分类号：（按中国图书分类法）　　单位代码：10636

密　级：（注明密级与保密期限）　　学　　号：20151304001

四川師範大學

硕士学位论文

中文论文题目：具有瀑布特征的可信虚拟平台信任链模型及其分析方法

英文论文题目：**Research on the Trust Chain Model with Waterfall Characteristic and its Analysis Methods Based on the Trusted Virtualization Platform**

论文作者：**

指导教师：**

专业名称：信息安全

研究方向：可信计算

所在学院：计算机科学学院

论文提交日期：　　年　　月　　日

论文答辩日期：　　年　　月　　日

四川师范大学学位论文独创性声明

本人声明：所呈交学位论文_____，是本人在导师_____指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不含任何其他个人或集体已经发表或撰写过的作品或成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本声明的法律结果由本人承担。

本人承诺：已提交的学位论文电子版与论文纸本的内容一致。如因不符而引起的学术声誉上的损失由本人自负。

学位论文作者：　　签字日期：　　年　　月　　日

四川师范大学学位论文版权使用授权书

本人同意所撰写学位论文的使用授权遵照学校的管理规定：

学校作为申请学位的条件之一，学位论文著作权拥有者须授权所在大学拥有学位论文的部分使用权，即：1）已获学位的研究生必须按学校规定提交印刷版和电子版学位论文，可以将学位论文的全部或部分内容编入有关数据库供检索；2）为教学、科研和学术交流目的，学校可以将公开的学位论文或解密后的学位论文作为资料在图书馆、资料室等场所或在有关网络上供阅读、浏览。

本人授权万方数据电子出版社将本学位论文收录到《中国学位论文全文数据库》，并通过网络向社会公众提供信息服务。同意按相关规定享受相关权益。

（保密的学位论文在解密后适用本授权书）

学位论文作者签名：　　导师签名：

签字日期：　　年　　月　　日　　签字日期：　　年　　月　　日

具有瀑布特征的可信虚拟平台信任链模型及其分析方法

信息安全专业

研究生 ** 指导教师 **

摘要 将虚拟化技术与可信计算相结合构建的可信虚拟平台及其信任链模型是目前的一个研究热点。目前大部分的研究成果是采用在虚拟平台上扩展传统信任链的构建方法，不仅模型过粗且逻辑不完全合理，而且还存在底层虚拟化平台和顶层用户虚拟机两条分离的信任链问题。并且目前的无干扰理论形式化方法只定义了动作所属域，没有针对云环境下系统动作的主体等进行定义，不能完全适用于云计算环境下的信任链传递模型。

为此，本文首先提出了一种具有瀑布特征的可信虚拟平台架构，该可信虚拟平台在层次上添加了可信衔接点层次，主要由虚拟机构建模块、虚拟可信平台模块构建模块、虚拟机和其虚拟可信平台模块绑定模块组成。当可信虚拟平台启动时，不仅可以以静态度量方式参与底层虚拟化平台的启动，也可以和虚拟可信模块共同作为虚拟机启动动态度量的虚拟可信根。在该可信虚拟平台中，可信衔接点具有承上启下的瀑布特征，把底层虚拟化平台的启动输出作为了虚拟机启动的度量输入，类似于软件设计中的瀑布特征。然后基于上述可信虚拟平台架构构建了信任链模型，该模型以硬件 TPM 为起点，在底层虚拟化平台和顶层用户虚拟机信任链之间加入可信衔接点。当信任链从底层虚拟化平台传递到可信衔接点时，由可信衔接点负责对用户虚拟机的 vTPM 进行度量，之后将控制权交给 vTPM，由 vTPM 负责对用户虚拟机启动的组件及应用进行度量。该模型中可信衔接点具有承上启下的瀑布特征，能满足虚拟化环境的层次性和动态性特征，保证了整个可信虚拟平台的可信性。基于 Xen 的信任链构建的实验结果表明本信任链传递方法可以保证可信虚拟化环境在整个运行过程是安全可信的。

然后，本文基于安全系统逻辑的形式化分析也证明了该信任链模型的安全性，分别对底层物理平台、可信衔接点等层次的本地验证以及远程证明的形式分析，证明了本文信任链模型的可靠性和安全性。最后，本文提出了一种扩展无干扰的信任链

形式化方法，针对目前的非传递无干扰理论均没有考虑到云计算运行中时的安全域、动作所属主体以及动作对安全域和系统状态的影响进行详细的说明，对无干扰理论在安全域动作所属主体等进行详细的扩展，并定义了云计算环境下的非传递无干扰安全定理，结合上述可信虚拟平台和信任链对无干扰进行了实例验证。

关键词：可信计算　云计算　信任链 无干扰 安全系统逻辑

Research on the Trust Chain Model with Waterfall Characteristic and its Analysis Methods Based on the Trusted Virtualization Platform

Major: Information Security

Postgraduate Student:** **Tutor**: **

ABSTRACT The trusted virtual platform which is constructed by the combination of virtualization technology and trusted computing and its trust chain have become one of the key focuses in the researched fields. But at present, most of the researching achievements construct the trust chain by extending the conventional trust chain model. As a result, the model is not precise and the logic is not completely reasonable. Moreover, two separated trust chains have existed, one is starting with the underlying virtual platform, and the other is the starting with the top-level user virtual machine.And the current no interference theory formalism method defines that the action only belongs to the domain. But something important belong to the cloud environment are not defined, such as the system action subject. So it cannot be fully applicable to the trust chain model under the cloud computing environment.

In order to solve this problem, this paper proposes a trusted virtual platform which has the characteristic of the waterfall. The trusted virtual platform add a trusted joined point, which is mainly built by virtual mechanism module, virtual trusted platform module build module, the virtual machine and its virtual trusted platform module of binding modules.When trusted virtual platform to start, not only can participated in the underlying virtualization platform in static measure, but also can be together as a virtual machine and virtual trusted module launch dynamic measurement of virtual trusted root. In the trusted virtual platform, the trusted connection point has a waterfall feature, which forms a connecting link between the preceding and the underlying virtualization platform to start the output as the measurement of the virtual machine to start input, which like the waterfall in the software design features.Then this paper build the trust chain model based on the trusted virtual platform. This model starts with the physical

TPM, and increases a Trusted-Joint Point called TJP between the chain of the underlying virtual platform and the chain of the top-level user VM. The TJP is in charge of the measurement of vTPM for VM after the trusted chain is transmitted from the underlying virtual platform to the TJP, and then the vTPM gets the control, and is in charge of the measurement of the related components and applications of the top-level user VM in the starting process. The TJP which has the waterfall characteristic between the underlying virtual platform and the top-level user VM can be viewed as a connecting link, and it can satisfy with the hierarchical and dynamic characteristics of the virtual platform, moreover guarantee the trust of the whole virtual platform. This trusted virtual platform is implemented on the Xen platform, the experimental results based on Xen show that this trust chain transfer method can guarantee that the trusted virtualization environment is safe and reliable in the whole operation process.

Then, this paper proves the safety of trust chain model based on the formal analysis of security system logic,which mainly contain underlying physical platform and credible connecting point, and level of validation and remote proof in the form of local analysis. All of them can prove the reliability and security of the trust chain model. Finally this paper proposes a trust chain formalism method, based on the extension of no interference theory. According to the present no interference theory don’t consider the security domain of the main body, action and the action of security domain and the influence of the system state to carry on the detailed instructions when the cloud environment in running. Subject to no interference theory in security domain action belongs to such as extension in detail, and defined the cloud computing environment is not passed no interference security theorem, combined with the virtual platform and credible trust chain for instance verification without interference.

Key Words: trusted computing; cloud computing; trusted chain; no interference theory; logic of safety system

插图和附表清单

目录

1 绪论

1.1 研究背景及意义

根据NIST定义[1]，云计算是可以使用户按照使用量付费并且获得高效的、快捷的网络服务的新型资源共享模式，其主要目的是在提高网络、服务器、硬盘存储、软件等共享资源利用率的同时，使云租户不再关注硬件资源的管理和维护，云租户只需要在硬件资源上投入很少的管理维护工作就可以得到很高的资源回报。云计算的高效的资源处理能力也带动了大数据、人工智能等相关领域的发展。目前的云计算提供商，比如国外的Intel、IBM、微软，以及国内的腾讯、阿里巴巴都拥有非常成熟的云计算技术和应用服务提供技术。云计算的快速发展同时也给云计算带来了除传统信息安全、网络安全之外的安全问题[2][3]，其中，如何向云租户证明云计算底层平台的安全性、虚拟机(Virtual Machine，VM)的安全性是一个非常重要的问题[4~8]。而可信计算是保障信息系统安全最为重要的技术手段之一[9][10]，它通过提供数据保护、身份认证、远程证明以及完整性度量等特性提高包括底层物理资源、应用软件等在内的计算平台的可信性和可靠性[11]。因此，将可信计算技术应用在提高云计算环境的安全性是工业界和产业界必须重视的地方。

根据中国信息通信研究院2017年7月发布的《云计算关键行业应用报告》[12]，近几年，云计算的发展十分迅速，并产生了以云主机为主要服务的云计算市场。其中，2016年，全球云计算市场的经济整体规模已经达到了654.6亿美元，较2015年增长25.4%，并且预计在2020年云计算市场规模将达到1453.3亿美元，平均每年增长率为21.7%。2016年我国的云计算市场达到514.9亿元，增速为35.9%，处于全球国家云计算发展的前列。2017年，工信部在官方网站发布通知《云计算发展三年行动计划（2017－2019年）》[13]中提到，我国云计算的发展目标到2019年我国的云计算规模预计达到4300亿元，该项内容为我国云计算和云计算安全技术创新和产业发展指明了方向，提供了政策保障和法律依托。并且，根据著名安全公司McAfee发布的“2017年全球云计算安全报告”[14]显示，在2016下半年到2017年，在参与调查的公司中，为把更多的精力投入到提高客户体验中，IT预算中有超过80%的预算被用来使用云服务及其解决方案。但是，仅有23%的企业完全信任云计算提供商。

目前的大部分云租户不能完全信任云提供商提供的云计算服务的原因，主要是由于云租户在使用云提供商提供的虚拟机时，并不能确认云计算平台上的物理主机是云提供商按照各自操作系统官方文件进行启动的，以及租户请求的虚拟机

是按照预期的配置和要求进行启动的。因为云计算环境下的虚拟机存在着包括传统信息系统安全以及新型网络安全等威胁，比如：虚拟机恶意代码攻击、虚拟机逃逸等，这些都会导致虚拟机在重新启动时的组件被篡改，在云租户对虚拟机进行重新启动时，可能无法判断虚拟机的操作系统、数据是否被篡改。而可信虚拟平台的构建可以利用可信平台模块（Trusted Computing Platform, TPM）中的可信度量、可信报告等技术向用户发送关于云计算平台的可信度量结果，并且证明自身的安全性。一方面，云计算架构中独特的虚拟机监视器（Virtual Machine Monitor, VMM）的隔离机制、安全机制、监控机制，为在物理服务器操作系统和应用服务建立可信计算环境提供了保障，也可以有效的防止外界对可信计算环境的侵扰和破坏；另一方面，可信计算技术为虚拟化技术中的虚拟机提供了完整性度量和信任链扩展的思路，为虚拟机对云租户提供云服务提供了保障。

信任链技术[15]是可信计算的关键技术，针对可信计算技术与云计算技术结合的可信虚拟平台的信任链构建更是十分有必要的。利用虚拟可信平台模块（Virtualization of Trusted Platform Module, vTPM）[16][25]在云计算环境中构建安全可靠的可信虚拟平台，并且利用可信计算中的关键技术——信任链技术对整个云计算平台进行信任链构建，建立从云计算平台底层物理服务器到提供服务的可信虚拟机的信任链，可以给目前的云计算安全问题提供一个新的解决思路，为构建安全的云计算环境提供更好的安全保障。

并且，针对信任链传递模型的形式化分析方法一直是可信计算研究领域中重点关注的问题，从安全系统的整体构建角度上来看，一个安全可靠的系统不仅仅在功能上是安全的，也必须利用形式化分析中的数学模型描述其组件和安全属性，并用语义明确的定义和安全定理证明其安全性。在针对信任链形式化分析方法中的众多理论中，无干扰理论一直是备受研究者关注的形式化分析方法。无干扰理论基于有限状态机描述系统的行为和动作，并从传递无干扰和非传递无干扰给出系统的安全定理。但是目前的针对信任链传递模型的形式化分析方法还不能完全适用于云计算环境下的可信虚拟平台，其缺点主要表现在其没有考虑到云计算运行中时的安全域、动作所属主体以及动作对安全域和系统状态的影响进行详细的说明，比如，系统的发出动作在整体上是属于某一域，但是实际上也存在发出该

动作的主体。

为此，本文对可信虚拟平台架构及其之上的信任链模型进行研究，并利用已有的形式化分析方法安全系统逻辑方法和本文提出的扩展无干扰理论分别对信任链进行形式化分析。首先提出了带有可信衔接点的可信虚拟平台架构 TVP-QT，其中可信衔接点由三部分组件构成，虚拟机构建模块，虚拟可信模块构建模块，VM 和 vTPM 绑定模块，该可信衔接点充分连接了云计算平台和虚拟机，保障信任链构建的安全性和完整性。然后构建了 TVP-QT 的信任链模型，该模型以硬件 TPM 为起点，在底层虚拟化平台和顶层用户虚拟机信任链之间加入可信衔接点。当信任链从底层虚拟化平台传递到可信衔接点时，由可信衔接点负责对用户虚拟机的 vTPM 进行度量，之后将控制权交给 vTPM，由 vTPM 负责对用户虚拟机启动的组件及应用进行度量。该模型中可信衔接点具有承上启下的瀑布特征，能满足虚拟化环境的层次性和动态性特征，保证了整个可信虚拟平台的可信性。最后，本文对目前的无干扰理论进行扩展，详细的定义了动作的主体，动作对系统安全域的影响等，并给出了非传递无干扰的云计算环境信任链安全判定定理。不仅在实验上证明了 TVP-QT 模型的安全性和可靠性，也在理论上证明了 TVP-QT 模型的安全性和可靠性。

1.2 国内外研究现状

可信计算技术与虚拟化技术的结合的可信虚拟平台（Trusted Virtualization Platform, TVP）一直以来都受到国内外学者的广泛关注。Intel的Stefan Berger[16]等人最先提出vTPM的概念，随后产生了很多关于TVP及其信任链构建的研究成果，其中，开源的虚拟机架构Xen[17]是最早支持vTPM的VMM，为学术界和产业界提供非常便捷的实验平台；2015年，云计算公司EMC[18]宣布在VMware vSphere 6.0中支持vTPM；2016年，微软[19]宣布在win10中加入可信计算，并提供Hyper-V技术。总之，学术界和产业界都已开始重视可信计算技术与虚拟化技术的结合。为更好的对本文研究内容的国内外研究现状进行阐释，本文将从可信虚拟平台架构、可信虚拟平台信任链模型、针对信任链的形式化分析方法三部分进行描述。

1.2.1 可信虚拟平台架构

国外研究者针对可信计算解决虚拟系统安全问题的研究比较早，在2000年后都被研究者用来解决虚拟系统平台安全的问题，这些平台包括Terra[20]，Perseus[21]等，这些平台的主要思想是把底层计算平台分为两部分，可信区域和不可信区域，

