Utilizing TPM Functionalities on Remote Server

Norazah Abd Aziz and Putri Shahnim Khalid

Abstract Trusted Platform Module (TPM) has become an essential functionality in the information security world today. However, there are legacy computers that do not have TPM onboard and would still want to use the TPM functionalities without having to replace the hardware. Also, TPMs are not available for virtual machines hence there is a need to provide integrity of the virtual machine platforms. This paper introduces a framework to provide a remote server with TPM capabilities for the legacy computer and also virtual machines to be able to utilize TPM functionalities. In this framework, there is also a need to provide fault tolerance mechanism to ensure reliability of the server and also scalability feature is incorporated to cater for growing number of users. The main component of the framework is the 'vTPM Manager' module which resides in the remote TPM server. This vTPM Manager handles the creation and deletion of virtual TPMs, providing fault tolerance mechanism and also scalability feature for the whole system. By using this framework, users who do not have a TPM residing in their device would be able to remotely access the TPM server to utilize the TPM functionalities with the assurance of a fault tolerance mechanism and the number of users is unlimited since it is scalable.

Keywords Fault-tolerance · Scalability · TPM instances · Migration · vTPM Manager · Virtual TPM

N. Abd Aziz (✉) · P. S. Khalid
MIMOS Berhad, Technology Park Malaysia, 57000 Kuala Lumpur, Malaysia
e-mail: azahaa@mimos.my

P. S. Khalid
e-mail: shahnim.khalid@mimos.m

S.-S. Yeo et al. (eds.), *Computer Science and its Applications*,
Lecture Notes in Electrical Engineering 203, DOI: 10.1007/978-94-007-5699-1_1,

1 Introduction

Trusted Computing (TC) is a technology developed and promoted by non-profit industry consortium. The technology aims to enhance the security of hardware and software building blocks. The consortium known as Trusted Computing Group (TCG) [1] has come up with specifications on Trusted Platform Module (TPM) which has potentials to be used for security and trust related services like remote attestation and key management. In order to utilize the Trusted Computing functionalities, new PCs and laptops are equipped with a TPM [2] on the motherboard by many hardware manufacturers.

However, legacy computers and older motherboards do not have TPMs onboard. This poses a problem for users wanting to utilize the Trusted Computing functionalities without having to replace all the equipments. Furthermore, most virtual machine environments are not equipped with a TPM. Since virtual machines are used widely in cloud computing environment, it is necessary to apply TPM functionalities to provide integrity of the virtual machine platforms.

In this paper, we introduce a framework to provide a remote server with TPM capabilities in the form of hardware TPM and/or software based TPM. Software based TPM in this context is referred as virtual TPM (vTPM) in this paper. Users can connect to the remote server and use the trusted computing capabilities. We will discuss about fault tolerance mechanisms to ensure users can connect to the server at all times through virtual TPM (vTPM) instances. In addition, we also discuss on the scalability of the servers in order to cater for high number of users and their associated vTPM instances at one time.

This paper is organized as follows. Section 1 starts with this brief introduction and followed by current related work in Sect. 2. Section 3 of the paper explains about the fault tolerance mechanism and scalability of the server. The basic framework of the attempt implementation containing the process flow of system requirement is presented in Sect. 4. The paper continues to describe the concept of remote server with TPM capabilities implementation handled by a vTPM Manager module. Finally Sect. 5 describes the current implementation. The paper ends with a conclusion.

2 Related Work

A system which enables the trusted computing for an unlimited number of virtual machines was proposed by [3]. Their approach is to virtualize the TPM, so the TPM functionalities are available to operating systems and applications running in virtual machines. We adopted their approach which provides added functions to create and destroy virtual TPM instances as well as to maintain the migration of a virtual TPM instance with its respective virtual machine. The difference is they

implement multiplexing of request from clients to their associated vTPM instances but our approach only interacts with the clients during initialization process.

The paper in [4] is extended from [3] by adding built-in attestation mechanism. They have introduced a ticket-based remote attestation scheme. Compared to our framework, their vTPM instances management resides in Virtual Machine (VM). But, similar with our approach, the software TPM is also always protected by the hardware TPM. During vTPM spawns, its PCR values are initialized with values from the underlying hardware TPM.

