SICS Technical Report T2010:05 ISSN 1100-3154

TCG Based Approach for Secure Management of Virtualized Platforms

State-of-the-art

(June 05, 2010)

Mudassar Aslam, Christian Gehrmann

{Mudassar.Aslam, Christian.Gehrmann}@sics.se

Abstract:

There is a strong trend shift in the favor of adopting virtualization to get business benefits. The provisioning of virtualized enterprise resources is one kind of many possible scenarios. Where virtualization promises clear advantages it also poses new security challenges which need to be addressed to gain stakeholders confidence in the dynamics of new environment. One important facet of these challenges is establishing 'Trust' which is a basic primitive for any viable business model. The Trusted computing group (TCG) offers technologies and mechanisms required to establish this trust in the target platforms. Moreover, TCG technologies enable protecting of sensitive data in rest and transit. This report explores the applicability of relevant TCG concepts to virtualize enterprise resources securely for provisioning, establish trust in the target platforms and securely manage these virtualized Trusted Platforms.

Keywords:

Security, Trusted Computing, Virtualization, Cloud Computing, Telecommunication Networks

Table of Contents

1 Introduction

There is a strong trend towards move of services into the “cloud” or virtualized resources. The main reason for this strong market move is that cloud computing allows users to launch advanced IT services without huge upfront investments or (in some circumstances) expertise IT knowledge. So far, this trend has *not* affected telecommunication networks or infrastructures to any large extent. There are good reasons to believe that this will change in the near future and even open up for a new business environment in the telecommunication market. Moving such sensitive services as telephone and computer communication services into a virtualized environment implies several security concerns:

- Risk of loss of sensitive information due to leakage between virtual machines
- Loss of service due to denial of service attacks from adjacent virtual machines on the same host

As pointed out in by the Trusted Computing Group (TCG) [1], these security concerns can be tackled using trusted computing technologies and we believe that trusted computing could play an important role in securing future virtualized telecommunication resources. In particular, TCG technologies enable protection of data in rest and transit, strong authentication of virtualized resources and platform configurations. In this report we give an overview of the state-of-the- art with respect to trusted computing and protection of virtualized resources. The purpose with the overview is to serve as basis for future design of trusted computing based security architecture for virtualized telecommunication equipment.

This report is organized as follows: First we introduce the virtualized telecommunication equipment scenario that we address. Next, in Section 3 we make a summary of the most important TCG concept we consider. We only discuss major functions and the TCG specifications we think are most relevant to our scenario. Section 4 summarize a chosen set of relevant research papers treating trusted computing based solutions for securing virtualized environments. Finally, in Section 5, we make reflections and conclusions from the treated state-of-the-art technologies and suggest the direction for the design phase of the project.

2 The scenario and problems addressed

In a conventional scenario, hardware manufacturers provision their hardware device usage to different operators and charge them based on the cost/functionalities of the device. In such scenarios, one hardware device is dedicated for one operator which means that the operator might be paying for the resources which it does not want. Moreover, dedicating a hardware device, results in under-utilization of hardware capabilities and resources. This and many other problems can be addressed if operators are provisioned with virtual resources instead of giving full access to real hardware devices. Leveraging expanding capabilities of virtualization, an upcoming scheme is to outsource virtual devices to operators. The key to adopt such a scheme is that operators are charged for the exact type and number of resources and operator is bound to use only provisioned resources. Following activities can be performed to achieve these requirements:

1. **Issuance of license to operator:**

   Operator must be given a tamper resistant license which contains ´required resource access tokens´ along with other license specific information. We will investigate license protection using TCG Technologies (See Section 3).

2. **Provisioning of Virtual Resources:**

   The operator communicates with Virtual Machine Monitor (VMM) with the issued license. VMM launches a virtual machine for requesting operator which provisions only those resources which are licensed. A license can be used *either* to enforce resource constraints or indicate the right to utilize resource based on actual resource consumption reporting. In both cases we need an infrastructure for secure license issuing and consumption.

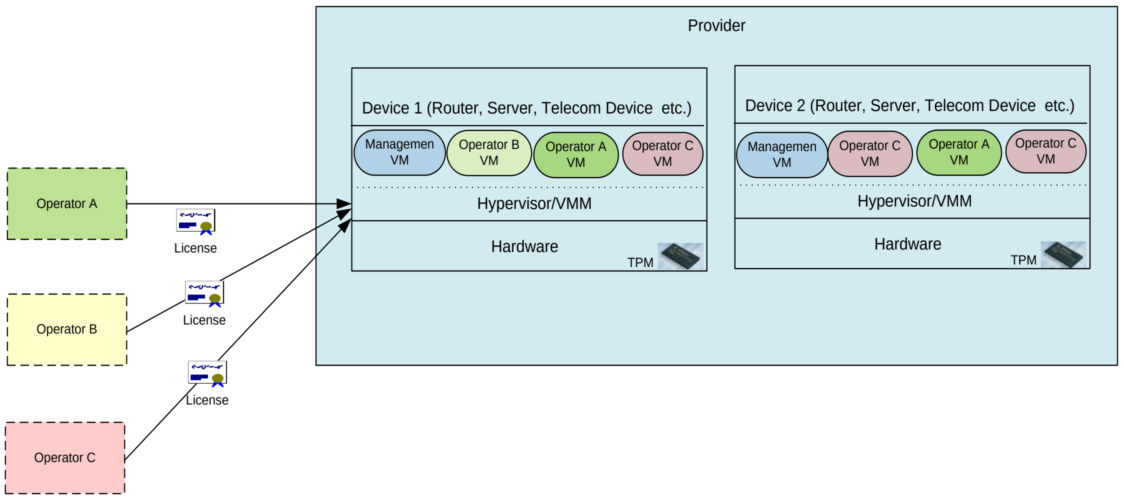


Figure 1 - Provisioning of Virtual Resources

3. **Resource consumption reporting:**

   Once a VM is launched for the operator, the operator communicates with that VM. The VM resource constraints are enforced by the VMM. Similar it is the responsibility of the VMM to measure operator VM resource consumption. Actual configuration and reporting of resources might be done through a Management VM, which runs in parallel with the Operator VM. At least we intend to investigate such architecture. But, alternative SW architecture will be investigated as well.

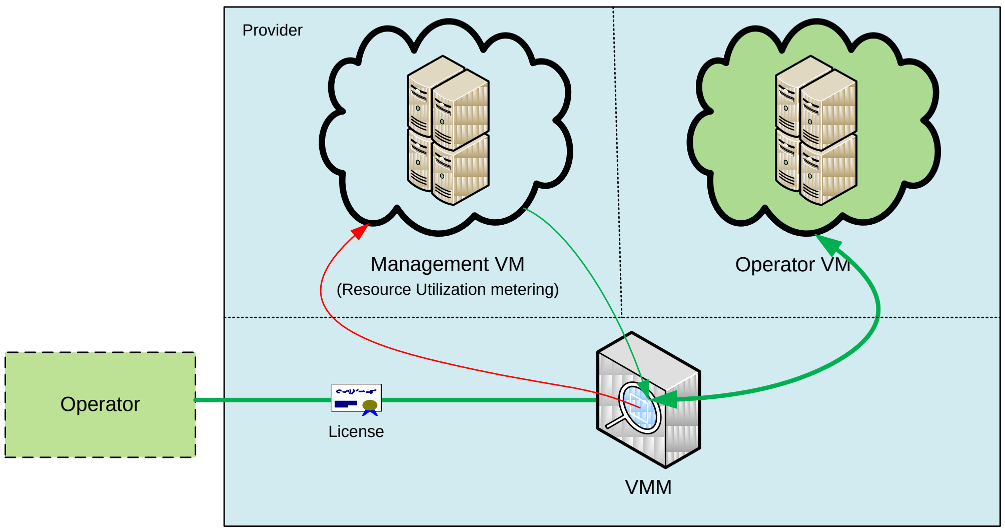


Figure 2 - Resource Consumption

4. **Trustworthiness of VMM (Secure bootstrapping):**
   Secure bootstrapping is the key to the whole system security. The operator needs to know that information not is leaked between virtual machines in the system. Furthermore, he needs some guarantees and evidence ensuring that his VM or VMs are booted into trusted states. We will investigate the applicability of Trusted Computing based solutions.

3 Trusted computing architectures

3.1 The TCG architecture

This section gives an overview of relevant parts of the TCG architecture.

3.1.1 Introduction

The Trusted Computing Group (TCG) is a non-profit industry-standards organization with the mission of proposing specifications and technology for trusted computing platforms. The most important among the TCG specifications is the definition of the Trusted Platform Module (TPM) [3].

3.1.2 TCG Usage Scenarios

The TCG technologies can be used for securing information assets. Integrity of the information asset can be protected by cryptographic hashing. Confidentiality can be applied by symmetric key encryption and authenticity of the information can be achieved by digitally signing it. For all these security services, keys are the cornerstones for information security which can be securely stored in ‘*Protected Storage*’.

The TPM can be used to assert asset ownership which means that system identity is protected and even if an asset is stolen, information cannot be accessed.

The TPM can also be helpful for E-commerce. It can be used to securely store and retrieve vendor-customer relationship context based on previous transactions which forms basis for trust. Finally, the TPM can also be used to monitor secure state of the platform by storing measurements (for hardware and software) in Platform Configuration Registers (PCR) which can be monitored anytime [3].

3.1.3 Trusted Platform Module (TPM)

The TCG defined Trusted Platform Module specifications can be implemented in hardware or software but hardware based TPM is considered more secure. The TPM is a fundamental security building block in the TCG architecture and therefore its components must function as intended. This can be ensured by following good engineering practices, manufacturing process and industry reviews. The main components of a TPM include:

- **Input/output (I/O):** This component controls information flow, encoding/decoding required for communication, routes messages to appropriate component and enforces access policy for TPM functions.

- **Non-volatile Storage:** Some values must remain in the TPM even if TPM is switched off. Endorsement Key (EK) and Storage Root Key (SRK) are RSA keys which must be stored in non-volatile storage.