其中可信区域上运行着高安全性需求的虚拟机，而存在着安全威胁的虚拟机被放在不可信区域。这些方案为可信虚拟平台的研究提供很好的理论基础。

TVP的概念首先由 Stefan Berger[16]等人提出，随后文献[22~25]等学者针对如何构建具体应用场景的TVP功能应用以及抽象和统一的TVP概念取得了很多较好的研究成果，并且达成了一些基本的共识。目前，研究此方面的学者绝大多数都认为，在物理上，TVP是一个可以支持可信计算虚拟化技术的物理主机，并且其与一般的带有TPM的可信主机的主要区别在于，一是拥有在物理硬件可信平台模块构建起来的虚拟可信信任根；二是可以并发的为可信虚拟平台之上的多个用户虚拟机（Virtual Machine, VM）提供可信虚拟信任环境。这种传统的TVP运行架构如图1.1所示。并且TVP架构可以被分为4个具有不同功能的层次。第一层为硬件信任根TPM，为整个可信虚拟平台提供物理保障。第二层主要包括虚拟机监视器VMM，以及建立在VMM上，包括内核和VMM管理工具的管理域，并且被当做TVP的可信计算基来启动。第三层是作为虚拟机启动的虚拟可信根（Virtual Root of Trust, vRT），并且虚拟可信根的加载方式以两种不同的实现方案来实现，一种是当做传统信任链的一部分进行静态加载，另一部分是利用动态度量机制（Dynamic Root of Trusted Measurement, DRTM）实现动态加载。所以虚拟可信根可以作为TCB的一部分，也可以当做TVP上的单独进程。第四层是与用户关系最为密切的用户虚拟机。

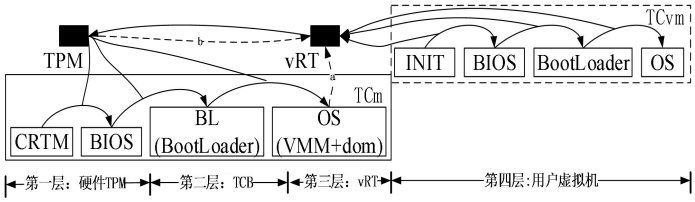

图 1.1 TVP 基本运行架构

根据文献[24]的vRT等概念，HP、IBM等研究机构分别提出并构建了相应的TVP[30][31]，其TVP架构可根据不同应用需求建立用户可定制的TVP，在很大程度上推动了TVP的发展。随后，Krautheim[24]等学者基于云计算环境建立了TVP，使其可以保护云计算环境下的虚拟机运行，以及保护虚拟机运行时上层服务软件的完整性、安全性。Zhang Lei[25]等提出一种具有可信域层次的TVP，通过可信云平台和可信虚拟机进行分离的TVP构建机制，并实现了对可信云平台以及可信虚拟机的安全保障。

国内的研究中，王丽娜[26]给出了基于信任链扩展的TVP架构，常德显[27]等根据TVP的功能层次给出了包括虚拟机和虚拟可信根的TVP定义，并细分为VMM、Dom0、TPM、vRT等组件。文献[28~33]也建立了类似以上的可信虚拟平台。可以说TVP在保证云计算环境安全、构建可信云平台上起到了重要的作用。

总结起来，目前针对TVP模型的研究尽管取得了很多成果并达成了基本共识，即TVP模型都包含基本组件vRT、vTPM等，但绝大多数已有的研究成果把TVP的VMM和管理域都作为TCB，一起作为虚拟机的vRT，这样的设计粒度明显过粗且逻辑上不完全合理的，因为管理域包含OS及大量的应用程序，不能采用链式度量所有的应用程序并存储其PCR值。

1.2.2 可信虚拟平台信任链模型

为更好的对信任链模型的研究现状进行更好的阐述，本文将从目前信任链模型三个不同类别的研究现状进行描述。

（1）通过对TCG链式信任链模型的扩展，实现TVP下可信度量以及信任传递。Scarlata[34]等提出在构建TVP时，通过可信测量构建从CRTM可信根到每个客户虚拟机的信任链，就可以证明每个客户虚拟机是可信的，显然这种信任链模型是不完善的，无法适应比较复杂的TVP环境。John[35]对信任链扩展上提出了“Transitive Trust Chain”信任链模型，并且简要的指出了信任链传递过程为TPM→VMM→TVEM manager→TVEM→VM OS→应用程序，但是此种信任链模型没有详细的描述特权域操作系统以及虚拟机操作系统的可信度量。Shen[14]等根据TCG动态度量方法提出了一种基于Xen的可信虚拟机在 DRTM下的信任链构建，其具体的构建过程为：CPU→可信代码→Xen VMM→Dom0（→vTPM Manager→Domain Builder）→Guest 0S→Guest Application，此种信任链模型也存在John[28]中的问题。

（2）通过研究可信云平台和可信虚拟机两部分的信任链，构建TVP下的信任链模型。常德显[27]等提出TVP信任链包括按照TVP的功能层次从硬件TPM层→TCB层→vRT层→用户虚拟机层的信任链模型，此信任链模型对vRT及层次间的连接定义比较模糊。Zhang Lei[25]等提出一种基于无干扰的可信域层次信任链模型，并且指出分别度量物理主机和VM的方式，即首先度量从物理的TPM到物理主机的应用程序，然后度量VM的vTPM和应用程序，显然此种信任链模型无法有效的对TVP下构建完整的链式信任链模型，不能向用户虚拟机呈现一条完整的信任链模型。

（3）树形或者星形的信任链模型。一部分学者认为TCG的链式信任链可信度量方式在虚拟化环境下是难以有效构建的。朱智强[36]提出了一种安全可扩展的星型信任度量结构，在信任度量时只需要信任根（RT）对管理域节点（Management Domain, MDn）进行度量即可，但是此信任链模型的关键节点RT需要对所有的管理节点进行度量，RT的负担重，无法高效的完成TVP下的可信度量以及信任传递。曲文涛[37]等提出了一种解决RT负担的改进方案，带链式结构的星型信任链模型，设计了MDn节点分担了RT的部分度量负担，但是此种信任链模型也存在负担重的MDn节点。总结起来，目前针对TVP的信任链模型的共同问题是信任链模型过粗且逻辑上不合理的问题，与具体云环境中虚拟化平台也不完全相符合，且目前研

究内容中的可信虚拟平台信任链与虚拟机信任链是两条不同的信任链，这两条信任链在度量层次和度量时间上均是分离的，不能向虚拟机用户展示一条从TPM到虚拟机应用的完整信任链。

总结1.2.1节和本小节，本文得出上述的TVP基本运行架构以及信任链传递模型存在过粗且逻辑上不完全合理的问题，与具体云环境中虚拟化平台也不完全相符合。如图1.1所示，为了便于叙述，本文将图1.1中从TPM到第三层的信任链称为可信虚拟平台信任链，将第四层的信任链称为虚拟机信任链。具体问题表现在：

（1）现有的TVP模型把整个第三层都作为TVP的TCB并作为虚拟机的vRT，是不精细的且逻辑上也不完全合理的。第三层包括VMM以及DOM管理域，信任链为CRTM→BIOS→BootLoader→VMM→DOM OS→Apps，DOM管理域包含OS及大量的应用程序，不能采用链式度量所有的应用程序并存储其PCR（Platform Configuration Module, PCR）值。

（2）虚拟平台信任链与虚拟机信任链是两条不同的信任链，即在整个TVP以及客户虚拟机启动过程中存在两条完全分隔的信任链，一条是可信虚拟平台在启动时的信任链，另一条是客户虚拟机在启动时的信任链，这两条信任链在度量层次和度量时间上均是分离的。如何向虚拟机用户展示一条从TPM到虚拟机应用的完整信任链，即这两条信任链存在如何衔接的问题。

1.2.3 无干扰理论

目前针对于信任链形式化建模与分析的方法有很多研究方向，其中最为热门的当属无干扰理论，本文在此对无干扰理论的研究现状进行阐述。针对于无干扰理论的研究，目前大部分的研究是基于有限状态机从系统内部域产生的动作和行为以及其运行结果的角度上建立了系统安全属性和定理[34][40]。其中，国外研究方向主要集中在系统外部和软件可信性上，Kai E[41]把无干扰利用扩展到非确定性系统，通过系统外部通道共享信息量。Baldan[42]等对无干扰中存在干扰的信息流添加了因果关系，建立了多级安全域的无干扰理论。

国内的针对无干扰理论的研究主要是将无干扰中的关键定义进行细分，比如，张兴[43]、赵佳[44]等在由Rushby提出的有限状态机的无干扰理论的基础上针对系统安全域进行了实例化，使其具体成为进程集合，然后给出了进程集合运行的可信条件和满足终端安全的可信定理，但是其模型中的没有针对动作的详细定义，不适合验证可信云环境信任链。刘鹏威[45]等在文献[40]的基础上重新定义了清除函数，增加了非传递的无干扰安全定理，然而同样存在赵佳中的问题。陈菊[46]等利用无干扰理论构建了从进程和代码完整性之间的安全操作。徐甫[47]等在动态干扰和静态干扰对无干扰理论进行了扩展，但是文章中的动态和静态的定义都是十分抽象的，无法与实际的操作进行匹配。秦晰[48]等从信任组件处理非信任组件的角

度对非信任组件的输出进行了域间隔离和干扰，但是并没有针对安全域进行详细的描述。

针对可信计算平台，张兴[39]等利用无干扰理论对传统可信计算平台信任链进行了建模分析，并且指出只有满足传递无干扰安全策略才可以构建安全的信任链。虽然Zhang[26]等人利用无干扰理论对可信云计算环境信任链进行了形式化分析和验证，但是此种信任链分析方法是建立在不连续的可信云计算信任链模型上，不能够对可信云计算环境进行正确的形式化验证。

上述对无干扰理论的研究，均没有考虑到云计算运行中时的安全域、动作所属主体以及动作对安全域和系统状态的影响进行详细的说明，比如针对于安全域，动作可能属于不同的主体，不同的主体也可发出不同的动作。

1.3 本文主要工作

（1）具有瀑布特征的可信虚拟平台信任链模型

由本文对可信虚拟平台的研究综述来看，目前的可信虚拟平台存在一些需要改进的地方。本文针对现有的可信虚拟平台，在可信虚拟平台和用户虚拟机之间加入可信衔接点，设计一种在逻辑上合理，vRT 精细的可信虚拟平台架构。

在原有的TVP架构中的第二层和第三层加入可信衔接点。可信衔接点位于Dom0，是Dom0的一组应用程序，包括VM的创建组件VM Builder、vTPM实例的创建模块vTPM Builder以及VM-vTPM映射组件VM-vTPM Binding，且作为vRT的一部分，在信任链上按照VM Builder、vTPM Builder、VM-vTPM Binding的顺序依次进行度量。可信衔接点可对TVP-QT的第一、第二层与第四、第五层进行有效衔接，保证TVP-QT信任链构建的连贯性，起到承上启下的作用，具有瀑布特征。

并且该可信虚拟平台按照第一层硬件TPM（CRTM）→第二层TCB（BIOS→OSLoader→VMM→Dom0 Kernel）→第三层可信衔接点（VM Builderv→TPM Builder→VM-vTPM Binding）→第四层vTPM（vTPM实例）→第五层可信虚拟机（VBIOS→VOSLoader→VMOS→APP）进行信任链构建。并且，本文基于安全系统逻辑的形式化分析方法对底层物理平台、可信衔接点等进行了形式化分析。

本文表达的瀑布特征引用自软件开发中的瀑布模型，其主要思想是整体过程中必须按照从上一项工作的输出当初本阶段的输出，中间不能有间隔。而本文提出的可信虚拟平台，在整体的结构中都是以这种方式进行信任链构建，尤其是可信衔接点部分，即可以作为底层物理平台的最后一个环节，又当做可信虚拟机的虚拟可信根，保证了整体的信任链构建是完整的、连续的。

（2）基于扩展无干扰理论的可信虚拟平台信任链分析方法

本文针对目前的无干扰进行扩展，该扩展无干扰理论不仅可以适用于TVP-QT信任链模型，也适用于目前云计算环境下的可信计算平台。首先，按照云计算环境运行特征，对原有无干扰理论中的安全域、动作等定义进行扩充，并将动作主体和动作对安全域以及系统状态的影响等扩展到无干扰理论中；最后应用此扩展的无干扰理论来分析可信云环境信任链传递模型，并定义了扩展无干扰的信任传递判断定理，当云计算安全域之间符合非传递无干扰时，信任链的传递才是安全可靠的。最后，基于本文建立的可信虚拟平台架构对扩展无干扰理论进行了分析和验证，证明了扩展后的无干扰理论验证信任链模型的有效性。

1.4 论文组织结构

本文共分为六章，每章的安排如下：

第1章主要介绍了论文的研究背景及意义，主要从目前国内外云计算的发展及云计算安全的发展；然后介绍了国内外的研究现状，主要从可信虚拟平台，信任链模型，形式化分析方法三个方面进行介绍；最后介绍了本论文的研究方向和研究内容。

第2章主要介绍了对本文研究提供思路的相关理论和技术。首先介绍了云计算中的关键技术和虚拟化技术，以及虚拟机和虚拟机监视器，目前最流行的VMM结构Xen和KVM；其次介绍了可信计算中的关键技术，主要介绍了虚拟可信平台模块，信任链的构建和完整性度量技术，以及虚拟可信平台模块的分类和目前信任链技术的不足；最后介绍了形式化方法，并简要的介绍了安全系统逻辑和无干扰理论形式化分析方法。

第3章主要介绍了本文提出的TVP-QT可信虚拟平台及其信任链模型。首先提出了基于可信衔接点的TVP-QT，可信衔接点由虚拟机构建模块，vTPM构建模块，VM-vTPM绑定模块组成，不仅可以当做可信云平台底层物理服务器的一部分也可以当做虚拟机启动的一部分，完成对底层物理和虚拟机的衔接。其次，介绍了构建在TVP-QT之上的信任链模型并对其信任属性进行定义。最后对信任链进行了功能和性能实验，针对该模型的信任链构建和传统信任链的对比进行了实验和分析。

第4章本文主要利用目前的安全系统逻辑形式化分析方法对第3章提出的TVP-QT信任链模型进行了形式化分析。首先给出了本文形式化分析的基本假定，然后从主机、可信衔接点信任链的的本地验证和远程证明上给出了基于安全系统逻辑的形式化分析。分析过程可以得出本文的信任链模型在理论上是安全可靠的，是一个在数学模型上安全的系统架构和信任链。

第5章主要介绍了本文提出的扩展无干扰理论，主要从安全域、组合安全域及其动作主体进行扩展，并从弱单步一致性、输出一致性等给出了云计算下安全系

统需要满足的性质，然后介绍了非传递无干扰安全策略，其中本文提出的扩展无干扰理论方法可以适用于目前的可信虚拟平台信任链的分析。

第6章主要是对本文的研究内容进行概括分析，并指出了论文的不足之处，对未来的研究方向进行了展望。

2 相关技术与理论

2.1 虚拟化技术

虚拟化技术是将实际的实体资源进行抽象，使单一的计算机资源可以被用来提供多个同类资源的资源管理方式，比如物理计算机服务器、内存、硬盘存储等；主要用来解决当前物理计算机资源利用率低的问题，从而最大化的利用物理资源。在虚拟化技术中，可以利用资源隔离的方式对虚拟出的计算机资源进行安全隔离，使其可以处在抽象的不同的操作系统中向外部提供服务。

2.1.1 虚拟化技术分类

为了满足当前云计算提供商需要提供不同的功能和需求，在实现层次和虚拟方式上都产生了不同类别的虚拟化解决方案，使得不同的虚拟化系统不仅具有最基本的虚拟化技术，也呈现了各自独特的个性[49][50]。 下面本文从实现层次和实现方式对虚拟化技术的分类进行介绍。

(1) 不同实现层次

指令集虚拟化，指的是指令集架构级虚拟化技术。该方式的虚拟化技术能够将操作系统在运行中指令集以软件模拟的方式进行实现，将虚拟机中运行过程中产生的指令转化为本地的指令集，然后在本地及其中执行这些被转换的指令集。目前，比较流行的指令集架构级的虚拟化系统主要由纯软件实现的模拟器Qemu系统和基于x86的开源模拟器Bochs系统。

硬件级虚拟化，这种虚拟化技术主要用来虚拟化具体的计算机主机，使得用户在操作计算机资源时，可以使抽象的计算机操作系统直接发送与真实机器相同的指令集进行资源操作。基于硬件级虚拟化技术，用户可以方便的使用被映射为物理资源的虚拟机资源，是目前云计算虚拟化技术中被重点研究的方向。目前基于硬件级的虚拟化技术有EMC公司的VMWare，以及由英国剑桥大学发起的开源框架Xen，都十分具有影响力。

