Deploying Virtual Machines on Shared Platforms

(Security Management - Design & Mechanisms)

(February 14, 2011)

Mudassar Aslam and Christian Gehrmann
Swedish Institute of Computer Science,
Box 1263, SE-164 29 Kista, Sweden
{mudassar.aslam,christian.gehrmann}@sics.se
http://www.sics.se

Abstract:

In this report, we describe mechanisms for secure deployment of virtual machines on shared platforms looking into a telecommunication cloud use case, which is also presented in this report. The architecture we present focuses on the security requirements of the major stakeholders' part of the scenario we present. This report comprehensively covers all major security aspects including different security mechanisms and protocols, leveraging existing standards and state-of-the art wherever applicable. In particular, our architecture uses TCG technologies for trust establishment in the deployment of operator virtual machines on shared resource platforms. We also propose a novel procedure for securely launching and cryptographically binding a virtual machine to a target platform thereby protecting the operator virtual machine and its related credentials

Keywords:

Security, trusted computing, virtualization, cloud computing, telecommunication networks

Table of Contents

1 Introduction

Past years we have seen a strong move in the market place towards usage of virtualization technologies[1]. Virtualization allows one to run several Virtual Machines in parallel on a common hardware platform and in some circumstances also to run legacy applications unmodified on new hardware platforms. This is realized through virtualization of memory and peripherals on the platform with the assistance of a so called hypervisor or Virtual Machine Monitor (VMM). A hypervisor runs in the most privileged mode in a system and has full control over vital system resources. A hypervisor-based system gives increased system utilization as multiple Virtual Machines (VMs) can run simultaneously on a single powerful hardware platform, opening for new business models and a new business landscape. This implies for example that existing services can rather easily be migrated into large computing clusters or what often is referred to as the cloud. This new flexibility obviously has a price: increased security risks. Systems previously physically isolated, might now run on the same machine and consequently opening up to new attacks between virtual machines running simultaneously on the same hardware. The hypervisor or VMM is a new target for attacks. Once a VMM is compromised, the whole system is compromised [18]. Hence, it is very important to make sure that the all security critical components including the VMM are trusted prior to launching a service on a platform. Protection mechanisms implemented on OS or application level utilizing specific hardware capabilities/features might become vulnerable when the actual hardware is virtualized. In addition to these direct risks, the trust model will to a large extent depend on the business model of the system, and the security architecture needs to be adapted to that model to serve the different stakeholders' needs. Despite these facts, few attempts have been made to make a detailed analysis and design of comprehensive security architectures for virtualized systems. In this report we make such an analysis and design through looking into a particular virtualization scenario: a telecommunication system where several different operators share the same physical infrastructure. The main issues addressed in this report are the following:

- We introduce a telecommunication cloud use case in which the telecommunication resource provider provisions virtual resources to different operators instead of physical resources.
- We designed a novel security architecture including trust models, roles and security protocols based on the Trusted Computing Group (TCG) [23] industry standard security frameworks.
- We provide secure VM deployment mechanisms and do prototype implementations accordingly.

This report is organized as follows. In Section 2 we describe the telecommunication cloud scenario. Section 3 presents a detailed description of the security architecture we suggest. In Section 4 we propose a VM launch protocol and finally we conclude in Section 5.

[1] The virtualization technology we discuss here is the approach when a complete software system (including OS) runs on top of a hypervisor. This makes the illusion to the guest system of actually running directly upon the real hardware and it is often also referred to as system virtualization [22].

2 Scenario – A Telecommunication Cloud Use case

The network scenario we look into has two major stakeholders - the *Provider* who provides telecommunication hardware resources which form ”the telecommunication cloud”, and the *Operator* who handles all end-customer relations and operates the telecommunication network utilizing the telecommunication resources offered by the provider. We consider the situation where operators are offered *virtual resources* instead of real hardware devices. In this system set-up, the operators are charged for the exact type and number of resources they need and the provider can enforce strong licensing techniques restricting operators to use only provisioned resources. An overview diagram of the stated scenario is shown in Figure 1.

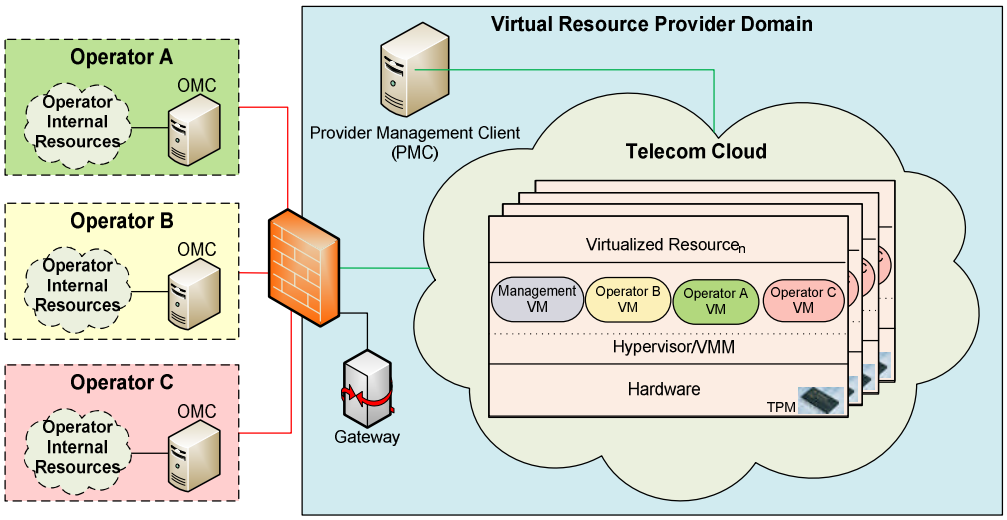

Figure 1: Our Proposed Telecommunication Cloud Scenario

The resource provider manages provisioned resources in a cloud in which each resource is virtualized to allow different operators to launch their virtual machines. The operators communicate using Operator Management Clients (OMCs) with their VMs through a gateway entity. The gateway entity protects the provider internal network from unauthorized external accesses. Similarly, we assume that the provider manages the virtualized telecommunication resources through a Provider Management Client (PMC).