- **Program Configuration Registers (PCR):** PCR can be implemented in non-volatile memory but it is better to implement PCR in volatile memory because they are reset when the platform restarts.

- **Attestation Identity Key (AIK):** Attestation Identity Keys are used for remotely attesting that this is a valid TPM-based platform. AIKs are recommended to be stored outside TPM and loaded in to TPM volatile memory when required.

- **Program Code:** Program code usually contains Core Root of Trust Measurement (CRTM) which should ideally be in the TPM. This code starts the root of trust chain.

- **Random Number Generator (RNG):** RNG is used for generating true random numbers which are used for key generation and nonce creation.

- **SHA-1 Engine:** Message digest engine to generate digest for digital signature and integrity services.

- **RSA Key Generation:** TCG has made RSA algorithm as standard for use in TPM modules due to its known strengths. RSA key generation engine generates 2048-bit modulus RSA keys for signing and storage (SRK, AIK).

- **RSA Engine:** RSA engine is used for signing and encrypting/decrypting keys for storage.

- **Opt-In:** This component is used for enabling/disabling or activating/deactivating TPM as desired by the customer.

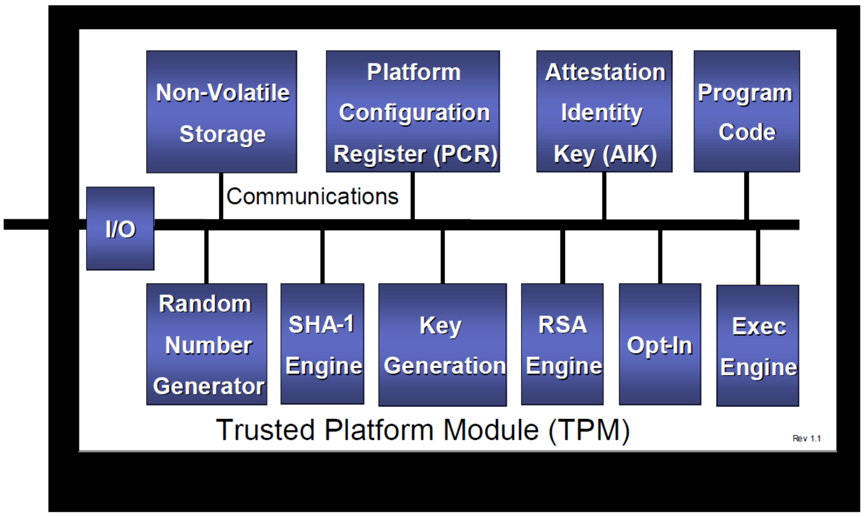


Figure 3 - TPM Component Architecture [2].

3.1.4 TPM – Key Management

Key management is an important aspect for any secure system. This includes key creation, storage and usage. TPM contains a key generation component which uses RNG for creating reliable symmetric and asymmetric key. TCG classifies these keys into following 7 types [2].

1. **Endorsement Key (EK)**: Every TPM has a unique Endorsement Key (EK) which is asymmetric and non-migrate-able. It is only used for getting platform ownership and for securing messages associated with AIK creation.

2. **Storage Keys**: These are asymmetric keys used for storing other keys or data securely external to TPM. An important non-migratable storage key is '*Storage Root Key (SRK)'* which forms the basis of Root of Trust for Storage (RTS) (See also Section 3.1.5.2).

3. **Signing Keys**: These are asymmetric keys which can be used for signing application data or messages.

4. **Binding Keys:** These are used to decrypt or unbind small amount of data on a remote entity.

5. **Identity Keys:** These are non-migratable keys and are also called Attestation Identity keys (AIK). These are exclusively used for signing TPM data like PCR register values. AIK attest TPM identity and its security level. AIK also accompany a trusted CA certificate.

6. **Authentication Keys:** These are symmetric keys which are used to secure communication channel with TPM.

7. **Legacy Keys:** These are legacy system keys which are created out of TPM and then imported in the TPM for signing or encryption operations.

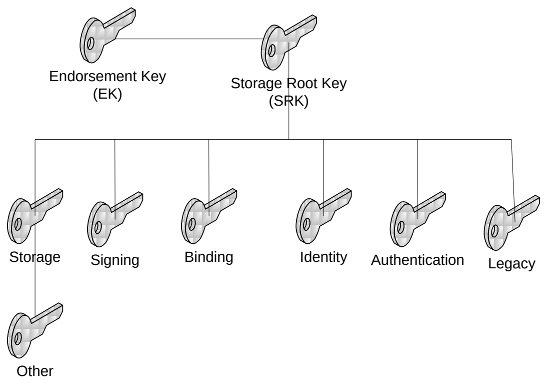


Figure 4 - Key Hierarchy

3.1.4.1 Key Storage

TPM memory, volatile and non-volatile, is not big enough to store too many keys. On the other hand, TCG usage scenarios expect unlimited storage capacity. Therefore, external storage must be used by RTS which packages keys into encrypted key BLOBs which are bound to the TPM and can also be sealed to a particular platform. Keys cached to external storage are 'managed' by Key Cache Manager (KCM) and can be 'used' by applications for encryption/decryption and other services. TPM exposes interfaces for key management and key usage.

3.1.5 Roots of Trust

Roots of trust are the components which are trusted and act as the starting points for overall trust assurance. There are three defined roots of trusts; a root of trust for measurement (RTM), root of trust for storage (RTS) and a root of trust for reporting (RTR).

3.1.5.1 Root of Trust for Measurement (RTM)

This is a computing engine capable of taking reliable integrity measurements [2]. Measurement is done by taking the hash of the entity (an executable, configuration file, data file, etc.). The code to calculate the hash must be stored in a secure and protected location. This is guaranteed by storing this code in PC BIOS or in '*Program Code*' component of the TPM. Since it is the first piece of software which is loaded and measures hash of subsequent softwares, therefore it is considered to be the Core Root of Trust for Measurement (CRTM). RTM is also the root of '*transitive trust'* which is a process in which '*trust boundary'* is extended from root to higher levels. Trust boundary is extended as follows:

1. RTM is trusted implicitly.
2. RTM takes integrity measurements of the next level entity which is done by taking hash of it. This creates 'fingerprint' of the measured entity.
3. [Optional but Recommended] This fingerprint is appended in the Stored Measurement Log (SML) which is stored outside TPM.
4. This fingerprint is then entered in PCR through an extend operation.

5. Control is passed to the next level entity thereby extending the trust boundary. Steps 2 to 5 can be repeated to extend trust from CRTM to user application level.

3.1.5.2 Root of Trust for Storage (RTS)

Root of trust for storage (RTS) is a computing engine which is capable of securely storing data (measured digests) and keys (signing key, binding key, storage key, etc.) [2]. The TPM does not have any large storage capacity to store many keys required for different purposes. Therefore, only the EK and SRK are stored in TPM non-volatile memory. SRK is then used to wrap other keys required for cryptographic operations and these are cached out of TPM. Trust of the whole TPM storage thus lies in SRK. Therefore, storage root key (SRK) forms the basis of root of trust for storage.

3.1.5.3 Root of Trust for Reporting (RTR)

Root of trust for reporting (RTR) is a computing engine which is capable of reliably reporting the information store by RTS. It has two functions; to expose protected locations for storing integrity measurements and to attest the authenticity of the stored information [2]. Attestation identity keys (AIK) are used to digitally sign and report PCR values reliably. Endorsement Key (EK) is used for issuance of AIK which means that EK can be considered to be RTR. EK is stored in the TPM and cannot be stolen which means that it can be trusted.

Attestation Capability

Attestation capability means that the accuracy of information is affirmed by the attesting entity. There are many dimensions of attestation. A challenger can ask for integrity of the platform which can be attested using '*Attestation Protocol'.* Attestation also allows the challenger to know about the authenticity of the platform which signed the message.

3.1.6 Protected Messages

TCG classifies protected messages in 4 categories. These are defined below [2]:

1. **Binding:** Binding is actually encryption of the message using the public key of the recipient. A message is said to be bounded because the corresponding private key is stored in the recipient's TPM (usually in key cache) and it is non-migratable.

2. **Signing:** Signing is also a traditional operation which ensures integrity and authenticity of the message. The signing key is used for this operation.

3. **Sealing:** Sealing is similar to binding in terms of encryption but it also includes platform metrics which must be satisfied to be able to unseal the message. Message is first encrypted using a symmetric key then platform metrics (PCR values) and symmetric key are encrypted with a non-migratable asymmetric key. The recipient can only decrypt message if he has the corresponding private key and system configuration matches the values specified by the sender.

4. **Sealed-Signing:** In sealed-signing, sender also includes PCR values in computing the digest to sign. This can be used to inspect the sender's platform configuration at the time of message creation.

3.1.7 TCG Execution Model

TPM execution/operational states are as follows:

1. **TPM receives power:** The TPM receives power for the first time during manufacturing stage when product is not even shipped to the customer. The Endorsement Key is generated and remains in non-volatile storage of the TPM and never leaves it.
2. **Disabled:** In disabled state, TPM can only report its capabilities and can accept updates to PCR values. All other TPM operations are restricted in disabled state. The TPM can switch states between enabled and disabled.
3. **Enabled:** All features of the TPM are available in enabled state. Usually TPM ownership is taken when it is enabled for the first time.
4. **Deactivated:** Deactivated state is similar to disabled state. However, state changes are allowed in deactivated state.
5. **Activated:** All features of TPM are available in activated state.
6. **Un-owned:** When the TPM is first shipped to the customer, it is in un-owned state. It should be owned after enabling it.
7. **Owned:** TPM ownership is taken by generating a storage root key (SRK). This operation can be performed in enabled or activated state. Ownership can be relinquished and new owner can be set if required.