操作系统级虚拟化，传统的操作系统中运行着来自不同程序的进程，如果操作系统的内核或进程管理出现缺陷或漏洞，不同用户之间相同类别的进程有可能会产生影响。操作系统级的虚拟化技术不仅给用户虚拟出一个真实的计算机资源还可以有效的防止进程之间相互影响的问题。目前最为流行的操作系统级虚拟化技术当属火热的容器技术[51]，其利用Linux的LXC技术，通过设定不同类别的namespace提供给不同用户相同的并且真实的操作系统环境。使得每个用户都可以在独立的用户空间进行活动，相互之间的进程、网络、文件等都通过namespace进

行隔离。这种方式由FreeBSD首创，目前技术上类似的操作系统级虚拟化技术主要由：OpenVZ，Linux-VServer，AIX Workload Partition，以及Docker等容器技术。

（2）不同实现方式

使用二进制翻译的全虚拟化。全虚拟化环境下的虚拟机可以和真实机器一样拥有者未经修改的操作系统。该虚拟的硬件环境能够在不需要修改当前虚拟机的操作系统内核便可以提供如同真实硬件环境提供的服务，虚拟机内核中也无法感知当前运行的硬件环境是否是在虚拟化环境下。目前类别为 Windows 的虚拟机大部分都是基于全虚拟化技术提供的，因为 Windows 系统由于没有被开放源码，无法进行内核修改，因此只能选择全虚拟化方式。当前可以提供全虚拟化的架构有 Xen、VMware、KVM 等。

操作系统辅助虚拟化的半虚拟化。和全虚拟化技术中不对内核进行修改的虚拟化方式不同，半虚拟化技术需要通过对虚拟机操作系统进行内核修改，才可以完成对操作系统的虚拟化。通过半虚拟化技术提供的虚拟机可以通过操作系统下的命令查看到虚拟机的架构。Xen 等都可以提供半虚拟化的虚拟机。

硬件辅助虚拟化。随着虚拟化技术的不断发展，Intel 和 Amd 都推出了各自的虚拟化技术，比如 Intel 的 VT-x，AMD 的 AMD-V 技术，这两个硬件辅助虚拟化技术都针对特权指令为 CPU 添加了一个独特的执行模式，使得 VMM 运行在 0 环，也就是特权域，使得由虚拟机发出的特权和敏感调用自动陷入到虚拟机监视器中。

2.1.2 虚拟机与虚拟机监视器

虚拟机是通过软件模拟的方式产生的具有真实硬件系统的并且运行在一个独立环境中的真实计算机系统。进入虚拟机之后，所有的操作都如同在真实的计算机系统里，可以在独立的系统桌面内安装使用软件，并进行数据的保存等，而且不会对底层的物理资源产生任何影响。虚拟机一般是由可以提供虚拟化技术的软件架构提供，比如 VMware、Xen、KVM[52]。从架构上看，VMM 对硬盘资源、内存、I/O 设备等硬件资源提供虚拟化，承担着管理所有物理资源的任务，并且 VMM 也负责着虚拟化虚拟环境的创建和管理，比如虚拟机的迁移、销毁、备份等。

虚拟机与 VMM 的基本结构图如图 2.1 所示：

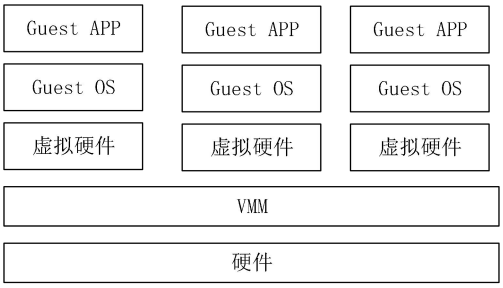

图 2.1 虚拟机与 VMM 基本结构

从图2.1中可以看出，VMM是一个位于操作系统和计算机硬件之间的特权域，可以保证对上层运行的多个虚拟机进行资源隔离，以保障每一个虚拟机都可以在安全的环境下运行，并且虚拟出与甚至硬件资源的虚拟硬件环境，实现了多个虚拟机运行在一个物理平台上，提高了计算机硬件的使用率。实际上VMM的出现不仅仅可以提供虚拟化技术，而且可以对虚拟化平台提供虚拟化相关的管理任务，使得虚拟化资源可以有序高效的提供资源。

2.1.3 Xen 与 KVM

（1）Xen

Xen 是由英国剑桥大学开发的开源虚拟机监视器，Xen 必须对 Linux 内核进行修改才可以在使其在 Linux 环境中充当 VMM 的角色，这种方式可以方便的对虚拟机操作系统发出的 CPU 指令进行进行理解和翻译，并且无需得到特殊硬件的支持便可以达到高效率的虚拟化目的。Xen 既支持全虚拟化又支持半虚拟化，在实际的 Xen 环境中，主要有两个组成部分。一个是位于硬件和虚拟之间的 hypervisor，在硬件启动之后，首先载入，然后启动管理虚拟机。管理域虚拟机也叫 Domain0，在 Xen 环境中是一个拥有很高特权的虚拟机，拥有原生的设备驱动，负责对虚拟机操作系统的管理和命令的转发，这样可以使得 hypervisor 能够把更多的工作用来进行虚拟化。在 Domain0 启动之后，还会载入一个对其他 DomainU 虚拟机进行管理的 Xend 进程，并提供对这些虚拟机进行访问的控制台，管理员可以通过控制台直接与 Domain0 进行对话。DomainU 没有真实的硬件驱动，不能直接对物理硬件资源进行访问，必须通过 Domain0 才能访问。

（2）KVM

KVM 是 Kernel-based Virtual Machine 的简称，也是一个开源的系统虚拟化架构，KVM 自 Linux 2.6.20 版本开始集成在 Linux 的各个发行版本中。KVM 作为 Linux 的一个内核模块，通过 Linux 内核自身的调度器进行模块管理，相对于 Xen，KVM 的核心源码很少，已经成为学术界和产业界主流的 VMM。KVM 需要硬件辅助才可以进行虚拟化，是一种全虚拟化的技术。KVM 自身被构建 Linux 内核模块中，需要在使用的时候进行内核的编译和添加才可以使用，并且 KVM 必须依赖 Qemu 才可以模拟 IO 设备等硬件资源。

2.2 可信计算

可信计算技术是一种保障信息系统安全的新技术，旨在把人类社会成功的基于信任的管理经验用于保障计算机系统安全。可信计算的基本思想是：

首先在计算机系统建立一个可以有物理系统安全技术进行保障的信任根；然后以该信任根为信任基础建立一条从硬件平台到操作系统应用程序启动的信任链[27]，一级度量一级，直到平台所有组件启动。

可信计算技术中最重要的技术是可信计算模块，旨在通过硬件安全模型采用软件和硬件相结合的方式来提高传统信息系统和新型计算平台的安全性和数据完整性。并且可以提供给第三方进行远程验证计算平台的远程证明方式来达到保护本地和远程终端的安全和可靠。

2.2.1 可信平台模块

可信平台模块是可信计算技术的核心技术，是可信计算技术实现的核心。首先，TPM作为可信计算技术实现数据加密、完整性度量的关键模块，必须严格保障自身的安全，并且保障可信计算功能的有效执行，防止外界的干扰。其次，TPM应具备远程证明。构建信任链等功能，并且具有密钥管理和数据加解密等基础功能。可信平台模块的功能应括平台数据保护、完整性存储与报告、身份标识三个部分。

2.2.2 信任链技术

当前世界很多信息的传递都离不开网络这一媒介，同时带来了新型的网络新型安全问题，比如网络传播的木马、病毒等。但是这些恶意代码的最终的作用目标都是计算机终端，并且破坏计算机终端数据的完整性。当前围绕着保护计算机终端的安全性出现了很多信息系统安全保障技术，比如访问控制、入侵检测等等。但是这些技术都是以在计算机启动之后给系统安装针对恶意代码的补丁的方式来提高终端设备的安全性，但是实际上这种方式并没真正的解决终端设备的安全问题。可信计算中的信任链技术能够对系统启动过程以及运行中的组件进行完整性度量和信任传递，从硬件度量和启动时解决安全问题。

可信计算组织通过构建从TPM芯片的CRTM开始，到系统应用启动完成的链式信任链，完成可信计算平台的信任度量。经过计算机物理硬件、系统引导、操作系统以及启动后的应用程序的逐级认证，将信任从硬件底层的静态信任根扩展到整个可信计算平台，保障其安全可靠。

TCG的链式信任链模型如图2.2所示：

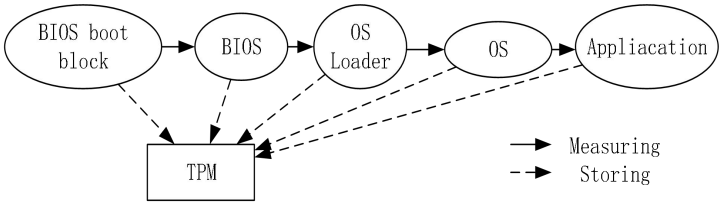

图2.2 TCG信任链模型

下面本文对PCR和完整性度量机制进行简要的介绍。

（1）PCR

平台状态寄存器，可以记录经过特殊算法处理后的系统的各种信息，比如系统进程信息等。但是TPM芯片由于储存的信息有限，所有在PCR中存放的往往是通过SHA-1算法得到的运行状态的哈希值。SHA-1作为一种密码学散列函数，对于任何长度的输入消息都可以生产固定长度的哈希值，一旦输入消息有1bit的差别，就会得出不同的哈希值。因此从PCR中记录的计算机软硬件配置信息是非常容易看到系统完整性的改变。

（2）完整性度量机制

在实际的机器进行启动时，往往都是从系统的BIOS开始，经过设备检查、系统引导、操作系统初始化、操作系统内核加载等主要的过程，直到应用程序加载，才会完成整个系统的启动。如果中间任一环节被恶意篡改导致系统的控制权被恶意的转向不同的过程，都会影响到系统的安全性。比如在系统系统过程中如果加载了一个已经被恶意篡改的操作系统内核，则有可能在系统启动完成之后，整个系统都会被入侵者控制，导致数据丢失、篡改等问题。可信计算中的完整性度量机制可以针对系统启动过程中的每个过程进行完整性度量，并把每个过程的度量信息进行存储。信任链机制可以通过完整性度量完成系统在某个过程启动时，首先判断下一个组件是否安全可信，如果是安全可信的，则将系统的控制权转移到下一个过程，整个系统的启动序列都必须遵守先进行度量然后进行程序执行的规则。

2.2.3 可信计算模块虚拟化

TPM虚拟化，就是在虚拟计算平台中将I/O设备虚拟化技术应用于TPM，使得该平台中的每一个虚拟机，都能安全、独立和自主地使用可信计算功能来提高自身运行环境的可信性和安全性，满足虚拟机用户的可信需求。TPM虚拟化的概念有三方面含义，首先，在虚拟化技术方面，TPM虚拟化本质上是I/O设备虚拟化；其次，在功能方面，每个虚拟机都能使用TPM提供的可信计算功能；第三，在安全目标方面，通过TPM虚拟化，每个虚拟机能够提高自身运行环境的可信性和安全性，满足虚拟机用户的可信需求。对于硬件级虚拟化，根据VMM采用的虚拟化实现方式不同，I/O设备虚拟化采用的实现方式也会不尽相同，其核心在于I/O设备原生驱动的存放位置以及VMM对I/O设备的处理方式。TPM是一个通过LPC总线挂接在主板上慢速I/O设备，采用什么样的I/O设备虚拟化实现技术当然依赖于VMM虚拟化实现方式。

TPM虚拟化的目的是为虚拟机提供可信计算服务，协助虚拟机建立可信计算环境，让使用虚拟机的用户认为和使用带有物理TPM的可信计算机系统没有明显区别。因此，在虚拟化TPM时，有以下基本要求。

等价性，是指虚拟机中的应用程序或用户在使用TPM功能时，除去时间因素外，其余都必须与单独拥有物理TPM的计算机系统一样，包括可信计算的度量、存储和报告等功能。安全性包，括两方面含义，其一是物理TPM应该由VMM全权管理，客户虚拟机操作系统、应用程序或用户不能直接访问TPM；其二是TPM虚拟化软件系统及架构的安全性。虚拟平台对TPM虚拟化必须满足这两方面的安全要求。方便性，是指在TPM虚拟化完成之后，方便对其进行维护、升级和迁移。

2.3 形式化分析方法

形式化分析方法是软件或系统设计时，适合针对软件系统和硬件系统的描述、开发以及验证的基于数学逻辑的特种技术。形式化分析方法的目的是期望软件和硬件设计过程中能够像其他工程学科能够利用适当的数学分析来证明设计系统的可靠性。通常形式化分析方法用于十分注重安全性和可靠性的高度整合的系统。形式化方法通常有一套十分严谨的定义和概念，比如一致性和完整性，以及拥有严谨的证明规范。形式化分析方法的本质是使软件设计拥有数学的特性，使得软件设计产生的系统能够拥有严谨的数学逻辑方法，以及可以被证明的安全数学。形式化分析方法的一般步骤是系统进行抽象定义，然后描述系统应该满足的安全属性，并对安全属性进行验证。

2.3.1 无干扰理论

1982年，Goguen和Meseguer[53]最早提出基于信息流的无干扰理论，但是目前使用的无干扰理论是1992年由Rushby提出，主要采用状态机的无干扰利用，并传递无干扰和非传递无干扰的安全定义。无干扰理论被用来对信任链进行形式化分析的主要原因是无干扰中当前状态只对下一状态产生影响，并且可以通过特定函数删除不存在干扰的因素，这与信任链的构建过程基本相似。本文给出Rushby的基于状态机的无干扰理论的基本定义。

系统中M主要包括：系统状态集合S、动作行为集合A、系统输出集合O、系统安全域集合D四个集合，这四个集合主要存在四种动作函数：

单步状态转移函数：$step: S * A \rightarrow S$；

系统运行函数：$run: S * A^{*} \rightarrow S$；

输出函数： $output: S * A \rightarrow O$；

主域函数：$dom: A$

并且在传递无干扰理论模型中，~为安全域D上安全域间的干扰关系，比如r，t属于D；则$r \sim t$表示安全域r的信息流向t。定义辅助函数$purge: A^{*} * D \rightarrow A^{*}$，$purge(a,v)$表示从动作序列$a$中删除所有从$v$发出的动作序列。

但是Rushby的无干扰理论并不能完全适用于云计算平台，云计算平台存在着大量安全域以及安全域中的组件，每一个组件中又包含各自相互影响的行为动作，这在Rushby的定义是没有涉及到的。

2.3.2 安全系统逻辑理论

目前的安全系统大部分是为抵御安全威胁设计的复杂的系统，例如：虚拟机管理器，安全内核，操作系统，可信平台模块等。由Anupam Datta[54]等人提出的安全系统逻辑理论建立了一套从系统启动开始，到系统程序依次运行期间对指定程序进行安全性分析的形式化分析机制。其针对关于内存等资源的写入、加锁等的机制类似于可信计算技术的PCR和完整性度量，因此可以被用来对可信计算信任链模型的形式化分析。下面本文简要的介绍本文用到的安全系统逻辑中的部分定义。

read l：读取定位l中的行为；

write l,e：向l中写入数据e；

extend l,e：向PCR l中扩展e；

lock l：向l加入写锁；

unlock l：释放l的的写锁；

send e :发送消息e；

Receive e：收到信息e；

sign e,K：用私钥 K 签名 e。

2.4 本章小结

本章主要对于本文研究内容相关的主要技术及理论进行分析和介绍。主要介绍了虚拟化技术，以及从虚拟化实现方式、虚拟化实现层次介绍了虚拟化技术的分类，以及从虚拟机管理的角度上介绍了虚拟机监视器 VMM 等；随后介绍了可信计算技术中的关键技术可信平台模块以及信任链技术，并简单介绍了信任链技术中的重要的概念 PCR 和信任链度量方式等。然后简单介绍了可信计算技术虚拟化及虚拟可信平台模块的基本要求。最后介绍了安全系统逻辑和无干扰理论中一些最基本的定义和函数，后文本文将利用这两种形式化方法或者其扩展对本文建立的信任链模型进行分析和验证。