Many other scenarios can also arise involving multiple entities performing distinct roles. One possible alternate scenario could have an intermediate entity which takes the responsibility of managing the telecommunication cloud. In such scenario, the manufacturer outsources its physical resources to the Cloud Management Entity (CME), which then virtualizes and provisions the available resources to the operators. The resulting infrastructure would allow a CME to offer virtual resources from different manufacturers. By introducing another entity, the extended scenario becomes more complex with respect to the rights and responsibilities of each stakeholder. Furthermore, such a business model must be investigated in detail because obvious impact could be on the manufacture whose selling rights might be shifted to the CME. This extended scenario will be addressed more specifically in future after getting expected results from the basic scenario presented earlier. The extended scenario is depicted in Figure 2.

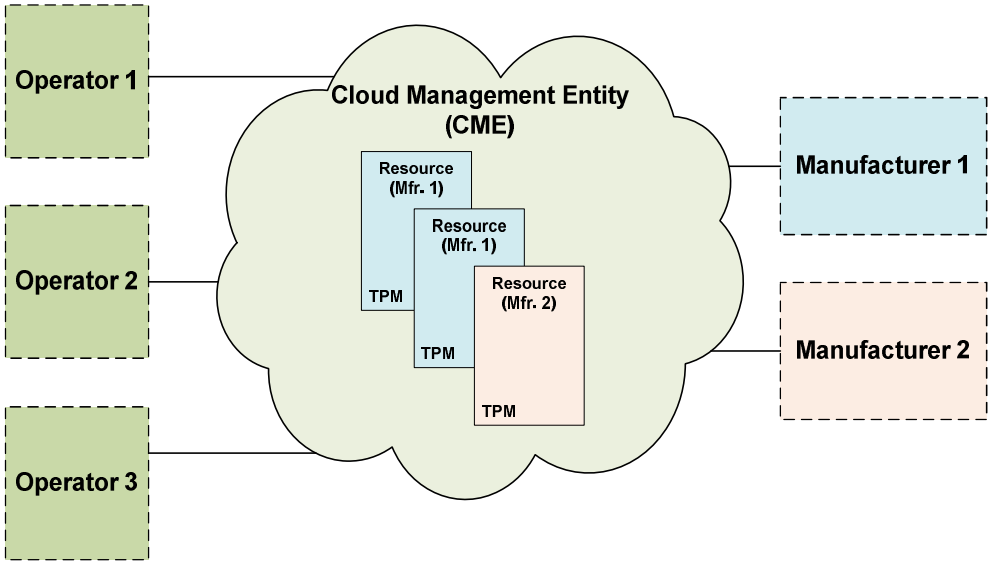

Figure 2: Extended Telecommunication Cloud Scenario with 3 roles

3 Security Architecture

Our security architecture is based on the TCG framework as it gives basic building blocks for trust establishment and platform integrity, which is needed to meet the security expectations of the stakeholders. This implies that the architecture assumes that all telecommunication target platforms are equipped with Trusted Platform Modules (TPMs) as defined by TCG [25]. The architecture we have defined consists of the resource platforms and a set of security components and protocols. We start by describing the resource platform architecture by introducing its different components and their roles. Next, we describe the procedures connected to the three resource management phases, i.e., initializations, VM deployment and VM operation.

3.1 *Resource Platform Architecture*

The basis for the architecture is the resource platform model we are using which is shown in Figure 3.

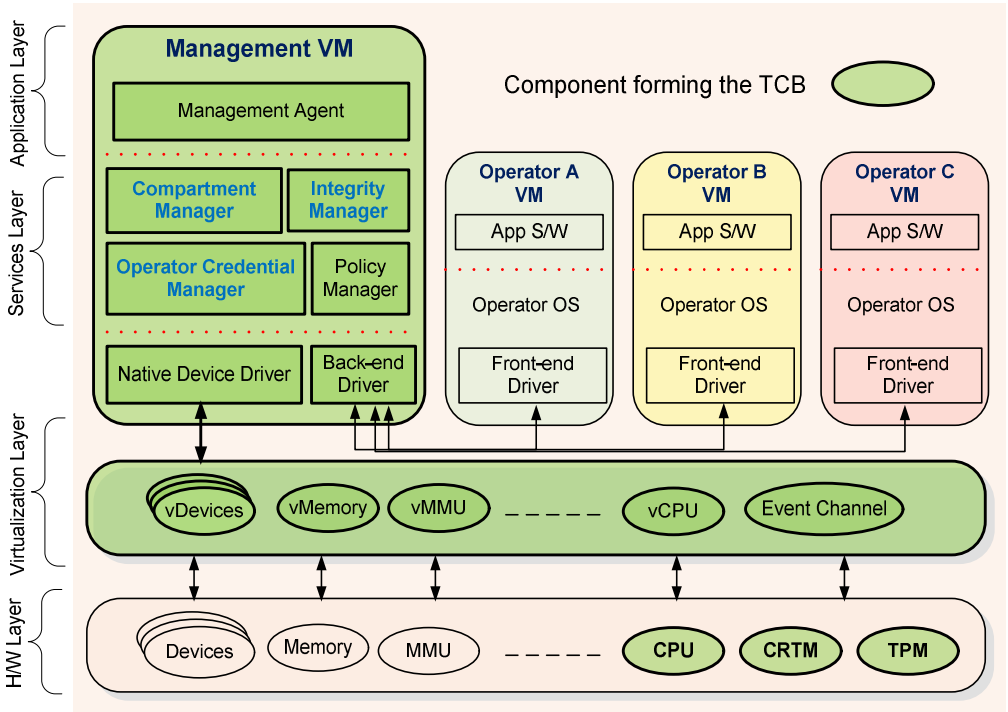

Figure 3: Security Architecture - Trusted Virtualized Platform

The resource platform is a virtualized platform in which platform virtualization is performed by a hypervisor. Here we assume a XEN hypervisor [4] but the architecture is generic and can easily be adapted to support other hypervisor solutions such as KVM [16] or VMware [26]. The main building block of the platform is the *Management Virtual Machine* which runs in dom0 as a privileged VM. Other VMs belong to different operators which are interested in using platform resources. The management VM has a Management Agent running at Application Layer which interacts with other entities of the network (internal and external) and perform services for them. There are four major security components in the Services Layer which interact with each other and perform all security management operations. The Compartment Manager (CM) performs the leading role by using services of other components. The main operations performed by the CM include creation of operator VMs, sealing a VM to the current platform, migration of a VM and protection of VM operation credentials for secure migration. The CM uses the services of the Integrity Manager (IM)