The security and reliability issues in client virtualization were also discussed in [5]. Their proposed solutions leverage on Intel vPro and TPM in order to overcome the issues and using trusted VM container through remote attestation protocol verification. Our approach is not limited to the Intel technology and platform and hence is more feasible.

In [6], the paper describes their approach to secure cloud-based system using trusted computing. Their design mainly focuses on the virtual DRTM (Dynamic Root of Trust for Measurement). By virtualizing the DRTM, they control the locality by modifying the Xen vTPM Manager. Locality is based on the memory addressing which corresponds to different levels in a system, for example security kernel at Locality 0 while application at Locality 3 and so on. Our approach differs in that we are not modifying the way the guest OS access the data based on these locality using a certain algorithm that has to be embedded in the hypervisor (such as Xen).

3 Fault Tolerance, Scalability and Attestation

3.1 Fault Tolerance

Fault-tolerance is the property of a software or hardware that enables a system to operate continuously in the event of failure of (or one or more faults within) some of its components [7]. In other word, it is designed to recover from failure immediately with no loss of service. There are a few levels of fault tolerance based on the ability to continue operation in the event of a power failure in time basis. The levels are defined by whether the fault tolerance feature is provided by software, embedded in hardware, or by both combinations.

A fault-tolerance mechanism consists of three types: replication, redundancy and diversity. Fault-tolerance replication is requesting or directing tasks in parallel from multiple identical instances of the same system or subsystem based on the best output [7]. Similarly, fault-tolerance redundancy is also providing multiple identical instances but switching to one of the remaining instances in case of a failure [8]. In other word, it uses multiple nodes that are ready to provide service in order to recover from service failure of a single node. In contrast to replication and

redundancy type, diversity provides multiple different implementations of the same specification.

In this paper we focus on fault tolerance using hardware which is provided to ensure the TPM server is available at all times. As mention earlier, trust and security is the main concern for the framework, hence replication of the vTPM instances is required to ensure users can connect to the server at all times even when there is a failure.

3.2 Scalability

According to [9], scalability is desirable in technology as well as business settings because both benefit significantly from the ability to easily increase volume without impacting the contribution margin. Scalability is the ability of a hardware or system to adapt to increasing demand due to the growing amount of context volume or size in order to meet user capacity. Our framework is designed to be scalable in the sense that the system can be upgraded easily and transparently to the users without shutting down the system. Hence, further investment to the system for adding new processors, devices and storage to the system has no additional cost.

Scalability can be measured in various dimensions [9], but this paper focus on load scalability. It means that the system easily expands and organizes its resource to sustain heavier or lighter number of inputs for modification and deletion activities. In our approach, the network scalability addresses the issue of retaining performance levels while adding additional servers to a network. Additional servers are typically added to a network when additional processing power is required.

3.3 Attestation

One of the most important uses of TPM is to enable a computing platform to attest its integrity to another entity. The attestation protocol involves measuring various ‘properties’ of the platform and storing the values in the TPM. When a remote entity asks for assurance of the integrity of the platform, the measurements are verified and sent over to the other entity. Our framework is designed to implement this feature in virtualization environment which is used to assure users of the integrity of the spawned vTPM. In our framework attestation protocol is run by enhanced virtualization API named as TMCI and a vTPM to verify the integrity of the associated VM. When a request for a VM is received, the TMCI will first ask the vTPM to attest the integrity of the VM. The VM will be created and given to the user if the attestation is successful. Otherwise another VM has to be created.

4 Framework

There arise desires for a service which provides TPM capabilities in the form of hardware and/or virtual TPM that would enable users who do not have TPM hardware in their device to access TPM functionalities as and when they require it. Therefore, our framework introduces a server which users can access remotely to use the TC functionalities. The server which is called TPM server is embedded with TPM hardware or a software-based TPM (vTPM) or both to cater for the users' needs. The concern is how to manage the multiple users and the multiple instances of vTPM. In some applications such as cloud computing, there can be millions of users connecting to the server at any one time. Hence, the TPM server must be scalable. The TPM server will also have to be available at all times since all the keys and state files are saved in the server and any disruption to the server may lead to loss of information on the user side. These functionalities are handled by a vTPM Manager (vMgr) on the TPM server which will be discussed further in the next section.

