



# Research and Development of TPM Virtualization

Liang Tan<sup>1,2(✉)</sup>, Huan Xiao<sup>1</sup>, and Juan Wang<sup>3</sup>

<sup>1</sup> College of Computer Science, Sichuan Normal University, Sichuan, China  
tanliang2008cn@126.com

<sup>2</sup> Institute of Computing Technology, Chinese Academy of Sciences,  
Beijing, China

<sup>3</sup> School of National Cyberspace Security, WHU, Wuhan 430072, China

**Abstract.** Combination of Cloud Computing and Trusted Computing is an important method to build a trusted cloud environment, and the most critical problem is the virtualization of TPM (Trusted Platform Module, TPM). But in view of the current research, TPM virtualization still not only does not meet the whole TCG specification, but also has a lot of security issues, and it is becoming the bottleneck of building a trusted cloud environment by combination of Cloud Computing and Trusted Computing. This paper introduces the basic concepts, types and basic requirements of TPM virtualization. The classification model of TPM virtualization is put forward by the I/O device virtualization technology. The main research work of the key technologies of TPM virtualization, such as architecture, key management, certification trust extension, migration and so on, are described in detail, moreover taking time as the clue, we can display a panoramic view of the evolution of related key technologies. Combined with the existing research results, the research direction and challenges of TPM virtualization under TCG architecture are discussed.

**Keywords:** Trusted computing · Cloud computing · Virtual platform module (TPM) · TPM virtualization · vTPM

## 1 Introduce

The Trusted Platform Module (TPM) is the core of trusted computing [1–9]. It is the root of trust in trusted computer systems and the key technology for trusted computing. TCG (Trusted Computer Group, TCG) [1] defines the core functions of the trusted computing platform: measurement, storage and reporting are all dependent on TPM; trusted roots of trusted computing platform: Root of Trust for Measurement (RTM), Root of Trust for Storage (RTS), and Root of Trust for Report (RTR) are all directly related to TPM. The RTM consists of a set of PCR memories in the CRTM (Core Root of Trust for Measurement, CRTM) and the TPM. The RTS consists of a TPM and a storage root key SRK. The RTR consists of a TPM and an EK. TCG is not only the initiator of trusted computing ideas, but also the organizer and promoter of trusted computing technology. In order to promote the widespread application and further extension of trusted computing, TCG has established a series of working groups, initially including Embedded System, Infrastructure, IoT, Mobile, PC Client, Server, Software Stack, Storage, Trusted Network

Communications and TPM, etc. [1], where the TPM Working Group is the basic and core working group are responsible for the drafting, revising and publishing the TPM specification, which has evolved from TPM1.0 and TPM1.2 to the current TPM2.0 [2–5].

In recent years, cloud computing [10–20], which is an emerging computing service, has rapidly emerged in various industries with its unique advantages of broadband interconnection, resource sharing, flexible configuration, on-demand services, and charging by volume. Users greatly reduce the burden of computing and storage by entrusting computing tasks and data to cloud service providers. It is worth noting that the operating environment provided by the cloud computing to the user (Tenant) is based on the virtual machine [10, 11]. The user's operating environment and data are stored in the cloud, thus users lose direct control of the physical environment. How to provide users with secure cloud computing services is an urgent problem [21]. Scholars in trusted computing at home and abroad are agilely aware of the credibility of that trusted computing can enhance the credibility of cloud computing environments. One of the effective ways to solve this problem is to build a virtual machine trusted running environment through TPM virtualization, such as vTPM (virtual TPM, vTPM) [22]. Trusted Computing Group TCG also quickly followed up on related work. First, in 2008, two new working groups, Cloud and Virtualized Platform, were added [1]. Cloud is mainly responsible for the drafting, revising and publishing the multi-tenant infrastructure, TMI specification, while the Virtualized Platform is mainly responsible for the drafting, revising and publishing the trusted virtual platform specification. One of its core tasks is TPM virtualization; secondly, the TPM 2.0 specification, announced in 2013, explicitly supports TPM virtualization [2–5].

The research on TPM virtualization has a great significance. In a cloud environment, TPM virtualization provides a trusted guarantee for the guest virtual machine running environment. TPM virtualization allows each guest virtual machine to logically have a single “unique” TPM, referred to as vTPM, which just like having a real physical TPM. The guest virtual machine environment can use features such as measure, storage, and reporting provided by vTPM. In particular, the trust chain transfer of the guest virtual machine environment can be implemented through the integrity check function of vTPM, the data protection function of the vTPM is used to implement the sealed storage of the guest virtual machine environment data, and the remote virtual certificate function of the vTPM is used to implement proof of identity of the guest virtual machine environment. In academia, since S Berger proposed the vTPM architecture in 2006 [22], research on TPM virtualization has gradually obtained widespread attention from many scholars and research institutions at home and abroad. In some international conferences, such as: TRUST2008 [23], ESI 2008 [24] (Proceedings of the 2008s International Conference on Emerging Security Information), UCC2015 [25] (IEEE/ACM 8th International Conference on Utility and Cloud Computing), CSS 2015 [27] (2015 IEEE 7th International Symposium on Cyberspace Safety and Security), TrustCom2012 [27] (2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications), MINES 2012 [28] (2012 IEEE 11<sup>th</sup> International Conference on Trust, Security and Privacy in Computing and Communications), CIS 2012 [29] (2012 Eighth International Conference on Computing Platform, Computational Intelligence and Security), Proceedings of the 2013 ACM workshop on Cloud computing security workshop [45], etc. There are a number of high-quality articles on vTPM issues, including vTPM system architecture, key management, certificate trust extension, and migration. TPM

virtualization issues (or related issues) at academic conferences on “National Trusted Computing and Information Security”, “National Computer Network and Information Security”, and “China Information and Communication Security” held in recent years in China is also a hot topic. In the industry, major IT giants also pay attention to TPM virtualization. A typical representative is Microsoft. Although Microsoft previously had the Next-Generation Secure Computing Base (NGSCB), in order to further improve the security of the cloud environment, vTPM is supported in Hyper-V [30] in Windows Server 2016 and Windows 10. Orchard’s virtualization platform Vbox [31] has supported vTPM in 2009. Most notable is that VMware, one of the virtualization industry giants, announced in 2015 that it supports vTPM in vSphere 6.0 [32]. In addition, well-known open source virtualization platforms such as XEN [33], KVM [34], etc., both support vTPM.

Despite this, the degree of convergence between trusted computing and cloud computing lags far behind market expectations. vTPM has not become the main security solution for major cloud platforms. TPM virtualization is one of the main bottlenecks affecting the application, promotion and popularization of trusted virtualization platform products, which seriously hinders the extension and expansion of trusted computing in the cloud environment. In particular, although China has developed a TCM (Trust Cryptographic Module) (TCM) related standard with independent intellectual property rights [9], similar to the TPM standard, compared with TPM2.0, the TCM standard has not considered the related issues of TCM virtualization. At present, there is no detailed and comprehensive review paper on TPM virtualization. In order to have a general understanding of the progress of this research and to promote domestic research in this direction, it is very meaningful to review the research progress of TPM virtualization.

This paper introduces the basic concepts, types and basic requirements of TPM virtualization, proposes a technical classification model of TPM virtualization, elaborates on the main research progress of key technologies such as system architecture, key management, certificate trust extension and migration of TPM virtualization, and summarize these work. Combined with the existing research results, the research direction and challenges of TPM virtualization under TCG architecture are discussed.

## 2 Basic Concepts of TPM Virtualization

In order to meet different functional requirements, there have been many different virtualization solutions for the x86 architecture. Combined with the computer system structure and the abstraction level adopted by virtual machines, virtual machine systems can be divided into instruction level virtualization, hardware level virtualization, operating program library virtualization [22–25]. For hardware-level virtualization, there are three main technologies, including full virtualization, para-virtualization, and hardware-assisted virtualization, all of which include CPUs, Memory and I/O virtualization. This article discusses TPM virtualization under the x86 architecture, which is a virtualized I/O device virtualization category.

### 2.1 Definition of TPM Virtualization

**Definition 1:** TPM virtualization, which is the application of I/O device virtualization technology to the TPM in the virtual computing platform, so that each virtual machine

in the platform can use trusted computing functions to improve securely, independently and autonomously the credibility and security of the operating environment to meet the trusted needs of virtual machine users.

As can be seen from Definition 1, the concept of TPM virtualization has three implications. First, in terms of virtualization technology, TPM virtualization is essentially I/O device virtualization. Second, in terms of functionality, each virtual machine can use the trusted computing capabilities provided by the TPM; Thirdly, in terms of security objectives, each virtual machine can improve the credibility and security of its own operating environment and meet the trusted requirements of virtual machine users through TPM virtualization.

## 2.2 Basic Types of TPM Virtualization

For hardware-level virtualization, depending on the virtualization implementation of VMM (Virtual Machine Monitor, VMM), the implementation of I/O device virtualization will be different. The core is the storage location of the I/O device’s native driver and how the VMM handles the I/O devices. TPM is a slow I/O device attached to the motherboard through the LPC bus. What kind of I/O device virtualization implementation technology is adopted depends on the method implementation of the VMM virtualization. Therefore, according to the different virtualization methods of VMM, we divide the TPM virtualization methods into three categories.

### 1. TPM virtualization under full virtualization

In full virtualization, the VMM can virtualize the virtual machine to the same physical environment as the physical TPM. All Guest OSs see a unified set of virtual TPM devices, and each I/O operation privileged instruction of the guest OS to the virtual TPM device will be trapped in the VMM, and then directly controlled by the TPM, so there is no need to modify the Guest OS. For x86 architectures, certain privileged instructions do not trap in the VMM, and at this time, binary code translation techniques will be used.

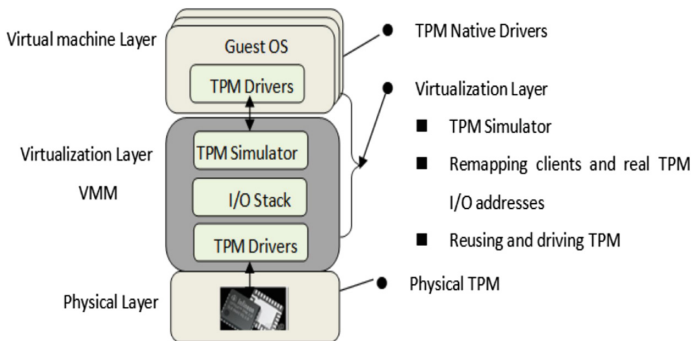


Fig. 1. TPM stand-alone Hypervisor model

According to the difference of the TPM native drive location, we can also classify this type of virtualization into a TPM independent model and a TPM host mode. The virtualization method of putting the TPM native driver directly into the VMM is called the TPM independent model, as shown in Fig. 1. The virtualization method of placing the TPM native driver in the host is called the TPM host mode, as shown in Fig. 2.

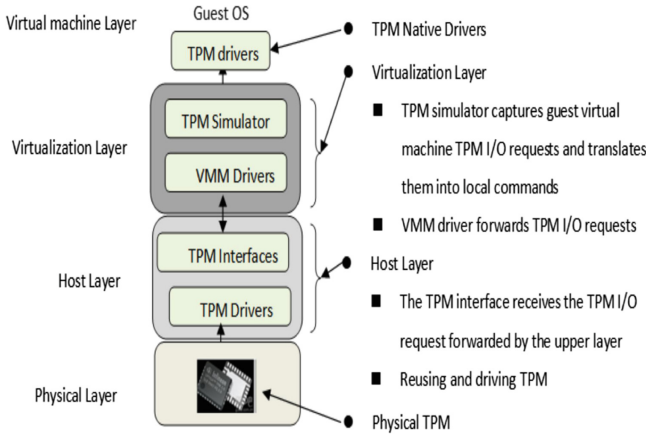


Fig. 2. TPM hosted-based model

## 2. TPM virtualization under para-virtualization

In the para-virtualization mode, all the privileged instructions accessing the TPM in the Guest OS need to be code-replaced, so the front-end virtual device drivers of different environments need to be developed. When the application makes an access request, the front and back drivers work together to complete the access. As shown in Fig. 3.

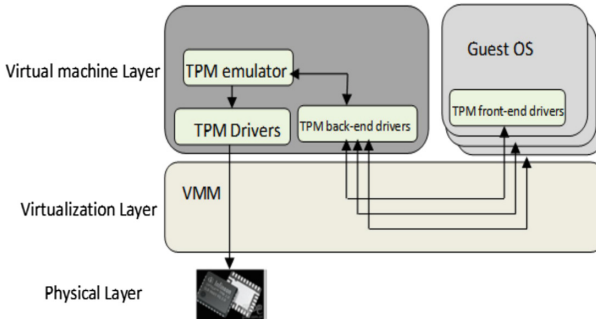


Fig. 3. TPM Para-virtualization

### 3. TPM virtualization under hardware-assisted virtualization

Under hardware-assisted virtualization, the VMM needs assistance of hardware to complete the virtualization of the TPM. At this time, the kernel of the Guest OS doesn't need to be modified. With the TPM virtual device model in the virtual machine management domain, the impact of the TPM privileged instructions on the host is eliminated by Intel's VT-x technology or AMD's AMD-V technology, which will run at different privilege levels. The VMM is implemented with different privilege levels to implement TPM virtualization, as shown in Fig. 4.

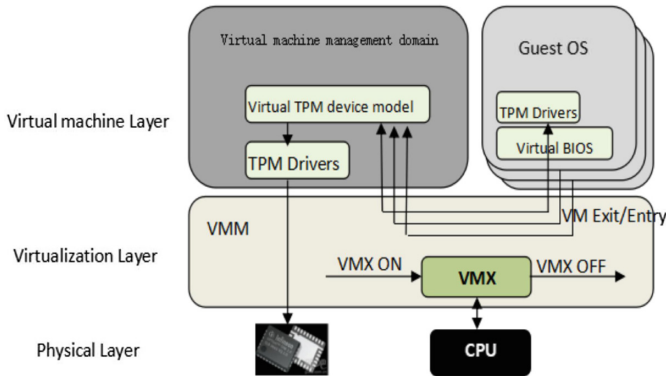


Fig. 4. Hardware-based TPM virtualization

### 2.3 Basic Requirements of TPM Virtualization

The purpose of TPM virtualization is to provide trusted computing services for virtual machines, to help virtual machines establish a trusted computing environment, and to make users who use virtual machines think that there is no significant difference between using a trusted computer system with a physical TPM. Therefore, when virtualizing TPM, there are four basic requirements [22–25].

#### 1. Equivalence

The so-called equivalence means that, except for the time factor, when the application or user in the virtual machine uses the TPM function, the rest must be the same as the computer system that owns the physical TPM alone, including the measurement, storage, and reporting of trusted computing.

#### 2. Relevance

The so-called relevance includes two aspects. One is that it should be related to the vTPM throughout the life cycle of the virtual machine, including the migration of the virtual machine. The second is that the vTPM maintains a correlation with the hardware TPM.