3 具有瀑布特征的 TVP 架构及信任链模型

本文提出了具有瀑布特征的 TVP-QT 架构及信任链模型，本章将对 TVP-QT 架构及信任链模型进行详细的描述。本文主要针对 TVP-QT 及信任链模型，而针对虚拟化平台固有的安全性机制，比如 VMM 资源隔离、VMM 特权域指令转发等特权域操作，可参考 Gilles Barthe[52]等研究者给出的形式化分析验证方法[27]。本文在本节对 TVP-QT 的功能组件以及 TVP-QT 信任链信任属性进行定义，本文在第 4 章将利用文献[27]提出的安全系统逻辑形式化方法对信任链进行形式化分析。

3.1 TVP-QT 系统结构

本文基于已有的TVP研究方案，提出了TVP-QT运行架构，如图3.1所示。

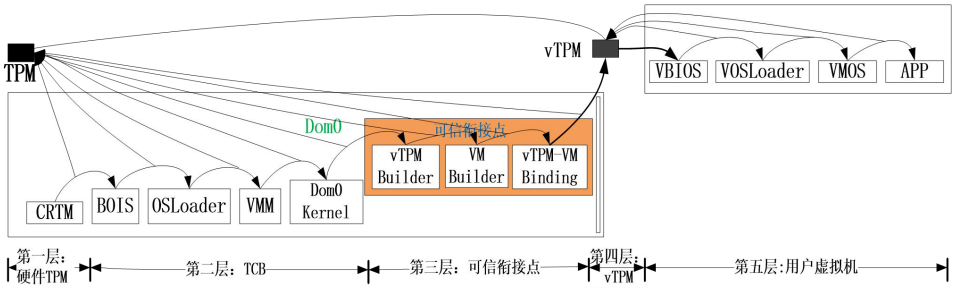

图 3.1 TVP-QT 运行架构

本文提出的TVP-QT运行架构在功能上可以分为5个层次。第一层是硬件信任根TPM构成的可信虚拟平台底层，可以从物理硬件层次保证虚拟化平台的可信；第二层主要包括VMM以及管理域Dom 0的相关组件，本文把管理域记作Dom0，针对不同的VMM，Dom0的启动会有不同的方式。例如，Xen的管理域启动相关组件包括运行中涉及的VBOIS、VOSLoader、VMOS等组件，这一层次可以作为TVP-QT的可信计算基。值得指出的是，与已有的TVP不同，本文只把Dom0 Kernel看成是可信基，这显然更为合理，因为Dom0实际上是整个虚拟化平台的管理域，含大量的应用程序，这些管理程序无法采用TCG链式度量，且也很容易受到攻击而改变[56][57]。第三层是本文重点设计的可信衔接点，可信衔接点位于Dom0，是Dom0的一组应用程序，包括vTPM实例的创建模块vTPM Builder、vTPM-VM映射组件vTPM-VM Binding以及VM的创建组件VM Builder，且作为vRT的一部分，在信任链上按照vTPM Builder→vTPM-VM Binding→VM Builder的顺序依次进行度量。可信衔接点可对TVP-QT的第一、第二层与第四、第五层进行有效衔接，保证TVP-QT信任链构建的连贯性，起到承上启下的作用，具有瀑布特征。第四层为vTPM，vTPM作为可信虚拟机部分的虚拟信任根，是由可信衔接点为虚拟机创建的vTPM实例，它可利用TCG的动态度量信任根（DRTM）机制启动，作为虚拟化平台应用进程的

一部分。最上层为可信虚拟化平台上与用户活动联系最大的用户虚拟机层次，其运行时组件主要包括VBIOS、VOSLoader、VMOS、应用程序（APP）等相关组件。基于上述对TVP-QT的分析，本文从功能角度给出TVP-QT的抽象定义。

定义3.1 TVP-QT主要包括两个部分，一是云计算平台下的主机以及组件的类型集合M，比如：VMM、用户虚拟机、可信衔接点等，主要是向用户提供云计算服务主要的组件和资源构成；二是构建可信虚拟平台的信任根RT（Root of Trust, RT），在本文中，RT主要包括硬件TPM，可信衔接点TJP，和虚拟可信平台模块vTPM。TVP-QT表示为TVP-QT:={M, RT}。

对于TVP-QT的主体集合M也可以分为两部分，M:={m, vm}，其中，m:={VMM, Dom0 Kernel, TJP}，特指底层的VMM、Dom0 Kernel和可信衔接点TJP，它们是 TVP 的TCB。vm表示云计算平台中提高计算服务的虚拟机集合，表示为vm:={vm_1, …,vm_n}（n代表第n个虚拟机）。

同时，TVP-QT的信任根RT也根据架构中不同的实现方式分为两大部分。一是作为底层硬件可信度量根的TPM，具有可信计算平台的有关数据加密、密钥管理等基本特性；二是作为虚拟可信根的vRT，在本文，vRT主要包括可信衔接点TJP和vTPM，根据目前最新的vTPM实现方式和本文的TJP，vRT作为可信虚拟平台上的独立的应用程序或者组件，并且通过软件的方式与硬件TPM进行关联，由TPM确保vRT的安全可信。其中vTPM是软件形式的TPM，具有TPM的安全功能；TJP是可信衔接点，TJP的可信依赖于物理TPM，用来衔接底层的虚拟化平台m和顶层的虚拟机vm。

因此，TVP从功能角度可定义为：

TVP:= {(TPM, m), (vRT_1, vm_1), … , (vRT_n, vm_n)}

= { (TPM, (vmm, Dom0 Kernel, TJP)), ((TJP, $vTPM_1$), vm_1), …,((TJP, $vTPM_n$), vm_n) }

其中，m必须使用TPM 来构建信任，而虚拟机vm则是利用TJP和其相应的vTPM来构建信任。

特别地，可信衔接点TJP可以划分为TJP:= { vTPM Builder, vTPM-VM Binding, VM Builder}，其中vTPM Builder, vTPM-VM Binding, VM Builder都作为可信平台管理域上应用程序的一小部分。vTPM Builder表示与创建和管理vTPM实例的相关组件，并负责提供给vm运行时的vTPM标识以及端口；而vTPM-VM Binding则表示对vm和vTPM实例间绑定关系的相关组件，VM Builder表示与创建用户虚拟机相关的配置文件、组件等。在TVP-QT涉及到的vTPM架构中，每个vm必须与唯一对应的vTPM实例绑定。可信衔接点TJP的来源如表3.1所示，其中vTPM管理组件可由VMM上的vTPM组件提供，比如目前存在与Xen上的vTPM Manager。

表 3.1 TJP 功能组件来源

TJP	主要组件	主要功能组件来源
vTPM Builder	vTPM 启动组件	vTPM 管理组件
	vTPM 配置文件	虚拟化平台
vTPM-VM Binding	vTPM-VM 绑定组件	vTPM 管理工具、VMM
VM Builder	VM 启动组件	VMM
	VM 配置文件	虚拟化平台

相对于已有的TVP，本文提出的TVP-QT运行架构具有如下特点：

（1）TVP-QT更加精细。已有的TVP把整个管理域作为TCB，包括Dom0 Kernel和所有应用程序，而TVP-QT模型仅把Dom0 Kernel、TJP及vTPM作为TCB，由于TJP及vTPM只是管理域中很小一部分应用程序，因此TVP-QT的TCB更小。

（2）TVP-QT在逻辑上也更合理。一方面，已有的TVP的TCB无法采用TCG链式度量机制进行安全保证，而TVP-QT的TCB可以实现链式度量。因此，TVP-QT更符合TCG的链式度量标准。另一方面，TVP-QT增加了TJP，从逻辑上比已有的TVP更加合理。

3.2 TVP-QT 信任链及属性

目前可信计算中的可信主要是遵从 TCG 组织给出的可信定义，即系统在郑体的加载和运行过程中，特定实体的行为是否可以达到系统预期的结果。根据这一定义，完整性度量的方案就可以被规范为系统组件运行前需要进行可信度量，如果和预期的结果一致的话才可以进行被系统整体所信任，以及系统控制权的转移。[27]。并且，云计算环境下的 TVP 的信任链模型也在整体上与普通可信计算平台类似，不仅需要保证底层物理平台能够按照原有的方式基于硬件可信根进行信任链构建，也要保证虚拟机能够基于虚拟可信根进行可信度量。但是在云计算平台中，每个虚拟机都会存在一个单独的信任链构建过程，原有的信任链扩展或者分离的信任链构建均不能达到对云计算平台信任链的链式构建的目的[27]。本文提出的具有可信衔接点的 TVP 架构能够在可信衔接点这一层次作为对虚拟机信任链构建的虚拟可信根。并且，只要虚拟化平台能够确保信任链构建过程的唯一性、正确性，以及能对任意的外部实体 R 证明确实构建了对应的信任链，那么整个虚拟化平台是可信的[19]。如图 3.2 所示，从外部实体来看，虚拟化平台仍然满足 TCG 最初建立信任环境的思想。

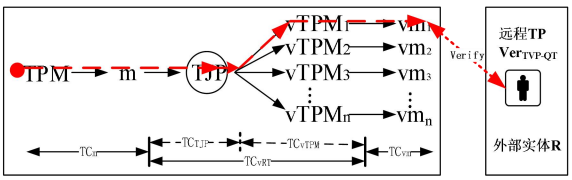

图 3.2 虚拟化平台可信环境信任链构建与验证

本文为验证 TVP-QT 信任链模型能保证信任链构建过程中信任传递的正确性，以及 TVP-QT 不仅能够在系统加载时程序控制权转移过程中每个过程都能够按照预期的行为进行执行，而且能够向外部实体对其信任属性和执行过程进行验证，对信任链构建过程中涉及的组件进行抽象。本文把待验证的目标统一为信任属性（Trusted Property, TP），其抽象定义如下。

定义 3.2（TVP-QT 信任链模型的信任属性 $TP_{TVP\text{-}QT}$）根据上文对 TVP-QT 信任属性的描述，TVP-QT 的信任属性应该定义为一个二元组 $TP_{TVP\text{-}QT}:=\{TC_{TVP\text{-}QT},Ver_{TVP\text{-}QT}\}$，其中 $TC_{TVP\text{-}QT}$ 表示在 TVP-QT 进行信任链构建过程中需要进行完整性度量以及控制权传递的程序序列，即上文对 TVP-QT 信任链模型具体构建过程的描述的各个组件序列。$Ver_{TVP\text{-}QT}$ 表示在对外部实体进行远程验证时所需要进行验证的程序或组件序列的属性。

按照本文 3.1 节对 TVP-QT 系统各个组件的抽象定义，TVP-QT 信任链的信任属性可以表述为如下方式：

$$TP_{TVP\text{-}QT}:=\{TC_{TVP\text{-}QT},Ver_{TVP\text{-}QT}\}=\{(TC_m,TC_{vRT},TC_{vm}),(Ver_m,Ver_{vRT},Ver_{vm})\}$$
$$=\{(TC_m,(TC_{TJP},TC_{vTPM}),TC_{vm}),(Ver_m,(Ver_{TJP},Ver_{vTPM}),Ver_{vm})\}$$

由定义可知 TVP-QT 信任属性可以分为三类：主机 m 的信任属性 TC_m，虚拟信任根 vRT 的信任属性 TC_{vRT}，以及用户虚拟机的信任属性 TC_{vm}。其中 TC_{vRT} 包含 TC_{TJP} 和 TC_{vTPM} 两个属性。本文将按照主机、虚拟信任根、虚拟机三个方面对 TVP-QT 的信任属性进行描述。

（1）主机的信任属性 TP_m 可以表示为 $TP_m:=\{TC_m,Ver_m\}$。其中，TC_m 表示表示以底层 TPM 为硬件信任根进行信任链构建的本地信任属性，在该部分信任链传递过程中不存在除信任链之外的其他组件的加载，即主机 m 正确的可信启动过程应该完整的表示为：$(CRTM\rightarrow BIOS\rightarrow OSLoader\rightarrow VMM\rightarrow Dom0\ Kernel)_{TPM\_Static}$，TPM_Static 表示以静态度量方式完成 m 的可信度量。其中 Ver_m 表示主机 m 在远程验证过程中向外部实体 R 证明 m 应该拥有的信任属性 TC_m，即 $Ver_m:=Verify(m, TC_m)$。

同理，虚拟可信根 vRT 的信任属性表示为 $TP_{vRT}:=\{TC_{vRT}, Ver_{vRT}\}$，$TC_{vRT}$ 表示为 vRT 的本地可信加载的信任属性，Ver_{vRT} 表示对外部实体证明的信任属性。并且，由定义 3.1 对 vRT 以及定义 3.2 对 TVP-QT 信任属性的定义，可对 TP_{vRT} 进行进一步细分：

$$TP_{vRT}:=\{(TC_{TJP}, Ver_{TJP}), (TC_{vTPM}, Ver_{vTPM})\},$$

其中，TC_{TJP} 表示基于硬件信任根构建的信任链衔接点，其信任传递的过程包括两种情况，其一是在 m 启动时，需要采用静态度量方式对 TJP 进行度量，其信任传递的过程为 $(\cdots\cdots\rightarrow Dom0\ Kernel\rightarrow TJP)_{TPM\_Static}$，完整地表示为：

（CRTM→BIOS→OSLoader→VMM→Dom0 Kernel→vTPM Builder→vTPM-VM Binding→VM Builder）$_{TPM\_Static}$；其二是在创建 vm 时，为了保证 TJP 的可信（由于 TJP 是应用程序，恶意程序容易篡改），从而使得信任关系可以传递到新建的 vm，需要采用动态度量方式对 TJP 重新度量验证，信任传递的过程为$(TJP)_{TPM_Dynamic}$，完整表示为：(vTPM Builder→vTPM-VM Binding→VM Builder)$_{TPM\_Dynamic}$。在这两种情况下 Ver_{TJP}: =Verify (TJP, TC_{TJP}) 表示对外验证可信衔接点所声称的信任属性 TC_{TJP}，使远程验证者 R 相信 TVP-QT 的可信衔接点拥有这样的信任链属性 TC_{TJP}；TC_{vTPM} 表示基于硬件信任根构建的用户虚拟机信任根 vTPM。本文在此需要强调的是，由于 vTPM 的实现方式可以是一个轻量级的微内核系统，比如 Xen4.4 之后的的 vTPM 实例域，也可以是一个通过 vTPM 管理器启动的进程，比如 Xen4.2 的 vTPM 的构建后产生的 vTPM 进程。但是，这两个实现方式都要保证 vTPM 和 TPM 之间的紧密联系，才可以保证 vTPM 功能的正常实现。TJP 到 vTPM 的信任传递，既可以采用静态度量，也可以采用动态度量，其信任传递的过程为：$(TJP→vTPM)_{TPM\_Staic}$ 或$(TJP→vTPM)_{TPM_Dynamic}$。Ver_{vTPM}:=Verify(vTPM, TC_{vTPM})，表示对外部实体验证 vTPM 所拥有的信任属性 TC_{vTPM}。

（3）同理，用户虚拟机 vm 的信任属性可表示为：TP_{vm}:={TC_{vm}, Ver_{vm}}，其中 TC_{vm} 表示基于 vTPM 构建的信任链，在创建 vm 时需采用动态度量方式对 TJP 进行度量，vm 从初始化到应用的可信启动过程：

