Para-Virtualizing the Trusted Platform Module: An Enterprise Framework Based on Version 2.0 Specification

Jiun Yi Yap and Allan Tomlinson

Information Security Group
Royal Holloway, University of London
Egham, Surrey
TW20 0EX, United Kingdom
Jiun.Yap.2012@live.rhul.ac.uk, Allan.Tomlinson@rhul.ac.uk

Abstract. This paper introduces a framework for para-virtualizing the newer Trusted Platform Module (TPM) version 2.0. The framework covers the design of a para-virtualized TPM 2.0 and the considerations when deploying it for use in an Enterprise Information Technology (IT) infrastructure. To develop this framework, a quick study of the TPM 2.0 specification was undertaken and a survey of para-virtualizing TPM techniques was carried out. The study found that TPM 2.0 core functions are suitable for para-virtualization. A set of requirements was then developed to guide the design of this framework. The framework includes components to support the para-virtualized TPM. The framework also covers external components that are essential for the proper functioning of the para-virtualized TPM in an Enterprise IT environment. Research challenges for this framework are then discussed at the end of the paper.

Keywords: Trusted Platform Module 2.0, Para-Virtualization, Framework, Enterprise IT.

1 Introduction

Virtualization is a fundamental technology that is widely used in Enterprise IT infrastructures. Users of virtualization technology need some level of assurance about the expected behavior of a virtual machine (VM) and its ability to protect confidential information from unauthorized disclosure. The TPM specified by the Trusted Computing Group (TCG) offers security properties that can be leveraged by the users of virtualization technology to increase the protection of the system and data from cyber security threats [1].

However, the TPM was originally designed for use with a computing system in a one to one relationship. In a virtualized system, the design will require enhancements to the TPM in order for it to work in an environment where a computer hardware platform hosts several VM. There are generally two types of technique to enable a TPM hardware chip to support multiple VM. The first type is full virtualization of the

R. Bloem and P. Lipp (Eds.): INTRUST 2013, LNCS 8292, pp. 1–16, 2013.

TPM which is exemplified by the work of Vincent Scarlata et al. [4]. In that paper, the authors described the creation of software virtual TPM instances contained in a privileged VM. Each virtual TPM will support a unique VM. This design is aligned to the virtual TPM framework proposed in TCG's Virtualized Trusted Platform Architecture Specification [5] and Open Trusted Computing's VTPM Architecture [6]. Although the designs often extend the root of trust from the TPM hardware chip to the virtual TPM, the security protection for confidential data provided by the TPM's hardware based protected storage location is not offered. A probable reason could be that these designs are based on the older TPM 1.2 specification and the limited amount of TPM memory is unable to support the requirements of the virtualized environment. In addition, the long chain of trust from the TPM hardware chip to the virtual TPM can be fragile as the attack surface is now wider compared to a non-virtualized implementation.

This paper will analyze the other type of technique which is para-virtualizing TPM. Paul England et al. wrote that TPM para-virtualization refers to the method of mediating guest VM access to hardware TPM using a software component [7]. The design will require no change to most of TPM functionality but some aspects of the device interface may change. A major advantage offered by this technique is the availability of TPM hardware based protected storage location and this feature is desired by organizations that requires hardware based security; for example, government Enterprise IT. Moreover, the chain of trust is now shorter as the VM can access the TPM hardware chip in a more direct manner. However, in a para-virtualization design, the use of the resources belonging to a single TPM by multiple VM has to be managed to ensure fair sharing and prevent cross-interference. On the other hand, the TPM has to provide sufficient resource to support the operation of several VM. With the advent of the newer TPM 2.0 specification, it is timely to examine if the newer specification can better support para-virtualization requirement. This para-virtualization framework will introduce components that leverage on new capabilities offered by TPM 2.0. The framework will also address the challenges for achieving TPM para-virtualization in Enterprise IT, for example, backup and migration.

This is a research paper rather than a presentation of an actual implementation. It contains a quick study of the new TPM 2.0 specification from the TCG and analyzes the state of the art regarding para-virtualizing the TPM. With this background knowledge, the paper will then examine the extent to which TPM 2.0 core functions are suitable for para-virtualizing. This is followed by a proposed framework for para-virtualizing TPM 2.0 in the context of an Enterprise IT infrastructure. The paper will end with a discussion on research challenges for the proposed framework.