#### 3. Safety

The so-called security includes two aspects. One is that the physical TPM should be fully managed by the VMM, the guest virtual machine operating system,

applications or users cannot directly access the TPM; the second is that the security of the TPM virtualization software system and architecture. Virtual platforms must meet these two security requirements for TPM virtualization.

4. **Convenience**

The so-called convenience refers to that after the TPM virtualization was completed, it is convenient to maintain, upgrade and migrate.

3 **Technical Classification Model of TPM Virtualization**

TPM virtualization is essentially I/O device virtualization. We divide TPM virtualization into three categories according to I/O device virtualization technology. One is software instantiation simulation, the other one is hardware sharing, and the third one is aggregation method. Below we introduce these three categories one by one.

3.1 **Software Simulation Method**

The so-called software instantiation simulation method means that the VMM creates a software simulation virtual TPM instance for each virtual machine that needs to provide a trusted execution environment for the user. When the user needs the trusted computing functions, most of them are provided by their corresponding software simulation virtual TPM instance. The physical TPM only provides a small number of functions, such as key management, certificate extension, and so on. The basic architecture is shown in Fig. 5.

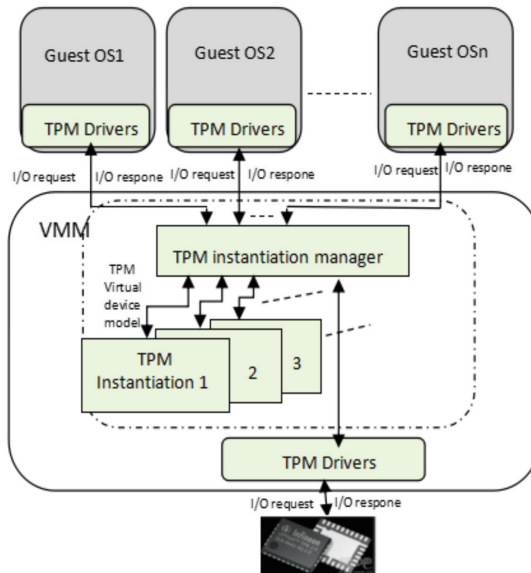


Fig. 5. Software simulation mode of TPM virtualization

This method can be done in either full virtualization, para-virtualization, or hardware-assisted virtualization. The essence of this method is to assign each virtual machine an instance of TPM software that has the same functions as the hardware TPM—vTPM. The vTPM usually runs in the VMM, virtual machine management domain, or isolated device domain, and can implement most functions of the TPM. The advantages of this method are simple implementation, flexibility, high performance and easy migration. The disadvantage is that the security isn't high, but because the vTPM runs in the VMM, virtual machine management domain or isolated device domain, therefore, in addition to privileged security threats and shared security, it can prevent general security threats<sup>1</sup>, that is, basic security can be guaranteed. Therefore, this method is currently the mainstream way to implement TPM virtualization on each virtual platform.

### 3.2 Hardware Sharing Method

The so-called hardware sharing method means that each virtual machine accesses the physical TPM in a time-sharing manner. Generally, the service requests and responses of the TPM are implemented in an asynchronous manner. When multiple client virtual machines issue TPM I/O requests to form a request queue, the VMM performs scheduling through certain policies and processes them in the TPM. After the TPM processed a request, the processing result is placed in the response queue, and the response result is returned to the corresponding virtual machine by the TPM virtual device model. The basic architecture is shown in Fig. 6.

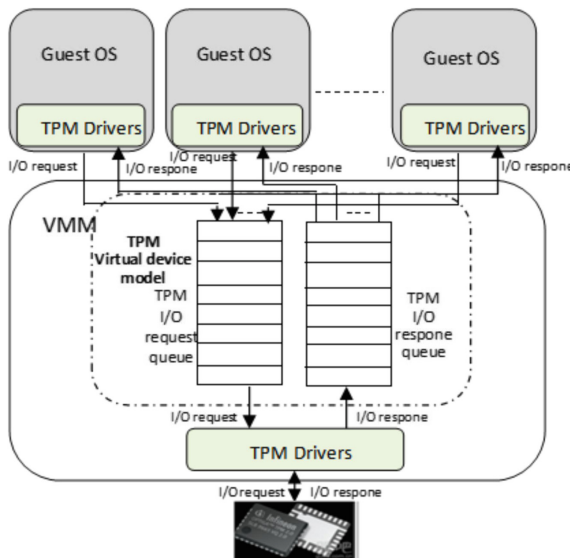


Fig. 6. Hardware share model of TPM virtualization

<sup>1</sup> The relevant connotations of general security threats, privileged security threats and shared security threats are described in the literature [61]



This method can be done in either full virtualization, para-virtualization, or hardware-assisted virtualization. The essence of this method is that the hardware TPM provides trusted computing functions for virtual machines. The advantage is high security, and it can prevent general security threats, privileged security threats, and some shared security threats, but the disadvantages are also obvious, it’s complicated, the mobility is poor, and since the TPM is a slow peripheral that is attached via the LPC bus, as the number of virtual machines increases, the sharing performance drops dramatically and it cannot be used at all.

### 3.3 Aggregation Method

The so-called aggregation method is to aggregate multiple TPM virtual functions in one TPM – VF-vTPM, and then directly allocate or schedule the vTPM to the virtual machine for use. The basic architecture is shown in Fig. 7.

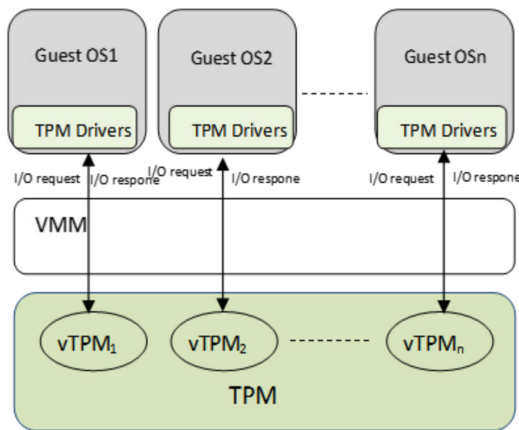


Fig. 7. Aggregation model of TPM virtualization

The aggregation method essentially allocates a VF-vTPM with the same functions as the hardware TPM for each virtual machine, and the vTPM runs inside the TPM. The advantages of this method are high security, good performance, and good scalability. However, the implementation and configuration are complex and the mobility is poor. In fact, the existing TPM 1.2 and TPM2.0 chips cannot achieve this method at all, because the LPC bus doesn’t support Single Root I/O Virtualization (SR-IOV) at all.

## 4 Key Technology Research and Development of TPM Virtualization

TPM virtualization involves key technologies in system architecture, key management, certificate trust extension, and migration, which are detailed in this section. Since the trust chain model and remote proof of trusted virtual machines belong to the category of “trusted computing platform”, this section will not discuss and analyze.

#### 4.1 System Architecture of TPM Virtualization

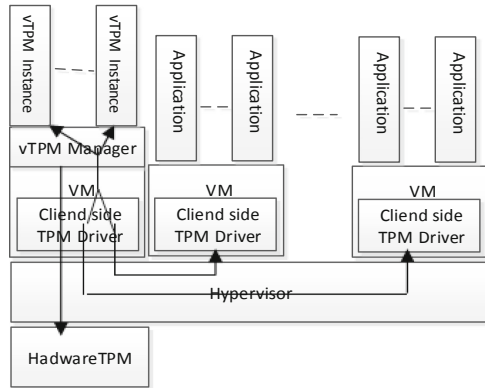
The so-called system architecture refers to the complete composition of the system and its reasonable logical function modules. According to the technical classification model of TPM virtualization in Sect. 3, we summarize the existing research results of TPM virtualization system architecture into three categories, one is the software simulation TPM virtualization system architecture; the other one is the hardware sharing TPM virtualization system architecture; the third one is the aggregated TPM virtualization system architecture. The so-called “software emulation type TPM virtualization system architecture” refers to the complete component architecture and reasonable functional modules for providing a complete composition architecture and reasonable functional modules for the virtual machine to access the software simulation TPM instances. The so-called “hardware sharing TPM virtualization system architecture” refers to the complete component architecture and reasonable functional modules that provide access to the hardware TPM for the virtual machine. The so-called “aggregated TPM virtualization system architecture” refers to the complete component architecture and reasonable functional modules that provide virtual machines with access to the TPM virtual function (VF).

**Software Simulation TPM Virtualization System Architecture.** For the “software simulation TPM virtualization system architecture”, whether domestic or foreign, whether academic or industrial, there are many research results.

In academia, the research results in this area were first proposed by S Berger<sup>2</sup> et al. of IBM TJ Watson Research Center in 2006 [22], as shown in Fig. 8. The system architecture includes vTPM, vTPM manager, and Client-Side TPM, Driver, Server-Side TPM driver, TPM and other entities, where vTPM, vTPM manager and Server-Side TPM driver are in the privileged domain, Client-Side TPM driver is in the virtual machine domain, and TPM is in the underlying hardware layer of the entire virtual system. vTPM is a software-simulated TPM that provides most of the trusted computing functions for the virtual machine. The vTPM manager is responsible for the lifecycle management of all vTPMs, including creating, suspending, restoring, and deleting vTPM instances, and forwarding the vTPM instance request to which the virtual machine is bound and returning the corresponding response. Each vTPM can be associated with the hardware TPM through the vTPM manager, and the associated content mainly includes PCR mapping and certificate chain extension. The Server-Side TPM driver is a privileged domain-side TPM driver. The Client-Side TPM driver is a virtual machine-side TPM driver. The Server-Side TPM driver communicates with the Client-Side TPM driver through the VMM. This literature is the foundation of TPM para-virtualization, and most of the research in this area is based on this idea.

---

<sup>2</sup> Although S Berger is a researcher at the IBM T. J. Watson Research Center, the published results are academic papers, so he is still attributed to academia.



**Fig. 8.** vTPM architecture

In 2007, the literature [36] also proposed a general system architecture GVTM for TPM virtualization, as shown in Fig. 9. The system architecture includes entities such as GVTM, GVTM manager, GVTM Factory, GVTM Protected Storage Service, etc. GVTM is a software entity that provides trusted computing functions. GVTM Manager includes Key & Session Manager and TPM Driver, is responsible for GVTM lifecycle management, GVTM binding to virtual machines and key and session management. The GVTM Factory is responsible for providing the GVTM manager with the ability to create and delete GVTM services. The GVTM Protected Storage Service is responsible for providing non-volatile storage for the GVTM manager. It can be seen from the above analysis that the basic ideas of the literature [22] and literature [36] are consistent, the difference is that the implementation of the vTPM manager function has changed. In the literature [22], the vTPM manager is responsible for the vTPM life cycle management and the protection storage of sensitive information, which is implemented by two other important entities, GVTM Factory and GVTM Protected Storage Service in the literature [36]. And the literature [36] doesn't specify that GVTM manager, GVTM Factory and GVTM Protected Storage Service are located in a certain specific area. The advantage of the system architecture proposed in literature [22] and literature [36] is that the vTPM instance is separated from the virtual machine, which is easy to implement, flexible to deploy, has little impact on the VMM, and is light burden for TPM. For example, the vTPM, vTPM manger, GVTM, and GVTM Manager instances are placed in the administrative domain, although which is vulnerable to privileged security threats, shared security threats, and heavy privileged domains, it can protect against general security threats.

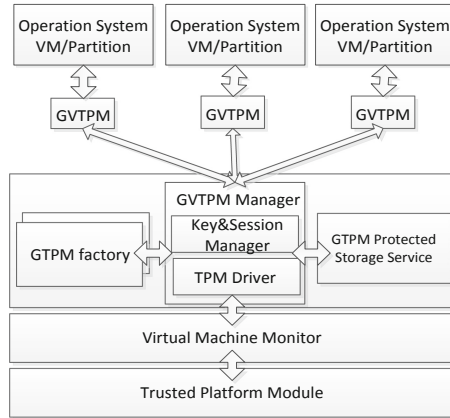


Fig. 9. vTPM general architecture

In order to improve the vTPM anti-privilege security threat capability, in the literature [22], proposing a vTPM architecture variant, as shown in Fig. 10, where the vTPM function is provided by an external security coprocessor plug-in, IBM’s PCIXCC, which can provide maximum security for sensitive data such as private keys in a vulnerable environment. The first virtual machine is the owner of the hardware. It obtains a vTPM instance for itself. Other vTPM instances are reserved for other virtual machines. A Proxy process is responsible for passing TPM I/O instruction information between the server driver and the external security coprocessor plug-in. It is worth noting that the vTPM running in PCIXCC is still a software type vTPM, and it’s relatively small. Many security functions, such as cryptographic operations, are provided by hardware in PCIXCC. The advantage of this system architecture is that the vTPM instance and the vTPM manager are located in the security processor and aren’t vulnerable to privilege security. But it needs to add additional hardware and cost.

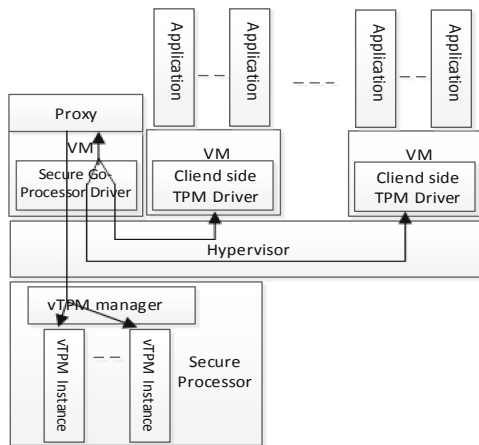


Fig. 10. Security coprocessor-based vTPM architecture

In 2007, the literature [37] based on Xen for the first time to separate the vTPM instance from the privileged domain, as shown in Fig. 11, the vTPM manager and the vTPMD daemon still run in Domain0, and the vTPM instances run in the LibOS-based Trusted VM domain. The applications in the DomU pass the TPM command to the vTPM instances through vTPMmanager and vTPMD. The vTPM instances can also return the running result to the applications in the DomU through vTPMD and vTPM manager. The privileged domain and the isolated domain exchange information through the inter-domain communication mechanism IDC. The biggest advantage of the system architecture is to improve the vTPM’s ability to resist shared security threats, reduce the burden on Domain0, and facilitate vTPM migration. However, vTPM has no connection with TPM, and the trust of the underlying virtual platform cannot be extended to virtual machines.