$(TJP)_{TPM\_Dynamic}$→{INIT→VBIOS→VOSLoader→VMOS→APP}$_{vTPM\_Static}$。

Ver_{VM}:=Verify(vm,TC_{vm})表示 vm 信任链的外部验证。

显然，相对于已有的 TVP 信任链模型，本文提出的 TVP-QT 信任链模型具有如下特点：

（1）TVP-QT 信任链模型具有瀑布特征。TJP 将分离的两条信任链连接起来，保证 TVP-QT 信任链构建的连贯性，起到承上启下的作用。

（2）TVP-QT 信任链模型具有动态性和层次性。动态性主要体现在两个方面，其一，从时间上看 m 的信任链和 vm 的信任链是两条分离的信任链；其二，可信衔接点 TJP 在 m 启动时采用的是静态度量，而在 vm 创建时，需要动态度量。这是因为为了防止 m 内的恶意程序对 TJP 进行篡改，破环新创建 vm 的可信性。层次性主要体现在 m 的信任链是基础，处于底层，而各 vm 的信任链是信任扩展，处于顶层。底层信任链和顶层信任链通过衔接点 TJP 链接，保证顶层信任链到顶层信任链的信任扩展。

（3）TVP-QT 信任链模型解决了虚拟平台信任链与虚拟机信任链的衔接问题。虚拟化平台存在两条信任链，其一是虚拟平台在启动时的信任链，其二是客户虚拟机在启动时的信任链，这两条信任链在度量层次和度量时间上均是分离的，存

在着两条信任链如何衔接的问题。已有的 TVP 信任链模型对这个问题没有具体回答，比较笼统；但是 TVP-QT 信任链模型回答得比较具体和清楚。

3.3 基于 Xen 的实例系统分析与讨论

为了在实际系统中检验 TVP-QT 及其信任链，本文选择已经构建的实例系统进行分析。该实例系统基于 Xen 半虚拟化平台，如图 3.3；

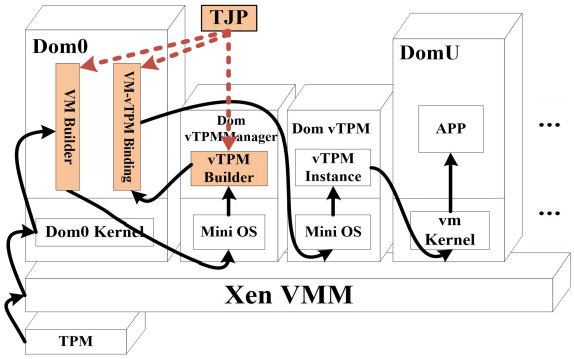

图 3.3 基于 Xen 的 TVP-QT 系统

其中，vRT 被分散在 Dom0、vTPMmanager 域和 vTPM 域。本节本文根据第 3.2 节中对 TVP-QT 信任链的描述：第一层硬件 TPM（CRTM）→第二层 TCB（BIOS→OSLoader→VMM→Dom0 Kernel）→第三层可信衔接点（vTPM Builder→vTPM-VM Binding→VM Builder）→第四层 vTPM（vTPM 实例）→第五层可信虚拟机（VBIOS→VOSLoader→VMOS→APP），将 TVP-QT 信任链分为三部分，第一部分就是虚拟化平台，包括 TVP-QT 信任链的第一层和第二层，第二部分是可信衔接点 TJP，就是 TVP-QT 信任链的第三层，第三部分是用户虚拟机，就是 TVP-QT 信任链的第四层和第五层。接下来本文结合 Xen 4.4 系统，对这三部分信任链进行实际的分析与讨论。

对于第一部分，在 Xen 平台硬件加电启动之后，把 CRTM 作为整个信任链的起点，并由 CRTM 首先度量物理平台 BIOS 和其他有关 BIOS 的配置，然后 BIOS 获得系统的控制权并度量 Xen 的引导程序 Grub，主要度量 grub-xen(head.S, tramp oline.S, x86_32.S)，Grub 获得控制权后会根据 Xen 的镜像头信息获得入口地址 Ox l0000 后读入 Xen 的镜像，并对此镜像和__startxen()并进行度量，然后把控制权交给 Xen，Xen 获得信任之后对 Dom0 相关组件进行度量，包括 construct_dom0()、_ start_32_、start_Kernel 和 LinuxOS 镜像等。然后把控制权交给 Dom0。至此，第一部分可信引导结束。

对于第二部分，Dom0 Kernel 获得控制权后首先度量 TJP 的 vTPM Builder，包括 Xen 中创建 vTPMManager 域的配置文件（.cfg）、vTPM Manger 域（主要是 MiniOS 镜像文件和 vTPM Manger 程序）以及启动 vTPM 的 vtpm-common.sh、vt

pm-impl.sh 等组件。然后把控制权交给 vTPM Builder，vTPM Builder 获得控制权，对 TJP 的 vTPM-VM Binding 进行度量，包括 Xen 中 xl、xenstore、vtpmd、tpm-xen、vtpm_manager_handle 等针对 vTPM-VM 绑定的组件。随后 vTPMBuilder 把控制权交给 vTPM-VM Binding，vTPM-VM Binding 获得控制权后，对 TJP 的最后的组件 VM Builder 进行度量，包括 Xen 的 xl、libxl（Xen4.1 之后 xl 作为默认的管理工具）等创建虚拟机所需的组件以及创建虚拟机的配置文件（.cfg）和虚拟机的镜像文件（.img）。完成 VM Builder 可信度量后，VM Builder 获得信任链控制权。至此，第二部分可信信任链传递结束。

对于第三部分，完成度量 VM Builder 后，可以采用两种方法对 vTPM 进行度量，其一是静态度量，其二是动态度量。如果采用静态度量，控制权在 VM Builder，如果采用动态度量，则控制权在物理 TPM。但无论是静态度量还是动态度量，度量的对象都是 vTPM 实例域，包括 vTPM 实例域的配置文件(.cfg)以及启动文件(.img)和 Mini OS、tpm instance 等组件。但是由于目前的 DRTM 机制在实现过程中会存在很多限制，比如要求进行动态度量的代码需要不依赖与其他组件，对本文的 vTPM 实例不太适用，因此本文采用静态度量方式实现对 vTPM 实例的完整性度量。~~V~~M Builder 完成对 vTPM 实例域的度量后，把控制权交给 vTPM 实例域，vTPM 实例域获得控制权，对最后的 DomU 部分进行可信度量，包括 DomU 启动的内核所需启动信息页的有关的 xen.h、start_info、qemu-dm、qemu-xen、pc-bios 等组件和 linux 镜像文件进行度量。需要说明的是，Xen 中虚拟机有关的 BIOS、引导等组件是利用封装在 Xen 中的 Qemu 实现的，所以需要对 Xen 中 Qemu 重要组件进行可信度量，如 qemu-io、qemu-img 等。在 DomU 启动相关组件完成度量之后，可信虚拟平台最后一部分信任链完成可信度量和信任传递。

以上述实例系统为例，本文完整展示了本文建立的通用抽象模型。值得注意的是，本实例系统的信任链得以正确传递需要满足以下前提：

（1）必须保障 vRT:=TJP+vTPM 自身的可信。在实例系统中，可信衔接点 TJP 包含的组件比较多，不仅大量应用程序、支持库和大量配置文件，而且还涉及 Dom0、vTPM mana- ger 和 vTPM 等域，需要度量的内容多，不允许出现遗漏，特别是 TJP 和 vTPM 关键的组件和配置文件必须是被度量的对象。

（2）必须确保 TJP 中的 vTPM Builder、vTPM-VM Binding、VM Builder 三个管理程序在启动时按顺序执行。尽管 vTPM Builder、vTPM-VM Binding 和 VM Builder 是 Dom0 中的应用程序，但必须保证按顺序执行才能度量结果。

3.4 实验及结果分析

本文基于 Xen 实现了 TVP-QT 的原型系统，并进行仿真实现和结果分析，对

TVP-QT 信任链进行有效性验证和性能测试。下面对仿真实验的环境进行描述。

3.4.1 实验环境

本文使用 TPM-Emulator 对 TPM 功能进行仿真模拟，实验的 Xen VMM 版本为 Xen4.4.0，实验物理平台的配置为 Intel Core i3 @3.4GHz 处理器，内存为 8GB，物理存储为 1T。Dom0 采用 Ubuntu LTS14.04，内核版本为 Linux3.19.0，DomU 使用类型为 Ubuntu LTS14.04 的半虚拟化虚拟机，内存为 4GB，并且部署不同的应用作为仿真实验的测试对比。

下表为 TVP-QT 实验环境所用物理平台和 DomU 类型为 Ubuntu 的具体配置信息：

表 3.2 物理平台(Dom0)和用户虚拟机(DomU-Ubuntu)配置

配置项	物理平台(Dom0 特权域)	用户虚拟机(DomU-Ubuntu)
CPU	Intel Core i3 @3.4GHz	Intel Core i3 @3.4GHz
内核版本	Linux3.19.0	Linux3.19.0
内存	8G	4G
二级缓存	4M	4M
硬盘容量	1T	30G

以下图示表示在 Dom0 上创建 DomU 类型为 Ubuntu LTS14.04 的配置文件部分参数，以及实验所需 vTPMManager 域以及 vTPM 实例域的配置参数。

图 3.4 为类型为 Ubuntu 的用户虚拟机配置：

UbuntuTest1.cfg

```
Kernel = "/boot/vmlinuz-3.19.0-25-generic"
ramdisk = "/root/xen-image/UbuntuTest1.img"
name = "UbuntuTest1"
memory = "4096"
disk = [ 'file:/root/xen-image/UbuntuTest1.img,sda1,w' ]
vtpm=["backend=vtpm-UbuntuTest1"]
.……
```

图 3.4 DomU-Ubuntu 配置部分参数

图 3.5 为 UbuntuTest1 对应的 vTPM 实例配置文件：

vtpm-UbuntuTest1.cfg

```
name="vtpm-UbuntuTest1"
Kernel="/usr/lib/xen/boot/vtpm-stubdom.gz"
extra="loglevel=debug"
memory=8
disk=["file://root/xen-images/vtpm-UbuntuTest1.img,sda1,w"]
```

```
vtpm=["backend=vtpmmgr,uuid=ac0a5b9e-cbe2-4c07-b43b-1d69e46fb839"]
.……
```

图 3.5 Ubuntu vTPM 实例配置部分参数

图 3.6 为 vTPMManager 域配置文件：

vtpmmgr.cfg

```
name="vtpmmgr"
Kernel="/usr/lib/xen/boot/vtpmmgr-stubdom.gz"
extra="tpmlocality=2"
memory=8
disk=["file:file://root/xen-images/vtpmmgr-stubdom.img,hda,w"]
iomem=["fed42,1"]
……
```

图 3.6 vTPMManager 配置文件部分参数

3.4.2 TVP-QT 信任链构建

TVP-QT 信任链在此虚拟化平台进行有效性测试时，利用哈希函数对信任链各层次的构建模块、功能组件或文件进行哈希值存储。按照 TCG 标准，采用迭代计算 Hash 值的方法对 PCR 进行扩展操作，将 PCR 的现值与新值相连，然后通过新的 Hash 函数计算方法对 Hash 值进行计算，并把新产生的值存储到相应的 PCR 中：

$$\text{New PCR}_i=\text{Hash}(\text{Old PCR}_i||\text{New Value})$$

其中，Hash 函数选用 SHA-1，||表示连接符号。在实验中成功运行虚拟机 UbuntuTest1。按照下表的顺序对 PCR 进行存储。其中 PCR[0]-[7]存储 TVP-QT 信任链第一层到第二层 TCB 的可信度量信息；PCR[8]-PCR[10]分别存储信任链中可信衔接点三个重要的组件的度量信息；PCR[11]-PCR[15]存储 vTPM 实例域和用户虚拟机信任链度量信息。具体的存储如表 3.3 所示：

表 3.3 仿真实验 PCR 存储简述

寄存器	存储内容	功能层次
PCR[0]-PCR[7]	BIOS 代码 可信云平台配置信息 Xen 引导 xen-grub (head.S,trampoline.S,x86_32.S 等) Xen VMM 内核代码(Xen Kernel) Dom0OS 启动相关信息 (construct_dom0()、_start_32_、start_kerne 等) Dom0 OS Kernel	第一层：硬件 TPM 第二层：TCB
PCR[8]	MiniOs 及 vTPMManager 配置文件(vTPM Builder，.cfg、.img 以及 vtpm-common.sh、vtpm-impl.sh 等组件)	第三层：可信衔接点
PCR[9]	负责 vTPM-VM 相关组件 (xl、xenstore、vtpmd、tpm-xen、vtpm_manager_handle 等组件)	
PCR[10]	VM 配置文件(VM Builder，xl、libxl 以及.cfg、.img 等组件)	
PCR[11]	MiniOs 及 vTPM 实例域（.cfg,.img,tpm instance 等组件）	第四层：vTPM
PCR[12]	VBIOS 及其他虚拟 BIOS 配置信息(qemu-dm、qemu-xen、pc-bios 等组件)	第五层：可信虚拟机部分
PCR[13]	VOSLoader(虚拟机启动引导文件)，如 Linux 系统的 initrd 和 vmlinuz 文件。	
PCR[14]	VM 启动的其他信息 (xen.h、start_info、qemu-io、qemu-img 等组件)	
PCR[15]	VM 中的应用程序	

按照 TVP-QT 信任链顺序存储的信任链信息结果如图 3.7 所示。根据 Hash 值计算基本不可逆的特性，可以反复对固定配置的系统进行信任链构建，存储的 PCR 值是不变的，一旦中间出现了 Hash 值改变的情况，则 PCR 对应的系统的层次某个组件或者文件处于被改变的状态，不再是初始的状态，以此可以判断信任链构建过程中，系统组件是否发生了恶意篡改的情况。

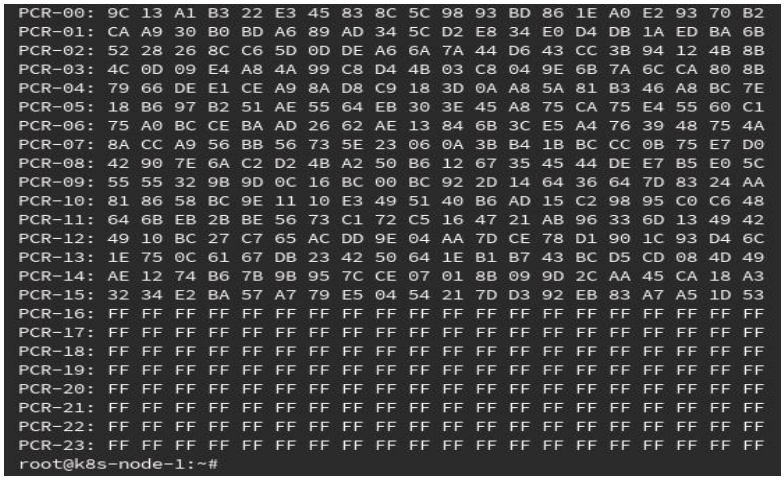

图 3.7 信任链 PCR 信息

本文在实验中人为的对虚拟机启动的配置文件添加一些无用的注释，实验结果如下，由图看出，针对于可信衔接点的 PCR 信息已经发生改变。如图 3.8 所示：

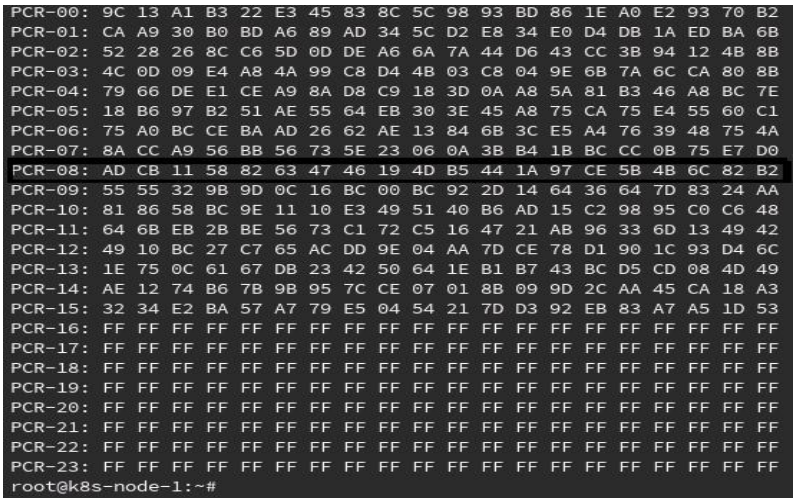

图 3.8 修改 VM 配置文件后的 PCR 信息

3.4.3 TVP-QT 性能测试及分析

与已有的 TVP 信任链模型相比，TVP-QT 信任链模型增加了可信衔接点 TJP。TJP 包含的组件 vTPM Builder、vTPM-VM Builder 和 VM Builder 无论作为 Dom0 的内核组件还是作为 Dom0 的应用，均需要再独立度量，因此，无论是对底层 m 信任链的构建还是顶层 vm 信任链的构建均会带来额外的开销。对于底层 m 信任链的构建，TVP-QT 信任链模型比已有的 TVP 信任链模型增加了对 TJP 的静态度量；对于顶层 vm 信任链的构建, TVP-QT 信任链模型比已有的 TVP 信任链模型增加了对 TJP 的动态度量。