which performs various tasks pertaining to platform trust establishment. It uses TPM features and performs tasks like taking runtime integrity measurements and reporting platform state for remote attestation. In principle, the CM calls IM functions for establishing trust before launching an operator VM. Moreover, the Provider Management Client can query the integrity state of the platform from the IM. The Operator Credential Manager (OCM) keeps the credentials of the licensed operators to make access checks of VMs prior to launch, logs launch events and transfer licensing information to the VMM layer. The Policy Manager (PM) handles different kinds of policies for the trusted platform like the boot policy which could activate secure boot or trusted boot. Similarly, the PM handles VM migration policy. These security components are shown in the class diagram shown in Figure 4.

The underlying Virtualization Layer makes virtual instances of hardware resources (an exception is the TPM, see the discussion below) so that every operator VM can have its own virtual hardware. As there is typically only one hardware TPM available on a VMM controlled platform, a ”trusted virtual platform” can be set up in two ways which we refer to as implicit trust establishment and explicit trust establishment. In implicit trust establishment, we limit the use of the single TPM to the hypervisor which is responsible for secure bootstrapping and platform integrity management. Moreover, all other TPM features (platform attestation, integrity reporting etc.) are only available to the VMM or Management VM. Every service/software running in the launched VM implicitly trusts that the VM itself was launched securely. In explicit trust establishment, instances of virtual TPMs [2] are required to render services to every VM. This approach allows the use of all TPM features by every VM, mandating it to perform integrity management services. The choice of approach to adopt for trust establishment varies depending upon the scenario and the stakeholders’ concern and in our case, the provider does not have any need for verifying what is running inside VMs. Similarly, the operator’s main concern is to launch the VM on a trusted remote resource platform (the VM configuration can be verfied prior to launch). Hence, we have adopted an implicit trust establishment model. This means that our proposed platform’s Trusted Computing Base (TCB) is up to the level of the Management VM. This is similar to the model in [9] where the authors suggest that the ”traditional” TCG based attestation mechanism are used to securely authenticate platform configuration up until the boot of the VMM layer. An architecture which establishes trust up until the level of the hypervisor is also proposed in [15].