2 Introducing TPM 2.0

The Trusted Computing Group (TCG) wrote in the Trusted Platform Module (TPM) version 2.0 specification [2] that trust conveys an expectation of behaviour from the computer system. In other words, a user can trust a computer if it always behaves as it is intended to. The assessment of trust always begins from some baseline, or "root of

trust". In a computing platform, the three roots of trust for measurement, storage and reporting provide the minimum functionality required to describe the attributes that contribute towards its trustworthiness. The TPM and supporting components aim to provide these 3 roots of trust.

TPM 2.0 is the latest specification from the TCG and it replaces the previous TPM 1.2 specification. The changes and enhancements to TPM 2.0 compared to the previous TPM version include: support for additional cryptographic algorithms, enhancements to the availability of the TPM to applications, enhanced authorization mechanisms, simplified TPM management and additional capabilities to enhance the security of platform services.

2.1 Architecture

TPM 2.0 is designed to be a self-contained computing device. This allows the TPM device to be trusted to carry out computations without relying on external computing resources. The following are short descriptions of the subsystems in a TPM 2.0 device while detailed explanation can be obtained from TPM 2.0 specification [2].

I/O Buffer – This component enables the host computing system to communicate with the TPM. It can be a shared memory. Data to be processed by the TPM will be validated at this point.

Cryptography Subsystem – The cryptographic engine supports commonly used cryptographic functions like hashing, asymmetric operations such as digital signature and key exchange, symmetric encryption, random number generator and key derivation function. These cryptographic functions can be used by the other TPM components or the host computer.

Authorization Subsystem – Before a TPM command is executed, this subsystem checks that proper authorization data has been given by the calling application.

Volatile Memory – This memory holds transient TPM data, including Platform Configuration Registers (PCR), data objects and session data. PCR contains the integrity measurements of critical components in the host computer. A data object can either be a cryptographic key or other data. The TPM uses sessions to manage the execution of a series of commands.

Non-Volatile (NV) Memory – This memory is used to store persistent TPM data that includes the platform seed, endorsement seed, storage seed and monotonic counter. Additional PCR banks can be created in this memory.

Management Subsystem – This subsystem oversees the operation of the various TPM states. Basic TPM states include power-off, initialization, start up, shut down, self-test, failure and field upgrade.

Execution Engine – This firmware contains the program instructions and data structures that are required to run a TPM command. These program instructions and data structures cannot be altered by the host computing platform. In the event of a firmware upgrade, there are security mechanisms to ensure that the update is authorized and the new firmware is checked for authenticity and integrity.

2.2 Core Functions

A Trusted Computing Base (TCB) can be a BIOS, Virtual Machine Monitor (VMM) or operating system that has proved to be highly secure and hence trustworthy. When a TCB is made to work together with a TPM, they can offer the capabilities of integrity measurement and reporting, protected data storage, certification and attestation and authentication. In integrity measurement, a hash function is performed by the BIOS on the first software component that is started when the computer powers up. The hash function will produce a digest of that software component. If that software component is altered, the digest will be different from the one obtained when the software component was first measured. This digest can be stored in the PCR located in either the volatile or non-volatile memory of the TPM. TrustedGRUB [3] is an application that implements this integrity measurement at system start. A TPM can have an authenticity certificate from the manufacturer and this feature is used in conjunction with the integrity measurement to report the "trustworthiness state" of a computing platform.

A unique feature of the TPM is the use of primary seeds to generate hierarchies of keys for use in cryptographic functions. The intention of this feature is to provide the flexibility to support different types of cryptographic functions without increasing the storage memory requirement. To establish trust in a key derived from a TPM primary seed, the TPM can produce with a certificate indicating that the processes used for creating and protecting the key meet the necessary security requirements. During attestation, a TPM can vouch for the authenticity and properties of either the host computing platform, a piece of software or a cryptographic key.

TPM non-volatile memory is typically used to store cryptographic keys that protect sensitive data. In this method, the sensitive data is encrypted with a cryptographic key derived from a root key inside the TPM chip. This cryptographic key is usually stored into the non-volatile memory of the TPM chip. To read the sensitive data, the user has to provide authorization data to the TPM chip and only upon successful authorization will the cryptographic key be released from the TPM chip. This is known as protected storage.

3 State of the Art for Para-Virtualizing the TPM

The following sub-sections will give brief descriptions of two projects on para-virtualizing TPM. It is important to note that these two projects are based on the older TPM 1.2 specification.

3.1 Para-Virtualized TPM Sharing

In this paper [7], the authors describe a para-virtualization design that allows a VMM to time share a TPM among its VM. The concept of associating a TPM context to a particular VM is proposed. A TPM context will contain the important data that defines a TPM state, for example, keys and sessions. When a particular VM wishes to use the physical TPM, the associated TPM context is loaded into that physical TPM.