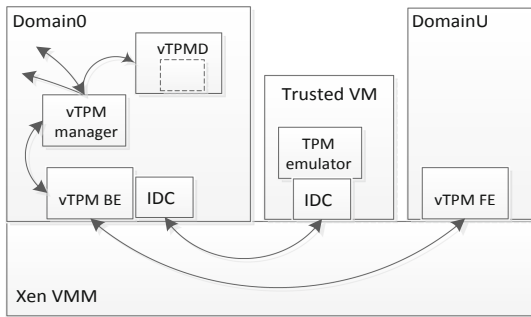
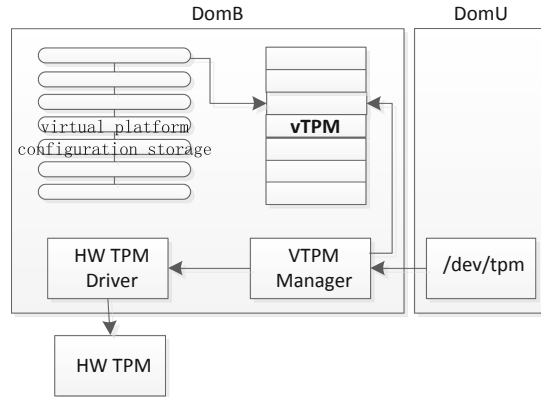


Fig. 11. Single isolated domain vTPM architecture for Xen

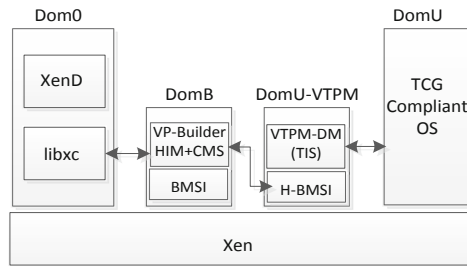
In 2008, the literature [38] further separated the vTPM manager, vTPM instances and privilege domain, as shown in Fig. 12. Firstly, an important function in the privileged domain, the domain management function, was separated. The Domain Builder domain is built based on MiniOS, which is called DomB, and the vTPM manager and vTPM instances are separated from the privileged domain to DomB. In addition, DomB also includes the TPM’s native driver and virtual platform configuration storage. The native driver is to facilitate the interaction between the vTPM manager and the hardware TPM. The main function of the virtual platform configuration storage is to save the status information of the vTPM for recovery. The DomU can directly interact with the vTPM manager in the DomB. The vTPM front-end in DomU is unchanged. Compared with the literature [37], although only one isolated domain has been added, the work of literature [38] goes further. The isolated domain of the literature [37] only runs one vTPM instance, the vTPM instance lifecycle management, binding to the VM, and integrity measurement for the virtual machine are still done in the privileged domain. The DomB in literature [38] not only separates vTPM instances from the privileged domain, but also separates vTPM instance lifecycle management, VM binding, and virtual machine integrity measurement from the privileged domain. This system architecture reduces the performance burden and shared security threats of privileged domain.



**Fig. 12.** Single isolated domain vTPM architecture

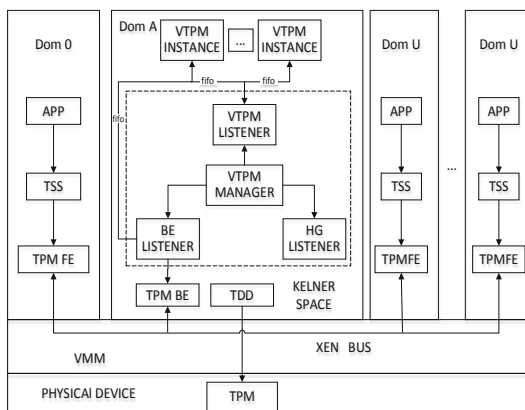
In 2009, OpenTC, a trusted computing project sponsored by the European Commission, presented a research report on the TPM virtualization architecture [39]. This report not only introduced the background and motivation of TPM virtualization, defined the vTPM status data structure, analyzed the existing vTPM architecture, restrictions, vTPM life cycle, vTPM and VM binding, etc., but also added vTPM attribute-based proof and migration functions [40] to the vTPM architecture, and a dual isolation domain is proposed on the Xen platform. The system architecture further separated the functions of DomB in literature [38]. As shown in Fig. 13. Two domains are included in the system architecture, one of which is DomB, which is similar to DomB in literature [38], but its functions is reduced. There are no vTPM manager and vTPM instances in the DomB. DomB, including VP-Bulider, HIM, CMS, and BMSI, is responsible for securely creating trusted virtual domains. VP-Bulider is responsible for creating virtual domain, HIM is trusted virtual domain integrity measurement management, CMS is virtual domain configuration, and BMSI is the basic management and security interface; second is the other one of which is DomU-vTPM. DomU-vTPM domain, including VTPM-DM, TIS and H-BMSI, runs vTPM instances, where VTPM-DM is the vTPM device model, TIS is trusted Computing interface specifications, H-BMSI is TPM-related basic management and security interfaces. When a vTPM instance is generated, the vTPM generates the EK through the H-BMSI. The private key is protected by the BMSI of the DomB and can only be accessed by the vTPM through the H-BMSI. The hash cipher of the public key, as an external nonce of the vTPM performing reference operations through the H-BMSI interface, implicitly associates with the integrity measurement of Xen, DomB, and vTPM; when starting a new virtual machine, first attached the vTPM domain ID associated with the virtual machine to the configuration table of the virtual machine, the VP-Builder then measures the vCRTM of the virtual machine, and the result of the measurement is stored in the HIM database as the vPCR0 value, and then the VP-Builder verifies the device model of the vTPM to ensure that the virtual machine can correctly access the TIS interface provided by the vTPM device model. Finally, VP-Builder VP-Builder updates the HIM database and establishes the interdependence between vTPM and VM. This

system architecture further reduces the performance burden and shared security threats of privileged domain.



**Fig. 13.** Double isolated domain vTPM architecture

In China, the similar literatures appeared in 2010. The literature [41] believes that considering the entire privileged domain as TCB will threaten the security of vTPM, because TCB is too large, it is easy to generate loopholes. In order to solve this problem, a new vTPM architecture is proposed in the literature [41], as shown in Fig. 14. The vTPM, vTPM manager, and TPM native drivers originally in the privileged domain are separated into the management domain DomA by creating a new administrative domain, DomA. There are two purposes for creating an administrative domain. One is to protect the vTPM and its related components from unauthorized access and invocation. The other one is to modify the access process of the TPM and protect the DomA through the TPM to improve the security of the vTPM and its related components. The idea of the system architecture is basically the same as that of the literature [37–39].



**Fig. 14.** Dom A -based vTPM architecture

In 2015, in order to prevent attackers in the cloud environment from using the virtual machine’s rollback mechanism (an important and common used function) to launch attacks, the domestic literature [42] proposed a trusted virtual platform module (rollback- resilient TPM, rvTPM) system architecture to resist rollback attacks based on Xen, and implemented the rvTPM prototype system in Xen, as shown in Fig. 15. There are four modules: (1) Information Collection Module, which is responsible for collecting information about rollback events. (2) An information registration module that records and shares status information collected by the information collection module. (3) Rollback module, which performs snapshot and rollback operations. (4) Rollback log module, which records the rollback events. As can be seen from Fig. 12, the information collection module, the rollback module, and the rollback log module are in Dom0, and the information registration module is in Xen VMM. The vTPM back-end driver interacts with information collection module, rollback module, and rollback log module, and the rvTPM manager interacts with the information registration module. Compared with the previous literatures, the system architecture follows the earlier TPM para-virtualization method, but it can prevent virtual machine rollback attacks, reduce general security threats, and further improve the security of the management domain and vTPM.

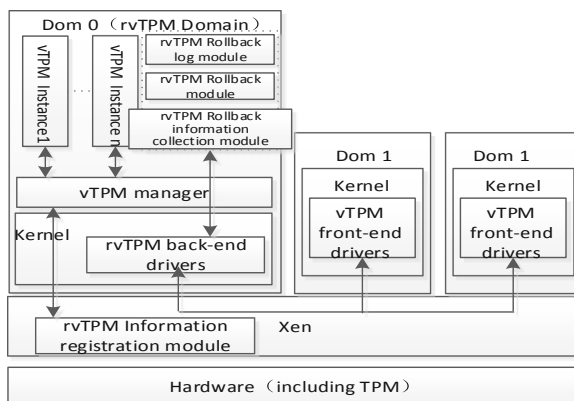


Fig. 15. rvTPM architecture for Xen

In the open source community, virtual platform systems Xen and KVM support TPM virtualization. Xen [34] is the earliest open source virtual platform system that supports TPM virtualization. Many academic researches on TPM virtualization use Xen as an experimental platform. In the beginning, Xen’s TPM virtualization system architecture is shown in Fig. 16. The system architecture includes vTPM instances, vTPM manager, vTPM back-end driver and front-end driver, TPM and other entities. The TPM instances, vTPM manager, vTPM back-end driver are in Dom0, vTPM front-end driver is in the virtual machine domain, and TPM is in the underlying hardware layer of the whole virtual system.



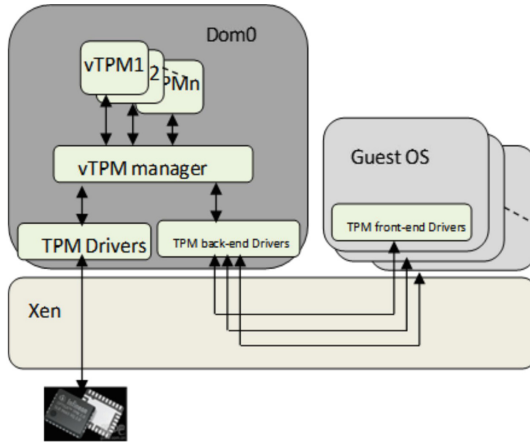


Fig. 16. Early vTPM architecture for Xen

In July, 2013, Xen published the version 3.2, and the TPM virtualization system architecture changed further. As shown in Fig. 17. Xen separates vTPM manager, vTPM and VM bindings, and other functions from Dom0, reducing performance burden of Dom0 and reducing shared security threats.

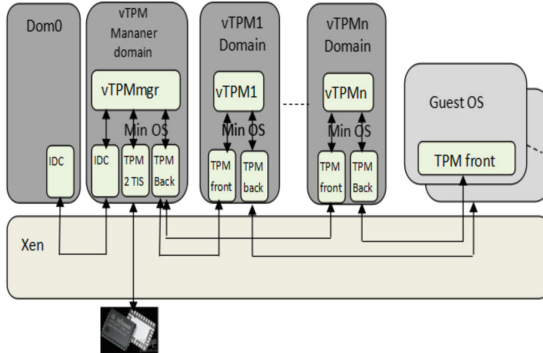


Fig. 17. Current vTPM architecture for Xen

KVM [35] supports TPM virtualization later than Xen, and KVM’s TPM virtualization relies heavily on QEMU. In 2007, QEMU supported access to the TPM emulator through patches, enabling virtual machines to use the trusted computing functions provided by the TPM emulator. In April, 2013, QEMU added new patches to enable virtual machines to directly use physical TPM through Pass Through technology, but in this way, only one virtual machine can use the TPM device on one physical machine, and the physical computer and the virtual machine cannot use the TPM chip at the same time, which limits the application of the trusted computing functions on the

virtual machines. In December, 2013, QEMU added new patches again, enabling multiple virtual machines to simultaneously use TPM functions simulated by software. The functions and interfaces of virtual TPM are consistent with physical TPM chip. The system architecture is shown in Fig. 18. The system architecture includes libtpms, QEMU with TPM, SeaBIOS, Kernel IMA, and other entities. The libtpms library is used to provide TPM functions to QEMU, including symmetric/asymmetric encryption, secure storage, integrity measurement, and security signatures. QEMU with TPM actually adds a TPM device type and a back-end driver for the TPM device in QEMU. The back-end driver of the TPM device receives the TPM device operation sent by the virtual machine, parses the operation type and calls the corresponding interface of libtpms. The TPM device operates and returns the results of the operation to the virtual machine. SeaBIOS is an open source implementation of a BIOS under the x86 architecture that completes the initialization of the hardware. The system architecture is complex, involving many components and entities, and libtpms has few links with the underlying physical TPM, and there are also privileged security threats and shared security threats.

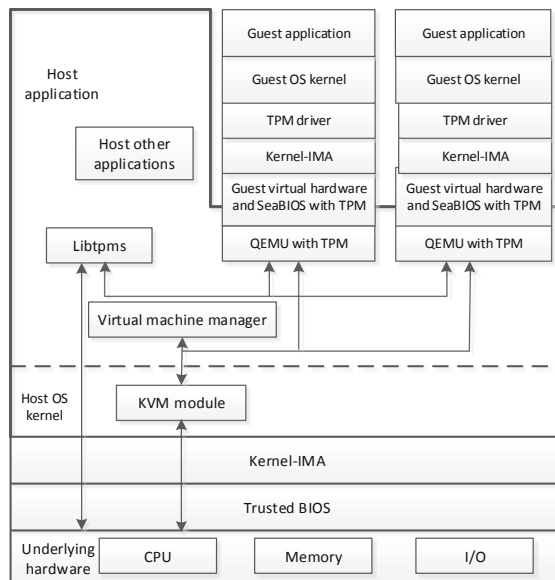


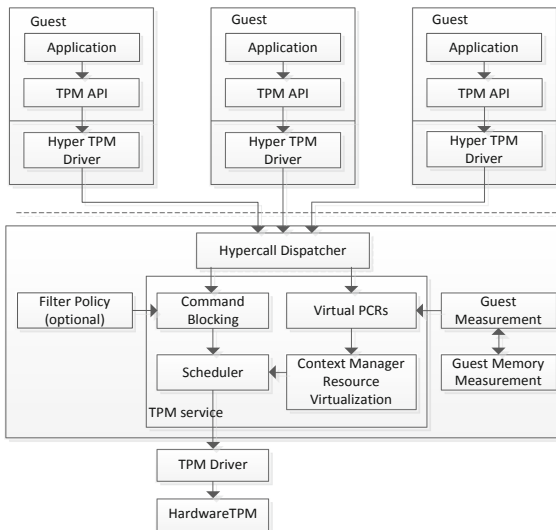
Fig. 18. vTPM architecture for KVM

For the industry, VMware [33] also announced that supporting for vTPM in vSphere 6.0 [33] in 2015; Microsoft [34] supports vTPM in its Windows sever 2016 Hyper-V, Orical [32] VBox supports vTPM through IBM PCIXCC. But neither VMware, Microsoft nor Orical has disclosed its system architecture.

In summary, because software simulation TPM virtualization has the characteristics of simple implementation, good flexibility, high performance and convenient

migration, whether it is academic, industrial or open source community, there are many researches on software simulation type TPM virtualization system architecture, and more research results have been obtained. To sum up, the software simulation TPM virtualization system architecture mainly focuses on two aspects. First, how to prevent various security threats, especially privileged security threats and shared security threats, to improve the security of the system architecture; secondly, how to improve the performance of TPM virtualization and reduce the security and performance impact on the virtual platform.