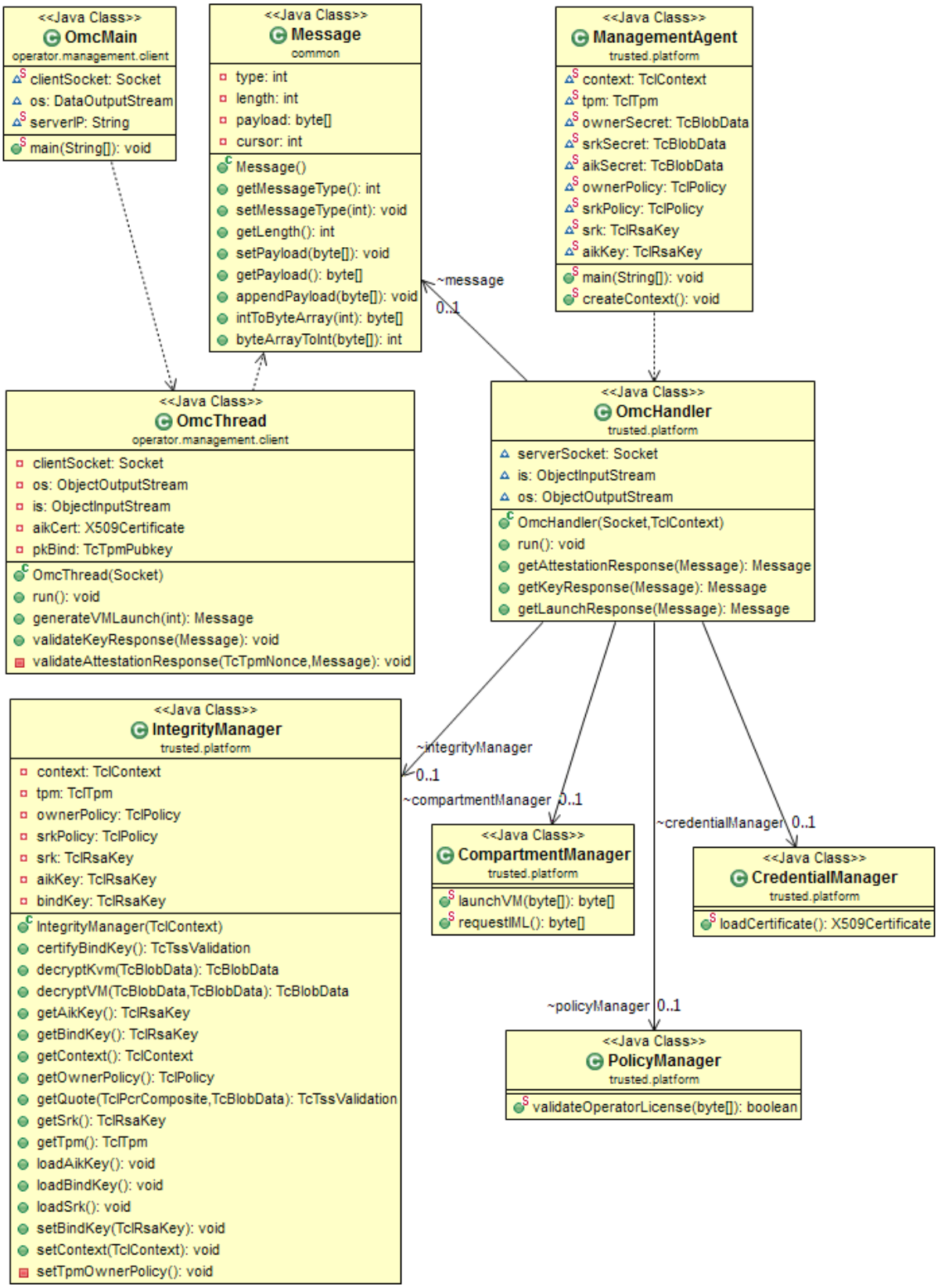

Figure 4: Security Components serving the Management Agent

3.2 *Initializations*

In order to meet the operator requirements of establishing trust in the offered platform resources, the operator should have mechanisms to verify the integrity and identity of the platform. In the context of trusted computing, the integrity of the platform is verified by comparing runtime integrity measurements (of the running software components) with good known reference measurements. This means that 1) the provider needs to publish the reference metrics which could be used by the operator for platform integrity-verification. 2) Another important initialization required to be done by the provider is platform identity registration with a trusted third party for authentication. 3) The basis for integrity verification and platform identity registration lies in establishing a trust hierarchy prior to these tasks. These initializations and eventually secure boot of the trusted platform are discussed in the following sections.

3.2.1 Certificate Management

There are two aspects of establishing trust in a virtualized platform. Firstly, there should be a mechanism to uniquely identify the platform (as hardware) which is referred to as platform authentication. Secondly, there should be a mechanism to verify that only trusted code is running on the platform which could be achieved by means of remote attestation. This is achieved by the operator client sending challenge for platform authentication and remote attestation and the platform responding with the response. For operator client to believe that the response is coming from the intended platform and is generated using a TPM, we follow TCG specified mechanisms for certificate management and trust establishment. TCG style certificate management follows that a platform is made up of many components (hardware and software), therefore, multiple entities must be involved (each entity to certify its component) in the process of making a unique platform identity. This necessitates multiple certificate issuing authorities which are briefly described below and are also modeled in our architecture. An abstract semblance of certificate hierarchy is shown in Figure 5.

3.2.1.1 Endorsement Key Certificate Authority (EK-CA)

The EK-Certificate contains the public Endorsement Key. The private part of the endorsements key is unique to each TPM never leaves the TPM and thus the public Endorsement Key can be used to secure identify a particular TPM. The EK-Certificate asserts that the holder of the private EK is a TPM conforming to TCG specification [27]. The TPM chip manufacturer is supposed to issue EK-Certificate for the TPM chip to be installed on the *to-be* trusted platform. However, not all chip manufacturers issue EK-Certificate therefore the platform manufacturer is sometimes required to issue EK-Certificate.

3.2.1.2 Platform Endorsement Certificate Authority (PE-CA)

A Platform Endorsement Certificate (PE) asserts that a specific platform contains a unique TPM and Trusted Building Block (TBB) [27]. This assertion is made by pointing to the EK-Certificate of the TPM which is installed on the platform. Usually, the platform manufacturer is the issuer of PE-Certificate who signs it only after ensuring that the platform contains the referred TPM and the TBB.

3.2.1.3 Attestation Identity Key Certificate Authority (AIK-CA)

The Attestation Identity Key Certificate contains public AIK key and is issued by a Privacy-CA. The idea of using AIK key is to assert that the operation (e.g. attestation) is performed by a TPM on an intended platform yet protecting privacy of the platform which could be exposed if Endorsement key is used. The role of Privacy CA is to attest the authenticity of the AIK key [27].