The loaded TPM context can be saved and cleared from the physical TPM to allow other TPM context to be loaded when required. TPM contexts are saved in the hypervisor. As a result, the design is able to support most TPM applications.

To implement this design, the authors located the TPM para-virtualization management software in the hypervisor. Figure 1 gives a high level view of this design.

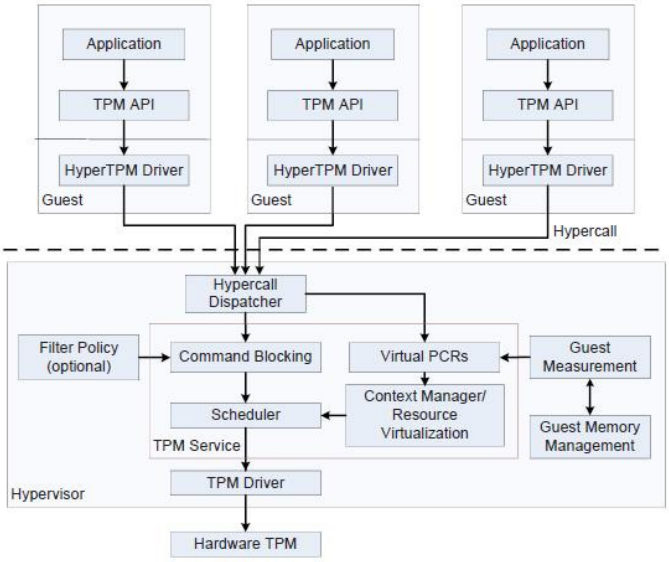

Fig. 1. Architecture for para-virtualizing TPM sharing from [7]

The TPM para-virtualization management software contains the following components:

Scheduler – Rosters shared access to the physical TPM.
Command Blocking – Filters TPM commands based on a pre-determined list of allowed commands. This is to ensure the safe operation of the TPM by disallowing applications in VM from executing certain TPM commands.
Virtual PCR – Each VM is assigned a set of virtual PCR and they are managed by the hypervisor.
Context Manager - This component inspects every TPM command and load the associated context into the physical TPM so that the VM can only access its own TPM resources.
Resource Virtualization – Certain limited TPM resources are virtualized, for example, key slots, authorization session and transport sessions. These virtualized TPM resources are tied to a context that is provided by the context manager.

The use of virtual PCR to store integrity measurements of the VM is not desired by organizations that require hardware based protected storage location. Furthermore, the authors do not elaborate on TPM migration and management of TPM endorsement credentials in this paper.

3.2 Enhancing TPM with Hardware-Based Virtualization Techniques

This paper [8] presented the design of a TPM that supports hardware-based virtualization. In this design, the VMM time multiplexes the hardware TPM in a manner similar to [7]. However, a difference is that the hardware TPM has to be modified to include additional non-volatile memory to store the various TPM contexts. The layout of this multi-context TPM is shown in figure 2.

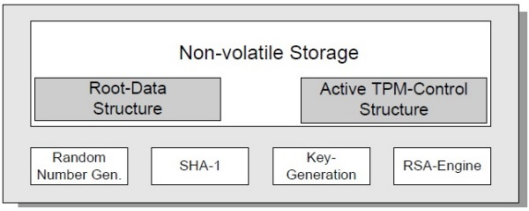

Fig. 2. Layout of the multi-context TPM from [8]

The VMM manages the transition of one TPM context to another using the TPM control structure. The VMM links every TPM context to its own TPM control structure. Figure 3 shows the content of the TPM control structure. When a new TPM context is to be loaded, the VMM will store the previous TPM control structure and load the new TPM control structure. To protect the confidentiality of the TPM control structure, it is encrypted and the cryptographic key is stored in the TPM root data structure.

Fields of the TPM Control Structure
Storage Root Key (SRK)
PCRs [16..23]
Attestation Identity Keys (AIK)
Endorsement Key (EK)
Endorsement Credential
Monotonic Counter Values
Values of the non-volatile storage
Delegation Tables
TPM context data
DAA TPM specific secret (f)
TPM_PERMANENT_FLAGS/DATA
TPM_STCLEAR_FLAGS/DATA
Authorization data

Fig. 3. TPM control structure from [8]

Another difference is the introduction of the concept of protection rings into the TPM. This TPM protection ring has a two level hierarchy that differentiate a privileged TPM mode from a non-privileged TPM mode. The TPM protection ring leverages the Intel VT architecture and hence can be considered as a form of hardware-based protection. There are two forms of CPU operation in the Intel VT architecture. The VMX root operation in which the VMM runs and the VMX non-root operation in which the VM runs. Only VMX root can run the privileged TPM mode

while the VMX non-root can only run in the non-privileged TPM mode. In addition, the authors extended the TCG specification for TPM to include extra commands to manage the transition between TPM modes and to control the different TPM contexts. These extra TPM commands can only be executed by VMX root.