3.2 TCG Integrity management

The TCG Infrastructure Working Group (IWG) has defined reference architecture for integrity management which is called *´Integrity Management Model'*. This model aims to establish trust in the ecosystem of trusted computing. TCG defines three important characteristics which must be satisfied in order to achieve trust [4] for which IMM is defined.

3.2.1 Unambiguous Identity

A platform is made up of subcomponents which could be hardware or software components. A platform can be declared trustable only if it's all components and the platform itself is unambiguously identifiable. Hardware components can be uniquely identified by their information like manufacturer, model, serial number, etc. whereas software components can uniquely be identified by their hashes. Therefore it is very important for a verifier to get this information from the original source in a secure way. The integrity management model ensures that this information is available to the verifier in pristine form [4].

3.2.2 Unhindered Operations

Any trusted platform component must perform its operations as intended. A platform cannot be trusted if any of its component is tampered, be it hardware or software. IMM ensures that all components which make up the platform are loaded in their original state and if any component is tampered, it can be identified by the verifier [4].

3.2.3 Attestation

Attestation is an action of bearing witness that the platform is trustworthy. A trusted platform must have some mechanism which attests that the platform is trustworthy. IMM provides the mechanism which allows a platform to report its integrity state at any particular time. This means that with the IMM, a verifier can be assured of consistent good behavior of a platform and integrity check can be performed at any time [4].

3.3 Integrity Management Model

The Integrity Management Model (IMM) revolves around integrity measurement and reporting. *Integrity,* which is the core concept in trusted computing, is defined as the pristine state of a component (software or hardware) within a trusted platform (as issued by the component designer/manufacturer), and also of the Trusted Platform as a whole [4]. This pristine state of the component and thus of the Trusted Platform should remain intact during the whole life of the component, from manufacturing stage to the usage stage. The IMM defines *phases* and *processes* which perform integrity measurements, integrity reporting and compare runtime measurements with reference measurements to verify integrity of a platform. The fundamental integrity management functions, measurement and reporting must be provided by the Trusted Platform. The root of trust for measurement (see section 3.1.5.1) and root of trust for reporting (see section 3.1.5.3) constitute the basis to perform these required functionalities. It is worthwhile to mention here that there are two facets of measurement; static or *´reference measurement´* and dynamic or *´runtime measurement´*. The reference measurements are taken and reported by component manufacturer/vendor whereas runtime measurements are calculated by Platform Trust Service (PTS).

PTS is a process running in Trusted Platform which provides interfaces to Integrity Measurement Collector (IMC) or other measurement controlling agent. The PTS provides many capabilities like runtime integrity measurement, canonicalization of integrity measurements, etc. The detailed PTS Interface specification can be found in [6].

Figure 5 extends the basic model of interacting entities - the requester, verifier and the relying party [5] - to portray the core of IMM which is integrity measurement and reporting.

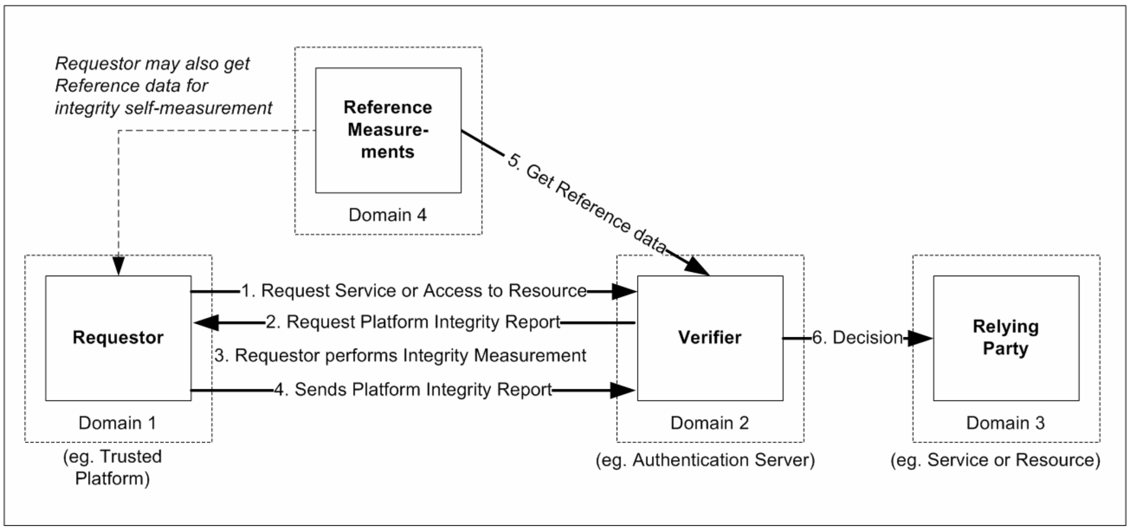

Requestor may also get Reference data for integrity self-measurement
Reference Measure-ments
Domain 4
5. Get Reference data
Requestor
Domain 1
(eg. Trusted Platform)
1. Request Service or Access to Resource
2. Request Platform Integrity Report
3. Requestor performs Integrity Measurement
4. Sends Platform Integrity Report
Verifier
Domain 2
(eg. Authentication Server)
6. Decision
Relying Party
Domain 3
(eg. Service or Resource)

Figure 6 - Core of Integrity Management Model [4]

3.3.1 General Phases and Processes in the Integrity Management Model

The architecture and working of Integrity Management Model can be divided into five chronological phases, each performing some tasks through defined processes. These phases and underlying processes are defined below.

1. Creation

In order to unambiguously identify any component or subcomponent of a trusted platform, its integrity data must be provided by its authoritative creator to achieve required level of trust. Authoritative creator could be component manufacturer or vendor. The creation phase involves *Harvesting Process* which takes raw integrity data as input and creates *Reference Manifest* which is formatted according to TCG Reference Manifest Schema. TCG reference manifest schema follows XML based semantics to achieve interoperability, extensibility and manageability [7]. The harvesting process can be performed by manufacturer/vendor or the manufacturer/vendor can feed raw integrity data to a trusted party who performs the harvesting process. Finally, the reference manifest is signed by the harvesting entity before it is made available for publication.

2. Collection

This phase refers to the collection of integrity data of every component along with the acquisition of physical components to assemble a Trusted Platform. The resulting collection of integrity data must be semantically meaningful to derive trust assertions. This collection is then made available to other parties for verification either by Trusted Platform manufacturer (e.g. OEM) or by a designated third party. In this phase, the '*Collection Process*' is responsible for collecting harvested integrity data which is submitted by the harvesting entity using submission interface (*IF-Submit)*. This step is deemed helpful because otherwise in order to verify a platform, the verifier would have to contact all vendors/manufacturers of subcomponents of the platform which could be very cumbersome. A third-party can perform this aggregation service and then communicate this integrity data (see phase 3) thus providing single point of access for the verifier.

3. Communication/Publication

The integrity data created in phase 1 (for individual components) and collected in phase 2 (for Trusted Platform) must be communicated to the verifier who wants to verify the integrity of a platform or a component. This can be done in number of ways. The component or Trusted Platform manufacturer who created and collected the data could publish it and verifiers can access that data for verification. Another approach could be that Integrity data is sent to a verifier along with the component on CD or by some other means. The most recommended approach is to communicate integrity data by a designated authority called Reference Authority (RA) who can publish this data for verifiers. This approach is helpful in a way that when a verifier wants to access data for multiple subcomponents, he is not required to browse through different points of access rather he can get the integrity data from single point. Hence RA acts as the aggregator.

In this phase, the '*Publication Process'* is responsible to publish the record in reference manifest database for verifier through publish interface (*IF-Publish*). The publishing entity is usually called

Reference Manifest Authority or Integrity Authority because it digitally signs the reference manifest records in the publication process.

4. Collation

The Collation phase refers to the gathering of integrity data of all components of a Trusted Platform in order to verify it by comparing it with runtime integrity measurements. This phase is different than collection phase in respect to the purpose of gathering such information. The collection is performed to eventually publish integrity data for verification whereas collation is performed to verify integrity data against runtime integrity measurements.

5. Evaluation

The Evaluation phase deals with ascertaining the integrity state of a platform by verifier. The verifier can do evaluation for himself or for some relying party. Therefore, the role of the verifier is use case dependent. The evaluation is done by *Verification Process* which compares runtime integrity measurements with reference measurements. The runtime measurements are taken by *PTS Process* and reference measurements are retrieved by the Integrity Measurement Collector (IMC) using IF-Publish interface.

Figure 7 summarizes the tasks performed during each phase of the Integrity Management Model.

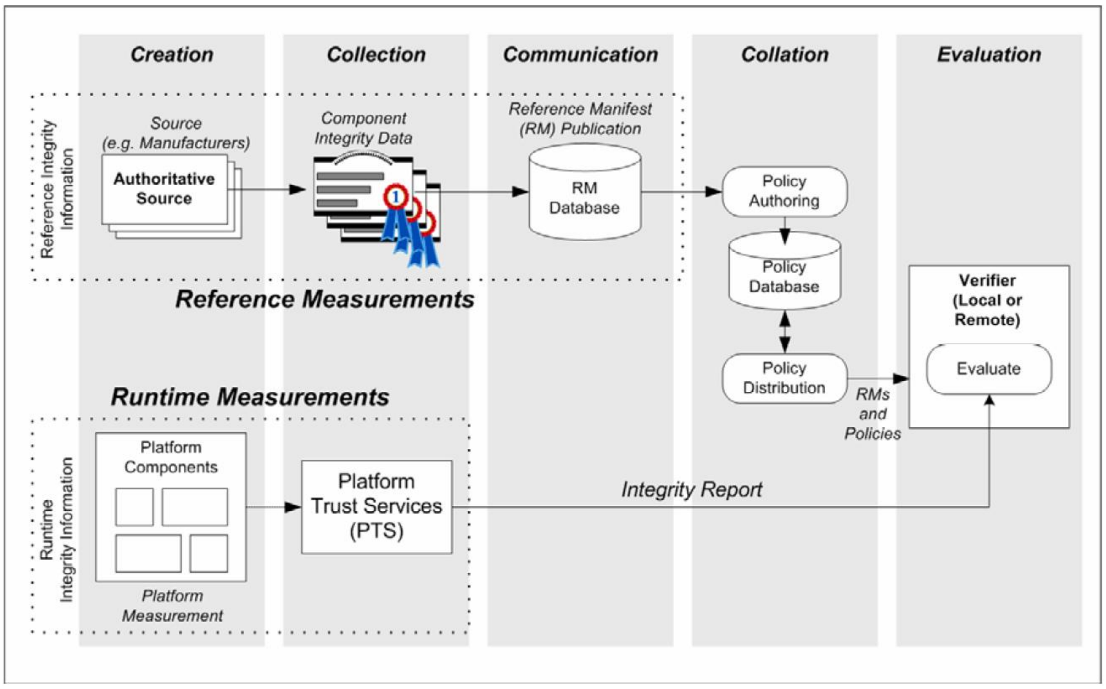


Figure 7 – The Integrity Management Model – General Phases [4]

Figure 8 shows the significant processes which perform the major task of each phase of IMM. The figure also shows supported interfaces for entities in different domains to communicate with each other.