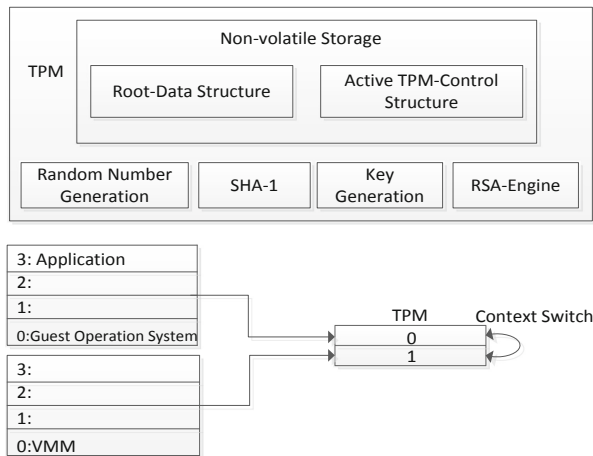
**Hardware Sharing TPM Virtualization System Architecture.** For the “hardware sharing TPM virtualization system architecture”, there are many studies in foreign academic circles, and there are relatively few studies in domestic academic circles. There is no similar design and implementation across the industry and the open source community.



**Fig. 19.** Hardware TPM sharing architecture

In the academic world, the earliest research results appeared in 2008. The literature [23] first proposed the physical TPM sharing system architecture, which belongs to “hardware sharing TPM virtualization”. As shown in Fig. 19, the system architecture includes the HyperTPMDriver in the guest VM, the HyperCallDispatcher in the VMM, the TPM service, the Filter Policy, the Guest Measurement, the Guest Memory Management, and the TPM Driver. The HyperCallDispatcher accepts requests from various guest virtual machines and distributes them. The TPM service, is the core of the entire system architecture, provides TPM function services for each guest virtual machine, including Command Blocking, Virtual PCRs, ContextManage, Resource Virtualization, and Scheduler. Because the physical TPM is an exclusive device with limited

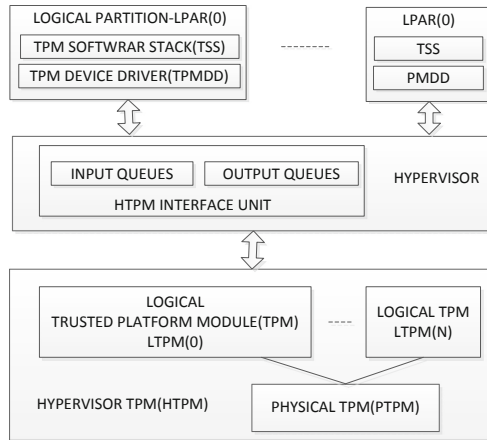
resources, when sharing a physical TPM, each guest VM not only needs Resource Virtualization to share limited key resources, such as key pool, authorization session pool, etc., but also needs to manage the context of its runtime and schedule it. Filter Policy is a filtering policy. Guest Measurement and Guest Memory Management are mainly responsible for the integrity measurement and memory management of the guest virtual machine. However, this literature is not specifically implemented. The advantage of this system architecture is that it has high security and can prevent general privileged security threats and shared security threats. However, the management and scheduling of TPM are both in VMM and the VMM becomes larger, which increases the general security threats and the performance is not high.



**Fig. 20.** Supporting shared hardware TPM architecture and CPU protection mechanism

In the same year, the literature [24] redesigned a new TPM to support virtualization, and its structure is shown in Fig. 20. Compared with the existing TPMs, it can support multi-client virtual machine sharing, that is, it is like owning a TPM for each virtual machine. As can be seen from the structure diagram, compared with TCG’s TPM, a non-volatile storage is added, which contains two key data structures, one is Active TPM-Control, and the other is Root-Data. The Active TPM-Control data structure includes: SRK, PCRs, AIK, EK, monotonic counters, other non-volatile storage values, delegate authorization tables, TPM context data, DAA parameters  $f$ , and associated authorization data. The Root-Data data structure contains specific sensitive data, such as seal keys. When a virtual machine issues a TPM I/O request, the VMM is scheduled to load the virtual machine’s TPM-Control data structure into the TPM and decrypt it. The TPM performs the virtual machine’s I/O request according to the TPM-Control data structure and The result is returned to the guest virtual machine. Once another client virtual machine issues a TPM I/O request, the VMM will use the key in the Root-Data data structure to encrypt the TPM-Control data structure inside the TPM and load the TPM-Control data structure of the next virtual machine. The CPU hardware

protection mechanism is also proposed in this literature. The VM runs on the CPU ring 1, the VMM runs on the CPU ring 0, the VM running on the ring 1 can only operate its own TPM-Control data structure, and the VMM can manage the TPM-Control data structure of all VMs. In addition, an extended command set for managing TPM-Control data structures and sensitive instructions is provided, and algorithms and processes such as sensitive instruction processing, TPM-Control context switching, and TPM scheduling are described in detail. The biggest difference between this literature and the literature [36] is that TPM sharing is implemented by adding new functions and hardware protection mechanisms of the CPU in the TPM, so that the TPM changes from an exclusive device to a shareable device. The advantage of this system architecture is that it is more secure, can prevent general security threats and shared security threats, but it needs to prevent privileged security threats; on the other hand, it has higher performance. Since the system architecture manages and schedules the TPM by means of the CPU hardware protection mechanism, the VMM increases less software code.



**Fig. 21.** Sharing hardware TPM architecture for data process system

Literature [43] is a patent granted by the US Patent Office in May, 2008, and it proposes a physical TPM sharing scheme for data processing systems. As shown in Fig. 21, the system architecture includes TPM TSS and TPM DEVICE DRIVER in the logical partition system LPAR, INPUT QUEUES, OUTPUT QUEUES and HTPM INTERFACE UNIT in VMM, multiple LTPMs (Logical TPMs) in HTPM, and a physical TPM. The two entities in the LPAR TPM TSS and TPM DEVICE DRIVER are mainly used to provide interfaces and drivers for applications in the LPAR access to the TPM. In the VMM, INPUT QUEUES indicates all request queues that access the TPM, and OUTPUT QUEUES indicates the response queues. HTPM INTERFACE UNIT indicates the interface to access HTPM in VMM. LTPM is responsible for providing vast majority trusted computing functions for LPAR, and TPM is mainly

responsible for sensitive data protection. When the VMM creates a LPAR, it immediately creates an LTPM to bind to it. During data processing, the application in the LPAR sends a TPM request to the VMM to form a request queue IPUT QUEUES. The VMM schedules the request in the request queue to the HTPM, and the LTPM and the TPM jointly process the request and return a response OUTPUT QUEUES. The biggest feature of the system architecture is that the TPM generates multiple LTPMs and binds the LTPMs to the LPARs scheduling through the VMM. Since LTPM is located in HTPM, it is safer.

In 2011, the literature [44] believed that with the development of technology and application updates, such as mobile computing and cloud computing, TPM has not adapted to the needs of new technologies and the diversification of user needs. To this end, the paper proposed a new TPM system architecture Dynamic-Context TPM (dcTPM), as shown in Fig. 22. The system architecture includes a FPGA system-on-chip, various peripheral control interfaces, and multiple physical TPMs. The FPGA on-chip system environment is uClinux, which has a dctmp daemon, which is not only responsible for linking with host PC, memory, I/O, etc., but also responsible for controlling multiple software emulation vTPMs, and controlling multiple hardware TPMs through the LPC bus. Whether it is a software vTPM or a physical TPM, its context is managed by dcTmp, and dcTmp is scheduled according to the specific requirements of the user. The system architecture is characterized by integrated software TPM and hardware TPM, which is more flexible, can better meet the requirements of users, and has high security.

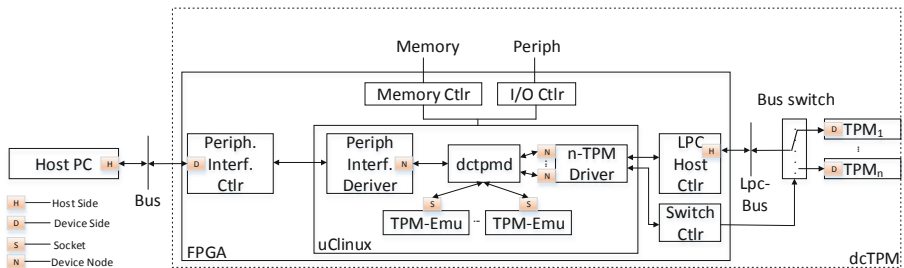
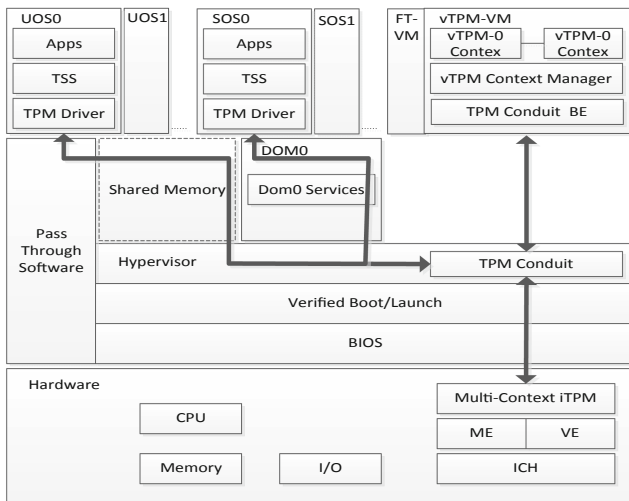


Fig. 22. dcTPM architecture

Literature [45] is a patent granted by the US Patent Office in September 2012, and proposes a system architecture that supports both software-simulated TPM virtualization and hardware-sharing TPM virtualization. As shown in Fig. 23, the system architecture mainly includes the UOS domain, the SOS domain, the vTPM-VM domain, the TPM Conduit, and hardware. UOS is the user virtual machine domain. The SOS domain is the service virtual machine domain. The vTPM-VM domain is a vTPM management domain, which includes vTPM0 to vTPMn Context, vTPM Context Manager and TPM ConduitBE. The vTPM0 to vTPMn Context mainly contains  $n + 1$  vTPMs and their contexts. The vTPM Context Manager is responsible for managing  $n + 1$  vTPMs and their contexts. The TPM Conduit BE is the back-end

driver for TPM Conduit and is responsible for interacting with TPM Conduit. The TPM Conduit is located in the VMM and is responsible for distributing TPM requests from UOS or SOS and returning responses from the vTPM-VM domain or Multi-Context iTPM. In the hardware layer, in addition to the traditional CPU, Memory and I/O, it also includes Muti-Context iTPM, ME, VE and ICH, where Muti-Context iTPM is a physical TPM, ME is its management engine, and VE is its virtualization engine, ICH is its input and output controller. The Muti-Context iTPM has two working modes, one is the independent device working mode, which is consistent with the traditional TPM. The second is the sharing mode, which supports TPM Context management. When a UOS or SOS sends a TPM access request, according to the specific requirements of this request, such as TPM version, performance requirements, etc., TPM Conduit forwarded the request to the vTPM-VM domain, which is processed and responded by the software type vTPM, or forwarded to the Muti-Context iTPM, handled and responded by the Muti-Context iTPM. Liking the literature [44], the system architecture also integrates the software TPM and hardware TPM, which is more flexible and can better meet the requirements of users, but the security is not as good as the literature [44]. In the literature [44], management of software and hardware TPM are implemented in FPGA, and this architecture is implemented by the VMM, which obviously makes the VMM larger, increases the general security threats.

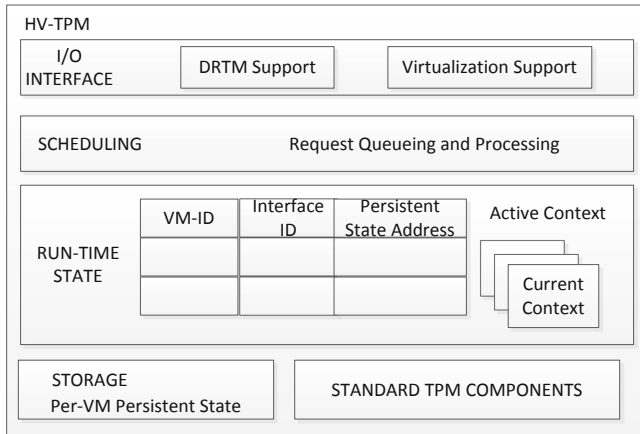


**Fig. 23.** Architecture for supporting both software simulation vTPM and Hardware sharing vTPM

In 2013, in order to create and manage a concurrent security execution environment for multi-core systems, the literature [46] proposed the HV-TPM system architecture, as shown in Fig. 24. The system architecture includes an I/O INTERFACE, SCHEDULING, and RUN-TIME. STATE, STORAGE Per-VM Persistent State and STANDARD TPM COMPONENTS, where I/O INTERFACE includes two entities,

one is DRTM Support, which supports dynamic integrity measurement; the other is Virtualization Support, which supports I/O virtualization. SCHEDULING is responsible for the management and scheduling of the request queue. The RUN-TIME STATE contains the VM status table and the current context of the TPM. The STORAGE Per-VM Persistent State is responsible for storing the status data that each virtual machine needs to maintain continuously. The system architecture is characterized by the creation and management of concurrent security execution environments for multi-core systems based on HV-TPM.

In 2013, the literature [47] proposed the para-virtualization architecture of TPM 2.0, as shown in Fig. 25. It is believed that the core functions of TPM2.0 are easy to support para-virtualized design. Therefore, it is not necessary to use software vTPM to simulate the functions of TPM 2.0 in VMM layer. Each virtual machine uses TPM Service to share TPM. The entire host computing platform includes VMM, privileged virtual machines, para-virtualized TPM service, and hardware. The TPM manager in the privileged virtual machines is responsible for the management of the hardware TPM. Para-virtualized TPM service includes Command Filter, Resource Manager, Log Manager, Sheduler, and Migration Manager. Command Filter is the TPM command parsing and filter, Resource Manager is responsible for TPM service resource management, Log Manager is responsible for TPM service log management, Sheduler is responsible for TPM scheduling, and Migration Manager is responsible for TPM virtualization migration. The advantage of the system architecture is that the security is high, but the management and scheduling of the TPM are both in the VMM and the VMM becomes larger, which affects the performance of the VMM and increases the general security threats of the VMM.

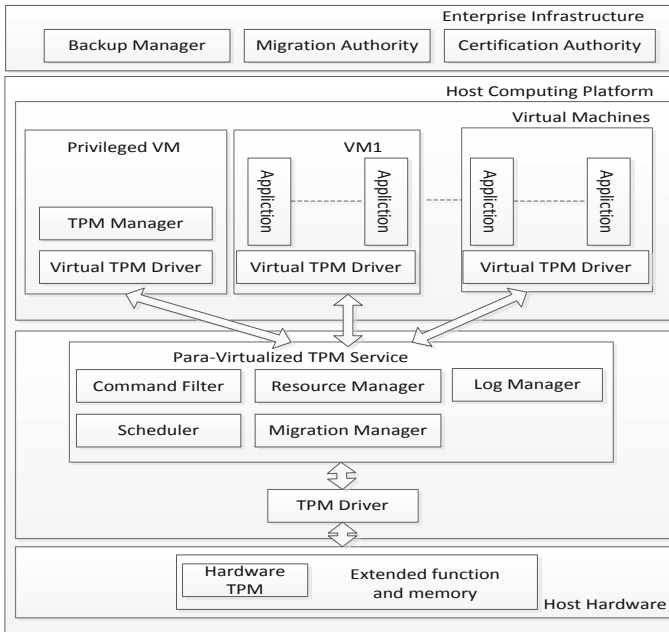


**Fig. 24.** HV-TPM architecture

In the industry, VMWare’s VMWare, Citrix’s XenServer, Microsoft’s Hyper-V, IBM’s PowerVM, etc., have not seen the release and display of relevant results. In the



open source community, including Xen, KVM, QEMU, Bochs, etc., this implementation has not been seen.