为此本文首先针对 TVP-QT 信任链构建过程中有关主机 m 的信任链构建进行性能测试和结果分析，并与传统的 TVP 架构（图 3.1 所示）进行对比；然后针对 TVP-QT 信任链构建过程中有关 vm 的信任链构建进行性能测试和结果分析。值得

注意的是，与传统的 TVP 信任链相比，vm 信任链的构建过程仅仅多了对 TJP 的动态度量。

本节性能测试的实验环境采用表 3.2 所描述的物理平台 Dom0(Ubuntu LTS 14.04)和用户虚拟机 DomU(Ubuntu LTS 14.04)，并且在 Dom0 和 DomU 分别安装一些常用软件来模拟云计算开发环境和云用户环境，比如 Firefox[57]、Wine[58]、WPS for Linux[59]、Eclipse[60]等。下面本文分别针对在 TVP-QT 和传统 TVP 下 m、vm 的信任链构建实验，对性能方面进行对比和分析。

（1）信任链构建的性能分析

传统 TVP 信任链中主机 m 的信任链构建过程为：

CRTM→BIOS→OSLoader→VMM→Dom0 OS Kernel →app；

TVP-QT 中主机 m 的信任链构建过程：

CRTM→BIOS→OSLoader→VMM→Dom0 OS Kernel →vTPM Builder →vTPM-VM Binding→VM Builder→ other_app

本文对以上两条信任链进行 50 次实验，并记录每次的完成时间。如图 3.9 所示。

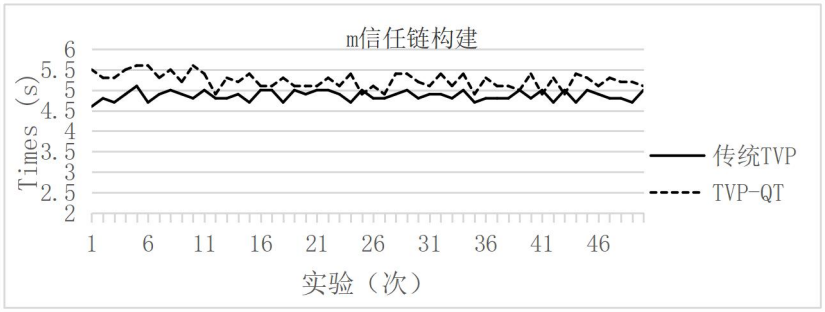

图 3.9 m 信任链构建时间

由图 3.9 可知，虽然 TVP-QT 在主机 m 上比传统 TVP 多了 TJP 的静态度量，但是在时间上并没有太大的多余开销，对可信虚拟的运行的影响不大。所以，TJP 的引入可以在保证可信虚拟平台 m 相关组件实现完整度量的情况下，不会给平台带来太多的开销。

（2） vm 信任链构建的性能分析

传统 TVP 信任链中 vm 的信任链构建过程为:

INIT→VBIOS→VOSLoader→VMOS→APP

TVP-QT 中主机 vm 的信任链构建过程：

$(TJP)_{TPM\_Dynamic}$→vTPM→VBIOS→VOSLoader→VMOS→APP

本文对以上两条信任链进行 50 次，并记录每次的完成时间。如图 3.10 所示。

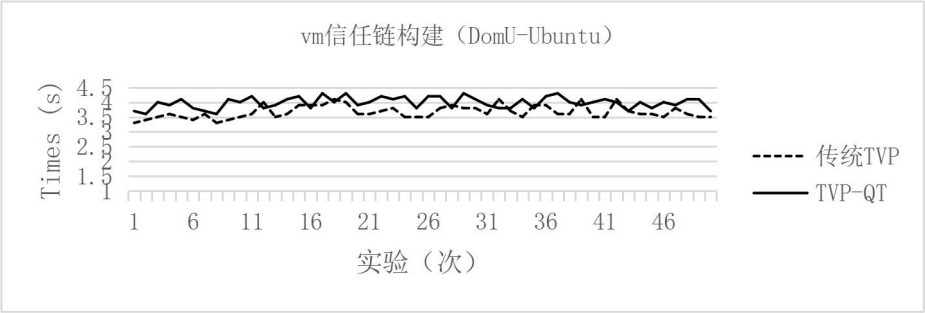

图 3.10 vm 信任链构建时间

由图 3.10 可知，TVP-QT 相比传统 TVP 下对 vm 的信任链构建过程，也仅仅多了由 TJP 带来的额外开销，可以保证对 vm 可信度量后的正常启动。

综上对 TVP-QT 与 TVP 信任链构建过程的对比实验来看，带有可信衔接点 TJP 的可信虚拟平台 TVP-QT 能够保证在对整个平台带来足够小的开销的情况下实现对平台的可信度量；并且可以保证本文在第 3 章提出的 TVP-QT 的信任链的优点，即在整体上以链式信任链构建整个可信虚拟平台信任链，以及拥有逻辑比较合理和粒度精细的虚拟信任根 vRT。

3.5 本章小结

本章为了解决已有 TVP 模型过粗且逻辑不完全合理，而且还存在底层虚拟化平台和顶层用户虚拟机两条分离的信任链等问题，提出了一种具有可信衔接点的 TVP-QT 模型，并对 TVP-QT 中的功能组件及其信任属性进行详细定义，并结合可信衔接点在 TVP-QT 建立从虚拟化平台硬件 TPM 开始的完整信任链模型 TVP-QT 信任链。最后，本章基于 Xen VMM 实现了 TVP-QT 原型系统，并对信任链 TVP-QT 信任链的构建过程进行了详细的描述，通过仿真实验对 TVP-QT 及其信任链的有效性和性能等进行了测试，证明了该信任链的正确性和有效性。

4 基于 LS² 的 TVP-QT 信任链分析

针对信任链的形式化建模和分析可以确保可信虚拟平台的可验证性。本文将采用已有的形式化分析方法“安全系统逻辑（Logic of Secure System, LS²）”对 TVP-QT 信任链模型进行形式化分析。

4.1 基本假定

本章对第 3 章定义的 TVP-QT 信任链的属性进行形式化分析前提的说明：

TVP-QT 包括主机和用户虚拟机启动所需的系统镜像文件都是经过完整性保护的，并且可以利用静态度量或者动态度量进行可信度量；（2）TJP 和 vTPM 组成的虚拟可信根可以经过 TVP-QT 的动态度量技术进行可信度量；（3）外部实体 R 和本地的 TVP 通过安全协议可以进行通信，并且远程验证阶段须根据 TCG 的完整性报告协议；（4）可信第三方已经对虚拟机的 vTPM 实例的平台身份密钥进行认证和颁发。[26]

从 3.2 的分析可知，本文对 TVP-QT 信任链的信任属性分析验证主要包括 3 部分：

（1）包括 TJP 在内的主机 m 的信任链属性验证以及向外部实体进行的远程验证；

（2）TJP 动态度量验证及远程验证；

（3）利用 vTPM 构建的 vm 信任链验证及远程证明；

在这 3 部分中，对（3）的验证分析与文献[26]相同，具体过程读者可参见文献[26]，本文不再论述；下面本文只对（1）（2）进行验证分析。

4.2 主机 m 信任链的本地验证及远程证明

4.2.1 本地程序执行

根据 3.2 节对 TVP 中 m 信任属性 TP_m 定义以及 TP_{vRT} 中对 TC_{TJP} 的定义，其信任链本地执行过程中涉及到的程序如图 4.1 所示。

SRTM(*m*)	≡ *b* = read *m.bios_loc* Extend *m.pcr.s*,*b*; Jump *b*
BIOS(*m*)	≡ *o* = read *m.os_loader_loc* Extend *m.pcr.s*,*o*; Jump *o*
OSLoader(*m*)	≡ *v* = read *m.vmm_loc* Extend *m.pcr.s*,*v*;

Jump *v*

VMM(*m*) ≡ *d* = read *m.dom0_Kernel_loc*

Extend *m.pcr.s*,*d*;

Jump *d*

Dom0 Kernel(*m*) ≡ *vb*= read *m.tjp_loc*

Extend *t.pcr.s*,*t*;

Jump *vb*

vTPM-Builder(*m*) ≡ *vv*= read *m.vtpm-vm-binding_loc*

Extend *m.pcr.s*,*vv*;

Jump *vv*

vTPM-VM Binding(*m*) ≡ *vmb* = read *m.vm-builder_loc*

Extend *m.pcr.s*,*vm*;

Jump *vmb*

VM-Builder(*m*) ≡ o_*app* = read *m.* o_*app* _*loc*

Extend *m.pcr.s*, o_*app*;

Jump o_*app*

Other_APP(*m*) ≡ ...

图 4.1 TVP-QT 中 m 信任链传递

程序执行流程：m 首先从 CRTM 启动执行，把 BIOS 的代码 *b* 从 *m.bios_loc* 中获取然后扩展到相应的 PCR 中；然后指令 Jump *b* 执行完毕之后，把控制权交给 BIOS，BIOS 读取 OS_Loader 的代码扩展到相应的 PCR 中；然后依次对 OSLoader、VMM、Dom0 Kernel 的控制权的转移和 PCR 扩展、指令执行等相关动作，直到可信衔接点 TJP 的成功加载。

4.2.2 本地可信属性描述

由上文描述的信任链传递所涉及的程序执行过程可以得到这样的结论，主机 m 的程序执行顺序可以反映在 PCR 的扩展顺序上，PCR 和程序的加载顺序是一致的。所以，本文基于定义 3.2 以及 PCR 顺序和程序执行顺序一一对应的关系可以将主机 m 的本地信任属性描述为：主机 m 信任链程序加载的顺序表现在 PCR 中度量值的序列，如果主机 m 的信任链组件都能按照预期进行执行，则 PCR 中的值也是正确的序列。即主机 m 信任链的构建顺序必须按照 BIOS、OSLoader、VMM、Dom0 Kernel、vTPM Builder、vTPM-VM Binding、VM Builder 这个特定的序列进行加载。按照安全系统逻辑方法的描述可以表示为：

$\mathrm{MeasuredBoot}_{\mathrm{SRTM}}(\mathrm{m},t)=$

$$\exists t_S.\exists t_b.\exists t_o.\exists t_v.\exists t_d.\exists t_{vb}.\exists t_{vv}.\exists t_{vmb}.\exists t_{o\_app} \wedge$$

$$\exists J.(t_S < t_b < t_o < t_v < t_d < t_{vb} < t_{vv} < t_{vvb} < t_{o\_app} < t)$$

$\wedge$ (Reset(m,J)@t_S) $\wedge$ (Jump(J,*BIOS*(m)) @t_b)

$\wedge$ (Jump(J,*OSLoader*(m))@t_o) $\wedge$ (Jump(J,*VMM*(m))@t_v)

$\wedge$ (Jump(J,*Dom0_Kernel*(m))@t_d)

$\wedge$ (Jump(J, *vTPM-Builder*(m))@ t_{vb})

$\wedge$ (Jump(J, *vTPM-VM Binding*(m))@ t_{vv})

$\wedge$ (Jump(J, *VM-Builder*(m))@ t_{vvb})

$\wedge$ (Jump(J, *VM-Builder*(m))@ t_{o_app})

$\wedge$ ($\neg$ Reset(m)on(t_S,t]) $\wedge$ ($\neg$ Jump(J)on(t_S,t_b))

$\wedge$ ($\neg$ Jump(J)on(t_B,t_o)) $\wedge$ ($\neg$ Jump(J)on(t_o,t_v))

$\wedge$ ($\neg$ Jump(J)on(t_v,t_d)) $\wedge$ ($\neg$ Jump(J)on(t_d, t_{vb}))

$\wedge$ ($\neg$ Jump(J)on(t_{vb}, t_{vv}))($\neg$ Jump(J)on(t_{vv}, t_{vvb}))

$\wedge$ ($\neg$ Jump(J)on(t_{vvb}, t_{o_app}))

上述形式化描述表示：在信任链构建期间，如果对主机 m 的信任链进行了正确构建，程序组件的启动序列一定是 BIOS→OSLoader→VMM→Dom Kernel→TJP，且必须保证没有其他的程序组件加入到信任链构建过程中。为保证该表示正确的，本文对主机 m 的程序组件启动的序列和其扩展的 PCR 之间的强关联关系进行证明。

定理 4.1 如果 m 从 CRTM 启动运行，且与该 m 启动过程对应的 PCR 值为：

seq(*BIOS*(*m*),*OSLoader*(*m*),*VMM*(*m*),*Dom0_Kernel*(*m*), *vTPM Builder*(*m*), *vTPM-VM Binding*(*m*), *VM Builder*(*m*))，

那么主机 m 的信任链传递就是安全可靠的，即按照 *BIOS*(m)到 *OSLoader(*m)再 *VMM*(m)、*Dom0 Kernel*(m)、*vTPM Builder*(m) 、 *vTPM-VM Binding*(m)、*VM Builder*(m)的顺序进行信任链构建。该信任属性形式化表示为

ProtectedSRTM(m)+

Mem(*m.pcr.s*,*seq*(*BIOS*(*m*),*OSLoader*(*m*),*VMM*(*m*),*Dom0_Kernel*(*m*),*vTPMBuilder*(*m*), *vTPM-VM Binding*(*m*), *VM Builder*(*m*))) $\supset$ MeasuredBoot$_{SRTM}$(m,*t*)

证明： 本文按照以下步骤进行证明：

首先，由前提条件可知在时间点 *t*，有公式：

Mem(*m.pcr.s*,*seq*(*BIOS*(*m*),*OSLoader*(*m*),*VMM*(*m*),*Dom0_Kernel*(*m*),*vTPM Builder*(*m*), *vTPM-VM Binding*(*m*), *VM Builder*(*m*)))成立，可以利用 LS² 的 PCR 公理可知，上述公式中的所有的子序列都会在时间 *t* 之前被扩展到 *m.pcr.s* 中，用形式化表示为：

$\exists t_S,t_1,t_2,t_3,t_4, t_5,t_6,J.(t_S \le t_1<t_2<t_3<t_4 <t_5 <t_6<t) \wedge$

(Mem(*m.pcr.s*,*seq*(*BIOS*(*m*),*OSLoader*(*m*),*VMM*(*m*),*Dom0_Kernel*(*m*),*vTPM Builder*(*m*), *vTPM-VM Binding*(*m*), *VM Builder*(*m*)))@*t*)

∧ (Mem(*m.pcr.s*,*seq*(*BIOS*(*m*),*OSLoader*(*m*),*VMM*(*m*),*Dom0_Kernel*(*m*),*vTPM Builder*(*m*), *vTPM-VM Binding*(*m*))) @t_6)

∧ (Mem(*m.pcr.s*,*seq*(*BIOS*(*m*),*OSLoader*(*m*),*VMM*(*m*),*Dom0_Kernel*(*m*),*vTPM Builder*(*m*), @t_5)

∧ (Mem(*m.pcr.s*, *seq*(*BIOS*(*m*),*OSLoader*(*m*),*VMM*(*m*),*Dom0_Kernel*(*m*)))@t_4)

∧ (Mem(*m.pcr.s*, *seq*(*BIOS*(*m*),*OSLoader*(*m*),*VMM*(*m*)))@t_3)