For VM migration, the authors described that their TPM context migration protocol is similar to the concept TCG introduced for migratable keys. On TPM credentials, the authors proposed to establish a certificate chain with the root Endorsement Key (EK). For every TPM context, the TPM generates a new EK which is then certified by the root EK. When the TPM context is migrated, the EK will then be re-certified with the root EK of the destination platform.

This design covers many technical aspects of para-virtualizing TPM. However, there are still gaps to cover before this design can be implemented in an Enterprise IT virtualization environment. For example, the support for business continuity plan has to be developed.

4 Examining TPM 2.0 Suitability for Para-Virtualizing

TPM 2.0 core functions are found to be generally suitable for use in a para-virtualized design and only some typical virtualization functions are required at the VMM level to allow a single TPM 2.0 device to support multiple VM. Based on a quick study of TPM 2.0 specification, the extent to which the core functions can be para-virtualized are described in the following paragraphs.

4.1 Endorsement and Storage Keys

Figure 4 shows how the TPM keys can be distributed to multiple VM. An endorsement key is derived from the endorsement primary seed located inside the TPM and it is the basis for the root of trust of reporting. Several endorsement keys can be generated and assigned to the various VM hosted on the computing platform. However, an endorsement key be migrated together with the associated VM. Although this may seem to be unfavorable, attestation, for example, will have to be done again, it is actually better to obtain a new endorsement key after VM migration from the destination TPM since the host computing platform has changed. Meanwhile, a new certificate will have to be obtained for the new endorsement key after VM migration.

For keys derived from the storage root key, they can be created to be migratable by setting the duplication flag. These keys are assigned to the VMs and kept outside the TPM. As an example, when a VM wishes to use the TPM for certification, the certifying key is loaded into the TPM using the command TPM2_Load. At the end of the session, the certifying key is unloaded from the TPM. The TPM can then start a new session with another VM. During VM migration, a certifying key can be packaged as a duplicable data object using TPM2_Duplicate and then migrated over to the designated TPM. TPM2_Import will load the migrated data object into the destination TPM. Meanwhile, the use of a TPM authorization session will ensure that only the

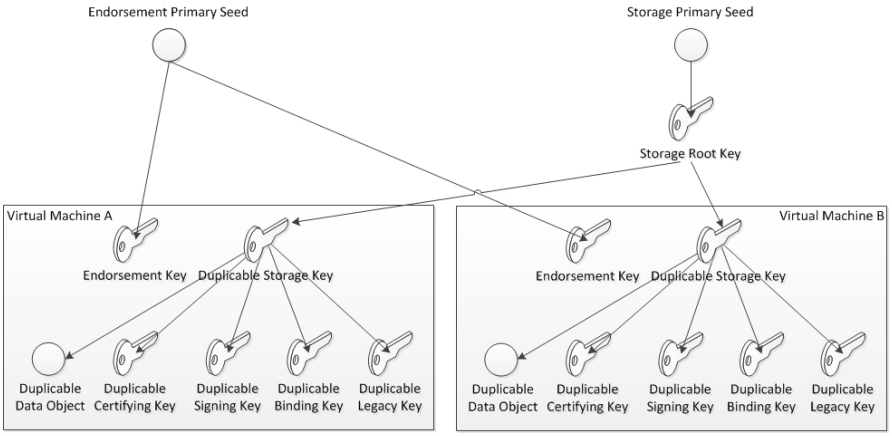

Fig. 4. TPM 2.0 keys distribution for multiple VM

right VM can access its certifying keys. The resulting effect is comparable to the notion of process and resource isolation between VM. Nevertheless, the credential for a certifying key may have to be re-issued after a migration as the host computer has changed.

4.2 Protected Storage

There are various TPM 2.0 commands that can move data in and out of either the volatile or non-volatile memory: for example, TPM2_Load, TPM2_LoadExternal, TPM2_Unseal, TPM2_NV_Write and TPM2_NV_Read. The use of a TPM authorization data will ensure that only the right VM can access its data in the protected storage location. The mechanism for the migration of these data in protected storage location is the same as those for certifying keys. For data that are encrypted and the access control depends on the host computing platform integrity measurements, this can cause a problem during VM migration when the host computer platform is different. The use of TPM 2.0 Enhanced Authorization will allow a more flexible access control policy for this type of protected data. For example, the authorization policy can either check the host platform integrity measurements or other security properties. In the meantime, the amount of volatile and non-volatile memory has to be managed to avoid a situation whereby there is insufficient memory for use by all the VM on that host computer.