**Fig. 25.** Virtualization architecture for hardware TPM 2.0

In summary, due to the advantages and disadvantages of hardware-sharing TPM virtualization are both obvious, the advantages are high security, the disadvantages are complex implementation, poor performance and inconvenient migration, so there are many academic studies, but in industry and the open source community, there are few related engineering achievements. To sum up, the hardware-sharing TPM virtualization system architecture is mainly developed in three directions. The first is how to share the TPM, which includes two aspects. On the one hand, the TPM is shared by the virtualization management layer by adding TPM context management, scheduling, and improving the system architecture. On the other hand, the purpose of sharing the TPM is implemented by designing a new TPM. The second is how to improve performance, mainly to reduce complexity, such as TPM context management mode, TPM scheduling policy, and so on. Third, in order to better meet user requirements and adapt to new technologies, software simulation TPM virtualization and hardware sharing TPM virtualization architectures have obvious convergence trends.

**Aggregated TPM Virtualization System Architecture.** For the “aggregate TPM virtualization system architecture”, no matter whether it is academia, industry or open source community at home and abroad, there is very little research work at present, and no results have been published or displayed.

The key technology for “aggregated TPM virtualization” is based on I/O hardware virtualization technology, SR-IOV [48], a hardware-based virtualization standard protocol proposed by the PCI-SIG team, and it’s a subset of the PCI-Express protocol. The SR-IOV standard allows for efficient sharing of PCI-Express devices between virtual machines, and embeds a module in the PCI-Express hard core (this module is implemented in hardware). It theoretically can achieve the I/O performance and utilization which is comparable to the performance of the machine. SR-IOV technology is mainly applied to network card devices, which improves the speed and efficiency of virtual machines access to the network. Since the TPM is not a PCI device but a LPC device, the existing PCI device virtualization specification cannot be applied to the virtualization of the TPM device.

At present, part of the work of the TCG VPWG (Virtualized Platform Work Group) also involves how to build multiple vTPMs within the TPM to support virtualization, but it has not disclosed related solutions [1]. The domestic literature [49] studied the I/O hardware virtualization technology – SR-IOV, and proposed a virtual environment security isolation model based on SR-IOV technology for the security isolation of virtual computing environment. The model uses encryption card and Pass Through technology to implement network data isolation and data encryption isolation, where the function implemented by the encryption card is one of the main functions of the TPM.

We believe that although the TCG TPM specification only specifies the functions and architecture of the TPM, there is no limit to the implementation technology. However, it does indicate the interfaces between the TPM and the computing system: LPC. This limitation actually hinders the improvement of communication performance between the TPM and the computing system, and cannot meet the performance requirements of multiple virtual machines, dynamics, and concurrent aspects of the cloud environment. Therefore, it is an urgent task to study the TPM and its system architecture that meet the hardware virtualization technology SR-IOV.

## 4.2 Key Management of vTPM

The key management of the vTPM, is the same as the key management of the TPM, also includes setting the key type, attributes, storage hierarchy, and key lifecycle management. In order to meet the consistency of user usage and the consistency of the program interfaces, the key management of vTPM and TPM should be nearly the same.

The earliest paper on the vTPM architecture [22] pointed out that the key organization structure of each vTPM is similar to the independent key hierarchy of the TPM, and the key type is the same as the key type of the TPM, that is, in addition to the endorsement key, the other six types of keys, including storage key, identity certificate, signature key, binding key, legacy secret, and symmetric key, form a key tree, and the vSRK is the root of the key tree. The private key of the child node key is protected by the private key of the parent node key. The management of various key lifecycles, including key generation, loading, registration, and destruction, is in the privileged domain and doesn’t depend on the physical TPM. The benefits are that it facilitates the rapid generation of keys and facilitates vTPM migration. The key in the key hierarchy is encrypted by the symmetric storage key in the TPM and stored on the peripheral.

However, this literature doesn't discuss vTPM key attributes. In fact, due to the implementation mechanism and software features of vTPM, the key attributes of vTPM present new features.

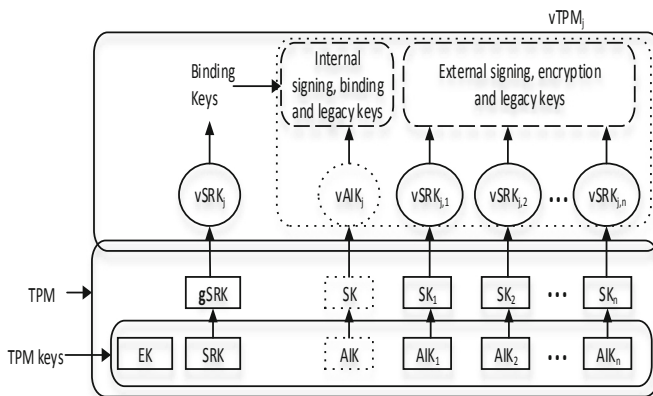
Literature [36] pointed out that the key attributes should change due to the different vTPM operating environment, which may violate the TCG specification's definition of the TPM key attributes. For example, the vEK key of vTPM, according to the TCG specification, should be a non-migratable key that can only be used to apply for an AIK certificate, but in vTPM, vEK is a migratable key in order to facilitate the migration of vTPM. It is used not only to apply for vAIK certificates, but also to encrypt or sign status data during vTPM migration. Therefore, vEK keys can only be legacy keys. Another example is the vAIK key of vTPM. According to the TCG specification, it should not be migrateable. It can only be used to sign the PCR value inside the TPM, but the vPCR of the guest virtual machine resides in the external memory, therefore, it happens outside the physical TPM that using the vAIK to sign the vPCR, which obviously doesn't conform to the TCG specification of the identity key. So the vAIK key can only be the signature key in the TCG specification. For another example, vTPM's storage root key vSRK. According to the TCG specification, it should be non-migratable. The key tree can only be protected inside the TPM, but in vTPM, the vSRK resides in external memory, and in order to facilitate vTPM migration, vSRK is also a migratable key. So vSRK should also be a legacy key (or encryption key). Therefore, although each vTPM has a separate key hierarchy similar to TPM, and its key attributes are shown in Table 1, this key tree hierarchy should be flatter. However, this literature doesn't discuss the storage structure of vTPM keys and key lifecycle management.

**Table 1.** Relationship between the key type of vTPM and the key type of hardware TPM

Virtual key type	Hardware TPM_Key_Type
Virtual endorsement key	TPM_LEGACY
Virtual storage key	TPM_LEGACY
Virtual attestation identity key	TPM_SIGNING
Virtual signing key	TPM_SIGNING
Virtual binding key	TPM_BINDING

Literature [50] proposed a key storage structure, which is suitable for VM-vTPM migration. As shown in Fig. 26. The vSRK is the root of the vTPM key tree, vAIK is the identity certificate key of the vTPM, SRK is the root of the TPM key tree, and AIK is the identity verification key of the TPM. gSRK and SK belong to the middle layer key. The vSRK is protected by the intermediate layer key, gSRK, which is protected by SRK and is a non-transportable asymmetric key. The purpose of introducing gSRK is to enable vSRK to migrate. SK is protected by AIK and is also a non-migratory key.

The purpose of SK is to associate vAIK with AIK to sign the migrated data. In addition, the vTPM key is divided into two categories, one is the internal keys, including vSRK, partial encryption, signature, binding and legacy key. Internal keys are not migrated when a virtual machine is migrated. The other type is an external keys, including partial signatures, encryption, and legacy keys, which are keys that sign and encrypt the migrated data when a virtual machine is migrated. In order to prevent link attacks of vTPM migration transactions when a virtual machine is migrated, a vAIK can only be used for one data migration of a VM. Therefore, vAIK is also non-migratorizable. However, this literature doesn't discuss the lifecycle management of vTPM keys.



**Fig. 26.** Key structure suitable for VM-vTPM migration

In October, 2013, TCG published TPM 2.0. The main change in key management is the generation method of master key. The TPM master key, including EK, SRK, etc., is generated by the key seed using the key derivation algorithm KDF. This not only saves TPM storage space, but also establishes a unified key management mechanism and migration mechanism for vTPM. After the TPM 2.0, vTPM has not changed in key type, storage hierarchy, etc. The main changes are reflected in the master key generation and master key attributes. Literature [47] is the first to propose that distributing the key for the vTPM of the virtual machine based on the key seed and generating an endorsement key and key tree, as shown in Fig. 26. As shown in Fig. 27, the vEK of the virtual machine vTPM is directly generated by the EPS, and the vEK cannot be migrated. The vSRK of the virtual machine vTPM is replaced by a general storage key, namely: a duplicateable storage key, and it can be migrated. SPS directly generates SRK instead of vSRK. Each virtual machine vTPM forms a key subtree with vSRK as the root, and the key subtree of all virtual machines is further composed of SRK as a larger key tree. However, this literature doesn't discuss vAIK.

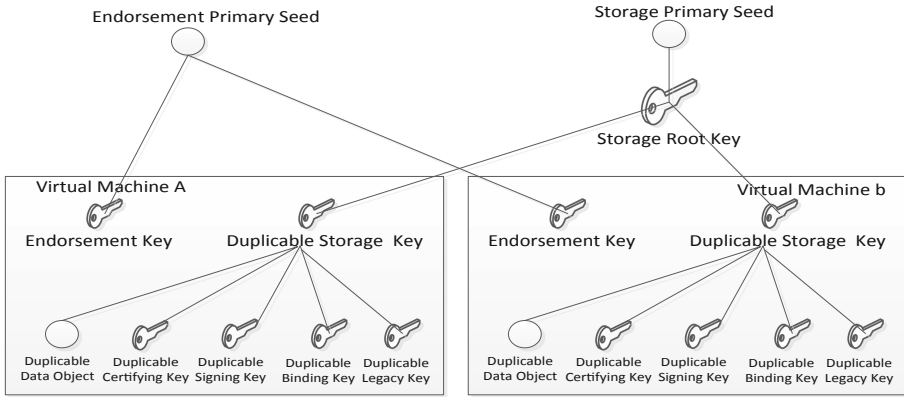
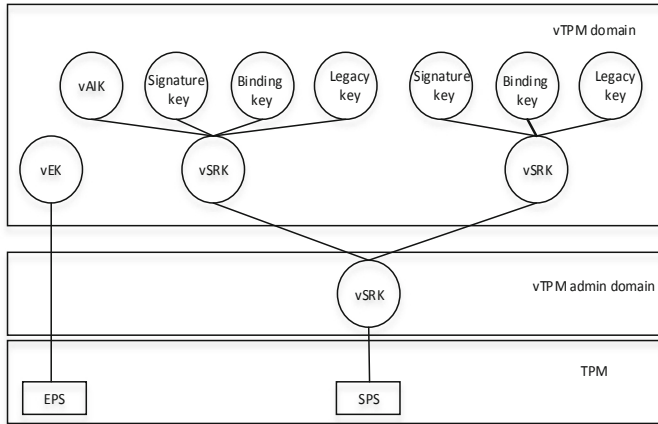


Fig. 27. vTPM key structure based on key seed

In 2015, the domestic literature [57] proposed a new generation of TPM virtualization framework design: Ng-vTPM, in which the vTPM key organization structure is designed, as shown in Fig. 28. In the Ng-vTPM, not only the virtual endorsement key vEK, the virtual storage root key vSRK, the virtual identity key vAIK (virtual attestation identity key), but also the signature key, encryption key, binding key and legacy key are designed. The vEK is generated by the physical TPM EPS and represents the real platform identity. It is used to identify the virtual machine Ng-vTPM and can defend against the fake platform identity. vSRK is the root key of the key hierarchy in Ng-vTPM, which is a migratable asymmetric storage key. The keys located in the vSRK storage hierarchy are identified as migratable keys. vAIK is the identity certificate key of the virtual machine, which is requested by the vEK to be verified by the certificate authority.. When each key is created and the parent key is used to encrypt the secret part of the subkey, different authorization sessions are used to ensure that the keys are not accessible without authorization. Moreover, according to the life cycle of the key, these keys are divided into three groups: vEK, internal migratable group and external non-migratory group. However, storing the SRK in the vTPM management domain violates the TCG specification, and a vTPM has multiple vSRKs, which is also inconsistent with the TCG specification.

From the above analysis, we can see that vTPM key management research mainly focuses on how to set key types and attributes, storage hierarchy and key lifecycle management. Since vTPM must meet the migration requirements, there is always a dilemma between setting the key type and meeting the migration requirements. Either it needs to change the key attributes that originally meets the TCG specification, or it needs to introduce key redundancy, which both can not meet the consistency requirements of vTPM and TPM key management, reducing the convenience of users using vTPM. In addition, the software emulation vTPM provides trusted computing services for virtual machines, especially cryptographic computing services are completed in the virtual platform’s memory (not within the TPM), and it are vulnerable to general security threats, privileged security threats, and shared security threats. It will seriously affect the security of virtual machines and vTPM.



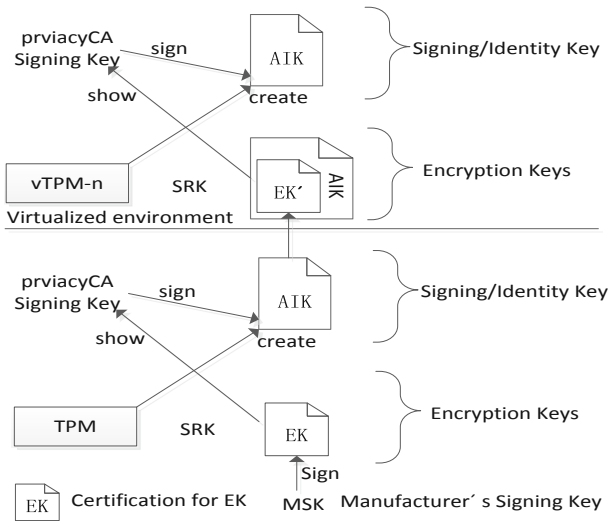
**Fig. 28.** Key structure of Ng-vTPM

### 4.3 vTPM Certificate Trust Extension

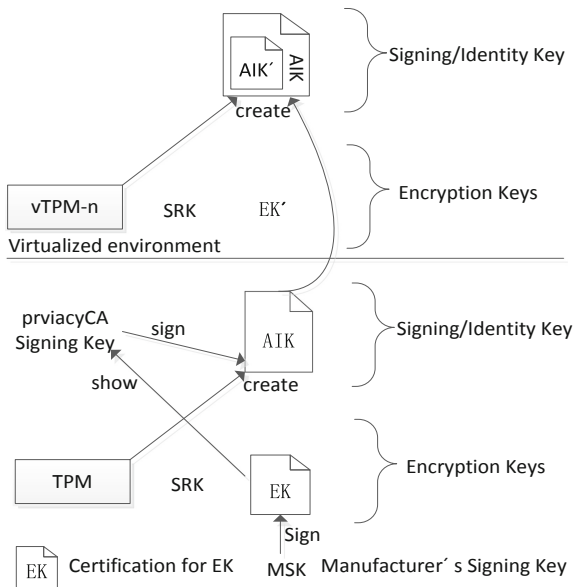
The credential trust extension of the vTPM refers to how to extend the certificate trust relationship of the physical TPM to the vTPM certificate and construct the certificate trust chain relationship from the TPM to the vTPM.