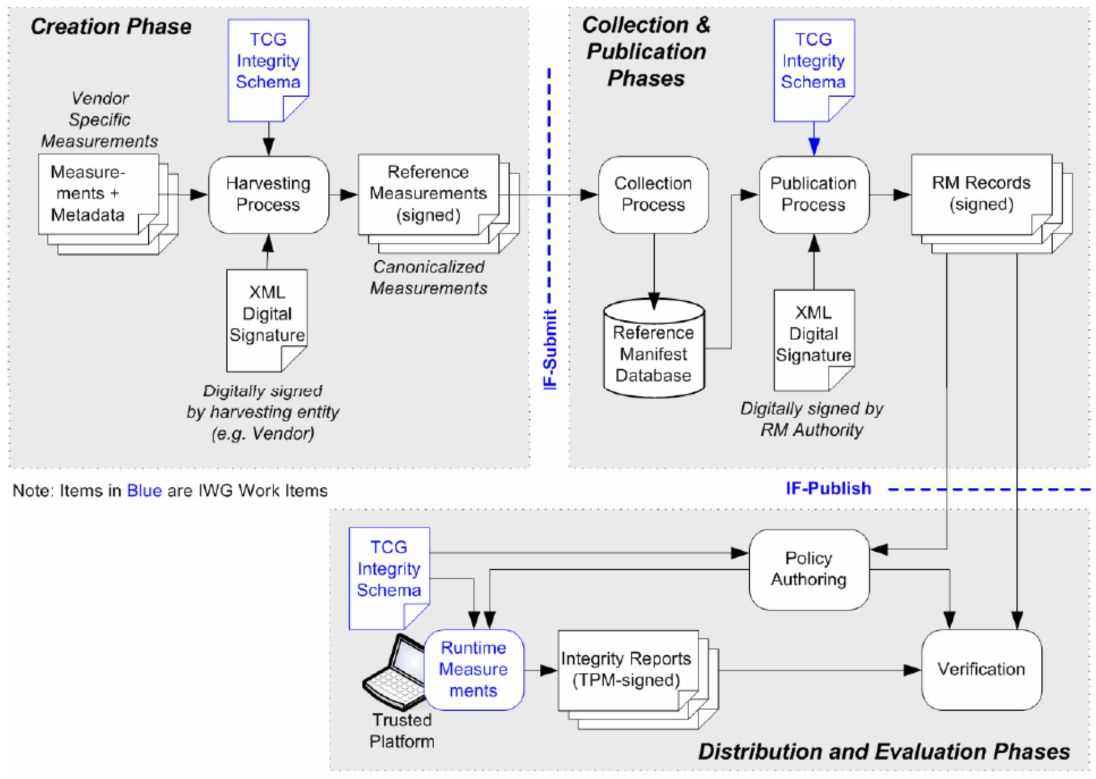

Creation Phase
TCG Integrity Schema
Vendor Specific Measurements
Measure-ments + Metadata
Harvesting Process
Reference Measurements (signed)
Canonicalized Measurements
XML Digital Signature
Digitally signed by harvesting entity (e.g. Vendor)
IF-Submit
Collection & Publication Phases
TCG Integrity Schema
Collection Process
Publication Process
RM Records (signed)
Reference Manifest Database
XML Digital Signature
Digitally signed by RM Authority
Note: Items in Blue are IWG Work Items
IF-Publish
TCG Integrity Schema
Policy Authoring
Runtime Measure ments
Integrity Reports (TPM-signed)
Verification
Trusted Platform
Distribution and Evaluation Phases

Figure 8 – The Integrity Management Model – Significant Processes [4]

3.4 TCG based resource management in virtualized systems

The TCG based resource management in virtualized systems necessitates a ´*Trusted Virtual Platform'*. The conventional non-virtualized environments can use typically only available TPM device to set up a Trusted Platform. But in a virtualized environment, a single TPM device may not work because like all other shared hardware devices, VMs would need (if required) to use a TPM to avail its services. Such a requirement would need a virtual instance of TPM as well just like virtual CPU, virtual Hard-disk etc. We can then dedicate a virtual TPM for a VM to render its services. Based on this discussion, we conclude that the depending upon the requirements, we can use TPM in two ways in virtualized environments to establish trust which we refer to as *Implicit Trust Establishment* and *Explicit Trust Establishment.*

In ***Implicit Trust Establishment***, we limit the use of TPM to 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 VM hypervisor. Every service/software running in the launched VM would trust implicitly that VM itself was launched securely.

In ***Explicit Trust Establishment***, instances of virtual TPMs [8] would be required to render services to every VM. This approach would allow the use of all TPM features by every VM, mandating it to report integrity measurements and platform attestation to the challenger.

The approach to adopt for trust establishment depends upon the scenario and stakeholders' concern. In our scenario, the provider and the operators are the main stakeholders. The operator's obvious concerns regarding *trust* would be that he is using trusted resources which could not be faked. The provider's concern is that the VMs launched for the operator should remain isolated and if operator VM is compromised, it should not affect the trustworthiness of the whole platform. This is similar to the model in [9] where the author suggests that the "traditional" TCG based attestation mechanism (see also Section 3.4.2.2) are used to securely authenticate platform configuration up until the boot of the VMM layer. Once the VMM is running instead of verifying running binaries, the VMM is responsible for attest VM properties (or what the author call semantic attestation). This model suits also the scenario we are working with and that speaks in favor for of using implicit trust establishment. It might also be worth looking into whether we could adopt attestation of some of the semantics properties suggest in [9].

The TCG based abstract solutions to the problems and scenarios discussed for secure management of trusted virtualized platforms (see section 2) will be discussed in following deployment phases.

3.4.1 Trusted Platform Configuration

The platform configuration phase includes setting up the whole environment. Here, we could make use of the Integrity Management Model presented in section 3.3. The design choice, to define entities performing all the processes required for IMM to work, is flexible which allows us to perform most of the configuration process within provider's domain in order to minimize dependency on other parties. This is required because we do not expect any entity performing these tasks (mentioned by IMM architecture) for system to work as yet. The proposed system design overview is shown in Figure 9 and is explained below.

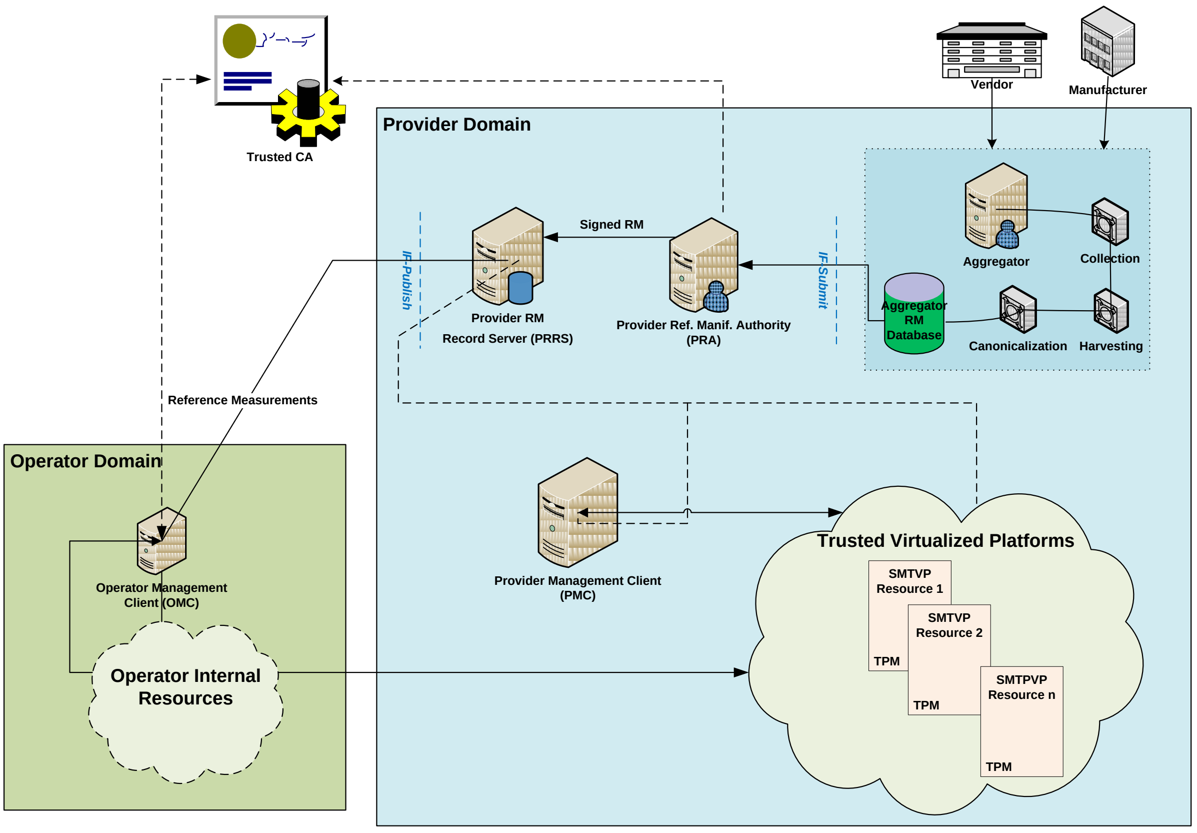


Figure 9 - Proposed System Design Overview

Following sequence of actions are performed before Trusted Virtualized Platform(s) could be deployed:

1. The provider (IT Purchase/Acquisition department) collects the raw integrity measurements, like hash values for software components and serial#, manufacturer id, etc. for hardware components, from manufacturers who are authoritative source of these measurements.