4.3 Integrity Measurement and Reporting

TPM2_NV_DefineSpace can be used to create PCR banks in the NV memory. TPM2_PCR_Extend will then be used to record the integrity measurements of the VM into the PCR located in the NV memory. For the host machine, TPM2_PCR_Extend will be used to record the integrity measurements into the PCR

located in the volatile memory. This arrangement will allow the integrity measurement of both the virtual and host machine be stored inside the TPM at the same time. As above, the amount of NV memory has to be managed to avoid the situation whereby there are more VM than the TPM can support. TPM2_Quote is used to report the integrity measurement stored in a particular PCR. When reporting the integrity measurements to a requestor, TPM2_Load is used to insert the relevant attestation key into the TPM. This key is then used to sign the integrity measurement. The mechanisms for the migration and access control of PCR are the same for those for certifying keys.

In addition to the points above, TPM 2.0 is designed with a context management feature that is intended to be used to manage TPM resources among various applications. In a virtualized environment, this feature can instead be used to manage TPM resources among various VM. TPM 2.0 specification states that the structure of the context is decided by the vendor. In other words, there can be a customized context structure to support TPM 2.0 para-virtualization requirements.

As with most hardware virtualization, TPM 2.0 will require some software components at the VMM level to allow it to support multiple VM. For example, a software component is required to provide some form of usage management to ensure that every VM has fair use of the TPM. Another software component is required to block state altering TPM commands issued by non-management VM. The architectures described in TCG's virtualized trusted platform specification [5] are more suited to the full virtualization technique although certain aspects such as TPM migration are applicable to this para-virtualized TPM framework.

5 Requirements for Para-Virtualizing TPM 2.0

As discussed in section 4, TPM 2.0 core functions are generally able to support para-virtualization. However, we noted that there has to be mechanisms to assign and ensure the fair use of TPM resources to multiple VM. In addition, issues pertaining to an Enterprise IT environment, such as migration, certification and logging have to be addressed as well. To this end, the design requirements presented in [1], [4], [7] and [8] were considered and we will like to suggest the following:

1. The way that an application uses the para-virtualized TPM should be the same as for a hardware TPM.
2. The para-virtualized TPM should be always available for use by the VM.
3. Strong association between the VM and its TPM resources. This association should be maintained after the migration of the VM.
4. TPM resources belonging to a VM should not be accessible by another VM.
5. The size of both volatile and non-volatile memory in the TPM should support the additional memory required to host multiple VM on a single physical computer.
6. Data stored in protected storage locations should be preserved unless instructed by their owners. These data should be moved together with the VM during migration and then stored into the protected storage location of the destination TPM.

7. The security strength of protected storage location in a para-virtualized TPM should be the same as for hardware TPM.
8. The activity of the para-virtualized TPM should be logged. The log file associated with a particular VM should be moved to the destination host computing platform during VM migration.
9. Non-privileged VM cannot execute commands that can alter the state of the TPM.
10. The para-virtualized TPM should have verifiable credentials at all times.
11. Individual VM interaction with para-virtualized TPM should be isolated from each other to avoid interference.
12. The para-virtualized TPM should be able to support business continuity plans.

6 An Enterprise Framework for Para-Virtualizing TPM 2.0

The framework shown in figure 5 contains components at the VMM and hardware level to support the para-virtualization of TPM 2.0. This framework allows the multiplexing of TPM 2.0 functions and resources for use by VM and their applications at