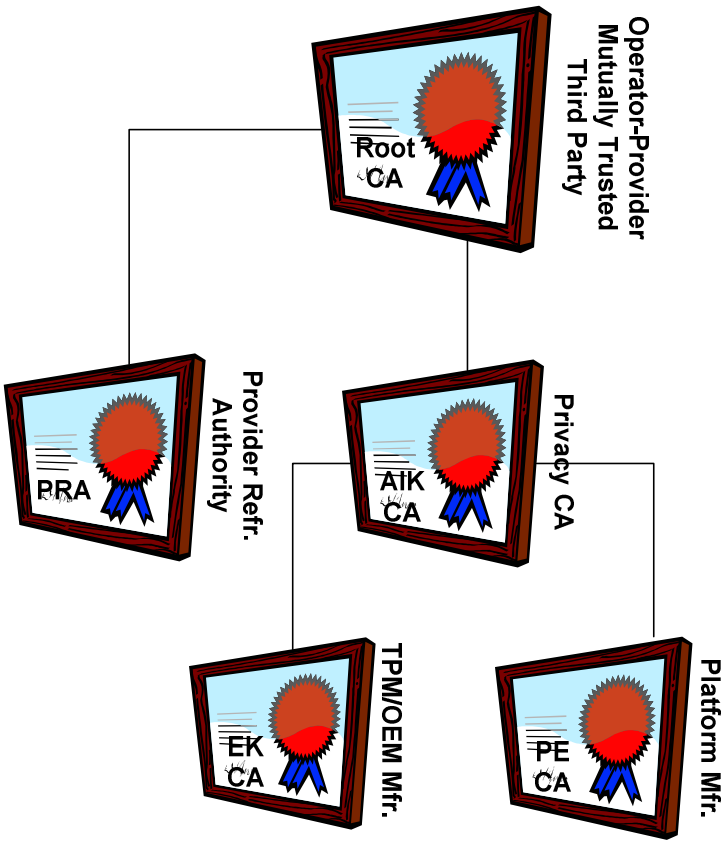

Figure 5: Security Components serving the Management Agent

3.2.2 Reference Metrics:

In order to fulfill the operators' expectations on distinguishing trusted platform configurations from non-trusted platform configurations, there must be well defined and secure means to obtain verified trust evidence values, so called reference metrics. TCG has defined the processes and formats for distributing protected reference metrics through the Integrity Management Model (IMM) [12], which we have chosen to adopt. The process of creating and publishing reference metrics may involve many entities depending upon their role. In our scenario, the provider is also the manufacturer of the target platform who can create and publish reference metrics for the platform or let a third party create and publish those on behalf of the manufacturer. Other than provisioning these reference metrics, the provider is also required to bind them with the platform which is discussed in the next section. The creation of reference metrics, called harvesting, means the transformation of raw integrity measurements or values for the component (e.g. serial number, manufacturer id, etc. for hardware components and hash values for software components) into TCG specified standard format called Reference Manifest (RM) [20]. Other than containing the raw data of the component, an RM

also contains a Component ID which distinguishes the RM of a particular component. Another important field of an RM is the list of pointers to the RMs (component IDs) of its subcomponents. The provider harvests RMs for the trusted virtualized platform and its subcomponents which make up the TCB. These include Core Root of Trust for Measurement (CRTM), hypervisor, device drivers and each service and application running in the management VM. After harvesting all platform-relevant RMs, the Provider Reference Manifest Authority (PRA) (see Figure 6) signs the RM records for their integrity protection and then publishes them through Provider Reference Manifest Record Server (PRRS). The trust between the PRA and the Operator is pre-established by a mutually trusted CA shown in Figure 5. For some subcomponents, the provider might get signed RMs from their manufacturers. In such a case, the provider acts as an aggregator who is only required to publish those RMs through PRRS. Finally, the operator can get all platform-relevant RMs using TCG defined IF-Publish interface exposed by PRRS.

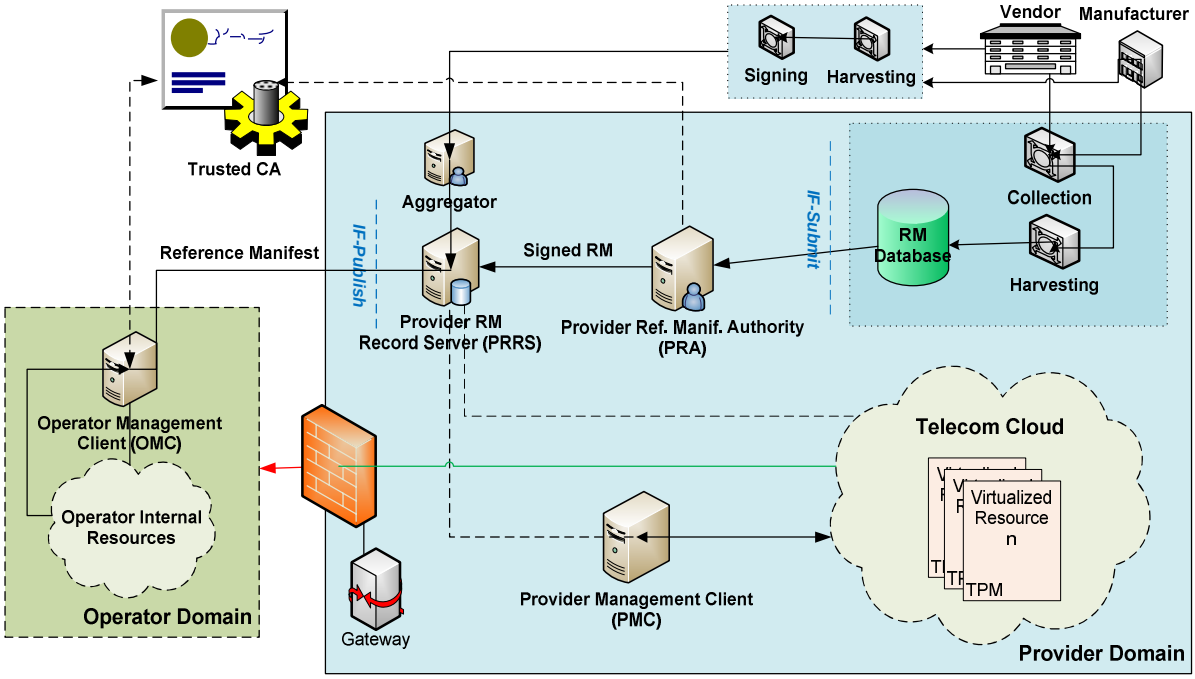

Figure 6: System Overview

3.2.3 Platform Credential Generation and Registration Procedures:

Next we discuss the credential configurations needed in order for the different trust verifications and identifications in our architecture. In order to report the platform integrity state in a trusted way, the trusted platform should have an asymmetric key pair for attestation, a so called Attestation Identity key pair, K_{AIK}. The secret part of the Attestation Identity Key (SK_{AIK}) is protected by TPM and is loaded and used only for remote attestation and key certification. The corresponding public part, PK_{AIK} needs to be registered with a Privacy CA who issues an identity certificate for the platform, a so called AIK Certificate according to the TCG attestation key and certificate principles [24]. Furthermore, as stated in the system requirements report[28], one of the major operator requirements is to be sure that a VM only is launched on a platform in a verified software state and preferably the VM should be cryptographically bound to a trusted platform state. As we will show in Section 4.3, this can be achieved through creating a non-migratable asymmetric key pair in the TPM

that is cryptographically boundto a specific defined integrity state of the platform and that never leaves the TPM unencrypted. This cryptographic binding of keys to a platform state identified by PCRs is called “Key Wrapping”. We denote this wrapped key pair by K_{bind}, the public part of this key is denoted by PK_{bind}, and the corresponding private part by SK_{bind}. The K_{bind} key pair is created through the TPM CreateWrapKey() **[25]** command at any time before a VM launch but after the creation of the K_{AIK}. In order to prove that K_{bind} is wrapped to certain PCRs, it is signed together with the defined PCRs using SK_{AIK} though the TPM CertifyKey() command [25].