∧ (Mem(*m.pcr.s*, *seq*(*BIOS*(*m*),*OSLoader*(*m*))@t_2)

∧ (Mem(*m.pcr.s*, *seq*(*BIOS*(*m*)))@t_1) ∧ Reset(m,*J*)@t_S

∧ (¬ Reset(*m*)on(t_S,*t*)) **(1)**

接下对图 4.1 中信任链程序的执行过程进行描述，主机 m 启动时首先加载组件就是 CRTM，表示为 Reset(m,*J*)，CRTM 加载完成之后对第一个信任扩展程序 BIOS 进行执行，需要写入的 PCR 会被加上写锁，防止其他程序写入 PCR，保障了期间不会对其他程序进行加载，及属性（2）是成立的。

$\forall t', b, o$

(((Mem(*m.pcr.s*,*seq*(*BIOS*,*b*,*o*)))@*t'*)

$\wedge(t_s < t' < t)) \supset \exists t_b \cdot ((t_s < t_b < t) \wedge (\text{Jump}(J,b)@\ t_b)))$

$\wedge(\text{IsLocked}(m.pcr.s,J)@\ t_b)$ **(2)**

类似地，接下来的信任程序：

OSLoader(*m*),*VMM*(*m*),*Dom0_Kernel*(*m*),*vTPM Builder*(*m*), *vTPM-VM Binding*(*m*), *VM Builder*(*m*)，

也利用 LS^2 规则，在某个时间 t_o、t_v、t_d、t_{vb}、t_{vv}、t_{vmb}、t_{o_app}，程序会跳转到 *o*、*v*、*d*、*vb*、*vv*、*vmb*、*o_app*，并且在跳转期间不会存在其他干扰的程序组件的加载，对应的 PCR 值也可以正常的被加锁锁定。这些时间也会存在类似属性（2）的表述，不再详细叙述。

根据式（2）及 t_o、t_v、t_d 等相似属性可知，在满足本文的假设前提下，m 信任链的执行顺序一定是从 BIOS(m)到 OSLoader(m)再到 VMM(m)、Dom0 Kernel(m)、TJP(m)。

$\exists t_S, t_b, t_o, t_v, t_d, t_{vb}, t_{vv}, t_{vmb}, t_{o\_app} \wedge (t_S < t_b < t_o < t_v < t_d < t_{vb} < t_{vv} < t_{vvb} < t_{o\_app} < t)$

∧ (¬ Reset(*m*,*I*)on (t_S,*t*]) ∧ (Reset(*m*,*J*)@ t_S)

∧(Jump(*J*,*BIOS*(*m*)) @t_b) ¬ (Jump(*J*,*BIOS*(*m*)) on(t_S, t_b))

∧(Jump(*J*,*OSLoader*(*m*))@t_o)∧(¬ Jump(*J*,*OSLoader*(*m*)) on(t_b,t_o))

∧(Jump(*J*,*VMM*(*m*))@t_v) ∧ (¬ Jump(*J*,*VMM*(*m*)) on(t_o, t_v))

$\wedge$(Jump(J,*Dom0_Kernel*(m))@t_d) $\wedge$ ($\neg$ Jump(J,*Dom0_Kernel*(m)) on(t_v, t_d))

$\wedge$(Jump(J, *vTPM-Builder*(m) @t_{vb}) $\wedge$ ($\neg$ Jump(J, *vTPM-Builder*(m) on(t_d, t_{vb}))

$\wedge$(Jump(J, *vTPM-VM Binding*(m))@ t_{vv})

$\wedge$ ($\neg$ Jump(J, *vTPM-Builder*(m) on(t_{vb}, t_{vv}))

$\wedge$(Jump(J, *VM Binding*(m))@ t_{vmb}) $\wedge$($\neg$ Jump(J, *VM Builder*(m) on(t_{vv}, t_{vmb}))

$\wedge$(Jump(J, *VM Binding*(m))@ t_{o_app})

$\wedge$($\neg$ Jump(J, *VM Builder*(m) on(t_{vmb}, t_{o_app})) **(10)**

定理 4.1 即得证。

在本文的信任链 m 的形式化验证过程中，并没有对攻击者的行为进行描述，但是中间会发生被攻击的行为，比如，在程序进行扩展 PCR 操作时，虽然对 PCR 进行了加锁操作，但是并不能保证 PCR 在加锁之前是安全的。但是可信计算中基于实体预期行为的表述可以推理出只有得到正确的预期的 PCR 值，信任链的构建才可以继续，程序才可以继续被加载。

4.2.3 信任链远程验证

主机 m 的信任属性不仅仅表现在本地属性的验证上，还需要对远程验证者证明自己在信任链传递过程中，所有程序的加载都是按照预期顺序进行的，从而建立了安全可靠的可信计算环境。LS² 中对远程验证的属性表示为：

$\mathrm{MeasuredBoot}_{SRTM}(m,t)$。

（1）远程验证程序执行

本文给出主机 m 在满足远程证明条件的情况下信任链传递的验证过程，如图 4.2 所示。

$TPM_{SRTM}(m)$ $\equiv w$ = read *m.pcr.s*;

r = sign(*PCR*(s),w),AIK^{-1}(m);

send r

Verifier(m) $\equiv sig$ = recieve;

v = verify *sig*, *AIK*(m);

match v ,(*PCR*(s),

seq(*BIOS*(m),*OSLoader*(m),*VMM*(m),*Dom0_Kernel*(m),*vTPM Builder*(m),*vTPM-VM Binding*(m), *VM Builder*(m))

图 4.2 TVP-QT 中 m 信任传递的远程验证程序

在远程验证过程中，必须保证外部实体和主机 m 是不同的实体，这样才能保证远程验证的必要性和有效性。具体的验证过程可描述为：主机 m 对本地存储的可以表示程序启动序列的 PCR 值用自身的额 AIK 进行签名，然后将其发送给外部实体 R 进行远程验证。随后，外部实体对收到的 PCR 值和预期的 PCR 度量值序

列进行比较，如果收到的 PCR 值可以与预期的序列进行匹配，则说明主机 m 是按照预期的执行序列进行信任链构建的，验证成功。即有下列公式成立。

$\Gamma_{SRTM} = \{\hat{V} \neq AIK(m), \text{Honest}(AIK(m), \{\text{TPM}_{SRTM}(m), \text{TPM}_{DRTM}(m)\})\}$

（2）信任链属性的远程验证

远程验证的信任属性也需要按照执行序列进行验证，本文对所需要证明的远程信任属性进行描述。

定理 4.2 如果外部实体 R 确认主机 m 的信任链构建的度量值是正确的，则 m 的 PCR 为：*seq*(*BIOS*(*m*),*OSLoader*(*m*),*VMM*(*m*),*Dom0_Kernel*(*m*),*vTPM Builder*(*m*), *vTPM-VM Binding*(*m*), *VM Builder*(*m*))。

形式化表示为：

$\Gamma_{SRTM} \vdash [\text{Verifier}(m)]_V^{t_b,t_e} \exists t.(t<t_e) \wedge$

$(\text{Mem}(m.pcr.s, seq(BIOS(m),OSLoader(m),VMM(m),$

$Dom0\_Kernel(m),vTPM\ Builder(m), vTPM\text{-}VM\ Binding(m),$

$VM\ Builder(m))@t)$ **(3)**

Γ_{SRTM}，$\text{Protected}_{SRTM}(m) \vdash [\text{Verifier}(m)]_V^{t_b,t_e} \exists t.(t<t_e)$

$\wedge \text{MeasureBoot}_{SRTM}(m,t)$ **(4)**

属性（4）可由属性（3）根据定理 4.1 直接得出，因此本文再次对属性（3）进行证明：

证明：首先根据前提假设及$[\text{Verifier}(m)]_V^{t_b,t_e}$，利用公理 *VER* 可得到：

$[\text{Verifier}(m)]_V^{t_b,t_e} \exists t_f,e,I.(t_f<t_e) \wedge \hat{I} = \hat{AIK}(m) \wedge \text{Contain}(e,SIG_{AIK(m)}^{-1})$

$\{|PCR(s), seq(BIOS(m),OSLoader(m),VMM(m),Dom0\_Kernel(m),vTPM$

$Builder(m), vTPM\text{-}VM\ Binding(m), VM\ Builder(m)))|\}) \wedge (\text{Sent}(I,e)@ t_f)$

$\vee \exists l.(\text{Write}(I,l,e)@ t_f))$

根据图 4.2 中的信任属性的远程验证，如果下列的程序不变量是成立的：对于程序前缀 $Q \in IS(\text{CRTM}_{SRTM}(m))$，有以下属性成立：

$[Q]_I^{t_b,t_e} (\forall l,e,t.(t \in t_b,t_e]) \supset \neg \text{Write}(J,l,e)@t)$

$\wedge (\forall t',e'.((t' \in t_b,t_e]) \wedge \text{Send}(I,e')@t') \supset (\exists e,t_R.(t_R<t') \wedge$

$(\text{Read}(I,m.pcr.s,e'')@t_R) \wedge e'=SIG_{AIK(m)}^{-1}) \{|PCR(s),e''|\})$

该式可以表示为：在远程验证过程中如果程序没有执行相关内存写入的操作，则数据 *e*'，*e*"的顺序一点是先读取了 *e*"然后才能进行数据 *e*'的发送，并且 *e*'是一个经过签名的值。

本文将利用推理规则 SEQ 和公理 Act1 证明上述不变量成立。利用诚实规则并进行简化后可得[26]：

$[\text{Verifier}(m)]_V^{t_b,t_e} \exists t_R,e,I.(t_R<t_e) \wedge \hat{I} = \hat{AIK}(m)$

∧Contain(e,$SIG_{AIK(m)}^{-1}$){|$PCR(s)$, $seq(BIOS(m),OSLoader(m),VMM(m),$ $Dom0\_Kernel(m)$, $vTPM\ Builder(m)$, $vTPM\text{-}VM\ Binding(m)$, $VM\ Builder(m)$)|})

∧(Read(I,$m.pcr.s$,e'') @t_R)∧e=$SIG_{AIK(vm)}^{-1}$){|$PCR(s)$,e''|}

分别利用等值公理 Eq 和 Read 公理，有

[Verifier(m)]$_V^{t_b,t_e}$ ∃t_R,e'',I.(t_R<t_e)∧Contain(e,$SIG_{AIK(m)|}^{-1}${|$SIG_{AIK(m)|}^{-1}${|$PCR(s)$,e''|},

$seq(BIOS(m),OSLoader(m),VMM(m),$ $Dom0\_Kernel(m)$, $vTPM\ Builder(m)$, $vTPM\text{-}VM\ Binding(m)$, $VM\ Builder(m)$)|})∧(Mem($m.pcr.s$,e'')@t_R) **(5)**

此时需要判定 e''的值，根据上述推理过程可知有两种可能：

(e''= $seq(BIOS(m),OSLoader(m),VMM(m),$ $Dom0\_Kernel(m),vTPM\ Builder(m),$ $vTPM\text{-}VM\ Binding(m)$, $VM\ Builder(m)$)∨ Contain(e,$SIG_{AIK(m)\text{-}1}$){|$PCR(s)$, $seq(BIOS(m),OSLoader(m),VMM(m),Dom0\_Kernel(m),$ $vTPM\ Builder(m)$, $vTPM\text{-}VM\ Binding(m)$, $VM\ Builder(m)$)|}) **(6)**

根据公理 PCRC：

├(Mem($m.pcr.s$,e'')@t) ⊃ ¬Contains(e,SIG_K{|e'|})

以及 Mem($m.pcr.s$,e'')存在的事实，可知式(14)中第 2 种可能不成立，故只有

e''= $seq(BIOS(m),OSLoader(m),VMM(m),$ $Dom0\_Kernel(m),$ $vTPM\ Builder(m),$ $vTPM\text{-}VM\ Binding(m)$, $VM\ Builder(m)$)成立。

利用等值公理 Eq 对式(14)进行变换可得

[Verifier(m)]$_V^{t_b,t_e}$ ∃t_R,.(t_R<t_e)∧(Mem($m.pcr.s$, $seq(BIOS(m),OSLoader(m),VMM(m),$ $Dom0\_ Kernel(m)$, $vTPM\ Builder(m)$, $vTPM\text{-}VM\ Binding(m)$, $VM\ Builder(m)$))@t_R)

即定理 4.2 中表达的属性式(3)可以被证明，并且继而利用属性（3）的结论和定理 4.1 直接证明（4）是成立的。

根据上述证明，可以得出，如果在向外部实体进行远程认证时，信任链的构建顺序是按照预期的构建顺序进行构建的，并且没有存在其他干扰程序和恶意代码的控制，此时的信任传递的结果一定是按照预期进行的，不存在信任的缺失。

4.3 可信衔接点 TJP 的本地验证及远程证明

本节根据 4.2 对 TVP-QT 中的相关定义和说明，对可信衔接点 TJP 的动态度量机制进行本地验证和远程证明的形式化描述。

4.3.1 本地程序执行

根据 4.2 节对 TVP-QT 中 TJP 信任属性 TP_{TJP} 定义以及 TP_{vRT} 中对 TC_{TJP} 的定义，可信衔接点 TJP 的本地执行过程中需要执行的程序如图 4.3 所示：

$Latelaunch_{DTRM}$(*vTPM-Builder*)≡*vtb*=read *m.vTPM-Builder_loc*; Extend *m.dpcr.d*,*m*;

Jump *vtb*

...

vTPM-Builder(*TJP*) ≡ *vvb* = read *m.vTPM-VM-Binding_loc;*

Extend *m.dpcr.d*,*vvb*;

Jump *vvb*

vTPM-VM Binding(*TJP*) ≡ *vvmb* = read *m.VM-Builder_loc*

Extend *m.dpcr.d*,*vvmb*;

Jump *vvmb*

VM-Builder(*TJP*) ≡ *vtpmb*= read *m.vTPM_loc*

Extend *m.dpcr.d*,*vtpmB*;

Jump *vtpmb*

vTPM(*m*) ≡ ...