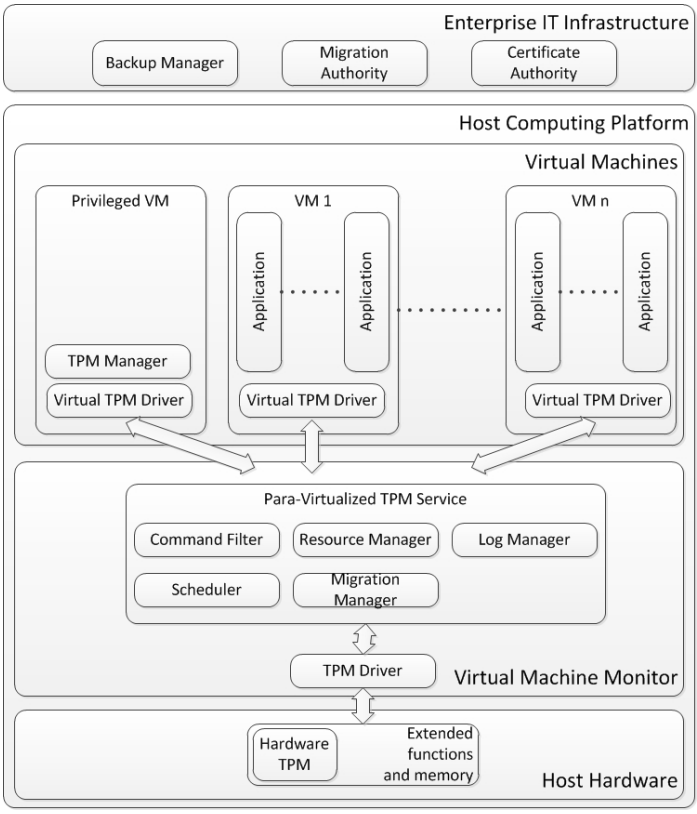

Fig. 5. Enterprise framework for para-virtualizing TPM 2.0

the same time while preserving the hardware based protected storage location. The development of this framework is based on the survey of TPM para-virtualizing works in section 3, the analysis of para-virtualizing TPM 2.0 core functions in section 4 and the design requirements from section 5 of this paper.

As shown in section 4, TPM 2.0 core functions are generally suitable for use in a para-virtualized design. Hence, the software components at the VMM level do not emulate TPM functions but instead focus on ensuring fair usage of TPM and to address requirement such as migration and logging.

A major difference from existing concepts is the removal of virtual PCR as the store for integrity measurements of VM because they will be stored in TPM NV memory. As a result, VM integrity measurements enjoy the security of hardware based protected storage location. In addition, a privileged VM is used to manage the hardware TPM and para-virtualized TPM service. This gives the system administrator a separate conduit to control the hardware TPM and para-virtualized TPM service. There are also provision for TPM hardware enhancements and a log manager. These are security features desired by a high security Enterprise IT infrastructure. Moreover, this framework covers external components that are essential for the proper functioning of the para-virtualized TPM in an Enterprise IT environment. The following paragraphs describe the components in this framework.

6.1 Extended Functions and Additional Memory

The framework allows modifications to be carried out on the TPM hardware to support the requirements of the virtualized environment. For example, the amount of memory can be increased to include more PCR banks to store the integrity measurements of multiple VM. Although hardware modifications can be costly, certain high security requirement, for example, government Enterprise IT, may demand and pay for this enhancement. In addition, the modified hardware can contain extended functions to support new virtualization technique such as the single root I/O virtualization standard specified by the PCI-SIG [12]. Hardware technique such as the use of field-programmable gate array described in [9] can be used to implement the modifications. Depending on security requirements and the amount of modification, a new platform certificate may have to be issued for the modified TPM to vouch that the platform contains a genuine TPM and that the communication path between the TPM and the host computer is trusted.

6.2 Command Filter

As there are commands that can alter the TPM state, for example TPM2_Shutdown, it is necessary that only the TPM manager in the privileged VM can execute such commands. All commands from the VM to the hardware TPM will be inspected and any state altering command from non-privileged VM will be blocked from execution. The command filter can make use of the hardware based virtualization technique described in [8] to enforce the restriction on using selected TPM commands.

6.3 Scheduler

This component sequences VM access to the hardware TPM in a time division multiplexing manner. This will ensure that every VM has an opportunity to use the TPM. The algorithm for sequencing can be either round robin or demand based. In round robin, the scheduler can poll every VM and ask if it likes to use the TPM. Alternatively, the scheduler can arrange access to the TPM whenever the VM makes a request. The time that a VM can use the TPM will be limited. If the TPM is currently servicing a request, any new request will be queued in a first-in-first-out memory cache.

6.4 Resource Manager

To achieve strong association between VM and their TPM resources, the resource manager will administer the assignment of TPM resources such as keys. It is envisioned that the VM will use keys derived from the primary seed in the hardware TPM. In turn, each VM will build their key hierarchy based on their assigned key. Meanwhile, the resource manager will create PCR banks in the TPM NV memory and assign them to the VM. PCR in the TPM volatile memory will be reserved for use by the host computer and VMM. For the purpose of attestation, the resource manager can provide the PCR contents in the TPM volatile memory to the VM. The other task carried out by the resource manager is to work together with the scheduler to isolate the TPM processes and resources of the VM. The mechanisms used to achieve this effect can include the use of TPM context management feature and authorization session. TPM commands such as TPM2_ContextSave, TPM2_ContextLoad, TPM2_FlushContext and TPM2_StartAuthSession will be used. This will prevent one VM from accessing the TPM resources of another VM. TPM contexts are saved in the VMM.