3.2.4 Boot Security

The integrity of a Trusted Platform lies in the fact that it is booted securely and remains in that state for the whole execution time. We distinguish between two basic security models for booting up the platforms, secure boot and trusted boot [12]. The concept of secure boot means that during the boot process, each component to be loaded is measured and verified against reference metrics before it is allowed to be executed. This requires secure verification code and protected reference metrics to be available to the Trusted Platform for boot and verification purposes. The implication of secure boot isthat the trusted platform may not boot at all, for example, because of a software attack or due to a software upgrade with missing related updated reference metrics. In contrast, in trusted boot, extended hash of every loaded code is kept in the TPM PCRs and also in the Integrity Measurement Log (IML) as Event Structure which contains the hash value and the component metadata. In order to fulfill the provider requirements, we propose a flexible boot model where the boot principle can be configured through a provider policy. This means that the boot policy (secure or trusted) is also stored in the TCB by the Policy Manager. All code blocks part of the TCB will be verified through the secure or trusted boot sequence by calculating hashes of the code blocks and either reporting them into TCG PCR registers (trusted boot) or comparing them with reference metric hashes (secure boot) according to the standard TCG transitive trust model [24]. Since we are using implicit trust establishment model every piece of code running launched after the code base forming the TCB will be trusted implicitly. From provider’s and operator’s point of view, this approach satisfies the requirement that the underlying virtual environment is trustworthy as long as the VMM provides strong isolation security (which will be a requirement from the operator perspective). Moreover, this approach is in line with the guiding principles suggested in [5] in which the authors explain security vulnerabilities due to hierarchical trust dependencies.

4 Secure VM Launch

The VM Launch can be performed after the security requirements regarding mutual authentication and platform integrity-verification are fulfilled. The following sections describe the three stages of the VM launch protocol which are shown in Figure 7.

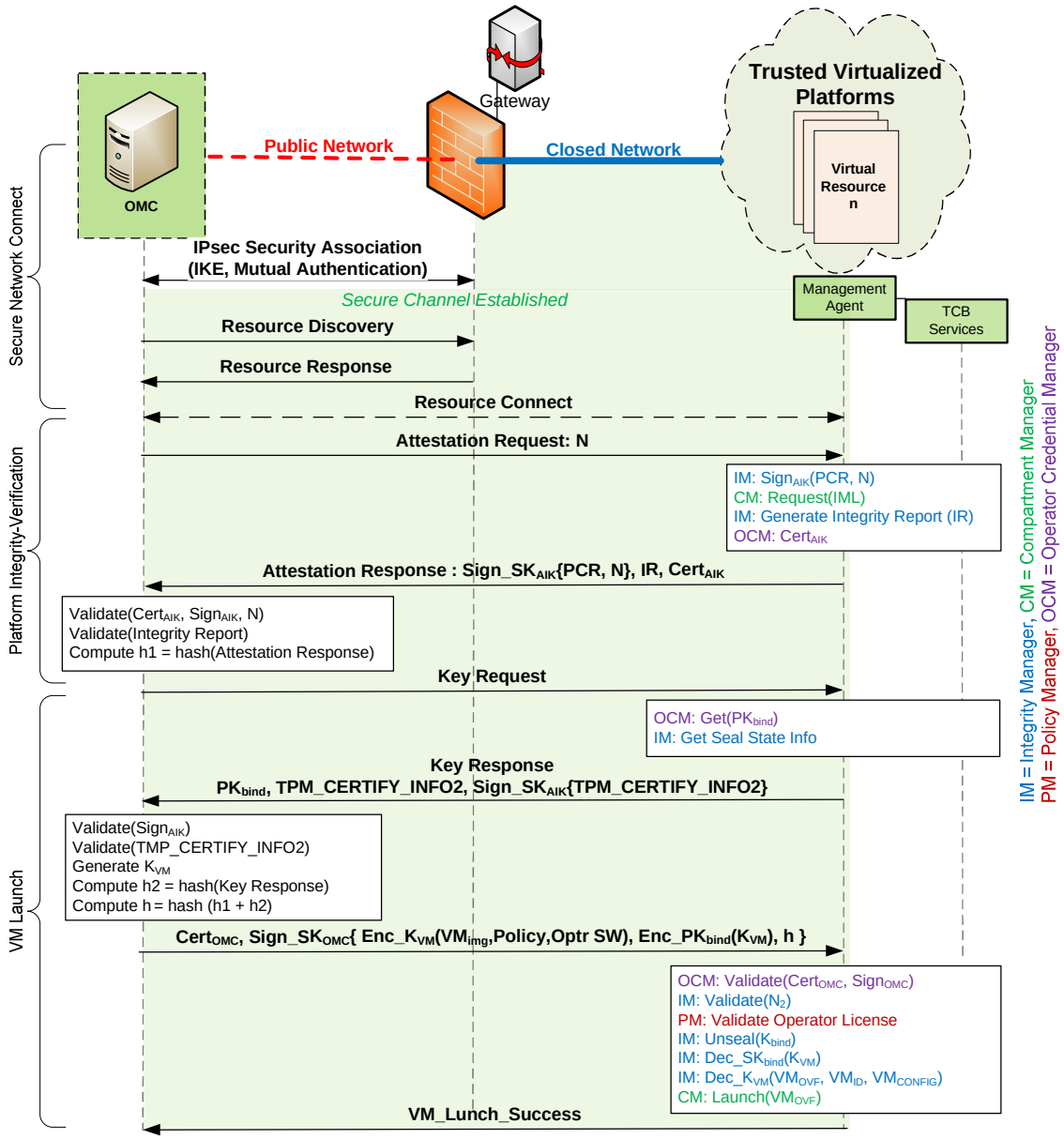

Figure 7: Secure VM Launch Protocol

4.1 *Secure Network Connect*

In order to protect the internal virtual resources, the provider needs to authenticate all resource requests towards the network through a provider gateway which authenticates the operator client and establishes a secure pipe with the operator client. Similarly, the Operator Management Client (OMC) would like to make sure that it connects to a trusted resource provider network. Standard VPN protection mechanisms are used to provide this and we use the Internet Key Exchange (IKE)

protocol [11] (mutual authentication and key exchange) in combination with IPsec [14]. The operator and the gateway are assumed to have their digital certificates issued by a mutually trusted CA. After the establishment of the IPsec connection, the OMC issues a resource discovery request searching for a particular resource or resource type and gets a response from the gateway with a handle that can be used by the OMC to connect to the provisioned platform.