2. These measurements are processed and canonicalized according to TCG Integrity Schema and are stored in Reference Manifest Database (RM DB).

3. The Provider Reference Manifest Authority (PRA) collects RM documents using *IF-Submit* interface, signs them and publishes them through Provider RM Record Server (PRRS). The operator can get these signed reference measurements using *IF-Publish* interface exposed by PRRS.

4. Trust between PRA and Operator is established by a common trusted CA.

Another important security configuration step is platform Identity Registration. In section 3.2.1, we have discussed the significance of unambiguous identity for a platform to acknowledge it trustworthiness. Therefore, the platform is registered with the provider infrastructure administrator. The infrastructure administrator who is also registered with Trusted Third Party, issues AIK-based identity certificate called AIK-Credential which asserts that a given platform is a Trusted Platform [5]. The generalized process of Identity Registration is depicted in Figure 10.

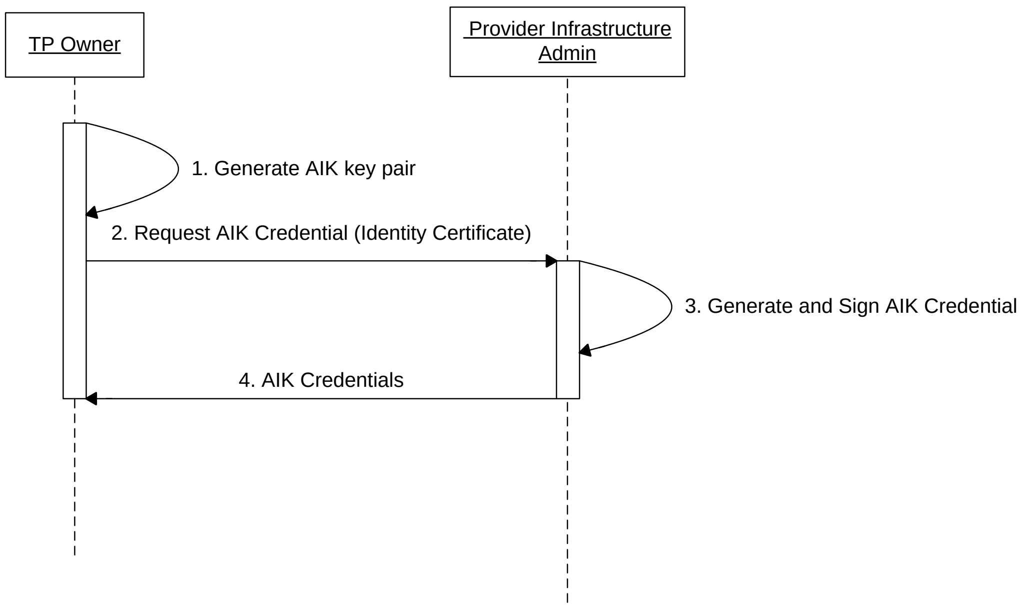

TP Owner
Provider Infrastructure Admin
1. Generate AIK key pair
2. Request AIK Credential (Identity Certificate)
3. Generate and Sign AIK Credential
4. AIK Credentials

Figure 10 - Identity Registration

The operator could also be registered at a Trusted Third Party so that mutual authentication can be performed. The registration scenario is depicted in **Figure 11**.

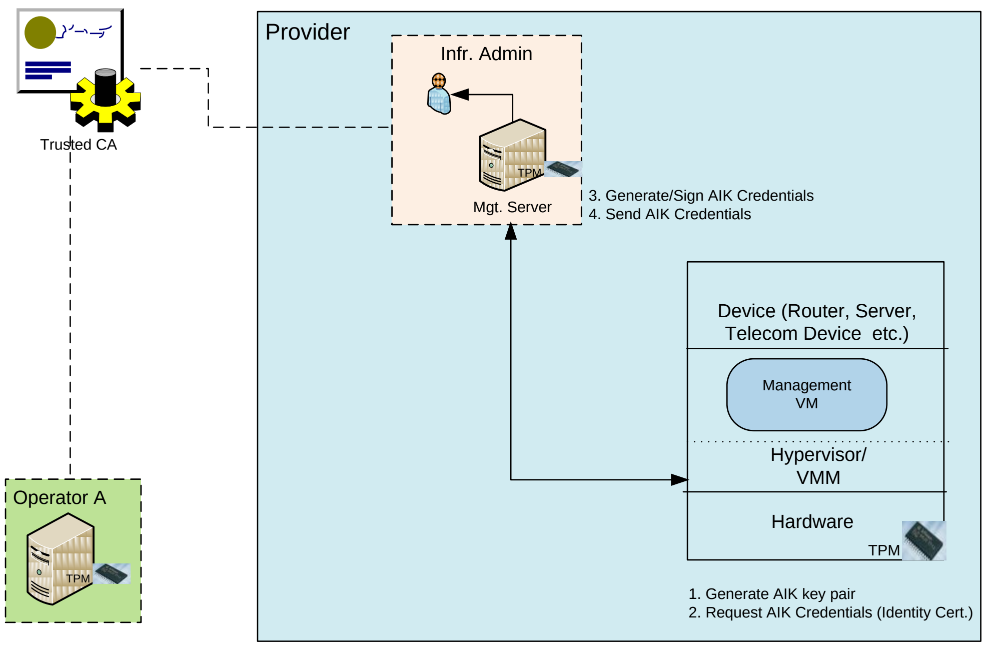


Figure 11 – Trusted Platform Identity Registration

The steps involved in Identity Registration are described below:

1. **Generate AIK Key Pair:** Attestation Identity Keys (AIK) can be generated by the owner of the platform. The AIK is an asymmetric, non-migratable key (see section 3.1.4) which is generated by the platform owner.

2. **Request AIK Credentials:** The platform, whose identity needs to be registered, sends the public part of the AIK key, EK Public, Platform Credentials and EK Credential to the provider infrastructure administrator. This request is encrypted with a symmetric key (which is encrypted with Public key of the infrastructure administrator).

3. **Generate/Sign AIK Credential:** AIK Credential is an Identity Certificate which binds the identity of the Trusted Platform to AIK. Before generating the AIK Certificate for the platform, the infrastructure administrator verifies signatures over platform credential and EK credential. The inf. admin. generates the AIK Credential and sends it back to the platform.

4. **Send AIK Credential:** The generated AIK Credential (Identity Certificate) is sent encrypted to the platform. The encryption is done using an EK encrypted symmetric key which can only be obtained by a valid platform containing the TPM which originated the AIK request.

5. **Trust Establishment:** The inf. admin. and the operator are registered with Trusted CA. This means that the identity certificate presented by TP to the operator (signed by inf. admin.) can be verified by the operator.

3.4.2 Trusted Platform Deployment

With reference to the Trusted Platform deployment, following use cases can contribute to secure management of these platforms.

3.4.2.1 Secure Boot

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. The concept of '*secure boot'* means that during the boot process, each component to be loaded is measured and verified against reference measurements before it is loaded. This requires secure verification code and protected reference measurements to be available to the Trusted Platform for boot and verification purposes. The reference measurements can be stored in non-volatile TPM storage which can only be accessed by trusted verification code. The trust of '*measure-verify-load*' code can be rooted in CRTM and extended in the boot-up process (see section 3.1.5.1).

The step-wise secure boot process is listed below and is depicted in Figure 12.

1. When system is switched on, it resets PCR.

2. The trusted CRTM code is loaded. The CRTM can for example be the extension of a normal BIOS or dedicated ROM code. The CRTM measures the system firmware code, performs PCR Extend operation and execution control is transferred to the system firmware.

3. The system firmware measures VMM/Hypervisor, performs PCR Extend operation and transfers control to the VMM.

4. The VMM launches a management VM (Kernel, libraries, files, applications etc.) measures and extend PCRs. The process of measurement, extend operation and control transfer is performed in sequence. All OS measurements are also saved in SML.