6.5 Migration Manager

During VM migration, this component will work together with the resource manager to oversee the packing of the associated TPM resources into duplicable data objects. The command TPM2_duplicate will be used to prepare the data object for migration. This component will also work with the migration authority to operate a migration protocol that securely moves the duplicated data object to the designated TPM. On the other hand, if the host computer is to receive a migrated VM, the migration manager can carry out the task of authenticating and verifying the trustworthiness of the migrating VM.

6.6 Log Manager

The availability of a log is crucial to forensic investigation in the event of a security incident. The log manager will log down all the operations performed by the VM on the hardware TPM. The log manager can make frequent integrity measurements of the log file and store the measurements in the hardware TPM. This will allow the

detection of unauthorized changes to the log file. Meanwhile, the log manager with work together with the migration manager and migration authority to move the log file associated with a particular VM to the destination host computing platform during VM migration.

6.7 TPM Manager

This resides in a privileged VM and is primarily used to manage the hardware TPM and configure the para-virtualized TPM service. The privilege status of this VM can either be enforced from the VMM or controlled by hardware based virtualization technique described in [8]. The TPM manager can check the integrity of the VMM components by querying the hardware TPM.

6.8 TPM and Virtual TPM Driver

The TPM driver contains the software stack that enables the VMM to communicate with the hardware TPM while the Virtual TPM driver contains the software stack that enables the VM to communicate with para-virtualized TPM service in the VMM. To detect unauthorized changes to these two components, integrity measurements are carried out when they are started. The integrity measurements are then stored in the TPM.

6.9 Backup Manager

This and the next two components are external to the computer platform but part of the Enterprise IT infrastructure. To support business continuity planning, the TPM keys for the VM should be archived and stored securely in a physically separate location. In the event of an incident that cause the TPM keys to be lost, the backup TPM keys can be retrieved to support the continuation of business operation.

6.10 Migration Authority

Besides administrating the migration of VM, the migration authority can work with the TPM migration manager to ensure that the accompanying TPM resources are moved to the correct destination TPM.

6.11 Certificate Authority

This component will verify the credentials of TPM keys provided by the VM and issue the appropriate certificates when it is satisfied with the credentials. After a VM has been migrated, it can work with the migration authority to recheck the credentials of the TPM keys and issue new certificates. More importantly, the certificate authority can revoke a particular certificate when required.

7 Requirements Revisited

To assess the thoroughness of the design of this framework, it is compared to the requirements listed in section 5. Firstly, there are no changes to the TPM commands and most TPM commands are available to the VM except for those that can alter TPM state. This meets the requirement of retaining the same TPM usage model. Secondly, the scheduler will sequence the TPM commands issued by the virtual commands. Besides ensuring that every VM can interact with the TPM, it works in conjunction with the resource manager to switch from one VM's TPM session to another session. The resource manager will link the TPM resource to the VM and access control is achieved by using authorization session. The resulting effect is equivalent to isolating each VM's interaction with TPM.

Meanwhile, the migration manager works with the resource manager to maintain the association between a TPM resource and its VM during migration. In addition, this framework allows for modifying the TPM hardware and this provision gives the flexibility to complement TPM hardware with additional memory to meet the TPM memory requirement of multiple VM. TPM 2.0 NV memory is a protected data storage location of this framework. The resource manager can store a certain amount of integrity measurements from numerous VM into this hardware memory. Hence, the security protection of integrity measurement in this para-virtualized TPM is the same as for a plain hardware TPM. The other requirement relating to logging is fulfilled by the log manager while the backup manager is used as part of the business continuity plan. As for certification, this framework uses an external certificate authority to check on the credentials of TPM keys. When it is satisfied with the credentials, it will issue a certificate to vouch for the TPM keys. Last of all, the command filter makes sure that non-privileged VM cannot execute commands that can alter the TPM state. To conclude, this framework meets the requirements set forth in the section 5.

8 Research Challenges

This framework is based on a paper study of TPM 2.0 specification. From the view point of a high level design, this framework can be considered to be reasonable. To validate this framework, the components in the VM monitor have to be implemented and tested. An ideal candidate for the VM monitor is the open source Xen hypervisor [10]. The Xen hypervisor can support para-virtualization and hardware based virtualization technology such as Intel VT and AMD V CPU architecture extension. The hypervisor has a driver for the TPM although it is for TPM 1.2 specification.