4.2 *Platform Integrity-Verification*

Before launching the VM to a particular platform, the OMC needs to verify the integrity of the TCB of the target platform. This is done according to the TCG attestation procedure [24]. We use an attestation protocol similar to the one described in [10] and [21] carried over HTTP. The attestation starts with the OMC sending a nonce (N) to the Trusted Platform (TP) with a request to report its integrity state. The Management Agent requests the Integrity Manager to generate an Attestation Response. The attestation response contains the current state of PCRs and the Integrity Report. The IM requests the TPM to get the current state of PCRs signed with SK_{AIK}. The Integrity Report is generated by the Platform Trust Service (PTS) [1] which is part of the Integrity Manager. The PTS uses the IML for integrity report generation and formats it according to the TCG Integrity Schema [13]; it is XML-coded for automated parsing and evaluation at the operator end. The integrity report is also signed using SK_{AIK} which ensures that it belongs to the Trusted Platform identified in the AIK-Certificate. The Integrity Report, current PCR state and the AIK-Certificate are sent back to the OMC. The OMC validates the AIK-Certificate, nonce and the signatures and compares the received Integrity Report with the Reference Manifests (RM) collected from Provider Reference Manifest Record Server (PRRS), potentially complemented with operator specific metrics (see Figure 6). Finally, if all checks went OK, the OMC computes hash (h1) of the received Attestation Response which is used in the next stage.

4.3 *VM Launch*

The final stage of the protocol is the actual launch of the Operator VM. As stated in Section 3, the operator VM should be sealed to the TP. We make use of the sealing feature of TPM for sealed launch of the VM which means that the VM will only be able to launch on the licensed platform with unique set of configurations. This sealed launch is done by using a specially created non migratable, asymmetric bind key which is created before the launch phase (see Section 3.2.3). The OMC requests for the public bind key, PK_{bind}, and the PCR state of the TP which needs to be satisfied for using the bind key. The TP returns PK_{bind} and the TPM CERTIFY INFO structure to the OMC. The OMC validates the signature and compares PCR state in the TPM CERTIFY INFO with the state reported in the integrity-verification stage to ensure that it is about to launch its VM on the TP attested earlier. The TPM CERTIFIY INFO structure also proves that PK_{bind} is either non-migratable or certified migratable key (CMK) because this structure is returned by CERTIFY_KEY command only if the key is non-migratable or CMK. We will explore the usage of CMK to support secure VM migration in future work. The OMC generates a symmetric key K_{VM} which is used to encrypt the VM along with the VM identity and related VM configuration parameters. K_{VM} is encrypted with PK_{bind} which ensures that the VM can only be launched on the intended TP, because only that TP can unwrap the SK_{bind} key used for K_{VM} decryption. In order to cryptographically bind the different protocol steps - attestation, bind key exchange and launch - the OMC computes the hash of Key Response message(h2), appends it to h1(computed in previous stage), and computes h which is also sent to the TP. The OMC must

sign the VM Launch message so that the operator cannot repudiate the launch later on. The TP performs necessary validations and checks operator license for VM launch rights, and (optionally) also updates corresponding charging records. It then loads the bind key, decrypts the K_{VM}, use K_{vm} to decrypt VM Launch bundle which contains VM Image, License or Policy, and operator software to be executed by the Operator VM. Finally, the management agent requests Compartment Manager to launch the operator VM. A success message is returned to the operator client if his VM is launched successfully who can then connect to his VM and perform Management Operations briefly discussed in the next section.

4.4 *VM operation*

Once the operator VM is launched on the Trusted Platform, the OMC needs to perform management operations remotely. All management tasks requests are sent to the Management Agent which first checks the access rights for the connected operator client (serviced by Policy Manager) and then performs the requested operation (serviced by Compartment Manager). The management session between OMC and the Management Agent is protected by TLS.. As the VM launch procedure provides protection of VM configurations including credentials, legacy management credentials can be used unmodified. An important security requirement mentioned in [28] for secure VM operation from both operator and provider's perspective is that the launched VM should remain isolated from other VM's running on the same platform. Strong VM isolation is the basic property offered by the hypervisor we use [4], [26], [16], therefore we assume that all VMs running on the TP will operate isolated from each other. Other important VM operations which need to be addressed are TP Software Update and VM Migration done by Provider Management Client (PMC). Any software update means the change of TP configurations which implies that the TP integrity will not be verified which is required for the launch of any new VM. Therefore, the provider would be required to re-perform some tasks such as the update of RM for the platform for integrity-verification (required for new VM launch) and creation of a new bind key (K_{bind}).

5 Conclusions

In this report we have presented a scenario in which virtualized telecommunication resources making up a telecommunication cloud, are provisioned to different telecommunication operators. We have considered the security requirements of the addressed scenario taking both stakeholders' concerns into account for every phase of the resource provisioning (boot, authenticate, launch, manage) [28]. We have identified the significance of *trust establishment* in the cloud environments for which we presented a comprehensive model which leverages *Trusted Computing* concepts. Our model necessitates prior configuration of *Reference Metrics* and *Identity Registration* of the target platform. The identity registration of the target platform with a mutually trusted certificate authority allows platform authentication; and the published reference metrics are used for comparison against runtime integrity measurements reported by the target platform in integrity verification of the platform. These two steps establish trust in the target platform for VM launch. Finally, we presented a novel VM launch protocol which allows cryptographic binding of the VM to the target platform. Such binding fulfills important security requirements related to the protection of operator program and data.. Overall, our presented architecture prevents unauthenticated users/clients from gaining access to the trusted platform or operator VMs and executing arbitrary actions, and also prevents authenticated users/clients from circumventing licensing and charging checks. This report presents the state-of-the-art in telecommunication clouds. The implementation of a demo system as proof of concept is also a complementary work. The telecommunication cloud scenario presented in this paper also requires secure licensing and charging mechanisms which is subject to future work. Furthermore, VM migration, which requires protection of VM credentials in transit, is another topic for future research.