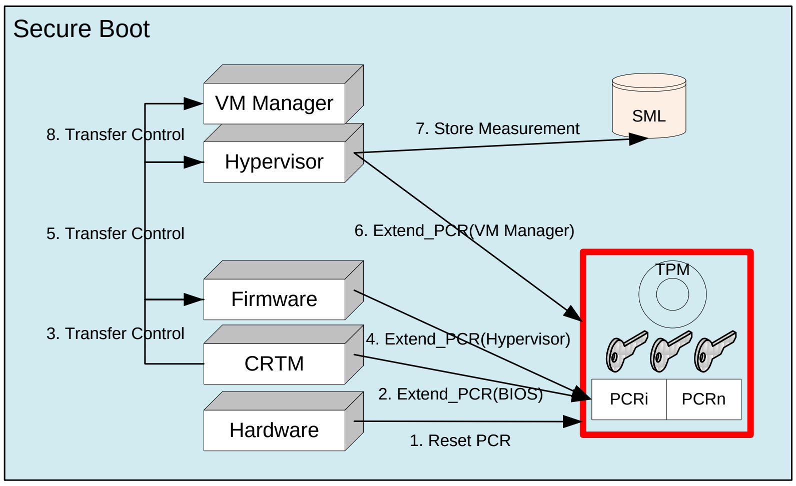


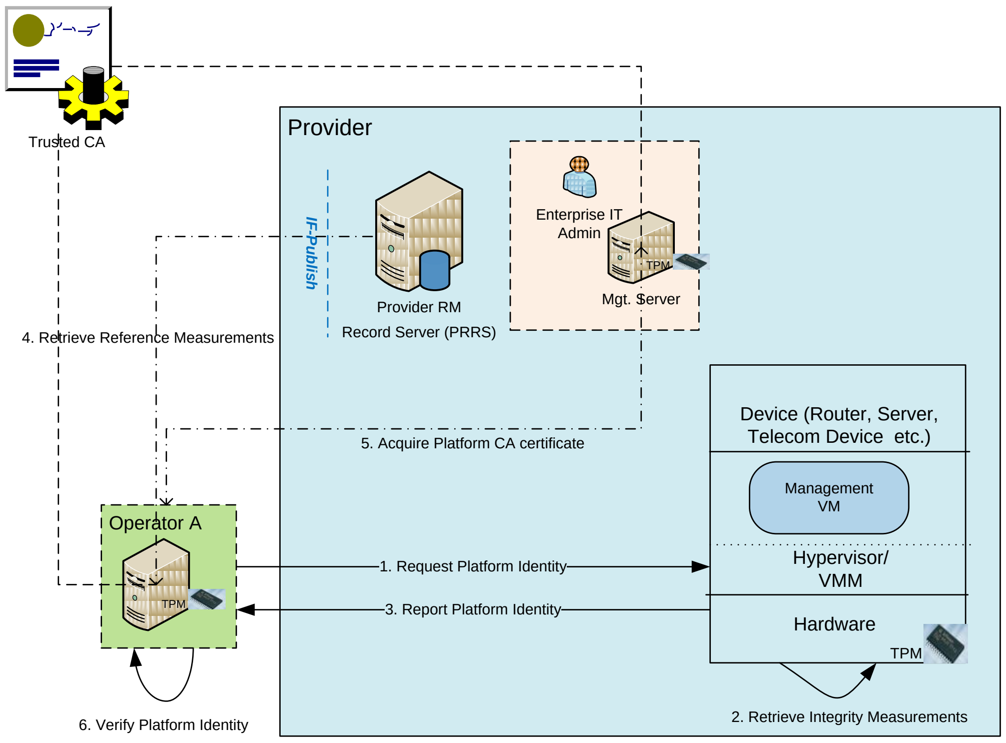
Figure 12 - TPM based Secure Boot

As discussed earlier, we are in the current study focusing on implicit trust establishment. This means that the integrity of VM Manager and every piece of code running below it can be attested. Consequently, every VM launched later would be trusted implicitly. From provider's and operator's point of view, this approach suffices 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).

3.4.2.2 Platform Authentication/Remote Attestation

When an operator wants to use resources (on a platform) provisioned by the provider, he is interested to authenticate that he is communicating with an authentic platform. Ideally, mutual authentication should be performed before establishing an actual session.

In the context of Trusted Computing, platform authentication or remote attestation relies on the fundamental features of a trusted platform namely; protected capabilities, attestation, integrity measurement and reporting [1]. Authentication of a Trusted Platform is possible only after registering the platform identity with a Platform CA. In the process of authentication, the Trusted Platform reports its PCR values to the verifier. The reported values are signed by the trusted platform using AIK key found in the TPM. On the other side, the verifier can validate the AIK credential (Identity Certificate of the Trusted Platform) by evaluating the signature of the Trusted CA to verify the integrity of the reported measurements. The outcome of the verification process is not limited to binary SUCCESS/FAIL rather the reported measurements could be interpreted on some pre-defined level of integrity called Trust Score (e.g. 1-100) and authentication could be granted if a specific threshold, as specified in the policy, is met. The process of platform authentication is depicted in Figure 13.



Trusted CA
Provider
IF-Publish
Provider RM
Record Server (PRRS)
Enterprise IT Admin
TPM
Mgt. Server
4. Retrieve Reference Measurements
5. Acquire Platform CA certificate
Device (Router, Server, Telecom Device etc.)
Management VM
Hypervisor/ VMM
Hardware
TPM
Operator A
TPM
1. Request Platform Identity
3. Report Platform Identity
6. Verify Platform Identity
2. Retrieve Integrity Measurements

Figure 13 - Platform Authentication

Following steps are performed for Platform Authentication.

1. **Request Platform Identity:** Identity request may contain set of PCR values required for authentication which depends upon the already stated policy. Platform identity request is essentially a request for the TP to report its configurations.

2. **Retrieve Integrity Measurements:** A TCG based Trusted Platform measures and stores integrity measurements in booting process. The measurements are logged in *Stored Measurement Log* (SML) and their digests are stored in PCRs. The TP retrieves SML and requests TPM to return signed PCRs.

3. **Report Platform Identity:** The identity of a Trusted Platform contains the retrieved integrity measurements in step 2 and AIK Credential to validate these measurements. A typical identity message looks like { Sig_{AIK} (PCR, N), SML, AIK_{cert} }.

4. **Verify Platform Identity:** The final step to authenticate a TP is verification of the Identity information. The authenticator first verifies the AIK Credentials by public key of the Platform CA. He then uses AIK to check signature on PCR values. The authenticator then validates the authenticity of reported measurements by calculating digest of SML and comparing with received signed PCR values. Finally, the authentication decision is taken after analyzing verification results and mapping them against the policy.

3.4.2.3 Virtual Machine Migration

VMs installed on virtualized equipment often needs to have access to sensitive information such as secret keys. *One* potential way of protecting this information is to define them as migratable TCG keys. The TCG infrastructure group has defined procedures and Web Services interfaces for secure key migration in administrative IT domains [10]. These mechanisms *might* also be used to protect VM credentials at VM installation and migration. At least we will investigate different solutions and options in the projects.

4 Overview of other important prior-art

Next, we give an overview of important related prior-art research papers. The main concept in the different papers is briefly described and we also discuss the applicability to the usage scenario we address in the project.

4.1 Terra: A Virtual Machine-Based Platform for Trusted Computing

Terra presents a trusted computing architecture using virtualization technique [12]. This work identifies the security related problems in non-virtualized platforms and addressed them using the concept of Trusted Virtual Machine Monitor (TVMM). TVMM allows Terra to give appearance of multiple boxes on a single hardware platform and run applications of varying assurance levels in appropriate boxes. The provision of open-box allows traditional OS and applications to run whereas closed-box allows only trusted applications to run. This model makes it similar to NGSCB [11] initiative which runs the platform in *Trusted Mode* (Nexus) and *Un-trusted Mode.* The difference between NGSCB and Terra is that NGSCB launches two VMs whereas Terra is flexible to launch many Trusted and Un-Trusted VMs.

The Terra architecture features attestation capability which is a valuable primitive for building secure distributed systems. The attestation is performed by building a certificate chain which starts from temper-resistant hardware which contains the private key embedded in the hardware. Next certificate is generated after receiving hash of the next level application (For example, the system firmware) attestable data. This hash is stored in the common name field of the certificate and the public key of the next level application is bound to this certificate. In this way, the tamper-resistant hardware certifies the system firmware; the firmware certifies the system boot loader, which certifies the TVMM, which in turn certifies the VMs that it loads.

The security model of Terra is based upon two components, TVMM and Management VM, each performing its designated security task. The TVMM is "root secure" which means that it is protected also from the platform owner who has root level access. Roles of TVMM include dictating policy for attestation, isolation of VMs from one another. The Management VM is responsible to start, stop and suspend other VMs. It also controls number of maximum VMs, sharing of resources (CPU, memory, Storage, etc.) among these VMs and management policies for these resources. It also defines platform access control and is responsible for distinguishing between Platform Owner and Platform User.

The Terra architecture allows applications to determine their level of assurance by running in securely isolated and application tailored OS running in a VM. The isolation property of VMs ensures that if an application is compromised, it does not affect the security of the whole platform. The assurance level of application running in an OS greatly depends upon the assurance of OS itself. Moreover, there are varieties of applications with different levels of required assurance. For example, an application could require very high level of assurance but may not require extensive feature-set of the underlying OS. In such a case, a minimal OS with less complexity and thus high assurance level, would be appropriate for the required application. Similarly, an application with moderate assurance requirements but high OS feature-set requirements would prefer a general purpose complex OS. The Terra architecture addresses these issues by allowing applications to run in application tailored OS thus providing a platform running multiple applications of varying levels of assurance in different VMs. An example model could be a closed-box VM running trusted OS which allows only specific applications to run and an open-box VM running traditional OS. This resulting model resembles NGSCB initiative.

The integrity measurement concept described in Terra architecture is somehow similar to TCG based integrity measurement mechanism presented in section 3.1.5.1. The major difference between the two is that IMM extensively relies on TPM device which forms the basis of roots of trust. The attestation mechanism in Terra uses certificate chains containing hashes of each level application as against PCRs used in IMM attestation process. Moreover, Terra architecture does not talk about reference integrity measurements which should be used to compare runtime integrity measurements. Terra assumes that the verifier already has the list of hashes of the applications which are whitelisted or blacklisted. The concept that TVMM is responsible to attest VMs to remote parties is similar to our proposed approach of *Implicit Trust Establishment* (see section 3.4).

The Terra architecture is a potential alternative to a TCG based architecture for protecting the virtualized platforms in our scenario, both as trusted boot and remote attestation. The TVMM as defined in the Terra architecture is *not a particular* VMM (they used a VM ware VMM in their own prototype), but the authors assume that it is possible to use a thin well analyzed VMM or hypervisor to support the architecture. In particular, the management and attestation interface proposed in the Terra architecture should be evaluated in our project.