Figure 1 illustrates components of our framework which contains a fault-tolerance mechanism in order to maintain the availability of TPM server. The framework consists of users which are running on different types of hardware, but not limited to desktops, virtual machines, laptops and mobile devices. Users who need to access the TPM server connects through the network. Other than vMgr to handle all the resources related to vTPM, there is also a migration controller component which manages the migration of the TPM server to another physical location. However, in this paper we will not discuss further on migration controller component. Since our approach for virtualization environment is specific for cloud computing, we have enhanced the virtualization API. The enhanced virtualization API is called TMCI and is placed in the framework as shown in Fig. 1.

TMCI is the virtualization API which processes the URI and then communicates bidirectional with vMgr and public API in Libvirt command. Public API will pass the request to the driver API and look into the corresponding Virtual Machine (VM) driver in Libvirt. In our approach, QEMU-KVM is used for the VM creation, spawn or destroy activities through Libvirt. Before creating and destroying the VM, TMCI will verify the integrity of the vTPM instances with vMgr. If the verification fails vMgr will send an error message to TMCI and TMCI will cut off the communication with Libvirt. Since the communication to TMCI is disconnected, Libvirt cannot create or destroy the VM.

The fault-tolerance component supported by TPM Server comprises of TPM Primary Server and TPM Backup Server. The TPM Primary Server is the main server and backup is provided by duplicating the server in the TPM Primary Backup Server. For scalability purposes, TPM Secondary Server is connected to the TPM Server Primary and its backup is provided in the TPM Secondary Backup Server. The scalability function does not limit the number of servers; hence there

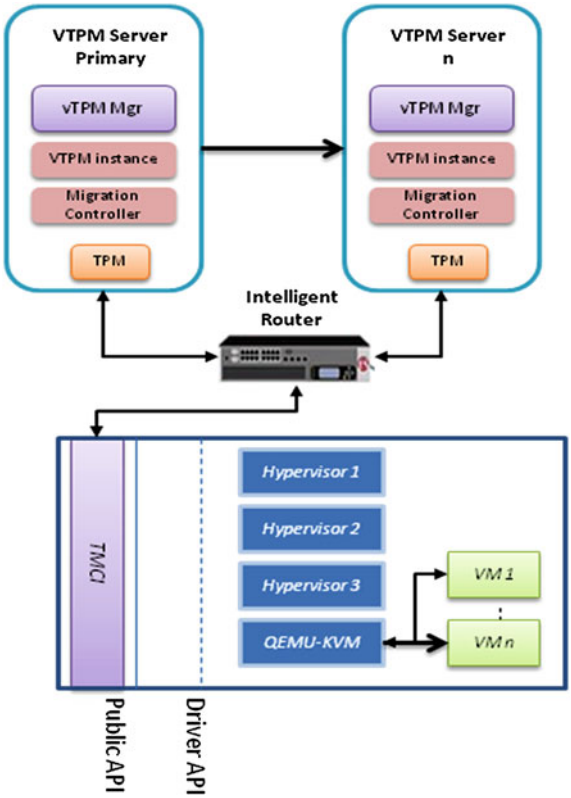

Fig. 1 Basic implementation framework

can be Tertiary Server and so on. All the servers are connected to the database where all the states are saved periodically.

4.1 vTPM Manager

vTPM Manager (vMgr) is the main feature in our framework. It is tasked to handle the creation or spawning of vTPM instances for individual users. Each user will be assigned a dedicated vTPM instance, which is linked to the TPM hardware. vMgr will assign dedicated port to the vTPM for the user to connect. vMgr is also responsible to update the storage or database with all the vTPM information and keys, which are also called the 'state files'. vMgr is the point of contact for the users to create, reactivate, suspend, resume, terminate and destroy the vTPM instances. The main process of vMgr is defined in Fig. 2.