The techniques described in [7] and [8] can be redeployed and used to implement this framework but significant challenges still exist. The performance of this para-virtualized TPM design is one area to be investigated further. Research can be carried to develop better algorithms in the scheduler to sequence VM requests for TPM resources. The strength of the isolation between each VM interaction with the para-virtualized TPM is another research area to be studied. The commonly used integrity measurement method of obtaining a digest from hashing a software component can be

difficult to implement in a virtualized environment. This is because VM migration can take place and the configuration of host computing platforms can differ. Attestation in such an environment will be tedious as there will be a variety of measurements to contend with. Hence, further research can be under taken to look at other approaches, for example property-based attestation in [11], to obtain a measure of the state of trustworthiness.

The development of use case for this framework is a task not to be neglected. It will be difficult to persuade organizations to take up this framework if there are no functional use scenarios. Meanwhile, threat modeling can be conducted on these scenarios. The results can be used by researchers to harden the design and organizations who wish to adopt the framework can put in mitigation measures recommended from the threat model to avoid the pitfalls. Lastly, the use of TPM monotonic counter by VM is not addressed in the proposed framework and further work can be done to study this matter.

9 Conclusion

The availability of hardware based protected storage location is an advantage that is desired by organizations that require high security for their Enterprise IT infrastructure. This paper found that TPM 2.0 core functions are generally suitable for para-virtualization. This indicates that the technical barrier to using TPM 2.0 in a virtualization environment can be potentially lowered. The proposed framework is holistic as it covers important considerations at different level of the virtualization environment. Differences from existing concepts include storing integrity measurements of VM in TPM NV memory and using a privileged VM to manage the hardware TPM and para-virtualized TPM service. There are also provision for TPM hardware enhancements and a log manager. Moreover, this framework covers external components that are essential for the proper functioning of the para-virtualized TPM in an Enterprise IT environment. To conclude, the studies and framework expressed in this paper provide a comprehensive basis for future work in para-virtualizing TPM 2.0 and integrating the design to an Enterprise IT virtualization environment.

References

1. Scarlata, V., Rozas, C., Wiseman, M., Grawrock, D., Vishik, C.: TPM Virtualization: Building a General Framework. In: Norbert, P., Helmut, R. (eds.) Trusted Computing, pp. 43–56. Vieweg (2007)
2. Trusted Computing Group: Trusted Platform Module Library Family “2.0” Level 00 Revision 00.96, March 15 (2013)
3. TrustedGRUB, `http://www.trust.rub.de/projects/trustedgrub/`
4. Berger, S., Caceres, R., Goldman, K.A., Perez, R., Sailer, R., van Doorn, L.: vTPM: Virtualizing the Trusted Platform Module. In: Proceedings of the 15th Conference on USENIX Security Symposium, vol. 15, pp. 305–320. USENIX (2006)

5. Trusted Computing Group: Virtualized Trusted Platform Architecture Specification "1.0" Revision 0.26, September 27 (2011)
6. Open Trusted Computing: VTPM Architecture Revision Final 1.0 Update, May 29 (2009)
7. England, P., Loeser, J.: Para-Virtualized TPM Sharing. In: Lipp, P., Sadeghi, A.-R., Koch, K.-M. (eds.) TRUST 2008. LNCS, vol. 4968, pp. 119–132. Springer, Heidelberg (2008)
8. Stumpf, F., Eckert, C.: Enhancing Trusted Platform Modules with Hardware-Based Virtualization Techniques. In: Cotton, A., Dini, O., Skarmeta, A.F.G., Ion, M., Popescu, M., Takesue, M. (eds.) Proceedings of the Second International Conference on Emerging Security Information, Systems and Technologies, SECURWARE 2008, pp. 1–9. IEEE Computer Society (2008)
9. Pirker, M., Winter, J.: Semi-Automated Prototyping of a TPM v2 Software and Hardware Simulation Platform. In: Huth, M., Asokan, N., Čapkun, S., Flechais, I., Coles-Kemp, L. (eds.) TRUST 2013. LNCS, vol. 7904, pp. 106–114. Springer, Heidelberg (2013)
10. Xen Hypervisor, `http://www.xenproject.org/`
11. Sadeghi, A.-R., Stüble, C., Winandy, M.: Property-Based TPM Virtualization. In: Wu, T.-C., Lei, C.-L., Rijmen, V., Lee, D.-T. (eds.) ISC 2008. LNCS, vol. 5222, pp. 1–16. Springer, Heidelberg (2008)
12. PCI-SIG: Single Root I/O Virtualization and Sharing Specification Revision 1.1, January 20 (2010)