4.2 Trusted Virtual Domains: Toward Secure Distributed Services

An abstract framework for secure distributed services is proposed in [13]. A basic concept in the architecture is so called Trusted Virtual Domain (TVDs). The principal parties forming a TVD include an *Initiator* and a *Responder.* TVD could be created for the initiator who requests for a sensitive but heavy computation to the responder, or it could be created for responder who offers its services by using state-of-the-art technologies. In both scenarios, the initiator can only rely on the responder if it can satisfactorily establish trust remotely. TVD realizes this by featuring variety of attestations which guarantee the security, functional and quality requirements of the initiator. These requirements are stated in the policy which is agreed upon by all members of the TVD.

Conceptually, TVD is an abstract union of initiator and responder(s) which join this union after agreeing to a set of mutual requirements. Both parties should have basic three components to establish a Trusted Virtual Domain (TVD). These include Mutually-Trusted Computing Base (MTCB), Attesting Virtual Environments (AVEs) and Execution Entities (EEs). The MTCB provides the basis for negotiating both parties about the identity and integrity of their platforms and thus authenticating each other. MTCB can be established by using secure hardware extensions like TPM which provides the root of trust. The AVEs work with underlying hardware to create environment which can make attestations for the initiator. AVEs form the basis for controlling dynamics of offered services. EEs are actual responders which perform the service for initiator. There can be more than one EE per AVE. Execution Entities provide the basis for service execution.

Trusted Virtual Domains are established in sequence of steps. First, an initiator creates its own TVD and identifies what task or role is desired. It then locates role taking responders and contacts the appropriate responder to perform the desired task. The initiator request includes the security requirements that must be fulfilled in order to establish MTCB. Similarly, responder can also specify its requirements to establish MTCB. Both entities need to generate attestations that satisfy mutual requirements in order to proceed as normal. The security requirements, which must be attested to establish MTCB, could arise from initiator and/or responder, or these could be pre-defined. The specification of such requirements which are abstract enough to be understood by system administrators and concrete enough to be enforced is presented as a challenge. However, this paper lists potentially attestable properties. These include data-related properties (confidentiality, integrity, etc.), processing-related properties (availability, cost and metering, auditing, etc.) and environment-related properties (redundancy, identity, administration, accountability). These properties could be specified in requirements and attested by the involved entities to setup a TVD.

In future, more dynamic, virtualized telecommunication networks, the TVD concept should be evaluated for example as a potential extension to a more basic TCG based security architecture. However, TVDs will not be a prioritized topic in the project.

4.3 TVD – Secure Foundations for Business and IT Services

The IBM research report on Trusted Virtual Domains [14] elaborates the concept presented in the related position paper on the same topic [13]. The motivation for this work comes from the indispensable requirement of developing secure, service oriented, on-demand distributed framework to address the needs of the future. According to the authors, the current distributed systems lack three important properties which hinder their adaptability to the future needs. These include *containment*, *trust* and *simplification* of security management. The TVD architecture uses state-of-the-art technologies to extend these features. Containment is achieved by using virtualization and overlay technologies providing strong isolation between different execution environments. Trust is established by leveraging trusted computing technologies which allows different entities in the TVD to convey trust evaluations and guarantees. Simplified security management is achieved by defining TVD policy at abstract level which can be methodically decomposed, enforced and verified. The policy statements may involve functional requirements (containment and access control), quality requirements specifying minimum level of acceptable trust assurances and implementations and mechanisms which map functional and quality requirements onto a real system. All the members of TVD must adhere strictly to the policy for that TVD before becoming member of that domain. This means that mechanisms to verify other TVD members must exist which are achieved by leveraging trusted computing technologies. The TVD architecture also requires components which support policy enforcement, TVD management and TVD operation. Policy enforcement is done by the nodes (physical platforms) which host containers (coordinates the platform security) containing execution environment (which performs the service).

The core TVD components which are containers and EEs provide multi-implementation-compatible APIs which allow their security management. EEs communicate with each other using these interfaces and are managed by containers which also use exposed interfaces. Containers are managed by domain controllers using container-exposed APIs. TVD-aware applications use these APIs to perform management and execution of TVD features.

TVD promises to provide On Demand control and metering across virtual enterprises, enabling a true eUtility model acceptable to all stakeholders. However, realizing full potential of TVD poses obvious research challenges which include achieving scalability, policy specification and management and maintaining simplicity. The concept of centralized domain controller might not scale well in a TVD in which many members forward their attestations to the central domain controller. Another research challenge is to define a way to specify and manage the operational policy which is a key to TVD framework. Finally, TVD users and administrators would desire to experience simple usage and administration. Providing a user-friendly solution which allows simple system configuration, operation and management is a challenge to be explored.

This work presents a broad and abstract architectural overview which provides the basis for solution for many scenarios presenting virtual resource provisioning. This allows the TVD architectural to map it to our scenario in which a TVD node could be interpreted as provider's virtualized hardware with TPM support, container could be mapped with a trusted secure hypervisor and Execution Environment could be mapped to isolated virtual machines provisioning virtualized resourcing to the operators. The operator in our context could be initiator and the provider could be responder. TVD does not explicitly depend upon TCG technologies but it presents a general infrastructure which could leverage TCG technologies for trust guarantees. However, as for the TVD concept in general, we do not foresee any work in this direction with any high priority.

4.4 An Open Trusted Computing Architecture – Secure Virtual Machines Enabling User-Defined Policy Enforcement

In the research report [15] secure virtualization technology solutions are discussed. The proposed architecture, called OpenTC, is an open and scalable solution leveraging virtualization and trusted computing technologies. It uses virtualization to achieve VM isolation and TCG technologies to establish trust and to perform platform attestations. The unique features of OpenTC are: Verifiable security using TCG technologies, support of multiple and different hypervisors and flexibility by means of configurable policies.

OpenTC [15] defines five layers of abstraction to achieve desired features. The lowest layer forms the actual hardware which is virtualized by the next layer called *virtualization layer.* Virtualization layer provides virtual machines and their basic policy enforcements capabilities. Xen and L4 hypervisors are extended to support security features. The security features provided by virtualization layer are verifiable by remote entities by means of trusted computing technologies. Virtualization layer exposes basic management interface (BMI) to the next higher layer called *Security Services Layer*. The security services layer primarily enforces security policies using multiple services. These services include *compartment security manager* which is responsible for enforcing security policy for a particular compartment and verifying selected security properties to remote peers. Other services running at security services layer are: *user security manager* which manages individual users and their roles; and *secure device virtualization*, which manages virtualizeable hardware.

The *Virtual Machine Layer* provides the execution environment for applications running in *Application Layer.* Virtual Machine layer contains application oriented VMs and management VM running management applications which can be used to interact and maintain the platform.

Like any other trusted computing based platform, OpenTC architecture allows trust management which is unique in the sense that it is based upon property-based attestations instead of cryptographic checksums. This distinction fulfils the scalability requirement of the any distributed system because it is practically infeasible for every system in the distributed environment to have complete list of software hashes to be able to compare them to evaluate attestations. Moreover, this approach addresses the privacy issues of exposing system status to remote parties. In the property-based attestation, the requester questions about the properties of the responder who responds with property attestation.

The open TC architecture is interesting in relation to the use case we address above all regarding policy handing in virtualized environments. Especially, policy handling with respect to VM launch and migration and should be evaluated by the project when designing the security mechanisms for these parts of the architecture.

4.5 TVDc: Managing Security in the Trusted Virtual Datacenter

This paper presents a 'multiple workload collocation architecture' on shared physical resources called *Trusted Virtual Datacenter (TVDc)[16]* which leverages virtualization and trusted computing technologies to provide isolation and integrity guarantees. The TVDc constitutes various components which include Trusted Platform Module (TPM) [3], the IBM virtual TPM [8], the IBM secure hypervisor (sHype) [17] and systems management software. The TVDc also uses TVD abstractions presented in [13].

The main focus of TVDc is to address added security concerns arising from the nature of virtualized environments. These include strong isolation protection and integrity guarantees required against malicious software penetrating into the virtualized platforms. The TVDc also focuses on the security management challenges which are compounded due to complexity of possible scenarios arising from such environments; like running a workload on different servers where each server is again shared for other possibly adversarial workloads.

The TVDc architecture solves these issues by utilizing state-of-the-art components mentioned in the preceding paragraph. The security management is simplified by using Trusted Virtual Domain (TVD) abstractions which allow security manager to define security policy at TVD level. Other components adhering to the active policy can then join this TVD. This approach simplifies resource assignment process and makes sure that a shared resource is not mistakenly re-assigned to another workload. Within a TVD, different VMs running on different physical platforms can communicate and share resources assigned to them without any restriction from the underlying hypervisor.

The strong isolation of a workload is critically important for its owner. The TVDc uses sHype to guarantee strong isolation based on workloads. The sHype is built upon the existing isolation provided by the core hypervisor. It controls access to inter-VM communication and resources according to the active policy. Inter-VM communication is controlled by means of access mediation hooks located inside core hypervisor. Access to resources is controlled by similar access mediation hooks located inside management VM. All access decisions requests are sent by these hooks to Access Control Module (ACM) which is implemented in the core hypervisor. The sHype overrides classical VM isolation mechanism to workload isolation which makes it easy to define and manage security policy by keeping focus on an individual workload. The access control policy used by sHype has two parts: (1) Label definitions and (2) Anti-collocation definitions. A security label is composed of a reference name and a set of types. Every VM and resource is assigned one security label reference name. The decision to grant or reject access is taken after looking at the label types of the subject and object. ACM grants access if common label type is found otherwise access is denied. Security labels are also used to authorize VMs to run on the system. A VM is allowed to run if its security label is subset of the security label of VMM. The other part of the security policy is anti-collocation definitions which enforce restrictions on VMs to run simultaneously on same system. This security policy can also provide network isolation (with virtual LANs) by following the same access principle as described above. For example, a VM can access a virtual LAN if both share a common security label.