6 References

[1] Platform Trust Services Interface Specification (IF-PTS). http://www.trustedcomputinggroup.org/resources (November 2006)

[2] Berger, S., C´aceres, R., Goldman, K.A., Perez, R., Sailer, R., van Doorn, L.: vTPM: Virtualizing the Trusted Platform Module. In: USENIX-SS'06: Proceedings of the 15th Conference on USENIX Security Symposium. USENIX Association, Berkeley, CA, USA (2006)

[3] Berger, S., C´aceres, R., Pendarakis, D., Sailer, R., Valdez, E., Perez, R., Schildhauer, W., Srinivasan, D.: TVDc: Managing Security in the Trusted Virtual Datacenter. SIGOPS Oper. Syst. Rev. 42(1), 40–47 (2008)

[4] Chisnall, D.: The Definitive Guide to the Xen Hypervisor (Prentice Hall Open Source Software Development Series). Prentice Hall PTR, Upper Saddle River, NJ, USA (2007)

[5] van Doorn, L.: Trusted Computing Challenges. In: STC '07: Proceedings of the 2007 ACM Workshop on Scalable Trusted Computing. pp. 1–1. ACM, New York, NY, USA (2007)

[6] Garfinkel, T., Pfaff, B., Chow, J., Rosenblum, M., Boneh, D.: Terra: a Virtual Machine-based Platform for Trusted Computing. pp. 193–206. ACM Press (2003)

[7] Gasmi, Y., Sadeghi, A.R., Stewin, P., Unger, M., Asokan, N.: Beyond Secure Channels. In: STC '07: Proceedings of the 2007 ACM Workshop on Scalable Trusted Computing. pp. 30–40. ACM, New York, NY, USA (2007)

[8] Griffin, J.L., Jaeger, T., Perez, R., Sailer, R., Doorn, L.V., Cceres, R.: Trusted Virtual Domains: Toward Secure Distributed Services. In: In Proc. of the First Workshop on Hot Topics in System Dependability (Hotdep05. IEEE Press (2005)

[9] Haldar, V., Chandra, D., Franz, M.: Semantic Remote Attestation -A Virtual Machine directed approach to Trusted Computing. In: USENIX Virtual Machine Research and Technology Symposium. pp. 29–41 (2004)

[10] Huang, X., Peng, Y.: An Effective Approach for Remote Attestation in Trusted Computing. In: WISA 2009 : Proceedings of the 2nd International Symposium on Web Information Systems and Applications. pp. 80–83. Academy Publisher, FIN-90571, OULU, FINLAND (2009)

[11] Internet Key Exchange (IKEv2) Protocol. Tech. Rep. RFC 4306, Internet Engineering Task Force (December 2005)

[12] TCG Infrastructure Architecture Part-II -Integrity Management. http://www.trustedcomputinggroup.org/resources (November 2006)

[13] Integrity Report Schema Specification. http://www.trustedcomputinggroup. org/resources (November 2006)

[14] Security Architecture for the Internet Protocol. Tech. Rep. RFC 4301, Internet Engineering Task Force (December 2005)

[15] Jansen, B., Ramasamy, H.V., Schunter, M.: Flexible Integrity Protection and Verification Architecture for Virtual Machine Monitors. In: The Second Workshop on Advances in Trusted Computing (WATC 06 Fall (2006)

[16] Kernel Based Virtual Machine. http://www.linux-kvm.org/page/Main_Page

[17] Landfermann, R., Kuhlmann, D., Kuhlmann, D., L, R., Ramasamy, H.V., Ramasamy, H.V., Schunter, M., Schunter, M., Ramunno, G., Ramunno, G., Vernizzi, D., Vernizzi, D.: D.: An Open Trusted Computing Architecture Secure Virtual Machines Enabling User-defined Policy Enforcement. www.opentc.net (2006)

[18] Ormandy, T.: An empirical Study into the Security Exposure to Hosts of Hostile Virtualized Environments. In: CanSecWest (2007)

[19] Open Virtualization Format. http://www.vmware.com/appliances/ getting-started/learn/ovf.html

[20] Reference Manifest (RM) Schema Specification. http://www. trustedcomputinggroup.org/resources (November 2006)

[21] Sailer, R., Zhang, X., Jaeger, T., Doorn, L.V., Sailer, R., Zhang, X., Jaeger, T., Doorn, L.V.: Integrity Measurement Architecture. In: The Proceedings of the 13th USENIX Security Symposium (Sec 04). pp. 223–238 (August 2004)

[22] Smith, J.E., Nair, R.: Virtual machines: versatile platforms for systems and processes. Morgan Kaufmann Publishers (2005)

[23] Trusted Computing Group. http://www.trustedcomputinggroup.org/

[24] TCG Specification Architecture Overview. http://www.trustedcomputinggroup. org/resources (August 2007)

[25] TPM Specification, TPM Main Part I-III Design Principles. http://www. trustedcomputinggroup.org/resources (July 2007)

[26] VMware Inc., Virtualization Solutions. http://www.vmware.com/ virtualization/

[27] TCG Credential Profiles Specification, http://www. trustedcomputinggroup.org/resources (May 2007)

[28] System Requirements Report - Secure Management of Trusted Virtualized Platforms (SMTVP), Project Internal Report, October 05, 2010.