The creation of a vTPM instance starts when the vMgr receives a request from user. vMgr is a service module which keeps running as a daemon to receive any request from the users. The request can be for creation, resume, suspend or destroy

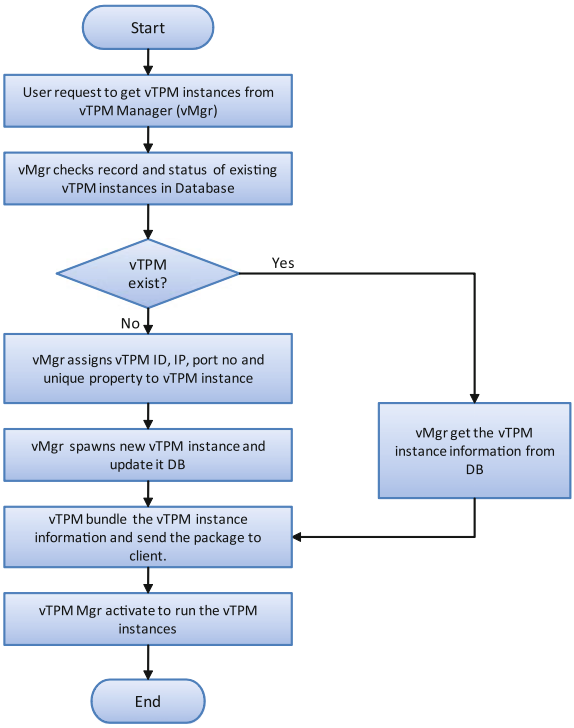

Fig. 2 Creation and management of vTPM instance flow diagram

the vTPM instance. Each new vTPM will be assigned parameters which include vTPM ID, port number, IP and unique property value by vMgr. Then, vMgr will spawn the new vTPM instance. The parameter will be stored in the database and also sent to the user.

In addition to the creation of vTPM instances and managing its status, vMgr also handle scalability issue. When a TPM server has run out of system resource, another TPM server is needed to answer the users' request to spawn a vTPM. Here, the main vMgr will manage and decide which TPM server should accept the request. For this, the main vMgr must first register itself to the system. Before the process of ensuring scalability in spawning that vTPM can happen, the main vMgr must register itself to the system. Once registration has been completed, as a request arrives, the main vMgr would either spawn the vTPM instance or pass the request to a Secondary vMgr if it has exhausted its resources.

The main vMgr handles all registration of the Secondary vMgr which has just joined to the system. Once the main TPM server has reached its maximum spawned vTPM, the vMgr that handles vTPM instances in that server interacts with the registered Secondary vMgr located at a different TPM server and request that vMgr to spawn a new vTPM instance using parameters given during the interaction process given by the main vMgr. A centralized database is used to store

all data of the Secondary vMgr. The data consist of the IP and port number, as well as an ID of the Secondary vMgr.

Therefore, when the main vMgr wants the Secondary vMgr to spawn a new vTPM instance, the main vMgr will retrieve the represented data of the Secondary vMgr from the database, and uses that data to send request to the Secondary vMgr to spawn a vTPM instance. Once the Secondary vMgr retrieved the request, it will then spawn a new vTPM based on the parameters given by the main vMgr and updates the centralized data indicating that the spawned vTPM ID is being spawned from that server and managed by the new vMgr.

Another function of vMgr is to handle the fault tolerance mechanism. This is separated into two processes: replication and fault tolerance itself. As mentioned earlier, the process starts with user sending request for a vTPM to the vTPM Manager Primary (vMP) to spawn a vTPM instance. Next, the vMP will generate IP and port number for the vTPM and record the information into the database. After the updating, vMP will spawn a vTPM instance. Then, the vTPM creates the state file and updates the vMP. The vMP will check updates of the state file periodically. If there is an update, vMP will send updated vTPM state file to vTPM Manager Primary Backup (vMPB). The vMPB will retrieve the virtual machine ID or user ID from the vTPM state file. The vMPB will get virtual machine data or user data from database. The vMPB will spawn vTPM accordingly and end the process.