The trust establishment in TVDc is based upon TCG integrity measurement and attestation process. The TVDc uses extended Grub bootloader to measure and report Grub configuration file and kernel to PCR in the TPM. The TVDc also uses Linux Integrity Management Architecture [18] to measure applications, driver code etc. , extend these SHA1 hashes to PCR and log them for remote attestation process. A remote challenger wanting to determine the trustworthiness of TVDc requests for the state of PCR and log of all loaded applications. After receiving TPM signed response from the challenged system, the challenger recalculates the SHA1 hash using the received hash-log and compares with the TPM reported hash (in PCR). A match of such comparison would only mean that the challenger has got an unchanged log report of every loaded code (kernel, drivers, applications, etc.). This step follows the verification step which requires a large collection of hashes of every kind of potential code running on the platform. The TVDc uses a central database containing those reference hashes along with annotations which describe the quality of that code. This database is accessible to challenger and the system manager to perform verification of system health. In TVDc, remote attestations can be done by every VM running on the platform by virtue of virtual TPMs [8]. A virtual TPM is software implementation of TPM specification and is based upon physical TPM. Virtual TPMs are loaded and managed in Management VM using vTPM Manager. Any host VM can use its vTPM instance to use TPM functionality which allows TPM-aware applications to run in host VMs.

The TVDc architecture has similarities with our architecture design but differs slightly in some functionalities and requirements. The TVDc uses TCG based trust establishment but does not discuss about the formats of reference measurements. A datacenter-oriented architecture is essentially required to follow some standards (e.g. TCG integrity schema, IF-Publish interface, etc.) for scalability and interoperability. The TVDc could therefore provide good proprietary security management mechanism but it lacks some open-architecture requirements. Comparing with our architecture, one main difference of trust establishment is that we follow implicit trust establishment according to our scenario requirements as opposed to explicit trust establishment used by TVDc. The remote management of the trusted virtualized platform done by IBM System Director is similar to Provider Management Client (PMC) in our architecture which performs the same role. As discussed earlier, contrary to the TVDc, our architecture follows the TCG specifications for providing reference measurements in standardized formats through standard interfaces. Our architecture also includes trusted CA which establishes trust between the challenger and the challenged platform.

4.6 Commercial Products

There are some commercial products featuring virtualization and trusted computing technologies which offer user and enterprise solutions. Key products come from Infineon Technologies, HP Technologies and VMware.

The HP management solutions software is datacenter-oriented and reaps the *business benefits* using virtualization. The solution pack comes from industry players in virtualization technologies including HP, VMware and nworks. The VMware provides the virtualization layer and HP provides virtualization management solutions running on VMware® Infrastructure 3. The management functionalities offered by these solutions is generally restricted to *operations management* targeted for IT managers and CIOs. Although such management solution is also necessary due to the inherent complexity of the virtual datacenters but there is also a whole new range of security management requirements arising due to the nature of virtualized environments and their remote usage. There is very little focus on security management in the management solutions provided by the HP which manifest the research work required in this area.

The Infineon Technologies specializes in embedded security solutions and is one of the main manufacturers of TPM hardware modules. Other than many chip card security solutions, Infineon Technologies offers Trusted Computing Management Server (TCMS) which enables enterprise deployment of trusted computing clients by providing automated and centralized management [19].

TCMS allows to perform administrative tasks on the remote TPM-equipped platforms. The TCMS is not targeted for security managements rather it supports the remote platform administration *securely*.

Some other similar solutions might exist which focus on virtualization management but security management in virtualized environments is relatively new area which requires further research before a commercial product for secure management of virtualized platforms is released.

5 Conclusions

In this report we have discussed the scenario in which virtualized telecom equipment is provisioned to the operators with the focus on security management and trust establishment. We have explored and presented an overview of the relevant trusted computing support with the motive of leveraging TCG technologies for trust establishment and secure management of virtualized platforms where applicable. We then presented an overview of the high level working system design which is mainly composed of configuration and deployment phases.

In the prior art section, we explored relevant literature and commercial solutions with a short reflection of each work on our state-of-the-art report. We have seen that a considerable amount of work has been done to address security problems using virtualization and trusted computing technologies. However, other than Trusted Virtual Data Center (TVDc), we could not find a complete architecture which could knit together all existing state-of-the-art technologies to propose a secure and trusted virtualized resource provisioning system. We presented a short comparison of the TVDc and our architecture and found some differences in the requirements and the followed design approach. In other prior art, we found some directions in OpenTC which need to be explored further to address policy specification and enforcement in our scenario.

The next step for this study is to present a detailed design for which we will dig into the individual components of the presented high level design. This includes actual protocols specification, defining policy and its enforcement mechanisms, identifying and explaining required services to perform remote attestations and other security operations, exploring the mechanisms to allow secure boot, managing trusted virtualized platforms remotely and finally presenting a prototype of the demo system.

6 References

[1] TCG, "Cloud Computing and Security – A Natural Match", April, 2010, http://www.trustedcomputinggroup.org/resources/cloud_computing_and_security__a_natural_match

[2] *TCG Specification Architecture Overview*, Specification Revision 1.4, 2nd August 2007, http://www.trustedcomputinggroup.org/resources/tcg_architecture_overview_version_14 (Last accessed: March 22, 2010)

[3] *TPM Specification*, TPM Main Part I Design Principles, Version 1.2, July 09, 2007, Available at http://www.trustedcomputinggroup.org/resources/tpm_main_specification (Last accessed: March 22, 2010)

[4] *TCG Infrastructure Architecture Part-II - Integrity Management*, Version 1.0, Revision 1.0, 17th November 2006, TCG Infrastructure Working Group (IWG), Available at http://www.trustedcomputinggroup.org/resources/infrastructure_work_group_architecture_part_ii__integrity_management_version_10 (Last accessed: March 29, 2010)

[5] *TCG Infrastructure Architecture Part-I – Interoperability Architecture*, Version 1.0, Revision 1.0, 16th June 2005, TCG Infrastructure Working Group (IWG), Available at http://www.trustedcomputinggroup.org/resources/infrastructure_work_group_reference_architecture_for_interoperability_specification_part_1_version_10 (Last accessed: March 29, 2010)

[6] *Platform Trust Services Interface Specification (IF-PTS)*, Version 1.0, Revision 1.0, 17th November 2006, TCG Infrastructure Working Group (IWG), Available at http://www.trustedcomputinggroup.org/resources/infrastructure_work_group_platform_trust_services_interface_specification_ifpts_version_10 (Last accessed: March 29, 2010)

[7] *Reference Manifest (RM) Schema Specification,* Version 1.0, Revision 1.0, 17th November 2006, TCG Infrastructure Working Group (IWG), Available at http://www.trustedcomputinggroup.org/resources/infrastructure_work_group_reference_manifest_rm_schema_specification_version_10 (Last accessed: March 29, 2010)

[8] Berger, S., Cáceres, R., Goldman, K. A., Perez, R., Sailer, R., and van Doorn, L. 2006. *vTPM: virtualizing the trusted platform module*. In *Proceedings of the 15th Conference on USENIX Security Symposium - Volume 15* (Vancouver, B.C., Canada, July 31 - August 04, 2006). USENIX Security Symposium. USENIX Association, Berkeley, CA

[9] N. V. Haldar, D. Chandra, and M. Franz, "Semantic Remote attestation: A Virtual Machine Directed Approach to Trusted Computing", *USENIX Virtual Machine Research and Technology Symposium,* May 2004, http://www.vivekhaldar.com/pubs/trustedvm-tr.pdf

[10] *Interoperability Specification for Backup and Migration Services*, Version 1.0 Final, Revision 1.0, 30th June 2005, For TPM family: 1.1b, Level:1, TCG Infrastructure Working Group (IWG),

Available http://www.trustedcomputinggroup.org/resources/infrastructure_work_group_interoperability_specifications_for_backup_and_migration_services_specification_version_10/ (Last accessed: April 09, 2010)

[11] *Next Generation Secure Computing Base (NGSCB),* Microsoft Corporation, http://www.microsoft.com/resources/ngscb/default.mspx

[12] Garfinkel, T., Pfaff, B., Chow, J., Rosenblum, M., and Boneh, D. 2003. *Terra: a virtual machine-based platform for trusted computing*. *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles* (Bolton Landing, NY, USA, October 19 - 22, 2003). SOSP '03. ACM, New York, NY, 193-206. DOI= http://doi.acm.org/10.1145/945445.945464

[13] John L., Trent J., Ronald P., Reiner S., Leendert v., Ramon C., *Trusted Virtual Domains: Towards Secure Distributed Services*, First Workshop on Hot Topics in System Dependability, Yokohama, Japan, June 30, 2005.

[14] *Trusted Virtual Domains: Secure Foundations for Business and IT Services*, November 09, 2005, Research Report, IBM Research Division.

[15] *An Open Trusted Computing Architecture – Secure Virtual Machines Enabling User-Defined Policy Enforcement*, June 28, 2006, Research Report, IBM Research Division.

[16] Berger, S., Cáceres, R., Pendarakis, D., Sailer, R., Valdez, E., Perez, R., Schildhauer, W., and Srinivasan, D. 2008. *TVDc: managing security in the trusted virtual datacenter*. *SIGOPS Oper. Syst. Rev.* 42, 1 (Jan. 2008), 40-47. DOI= http://doi.acm.org/10.1145/1341312.1341321

[17] R. Sailer, T. Jaeger, E. Valdez, R. Cáceres, R. Perez, S. Berger, J. L., Griffin, and L. van Doorn. *Building a MAC-based Security Architecture for the Xen Opensource Hypervisor*. 21st Annual Computer Security Applications Conference (ACSAC), December 2005.

[18] Reiner Sailer, Xiaolan Zhang, Trent Jaeger, Leendert van Doorn. *Design and Implementation of a TCG-based Integrity Measurement Architecture*. 13th Usenix Security Symposium, San Diego, California, August, 2004.

[19] *Trusted Computing Management Server (TCMS),* Infineon Technologies, http://www.infineon.com/cms/en/product/channel.html?channel=db3a304317a748360118194690b821d7 (Last accessed: May 7, 2010)