At present, a lot of research has been done on vTPM certificate trust extension at home and abroad. Literature [22] proposed four design ideas for constructing vTPM certificate chain in the research, but the last one is related to special hardware, so as not to lose generality, this paper does not discuss. We only discuss the first three:

- (1) vTPM vEK to hTPM AIK Binding, as shown in Fig. 29. In this method, both vEK and vAIK are generated by vTPM, vEK is bound by TPM’s AIK signature, and vAIK is used by Privacy CA. The private key is signed and the authenticator authenticates with the public key of the Privacy CA. This method not only extends the underlying certificate trust to the virtual machine, but also the vTPM certificate structure is consistent with the TPM, which is easy to understand and porting the existing results, but the disadvantages include two aspects. One is to use the TPM AIK to sign the vEK, which violates the TCG specification. In the TCG specification, AIK can only sign the information generated inside the TPM, but vEK is the external information of the TPM. Second, the validity period of the AIK is usually very short. The failure of the AIK causes the signature of the vEK to be invalid, which causes the vAIK to be invalid, which requires frequently re-apply vAIK from Privacy CA, so the performance burden of the Privacy CA is heavy.
- (2) hTPM AIK signs vTPM vAIK, as shown in Fig. 30, signing the vAI of the vTPM directly with the AIK of the TPM does not require a Privacy CA. This approach not only extends the underlying certificate trust to virtual machines, but also reduces the burden on the Privacy CA. However, the scheme relies on the APM of the TPM to sign the vAIK, which violates the TCG specification, and the failure of AIK will also invalidate its signature on vAIK. Therefore, needing to frequently



**Fig. 29.** vTPM vEK to hTPM AIK binding



**Fig. 30.** hTPM AIK signs vTPM vAIK

update the signature of the AIK on the vAIK, which increases the performance burden.

- (3) Local CA issue vEK Certificate, that is, the vEK for vTPM is issued by the local certificate authority (Note: not the Privacy CA), as shown in Fig. 31. The advantage of this solution is that the vEK is relatively stable and does not change

with the changes of the underlying virtual platform and the TPM. The disadvantage is that not only the additional certificate authority needs to be added, but also the TPM is not bound, that is, the trusted certificate extension of the TPM is not passed to vTPM.

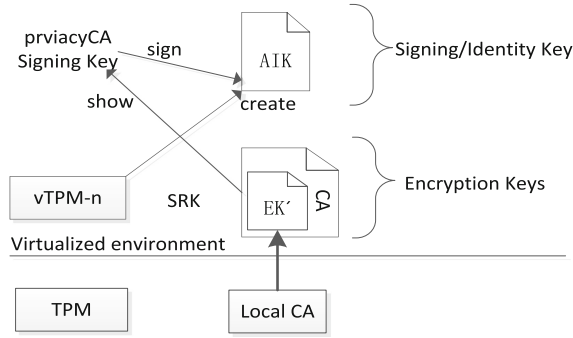


Fig. 31. Local CA issue vEK certificate

The literature [52] proposed the “vTPM vEK to hTPM EK Binding” scheme, that is, the vEK is bound by the TPM EK signature, as shown in Fig. 32. The advantage of this scheme is to avoid invalidation of the vEK signature due to the failure of AIK, the binding relationship with the TPM is clear and simple, and the underlying TPM certificate trust is extended to the virtual machine. But the disadvantage is that it violates the TCG specification, that is, the EK certificate cannot be used for signature.

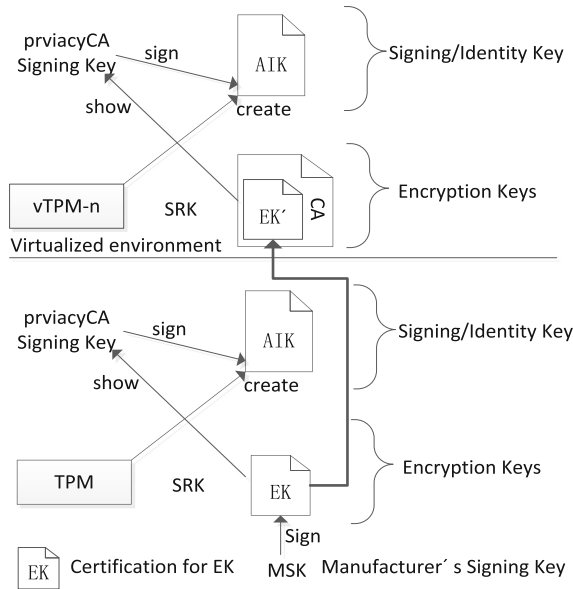
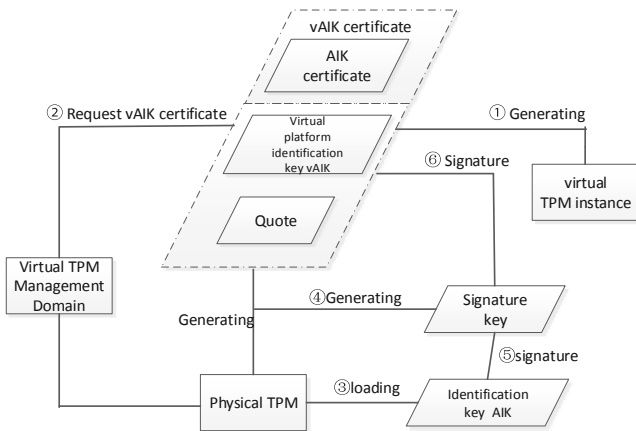


Fig. 32. vTPM vEK to hTPM EK binding



In addition, in order to improve the first four schemes, the literature [53] proposed the “vTPM vAIKto hTPM SK Binding” scheme, which introduces the signature key SK as an intermediary to implement the indirect signature of the AIK on vAIK, so that the AIK no longer sign TPM external data, which can better meet the TCG specification. And generating vAIK doesn’t depend on PrivacyCA, which reduces the burden of PrivacyCA, as shown in Fig. 33. However, this solution not only increases the complexity of generating vAIK certificates, but also doesn’t solve the problem that AIK failure will cause vAIK to failure. And refactoring vAIK needs to regenerate SK, which brings the new performance burden. In addition, each vAIK certificate corresponds to one SK, which generates a large amount of key redundancy.



**Fig. 33.** vTPM vAIKto hTPM SK binding

Literature [57] combined with the new features of TPM 2.0, proposed the “hTPM EPS Product vEK” scheme, as shown in Fig. 34. In this scheme, the vTPM’s identity certificate vAIK is generated by the EPS derived vEK to the Privacy CA verification. The literature believes that based on the mapping relationship between vEK and EPS, it can directly identify the real physical identity in the virtual machine and establish a trust chain from the physical platform to the virtual platform. But it is actually not working. EPS is only the basic key seed of Endorsement Key. Using KDF algorithm to generate Endorsement Key, which is simple and easy, but it only generates the key pair of vEK, and there is no problem of trust extension and delivery. The outside world cannot derive from the public key of the key pair that the public key is generated by the EPS of the underlying TPM. It is still necessary to pass the public key of the vEK generated by the EPS, the signature information of the underlying virtualization platform, and the vTPM information to the Privacy CA to generate the vEK certificate, and then generate the vAIK by using the vEK certificate from the Privacy CA to implement the extension of the vTPM certificate trust chain.

It can be seen from the above analysis that no matter what method is adopted to extend the certificate trust relationship of the underlying physical TPM to the vTPM

certificate is not perfect, there is a violation of the TCG specification, or increasing the key redundancy, or increasing the performance burden of the Privacy CA. Some schemes can't even extend the certificate trust. These will reduce the convenience and trust of the user.

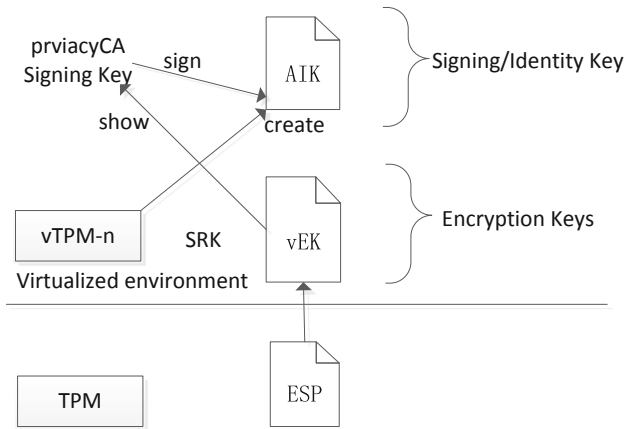


Fig. 34. hTPM EPS product vEK

#### 4.4 vTPM Migration

vTPM migration refers to migrating vTPM from the source virtual platform to the destination virtual platform. The vTPM migration can be classified according to different classification methods. According to whether the source virtual platform and the destination virtual platform are in the same cloud, it can be divided into vTPM intra-cloud migration and vTPM inter-cloud migration; according to whether the source virtual platform and the destination virtual platform are the same, it can be divided into vTPM homogeneous migration and vTPM heterogeneous migration; according to the migration mode, it can be divided into static migration and dynamic migration; at present, the most distinguishing methods used in the study are static migration and dynamic migration. Therefore, this section will be explained in terms of vTPM static migration and dynamic migration.

**Static Migration of vTPM.** The so-called vTPM static migration is to lock the vTPM, stop the vTPM service, and then migrate the vTPM related status data. The vast majority of current research is mainly to design vTPM static migration protocol, including vTPM single migration protocol, and VM and vTPM bundle migration protocol in the existing research results. Regardless of the vTPM single migration protocol or the VM and vTPM bundle migration protocol, the basic steps of the protocol are nearly the same. The only difference is that the vTPM is locked while the VM is locked, and the vTPM related state data is migrated while the VM related state data is migrated. Therefore, in this section, we all regard as the same type of research for vTPM individual migration and VM-vTPM bundle migration.

In 2006, the literature [22] first designed the static migration protocol of vTPM. In the migration protocol, transmitting the source vTPM instance state data to the destination platform securely by using session key encryption. The destination platform restores the vTPM instance and deletes the source vTPM instance. As shown in Fig. 35. First, the platform migration control process engine creates a vTPM instance with an empty state on the destination platform, generates a unique identifier Nonce associated with the instance, and passes the encrypted Nonce to the source platform. Nonce’s goal is to prevent replay attacks and the uncertainty of migration destinations, and Nonce is effective throughout the migration process. The source platform then locks the vTPM that needs to be migrated and associates it with the Nonce passed from the destination platform. Third, the source platform collects state data of the source vTPM, including: NVRAM, key session authorization and transmission callback status, authorization data, counters, and other related permanent flag bits and data. In order to ensure the integrity and confidentiality of the state data during the migration process, the source platform serializes the state data and signs it with an asymmetric key, and then generates a symmetric key pair to encrypt the serialized state data and the signature value to generate the migration data. The symmetric key is encrypted and protected by a storage key of the parent vTPM of the source vTPM. Fourth, the source platform deletes the source vTPM instance and delivers the migrated data to the destination platform. Finally, the destination platform migrates and decrypts the state data of the source vTPM, performs integrity verification and Nonce verification, and finally restores the target vTPM instance. It is worth noting that in order to the destination platform to be able to decrypt the migration data of the source vTPM, the parent vTPM storage key of the source platform must be migrated to the destination platform.

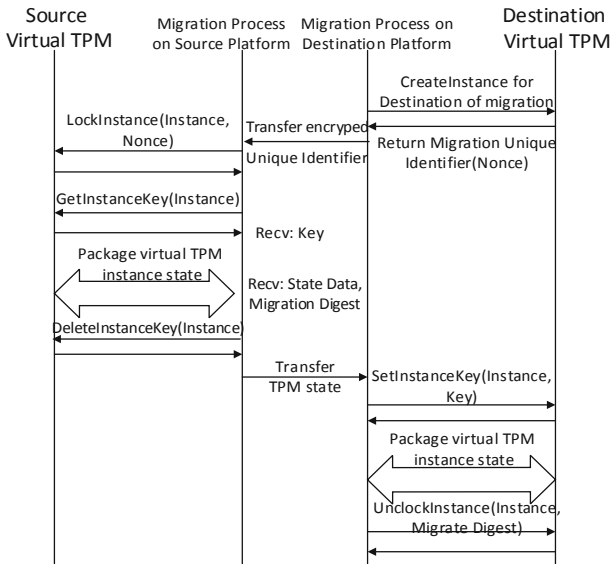


Fig. 35. Static migration protocol of vTPM

In 2007, the literature [52] analyzed the vTPM migration protocol in literature [22], and pointed out three shortcomings in the design of the vTPM migration protocol. First, the literature [22] uses the symmetric key encryption source vTPM state data, and uses the storage key of the source platform parent vTPM to protect the symmetric key. However, the protocol doesn't describe the migration of the storage key; secondly, the protocol is not well designed. For example, what key is used to encrypt the unique identifier Nonce of the destination platform vTPM, which is not stated in the protocol; finally, whether it is vTPM migration or key migration, it belongs to the TCG-IWG (Infrastructure Working Group: IWG) working group. The scope of work of the DMTF (Data Migration Task Force: DMTF) task group, but the design of the protocol is nearly not related to the work of the task group.

In 2008, the literature [40] improved the vTPM migration protocol in literature [22]. The biggest difference from the literature [22] is that the protection of the session key no longer by using the storage key of the parent vTPM. After the source migration control process engine initiated the migration and the destination migration control process engine created a new vTPM instance, the source vTPM requires to establish a trusted channel between the source platform and the destination platform. The session key is negotiated through the trusted channel. The subsequent process is basically the same as the literature [22], as shown in Fig. 36.