The process above will ensure that the backup server will always have updated state files of all the vTPM instances. In the event of failure of the main TPM server, an intelligent device routes the request to the backup server. Hence the users can still access their vTPM instances even though main TPM server is down. When this happens, the backup TPM server will take over the functions of the main TPM server. Once the faulty machine is has recovered, the vMgr will ensure that the record is synchronized again between the main TPM server and its backup server.

4.2 Prototype Implementation

Referring to the above framework, installation of all components must be established in order to proceed with the development. The current phase of implementation setup is focusing on providing the environment development for TPM server and its components. Some of the components are based on [10, 11] implementation such as OpenSSL engine, TrouSerS TCG Software Stack, TPM Tools, the TPM Device Driver and the modified TPM Emulator. The main components are vMgr daemon and database such as using mySQL. The vMgr daemon is configured to be able to accept requests from any virtualization API for future implementation of cloud computing.

In our prototype, we use Libvirt [12] as the virtual machine control engine. Libvirt is a library which provides the functions to provision, create, modify,

monitor, control, migrate and terminating an operating system running on a virtual machine. Due to the use of Uniform Resource Identifier (URI), Libvirt is able to support different types of hypervisors such as QEMU-KVM and Xen. In the prototype, Libvirt is used directly by TMCI.

We have conducted the system test on the components after it is integrated into our proposed framework to ensure the system runs well. The test involved up to 1000 instances of vTPM and works as expected. Each component after the integration in our framework can still run its original and enhanced functions.

5 Conclusion

In this paper, we have provided a framework for users to access TPM functionalities on a remote server. Typical scenario would be when the users of the system are using legacy computers or devices which does not have a built-in TPM but would still want to benefit from the Trusted Computing technology. By providing this framework however, the system needs to cater for the growing number of users and also the recovery from failure. Hence we have provided ways for the server to overcome this using the scalability feature and also the fault tolerance mechanism. With this approach, users can access the servers at any time with the ease and reliability that comes with the whole framework.

References

1. Trusted Computing Group: http://trustedcomputinggroup.org
2. TPM Main: Part 1 design principles. 1.2 revision 85 edition, (2005)
3. Berger, S., Caceres, R., Goldman, K.A., Perez, R., Sailer, R., Doorn, L.v.: vTPM: virtualizing the trusted platform module. In: 15th USENIX security symposium (2006)
4. Stumpf, F., Benz, M., Hermanowski. M., and Eckert, C.: Approach to a trustworthy system architecture using virtualization. ATC 2007, LNCS 4610, pp. 191–202, Springer (2007)
5. Wang, W., Zhang, Y., Lin, B., Wu, X.Y., Miao, K.: Secured and reliable VM migration in personal cloud, 2nd international conference on computer engineering and technology (ICCET), IEEE (2010)
6. Dai, W., Jin, H., Zou, D., Xu, S., Zhen, W. and Shi, L.; TEE: A virtual DRTM based execution environment for secure cloud-end computing. Proceeding CCS'10 proceedings of the 17th conference on computer and communications security, ISBN: 978-1-4503-0244-9, ACM (2010)
7. Shilin, Z., Mei, G.: Distributed multimedia content processing based on web service. Proceeding of international forum on computer science-technology and applications, ISBN: 978-0-7695-3930-0, IEEE (2009)
8. Morel, G., Pétin, J.F., Johnson, T.L.: Reliability, maintainability, and safety. Springer handbook of automation (2009)
9. Clarke, J., Dede, C.: Robust designs for scalability. AECT research symposium, June 22–25, Bloomington, Indiana (2006)

10. Norazah, A.A., Lucyantie, M.: Identity credential issuance with trusted computing, 2nd international conference on computing and informatics, ICOCI'09 (2009)
11. Lucyantie, M., Norazah, A.A., Habibah, H., Mohd Anuar, M.I.: Attestation with trusted configuration machine. Proceeding of international conference on computer applications and industrial electronics ICCAIE, ISBN: 9781457720574, IEEE (2011)
12. The virtualization API, http://libvirt.org/