] E. Gallery, A. Nagarajan, and V. Varadharajan, “A Property-Dependent Agent Transfer Protocol,” in Second International Conference on Trusted Computing, Oxford, 2009.

[46] F. J. Krautheim, D. S. Phatak, and A. T. Sherman, “Introducing the Trusted Virtual Environment Module: A New Mechanism for Rooting Trust in Cloud Computing,” in 3rd International Conference on Trust and Trustworthy Computing, Berlin, 2010.

[47] F. J. Krautheim, D. S. Phatak, and A. T. Sherman, *Private Virtual Infrastructure: A Model for Trustworthy Utility Cloud Computing*, TR-CS-10-04, University of Maryland Baltimore County, Baltimore, MD, 2010; http://www.cisa.umbc.edu/papers/krautheim_tr-cs-10-04.pdf.

[48] V. Scarlata, C. Rozas, M. Wiseman, *et al.*, "TPM Virtualization: Building a General Framework," *Trusted Computing*, N. Pohlmann and H. Reimer, eds., pp. 43-56, Wiesbaden, Germany: Vieweg+Teubner, 2008.

[49] D. Abramson, J. Jackson, S. Muthrasanallur, *et al.*, “Intel Virtualization Technology for Directed I/O,” *Intel Technology Journal*, vol. 10, no. 3, pp. 179-192, 2006.

[50] "Intel Trusted Execution Technology," http://www.intel.com/technology/security/.

[51] "Citrix XenDesktop," http://www.citrix.com/virtualization/desktop/xendesktop.html.

[52] http://www.trustedcomputinggroup.org.

[53] P. England, and J. Loeser, “Para-Virtualized TPM Sharing,” in 1st International Conference on Trusted Computing and Trust in Information Technologies, Villach, Austria, 2008.

[54] A.-R. Sadeghi, C. Stüble, and M. Winandy, “Property-Based TPM Virtualization,” in 11th International Conference on Information Security, Taipei, Taiwan, 2008.

[55] M. Strasser, “A Software-based TPM Emulator for Linux,” Department of Computer Science, Swiss Federal Institute of Technology, Zurich, 2004.

[56] X. Wang, Y. L. Yin, and H. Yu, "Finding Collisions in the Full SHA-1," *Lecture Notes in Computer Science*, pp. 17-36, Heidelberg: Springer, 2005.

[57] S. Smith, *Trusted Computing Platforms: Design and Applications*, New York: Springer, 2005.

[58] Q. Dang, "Recommendation for Applications Using Approved Hash Algorithms," *NIST Special Publication*, 800-107, NIST, 2009.

[59] D. Chisnall, *The Definitive Guide to the Xen Hypervisor*, Upper Saddle River, NJ: Prentice Hall, 2008.

[60] "Hadoop," http://hadoop.apache.org/.

[61] F. Chang, J. D. S. Ghemawat, W. C. Hsieh, *et al.*, “Bigtable: A Distributed Storage System for Structured Data,” in Seventh Symposium on Operating System Design and Implementation, Seattle, WA, 2005.