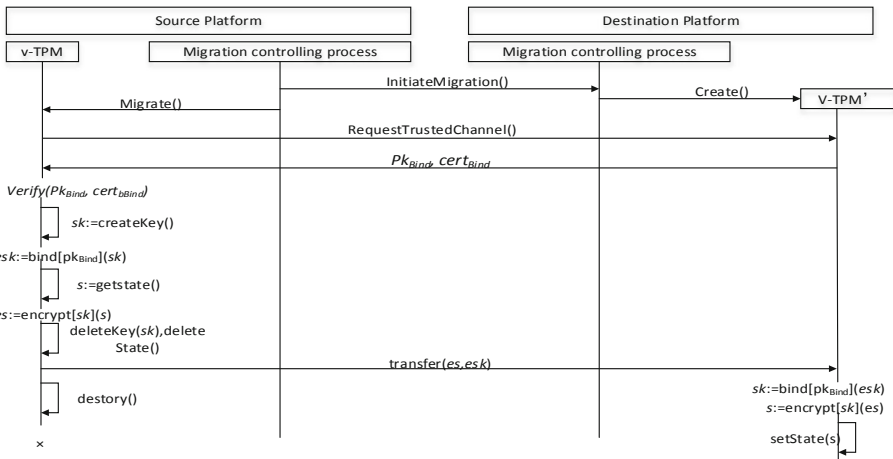
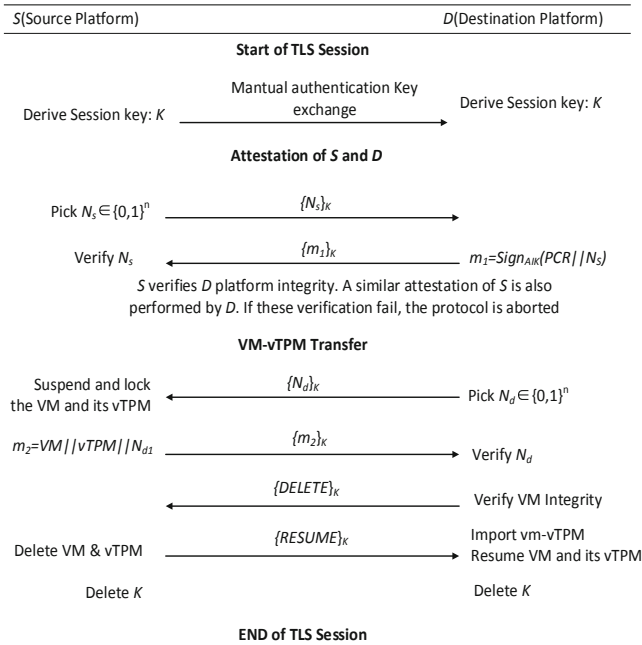


Fig. 36. Static migration protocol of vTPM based on trusted channel

In 2011, the literature [54] again improved the static migration protocol proposed in literature [22], as shown in Fig. 37. This literature proposed a 4-stage VM-vTPM migration process. In the first stage, establishing a TLS session; in the second stage, proving the source platform and the destination platform; in the third stage, migrating VM-vTPM state data, and starting the VM and vTPM of destination platform, deleting the VM and vTPM of source platform; in the fourth stage, ending the TLS session.

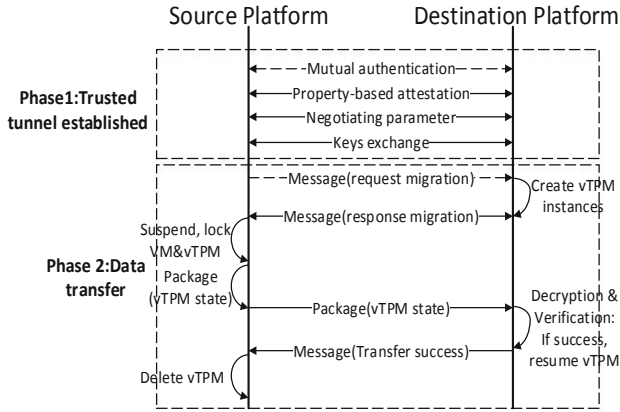
Compared with the literature [40], the source and destination credibility certificates have been added.

In 2012, the domestic literature [55] improved the literature [22], [40] and [54]. This literature proposed a two-stage vTPM migration. In the first stage, establishing trusted channel, and in the second stage, migrating the data. In the establishing trusted channel satge, requiring mutual authentication, attribute-based identity verification, parameter negotiation, and session key exchange. In the data migration satge, it is nearly the same as the literature [22] and [40]. As shown in Fig. 38.



**Fig. 37.** Static migration protocol of vTPM based on TLS

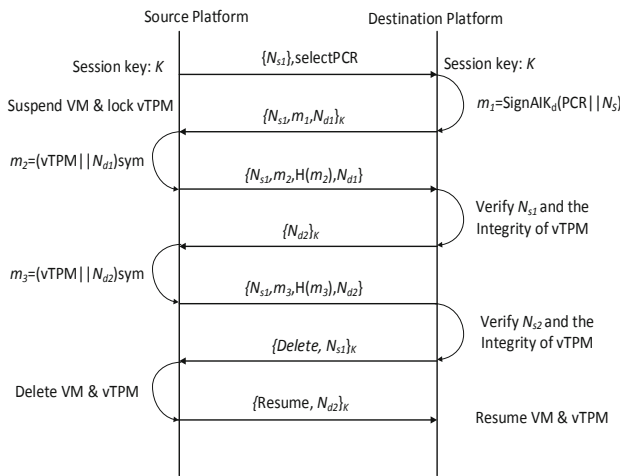
When designing the protocol, the literature also considers the requirement that the source vTPM has non-repudiation and needing to guarantee the transaction atomicity of the entire migration process, which is more perfect than the previous research results. In the same year, the domestic literature [56] also proposed the design and implementation of a secure VM-v TPM migration protocol, including 4 stages. In the first satge, establishing secure session; in the second satge, remoting proofing of the destination platform; in the third satge, transporting vTPM and VM state data; in the fourth satge, deleting the VM and its vTPM on the source platform and restore it on the destination platform. It is nearly consistent with the previous research results.



**Fig. 38.** vTPM static migration protocol for high security requirements

In 2015, the domestic literature [57] redesigned the vTPM migration protocol based on TPM 2.0. In this literature, dividing vTPM into two parts, one is the key migration of vTPM; the other is the state data migration of vTPM. In order to solve the migration timing problem of VM-vTPM, proposing a VM-vTPM instance pair migration protocol, which is divided into four stages: two-party identity authentication stage, two-party remote certification stage, data transmission stage and subsequent processing stage. The protocol process is shown in Fig. 39. And theoretical proof was carried out in this literature. However, this literature is too crude for the description of vTPM key migration.

In addition, the migration of TPM is also studied in literature [24] and [47]. But these two literatures are mainly aimed at the migration of physical TPM, not the migration of vTPM. Therefore, it is not analyzed here.



**Fig. 39.** VM-vTPM static migration protocol

In summary, the vTPM static migration protocol has achieved more research results and is maturing. However, most of the results transfer the vTPM key structure as one of the vTPM status data, which is unreasonable, and vTPM key migration must comply with the TCG specification. In addition, for the vTPM static migration, there are many research results in the academic, but the engineering achievements related to the industry and the open source community are still few.

**Dynamic Migration of vTPM.** The so-called vTPM dynamic migration, that is, the migration without stopping the vTPM service. At present, there are many dynamic migration results for VMs, including pre-copy, post-copy, checkpoint recovery, and log playback technologies. The vTPM dynamic migration is also a memory copy technology, which can learn from the existing research results of VM.

In 2015, the literature [58] designed the VM-vTPM dynamic migration protocol. As shown in Fig. 40. The VM-vTPM dynamic migration protocol is divided into two stages. The first stage is the integrity verification stage, and the second stage is the data

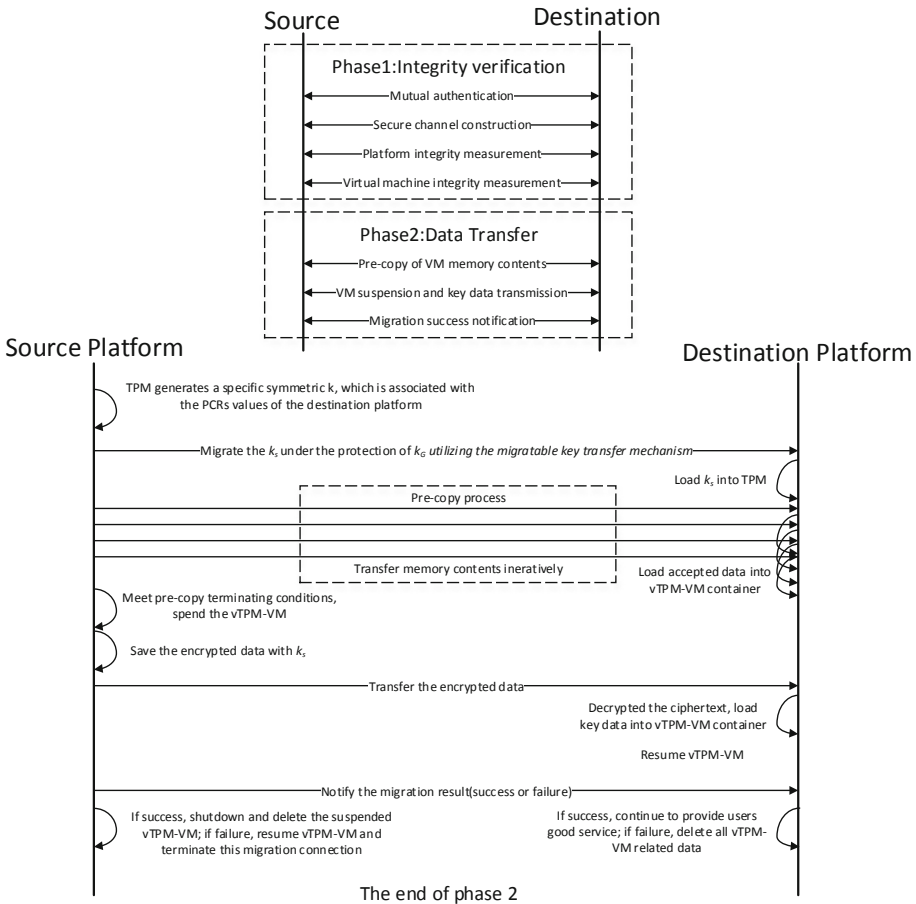


Fig. 40. VM-vTPM live migration protocol

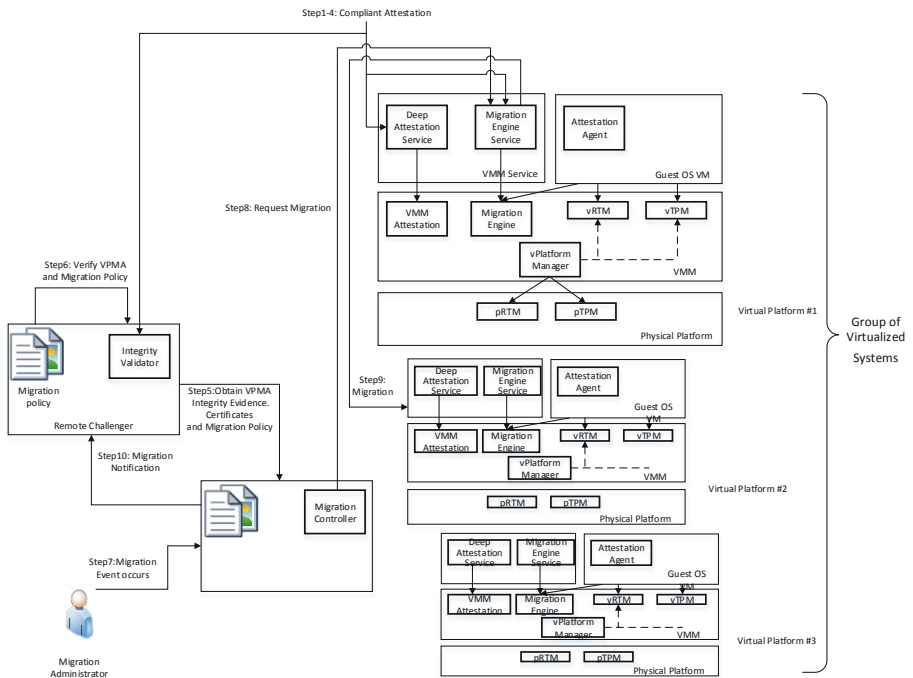
migration satge. In the first satge, the mutual authentication of the source platform and the destination platform, the construction of the secure channel, the integrity measurement of the platform, and the integrity measurement of the virtual machine are included. In the second satge, the migration of the VM-vTPM and the VM-vTPM lock using the pre-copy method, key data migration and migration success notification are included.

In addition, the literature [59] has done similar work on the KVM platform, and the device state of the vTPM is further considered in the migration process. In addition, other research results of vTPM dynamic migration have not been seen yet.

In summary, the existing results in the VM and vTPM memory copy, the discussion is not detailed enough, whether it is to copy the VM first, then copy the vTPM, or copy the vTPM first, then copy the VM, or copy the VM and vTPM at the same time, which is no explanation in the results.

**vTPM Migration System.** The so-called vTPM migration system refers to the software and hardware systems that manage, control and implement vTPM migration. It usually includes the roles, entities, architectures, and interoperability protocols that implement the migration.

In the academic, there has not been a publicly available, practical vTPM migration system. In the industry, whether it is VMware’s vSphere 6, Microsoft’s Hyper-V, or Orical’s VBox, there is no public display of the commercial vTPM migration system. In the open source community, only Xen gives a good vTPM migration. Since Xen 3.2,



**Fig. 41.** vTPM migration architecture of VPWG



vTPM can run in a separate domain of Unikernel, the separate domain is essentially a lightweight virtual machine that can be migrated using Xen's virtual machine migration system, but Xen can only migrate vTPM between similar platforms.

In 2011, TCG's VPWG (Virtual Platform Working Group: VPWG) proposed the vTPM migration system architecture and migration protocol under the trusted virtual platform in the Virtualized Trusted Platform Architecture Specification [58]. As shown in Fig. 41. The migration architecture has a total of 12 entities, including Migrate Controller, Integrity Validator, Deep Attestation Service, VMM Attestation, Attestation Service, Migration Engine Service, Migration Engine, Virtual Platform Manager (vPlatform Manager), Virtual Trusted Root (vRTM), Virtual TPM (vTPM), and Physical Trusted Root (PRTM) and Physical TPM (pTPM). The roles involved include remote challengers, migration managers, and more. In the first 4 steps of the migration protocol, the migration challenger proves compatibility between the source platform and the destination platform; in the steps 5 and 6, the Integrity Validator in the migration challenger obtains VPAM integrity evidence, certificates, and migration policies from the Migration Control (Migrate Controller) and verifies them; in the step 7, once the migration manager receives the migration event, a migration command is issued to the migration engine in step 8; in step 9, the migration engine migrates the VM-vTPM that needs to be migrated. After the migration was completed, the Migration Manager notifies remote challenger that migration was successfully. Compared with the existing research results, the specification details the entities involved in the migration, which is closer to the practical system, but the migration protocol itself is less discussed.

To sum up, in the aspect of vTPM migration, the researches on vTPM static migration protocol are nearly mature, however, there are few researches on vTPM dynamic migration, especially the timing problem of memory copy during vTPM and VM bundle migration has not been solved. The practical and mature vTPM migration system has not yet appeared, and further research and development are urgently needed.

## 5 Problems and Challenges Needed to Be Solved in TPM Virtualization

At present, various TPM virtualization solutions have different advantages and disadvantages, and there are still many problems that need to be solved in terms of security, performance and compliance with TCG specification. Mainly including the following 4 aspects:

- (1) On the TPM virtualization system architecture, there is a lack of a smart adaptive TPM virtualization architecture with high security, high performance and dynamic adjustment according to user security requirements. In addition, the research on the aggregated TPM virtualization architecture is also lagging behind. Our

thinking cannot be limited to the details of the TCG specification. We should learn from the basic ideas of trusted computing and boldly innovate in technology.

- (2) In the key management of vTPM, the problems and challenges include two aspects. On the one hand, since vTPM must meet the migration requirements, there is always a dilemma between setting the key type and meeting the migration requirements. It is necessary to change the key attributes that originally satisfies the TCG specification, or introduce key redundancy, which cannot meet the consistency requirement of the key management of the vTPM and the TPM, and reduces the convenience of the user to use the vTPM. On the other hand, the trusted computing services provided by the software emulation vTPM for virtual machines, especially the cryptographic computing services, are all completed in the virtual platform's memory (not within the TPM), which is vulnerable to general security threats, privileged security threats and sharing security threats, which will seriously affect the security of user virtual machines and vTPM.
- (3) In terms of certificate trust extension, in order to extend the certificate trust relationship of the physical TPM to the vTPM certificate. No matter what way is used, there is a case where the certificate usage violates the TCG specification, or a key redundancy is introduced, and no Satisfactory solution has not been found.
- (4) In the vTPM migration solution, the current researches focus on the vTPM static migration protocol between the same virtual platforms in the cloud. However, the vTPM static migration protocol, the vTPM dynamic migration protocol, and the vTPM migration system between heterogeneous virtual platforms in the cloud require further research and development.

## 6 Conclusion

The combination of trusted computing and cloud computing technology can guarantee the credibility of the cloud computing environment in a certain extent, and promote the expansion and extension of cloud computing in a wider range of application. One of the effective ways to solve this problem is to build a virtual machine trusted environment through TPM virtualization. How to securely and efficiently virtualize TPM and improve users' trust in cloud environment is the key to the integrated development of trusted computing and cloud computing. At the same time, TPM virtualization is not only a technical issue, it also involves many aspects such as standardization, supervision mode, application mode, etc. Therefore, it is not enough to explore TPM virtualization from a technical perspective, and it requires the joint efforts of the information security academic community, industry and relevant government departments to achieve this.

**Acknowledgements.** This work was supported by the National Natural Science Foundation of China (Grant No. 61373162), and the Sichuan Provincial Key Laboratory Project (Grant No. KJ201402).

## References

1. Trusted Computing Group. <https://www.trustedcomputing-group.org>. Accessed 08 Sept 2019
2. TPM Specification, Version 2.0/Part 1, Architecture. [https://www.Trustedcomputinggroup.org/wp-content/uploads/TPM-Rev-2.0-Part1-Architecture-01.36\\_public-review.pdf](https://www.Trustedcomputinggroup.org/wp-content/uploads/TPM-Rev-2.0-Part1-Architecture-01.36_public-review.pdf). Accessed 08 Sept 2019
3. TPM Specification, Version 2.0/ Part 2, Structures. [https://www.Trustedcomputinggroup.org/wp-content/uploads/TPM-Rev-2.0-Part2-Structures-01.36\\_public-review.pdf](https://www.Trustedcomputinggroup.org/wp-content/uploads/TPM-Rev-2.0-Part2-Structures-01.36_public-review.pdf). Accessed 08 Sept 2019
4. Specification, Version 2.0/ Part 3, Commands. [https://www.trustedcomputinggroup.org/wp-content/uploads/TPM-Rev-2.0-Part3-Commands-01.36-code\\_public-review.pdf](https://www.trustedcomputinggroup.org/wp-content/uploads/TPM-Rev-2.0-Part3-Commands-01.36-code_public-review.pdf). Accessed 08 Sept 2019
5. TPM Specification, Version 2.0/Part 4, Supporting Routines. [https://www.trustedcomputinggroup.org/wp-content/uploads/TPM-Rev-2.0-Part3-Commands-01.36-code\\_public-review.pdf](https://www.trustedcomputinggroup.org/wp-content/uploads/TPM-Rev-2.0-Part3-Commands-01.36-code_public-review.pdf). Accessed 08 Sept 2019
6. Changxiang, S., Huanguo, Z., Dengguo, F., et al.: Information security review. *Scientia Sinica Ser. E: Inf. Sci.* **37**(2), 129–150 (2007)
7. Changxiang, S., Huanguo, Z., Huaiming, W., et al.: Research and development of trusted computing. *Scientia Sinica Ser. E: Inf. Sci.* **40**(2), 139–166 (2010)
8. Dengguo, F., Yu, Q., Dan, W., et al.: Research on trusted computing technology. *J. Comput. Res. Dev.* **48**(8), 1332–1349 (2011)
9. State Cryptography Administration: Trusted Computing Cryptographic Support Platform Function and Interface Specification (2007)
10. Rimal, B.P., Choi, E., Lumb, I.: A taxonomy and survey of cloud computing systems. In: *Proceedings of the 2009 Fifth International Joint Conference on INC, IMS and IDC*, pp. 44–51. IEEE, Seoul (2009)
11. Yao, S., et al.: An efficient multi-objective scheduling method for data flow in cloud environment. *J. Softw.* **28**(3), 1–19 (2017)
12. Siyao, X., Weiwei, L., Zijun, W.: Virtual machine placement algorithm based on peak load characteristics. *J. Softw.* **27**(7), 1876–1887 (2016)
13. Wei, W., Zeyu, G., Wenbo, Z., et al.: A cloud computing system fault detection method based on adaptive monitoring. *Chin. J. Comput.* **39**(163), 1–15 (2016)
14. Guofeng, W., Chuanyi, L., Hezhong, P., et al.: Overview of internal threats in cloud computing models. *Chin. J. Comput.* **39**(145), 1–21 (2016)
15. Lifang, R., Wenjian, W., Xing, X.: Adaptive cloud computing service portfolio with uncertain perception. *J. Comput. Res. Dev.* **53**(12), 2867–2881 (2016)
16. Junjie, L., Fenghua, L., Qiongni, L., et al.: Optimized high-dimensional index and KNN query under MapReduce framework. *Acta Electronica Sinica* **44**(8), 1873–1880 (2016)
17. Amazon Elastic Compute Cloud (EC2). <http://aws.amazon.com/ec2/>. Accessed 08 Sept 2019
18. Google App Engine (GAE). <https://appengine.google.com/>. Accessed 08 Sept 2019
19. Microsoft Azure Services Platform. <http://www.microsoft.com/azure/>. Accessed 08 Sept 2019
20. Elastic Utility Computing Architecture for Linking Your Programs To Useful Systems (Eucalyptus). <http://www.eucalyptus.com/>. Accessed 08 Sept 2019
21. Chuang, L., Wenbo, S., Kun, M., et al.: Cloud computing security: architecture, mechanism and model evaluation. *Chin. J. Comput.* **36**(9), 1765–1784 (2013)

22. Berger, S., Cáceres, R., Goldman, K.A., et al.: vTPM: virtualizing the trusted platform module. In: Proceedings of the 15th USENIX security Symposium, pp. 305–320. ACM, Vancouver (2006)
23. 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). [https://doi.org/10.1007/978-3-540-68979-9\\_9](https://doi.org/10.1007/978-3-540-68979-9_9)
24. Stumpf, F., Eckert, C.: Enhancing trusted platform modules with hardware-based virtualization techniques. In: Proceedings of the 2008 Second International Conference on Emerging Security Information, pp. 1–9. IEEE, Cap Esterel (2008)
25. Lei, S., Deqing, Z., Hai, J.: Xen Virtualization Technology. Huazhong University of Science and Technology Press, Hangzhou (2009)
26. AlBelooshi, B., Salah, K., Martin, T., et al: Securing cryptographic keys in the IaaS cloud model. In: 8th International Conference on Utility and Cloud Computing (UCC), pp. 397–401. IEEE, Limassol (2015)
27. Zhilou, Y., Qiao, W., Weipin, Z., et al.: A cloud certificate authority architecture for virtual machines with trusted platform module. In: IEEE 7th International Symposium on Cyberspace Safety and Security (CSS), pp. 1377–1380. IEEE, New York (2015)
28. Dexian, C., Xiaobo, C., Yu, Q., et al.: TSD: a flexible root of trust for the cloud. In: IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications, pp. 119–126. IEEE, Liverpool (2012)
29. Xin, W., Zhiting, X., Yi, R.: Building trust into cloud computing using virtualization of TPM. In: Fourth International Conference on Multimedia Information Networking and Security, pp. 59–63. IEEE, Nanjing (2012)
30. Dongliang, X., Xiaolong, W., Yunwei, G., et al.: TrustVP: construction and evolution of trusted chain on virtualization computing platform. In: Eighth International Conference on Computational Intelligence and Security (CIS), pp. 623–630. IEEE, Guangzhou (2012)
31. Microsoft MVP. <http://anilerduran.com/vtpm-in-windows-server-2016-hyper-v/>. Accessed 08 Sept 2019
32. Oricale. <https://www.virtualbox.org/>. Accessed 08 Sept 2019
33. VMware. <http://www.vmware.com/>. Accessed 08 Sept 2019
34. Xen project. <http://www.xenproject.org/>. Accessed 08 Sept 2019
35. KVM project. <http://www.linux-kvm.org/>. Accessed 08 Sept 2019
36. Scarlata, V., Rozas, C., Wiseman, M., et al.: TPM virtualization: building a general framework. In: Pohlmann, N., Reimer, H. (eds.) Trusted Computing, pp. 43–56. Springer, Heidelberg (2007). [https://doi.org/10.1007/978-3-8348-9452-6\\_4](https://doi.org/10.1007/978-3-8348-9452-6_4)
37. Anderson, M.J., Moffie, M., Dalton, C.I.: Towards trustworthy virtualization environments: Xen library OS security service infrastructure. Hewlett-Packard Lab. **2007**(1), 43–51 (2007)
38. Murray, G., Milos, G., Hand, S.: Improving Xen security through disaggregation. In: VEE 08: Proceedings of the Fourth ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, pp. 151–160. ACM, Seattle (2008)
39. David, P., Serdar, C., Chris, D., et al.: TPM virtualisation architecture document. Open Trusted Computing (2009)
40. 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). [https://doi.org/10.1007/978-3-540-85886-7\\_1](https://doi.org/10.1007/978-3-540-85886-7_1)
41. Xin, J., Lina, W., Rongwei, Y., et al.: Administrative domain: security enhancement for virtual TPM. In: International Conference on Multimedia Information Networking and Security, pp. 767–771. IEEE, Nanjing (2010)
42. Weiqi, D.: Research on key issues of trusted construction of cloud computing execution environment. Huazhong University of Science and Technology (2015)

43. Bade, S.A., Betz, L.N., Kegel, A.G., et al.: Method and system for virtualization of trusted platform modules. US Patent 7 380 119, May, 2008
44. Feller, T., Malipatlolla, S., Kasper, M., et al.: dcTPM: a generic architecture for dynamic context management. In: International Conference on Reconfigurable Computing and FPGAs, pp. 211–216. IEEE, Cancun (2011)
45. Smith, N.M.: Method and apparatus for virtualization of a multi-context hardware trusted platform module (TPM). US Patent 2009/0 055 641 A1, February 2009
46. Jayaram Masti, R., Marforio, C., Capkun, S.: An architecture for concurrent execution of secure environments in clouds. In: Proceedings of the 2013 ACM workshop on Cloud computing security workshop, pp. 11–22. ACM, Berlin (2013)
47. Yap, J.Y., Tomlinson, A.: Para-virtualizing the trusted platform module: an enterprise framework based on version 2.0 specification. In: Bloem, R., Lipp, P. (eds.) INTRUST 2013. LNCS, vol. 8292, pp. 1–16. Springer, Cham (2013). [https://doi.org/10.1007/978-3-319-03491-1\\_1](https://doi.org/10.1007/978-3-319-03491-1_1)
48. Pci-sig-single root iov. [http://www.pcisig.com/specifications/iov/Single\\_root/](http://www.pcisig.com/specifications/iov/Single_root/). Accessed 08 Sept 2019
49. Mingda, L., Longyu, M.: A virtual environment security isolation model based on SR-IOV technology. In: Proceedings of the 31st National Computer Security Academic Exchange Conference, pp. 84–89. CNKI, Xiamen (2016)
50. Xinlong, L., Rui, J., Huafeng, K.: Secure and reliable VM-vTPM migration in private cloud. In: 2nd International Symposium on Instrumentation and Measurement, Sensor Network and Automation (IMSNA), pp. 510–514. IEEE, Toronto (2013)
51. Yongjiao, Y., Fei, Y., Junpeng, M., et al.: Ng-vTPM: a new generation of TPM virtualization framework design. J. Wuhan Univ. (Nat. Sci. Ed.) **61**(2), 103–111 (2015)
52. Goyette, R.: A review of vTPM: virtualizing the trusted platform module. In: Network Security and Cryptography Symposium, pp. 1–17 (2007)
53. Lina, W., Hanjun, G., Rongwei, Y., et al.: Research on the construction method of trusted virtual execution environment based on trust extension. J. Commun. **32**(9), 1–8 (2011)
54. Danev, B.: Enabling secure VM-vTPM migration in private clouds. In: ACSAC 2011 Proceedings of the 27th Annual Computer Security Applications Conference, pp. 187–196. ACM, Orlando (2011)
55. Xin, W., XinFang, Z., Liang, C., et al.: An improved vTPM migration protocol based trusted channel. In: International Conference on Systems and Informatics (ICSAI), pp. 870–875. IEEE, Yantai (2012)
56. Yinchao, Y., Zai, L., Zuoning, C.: Design and implementation of a secure VM-v TPM migration protocol. Appl. Electron. Tech. **38**(4), 130–133 (2012)
57. Armbrust, M., Fox, A., Grith, R., et al.: A view of cloud computing. Commun. ACM **53**(4), 50–58 (2010)
58. Peiru, F., Bo, Z., Yuan, S., et al.: An improved vTPM-VM live migration protocol. Wuhan University J. Nat. Sci. **20**(6), 512–520 (2015)
59. Yuqing, H., Bo, Z., Jue, X., et al.: A KVM-based v TPM virtual machine dynamic migration scheme. J. Shandong Univ. (Nat. Sci. Ed.) **52**(6), 69–75 (2017)
60. Virtual platform working group (VPWG) on virtualized trusted platform architecture specification. [https://www.trusted-computinggroup.org/wp-content/uploads/TCG\\_VPWG\\_Architecture\\_V1-0\\_R0-26\\_FINAL.pdf](https://www.trusted-computinggroup.org/wp-content/uploads/TCG_VPWG_Architecture_V1-0_R0-26_FINAL.pdf). Accessed 08 Sept 2019
61. Yan, D., Huaiming, W., Peichang, S., et al.: Trusted cloud service. Chin. J. Comput. **38**(5), 133–149 (2015)