图 4.3 TVP-QT 中 TJP 信任链传递

程序执行流程：首先确保 TJP 的 vTPM Builder 能正常加载。然后利用 DRTM 度量 TJP 的三个组件 vTPM Builder、vTPM-VM Binding、VM Builder，从主机内存地址中读取 vTPM Builder 的代码，将其扩展到一个 PCR 中（其中，*m.pcr.d* 表示 TJP 的度量值存储与动态度量的 PCR 中）；之后执行命令 Jump *vtb* 将控制权交给 vTPM Builder，按照上面的过程依次度量 vTPM VM Binding、VM Builder。

在此过程中，对 TJP 的动态度量必须在 m 启动之后且创建 vm 之前，否则会导致 TJP 无法按顺序正确度量。将其表示为

$\mathrm{Honest}(\mathrm{TPM}_{\mathrm{SRTM}}(m) \succ \mathrm{TJP}_{\mathrm{DRTM}}(m) \wedge \mathrm{TJP}_{\mathrm{DRTM}}(m) \succ \mathrm{vTPM}_{\mathrm{SRTM/DRTM}}(\mathrm{vm}))$。

除此之外，TVP-QT 在启动 TJP 时，系统产生的相应的进程 K 必须对 TJP 所要写入的 PCR 进行加锁，可以防止别的程序或者恶意攻击者对 PCR 的值进行修改。形式化表示为

$\mathrm{ProtectedSRTM}(m) = \forall t,K.(\mathrm{Reset}(\mathrm{m},K)@t) \supset (\mathrm{IsLocked}((m.pcr.s,m.pcr.d),K)@)t$

在本文中，TJP 的 vTPMBuilder 无论是作为一个轻量级的虚拟域或者单独的应用进程，都需被动态加载机制保证其安全可信执行，即利用 Latelaunch(vTPM Builder)确保其可信执行，即 K_{DRTM} 成立[27]。

4.3.2 本地可信属性描述

基于定义 3.2 及 TJP 度量后的 PCR 和其中的每个组件存在的唯一性、确定性映射关系可以得到 TJP 的本地信任属性：如果 vTPM Builder、vTPM-VM Binding、VM Builder 等都能按确定的先后顺序加载，则在 PCR 中观察到的度量值的顺序也是按照上述先后顺序进行扩展的。形式化表示为

$$\exists K.(t_{vtb} < t_{vvb} < t_{vvmb} < t_{vtpmb} < t)$$

$\forall I \in m.\exists t_{\text{vtb}}.\exists t_{vvb}.\exists t_{vvmb}.\exists t_{vtpmb}.$

$\text{MeasuredBoot}_{\text{DRTM}}(\text{TJP},t)=$

$\wedge(\text{Jump}(K,vTPM\ Builder(TJP)))@t_{vtb})$

$\wedge(\text{Jump}(K,vTPM\text{-}VM\ Binding(TJP))@t_{vvb})$

$\wedge(\text{Jump}(K,VM\ Builder(TJP))@t_{vvmb})$

$\wedge(\text{Jump}(K,vTPM(TJP))@t_{vtpmb})$

$\wedge(\neg\ \text{Reset}(m)\text{on}(t_{vtb},t])(\neg\ \text{Jump}(K)\text{on}(t_{vtb},t_{vvb}))$

$\wedge(\neg\ \text{Jump}(K)\text{on}(t_{vvb},t_{vvmb}))\wedge(\neg\ \text{Jump}(K)\text{on}(t_{vvmb},t_{vtpmb}))$

上述公式表示为：如果 TJP 的启动顺序是按照 vTPM Builder→vTPM-VM Binding→VM Builder 的顺序进行执行和完整性度量，则 TVP 正确完成了 TJP 部分信任链的构建过程，并且在此期间没有其他程序和代码执行。如果此描述成立，则必须证明扩展的 PCR 的值和 TJP 的执行顺序是一一对应的。根据前文的假定，则需要进行证明的 TJP 的本地信任属性如下所示：

定理 4.3 如果可信衔接点 TJP 是被成功加载和执行的，并且该 TJP 加载过程对应的 PCR 值为 *seq*(*vTPM Builder*(*TJP*),*vTPM-VM Binding*(*TJP*),*VM Builder*(*TJP*))，那么 TJP 的本地信任链传递过程就是可以被唯一确定的，不存在其他程序的加载。即确定地从 vTPM Builder(*TJP*)到 vTPM-VM Binding(*TJP*)再到 VM Builder(*TJP*)。该信任属性形式化表示为

$\text{ProtectedDRTM(TJP)}+$

$\text{Mem}(m.dpcr.d,seq(vTPM\ Builder(TJP),$

$vTPM\text{-}VM\ Binding(TJP)),VM\ Builder(TJP))$

$\supset \text{MeasuredBoot}_{\text{DRTM}}(\text{TJP},t)$

证明过程类似 m 的信任链本地可信属性的证明，在此不再叙述。

4.3.3 信任链远程验证

类似主机 m 的远程验证过程，TVP-QT 的 TJP 部分也应向外部实体证明自己的信任链构建过程是安全预期顺序进行构架的，并且对相对应的 PCR 的值也是预期的度量值和序列，如果得到正确的远程验证，则需要证明 $\text{MeasuredBoot}_{\text{DRTM}}(\text{TJP},t)$ 是成立的。本文从远程程序执行和远程信任属性进行描述。

（1）远程验证程序执行

TJP 涉及到的远程验证程序如图 4.4 所示。

$TPM_{DRTM}(TJP)$ $\equiv w = \text{read } m.pcr.d;$

$r = \text{sign}(PCR(s),w),AIK^{-1}(m);$

$\text{send } r$

$Verifier(TJP)$ $\equiv sig = \text{recieve};$

v = verify sig, $AIK(m)$;
match v ,($PCR(s)$,
seq(*vTPM Builder*(*TJP*),*vTPM-VM Binding*(*TJP*),*VM Builder*(*TJP*))

图 4.4 TVP-QT 中 m 信任传递的远程验证程序

首先，m 读取本地 TJP 的 PCR 值，用 AIK 签名（$AIK^{-1}(m)$）并将其发送给挑战者。然后，挑战者验证该签名，并用预期的度量值序列与收到的值进行对比，如果匹配，则表明该主机 m 的 TJP 拥有所声称的可信属性，否则验证失败。同样的，为防止远程验证的合理性，外部实体和 TJP 也应是不同的实体。

这些前提条件形式化表示为

Γ_{DRTM} = {Honest(AIK(m)), $\hat{V} \neq AIK(m)$}

（2）信任链属性的远程验证

根据相应的远程验证的执行流程，TJP 远程证明的信任属性的验证目标可表示为定理 4.4。

定理 4.4 如果 PCR 中存储 TJP 度量值得序列是 seq(*vTPM Builder*(*TJP*),*vTPM-VM Binding*(*TJP*),*VM Builder*(*TJP*))，则 TJP 的信任构建和完整性度量一定是安全预期顺序进行执行的。

形式化表示为：

$\Gamma_{DRTM} \vdash [\text{Verifier}(m)]_V^{t_B,t_E} \exists t.(t<t_e) \wedge$

(Mem(*m.pcr.d*,*seq*(*vTPM Builder*(*TJP*),

vTPM-VMBinding(*TJP*),*VM Builder*(*TJP*)))@t)

Γ_{DRTM}，$\text{Protected}_{SRTM}(m)$

$\vdash [\text{Verifier}(m)]_V^{t_B,t_E} \exists t.(t<t_E) \wedge$

$\text{MeasureBoot}_{DRTM}(m,t)$

证明过程类似 m 的信任链远程验证的证明，在此不再叙述。

4.4 本章小结

本章内容主要是利用安全系统逻辑形式化分析方法对 TVP-QT 信任链模型进行形式化分析。首先，介绍了主机 m 信任链构建本地执行的形式化表示，从程序执行的角度证明了主机 m 信任链模型的有效性；并且介绍了主机 m 的信任属性表示，即如果 PCR 中度量值的序列是按照主机 m 程序加载的进行扩展的，则主机 m 的信任链构建一定是按照预期的顺序进行的；然后结合远程验证，证明了从外部实体的角度上看，主机 m 的执行顺序也是可以信任的。然后用相同方法介绍了可信衔接点的本地验证和远程验证。

5 基于扩展无干扰理论的信任链分析方法

本文针对第 1 章中提到的目前无干扰理论中没有考虑到云计算运行中时的安全域、动作所属主体以及动作对安全域和系统状态的影响进行详细的说明的问题，对目前无干扰理论中的基本定义进行扩展并定义非传递无干扰安全判定定理。

5.1 扩展无干扰理论基本假定及定义

遵从 Rushby 等人的描述，本文以自动机（或叫状态机）的形式，并在安全域、动作等对无干扰进行扩充，在动作主体、动作主体对安全域和系统状态的影响扩展到无干扰理论中，给出无干扰的基本定义如下。

定义 5.1 系统 M 由如下要素构成：

（1）一个包含唯一初始状态 s_0 的状态集 S。约定使用 $...s,t...$ 等来表示系统 M 的不同状态（处于不同时间或者经过动作执行后）；

（2）一个由系统中可能对系统状态产生影响的所有不可再分的原子动作集合 A，并且约定用 $a, b, c...$ 表示单独的原子动作；

（3）一个由系统中所有可以进行动作执行的动作所属的主体集合 O，主体和动作之间并不是一一映射的关系，因为相同的动作可能由不同的动作主体发出，而一个特定的主体可以发出不同的动作。即每一个属于 A 中的动作都有一个包含于 O 中的主体集。即动作 $a \in A$ 的主体集 $O_a \in O$；

（4）一个由系统中所有由特定动作集组成的行为序列集 B。行为一般表示为几个动作的组成，行为类似于程序的存在，可以包含不同的进程。在本文中约定用希腊字母 α，β，$\gamma...$ 等表示行为。一个行为可以表示为：$\alpha = a_0 \circ ... \circ a_m \circ ... \circ a_n \in A^*$，其中$\circ$是连接符；

（5）系统经过动作 $a \in A$ 之后观察到的系统的输出集合( )；

（6）系统发出的每一个原子动作，都在当前状态下可以找到自身所属的，并且不可以被再分的 Min 安全域，并且这些原子动作的主体也属于此安全域。本文将这些安全域集表示为 Min 安全域 MD；MD 的动作主体集合中可以产生一个或者多个动作与系统产生信息交互。并且，安全域的之间的操作可以通过相关安全策略进行信息的隔离，彼此之间只存在干扰的信息流进行交互。并且根据云计算环境运行机制，安全域集 MD 的某一子集也可能单独成为运行的一个组合安全域，这些由 MD 子集组成的组合安全域称为 CD（$Combination-Domain$），且 CD 中任意两个元素可能存在交集。即对于 $\forall CD_i \in CD, \forall CD_j \in CD$，有 $CD_i \cap CD_j = \phi$。且对应与系统状态，域也有相应的状态，用 DS 表示域状态集合，约定使用 ds_s^u, ds_t^u 分别表示安全域 u 的状态在系统状态为 s,t 本安全域状态，显然有

$S=\ldots ds_s^u \cap ds_t^u \ldots \cap ds_s^v \cap ds_t^v \ldots$。

（7）系统安全策略~>和$\not\sim>$。安全域相互之间可以存在特定的干扰信息流，具有无干扰关系的信息量彼此之间不能影响。~>和$\not\sim>$分别称为信息流的干扰和无干扰关系，一旦两个信息流存在干扰关系，则无干扰就不复存在；

（8）动作主体到动作的映射函数：own：$A \to O$。own返回一个特点动作a所属的动作主体$own(a)$；

（9）安全域MD到动作的映射函数：$dom: A \to MD$。dom返回一个特定动作a所属的 Min 安全域$dom(a)$，并且必然$\exists CD_i \in CD$，使得$dom(a) \subset CD_i$。

（10）单步系统状态函数：$step: \frac{S \times A}{MD \times O} \to S$。.单步函数描述的是系从前一个状态，由某一安全域$MD$某一动作主体执行某个动作之后，应该到达的后一个状态；在系统s存在安全域u中的主体O_1，执行动作a之后，对系统的改变为$step(\frac{s,a}{u,o_1})$。

同理，动作对动作所属安全域的改变单步域状态函数为$stepd: \frac{ds \times A}{MD \times O} \to ds$，在安全域$u$在$ds_s^u$下的主体$O_1$，执行动作$a$之后，对安全域$u$改变为$stepd(\frac{ds_s^u, a}{u, o_1})$；

且单步状态函数满足以下条件：

$\forall a, if: dom(a)$变化，$own(a)$变化，$step(\frac{s,a}{u,o}) = step(\frac{s,a}{dom(a),own(a)})$也会变化。

（11）行为结果函数：$behcon: S \times A \to OP$。行为结果函数：$behcon\ (behavior\ .consequenc\ e)$给出了：在状态$s \in S$使用特定的动作$a \in A$所观察到的系统的输出集合；

（12）行为执行函数：$exec: S \times B \to S$。如果用 Λ 表示空动作序列，则$exec$可以表示为：

对于系统状态：$\begin{cases} exec(s, \Lambda) = s \\ exec(s, a \circ \alpha) = exec(step(\frac{s,a}{u,o}), \alpha), \exists u = dom(a), \exists o = own(a) \end{cases}$

对于安全域状态：$\begin{cases} exec(ds, \Lambda) = ds \\ exec(ds_s^u, a \circ \alpha) = exec(stepd(\frac{ds_s^u, a}{u,o}), \alpha), \exists u = dom(a), \exists o = own(a) \end{cases}$

需要强调的是，无干扰模型将输入定义为行为，也就是原子动作的连接序列。本文遵从原有的无干扰理论，对系统进行描述。

定义 5 .2 系统视图。

系统视图表现在机器 M 的各个属性，比如 M 的内存状态，存储状态，CPU 状态等存储单元，每个存储单元都会在系统的某个时刻存在着不同的值。并且可以通过观察和修改等操作单元对其进行修改。

（1）系统的存储单元集合 N，该存储单元集合及其取值构成了系统在特定时刻下的系统状态；

（2）系统的视图值集 V。集合 N 中的每一个存储单元 $n \in N$ 在系统的状态 $s \in S$ 都会有一个特定的可以表示系统目前状态的值 $value \in V$。本文定义内容观察等函数表示目前状态的值；

（3）系统视图观察函数 $contents : S \times V \to V$。

定义 5.3 域视图。

域视图对应于 M 中的安全域 MD，是 M 视图的子集。

对应于系统状态等相关内容函数，域视图有以下表示：

（1）安全域视图集合 DS 和对弈的视图值集合 DV。安全域 u 中的存储单元在安全域的特定状态都会有一个特定的视图值集合；

（2）域视图内容函数 $contentsd : DS \times DV \to DV$。

（3）域视图观察函数 $observed : MD \to P(N)$ 和域视图修改函数 $alterd : MD \to P(N)$。域视图的观察函数和修改函数分别给出了安全域自身所拥有的存储单元的集合和可以进行修改操作的存储单元值的集合。

定义 5.4 主体视图。

（1）在整个系统 M 中，域中有很多可以发出动作的主体，即

$\forall g \in O, \exists u \subset MD$，使得 $g \in u.$。

亦有 $\forall a \in A,$ 有 $own(a) \in dom(a).$。

仅次于安全域的单位为动作主体，动作主体对应系统中的存储单元在域不同状态下，由动作发出后，其存储单元也会改变。

主体视图值集 OV。动作主体 g 每一个存储单元 $n \in N$ 在特定的状态下 $ds_s^u \in DS$ 都会有一个特定的值 $value \in OV^g$，其中有 $g \in u$。、

（2）主体视图内容函数 $contentso : DS \times OV \to OV$。

（3）主体视图观察函数 $observeo : O \to P(N)$ 和主体视图函数 $altero : O \to P(N)$。不同的主体能观察和修改的存储单元的集合不同。

定义 5.5 如果存在等价关系 $\overset{u}{\sim}$ 则系统必须同时满足输出一致性和单步一致性，只有这样在系统运行时才会两个不同的域或者主体是等价的。

（1）输出一致性：

在系统视图上：$s \overset{u}{\sim} t \supset behcon\ (s,a) = behcon\ (t,a)$。

并且由 $ds_s^u \in s, ds_t^u \in t,$ 可得域视图的输出一致性：

$ds_s^u \overset{u}{\sim} ds_t^u \supset behcon\ (ds_s^u, a) = behcon\ (ds_t^u, a)$ 。

（2）系统单步一致性和弱单步一致性：

对于云计算环境下的传递安全策略，必须满足以下描述的单步一致性：

$dom(a) \sim> u \wedge s \overset{u}{\sim} \supset step(\frac{s,a}{u,o}) \overset{u}{\sim} step(\frac{t,a}{u,o})$ 。

而对于非传递安全策略，则需要满足以下的弱单步一致性：

$dom(a) \sim> u \wedge s \overset{dom(a)}{\sim} t \wedge s \overset{u}{\sim} \supset step(\frac{s,a}{u,o}) \overset{u}{\sim} step(\frac{t,a}{u,o})$ 。

从单步一致性和弱单步一致性可看出，其实弱单步一致性仅仅增加了一个很重要的条件 $s \overset{dom(a)}{\sim} t$。因为，弱单步一致性是在满足单步一致性的前提下的安全策略。

定义 5.6 云计算环境下的间接干扰关系 $\overset{\approx}{\sim>}$。

对于云计算环境下的非传递安全策略 $u \sim> v_1 \wedge v_1 \sim> v_2 \wedge ... \wedge v_n \sim> w \wedge u \not\sim> w$ 而言，系统中的安全域 u 虽然不能直对 w 产生干扰，，但是，仍然可以间接对进行干扰(因为 $u \sim> v_1 \wedge v_1 \sim> v_2 \wedge ... \wedge v_n \sim> w$)。因此本文在定义 5.1 中的可以被当做直接干扰关系的基础上对间接干扰关系进行定义。间接干扰关系“ $\overset{\approx}{\sim>}$ ”定义如下：

$u \sim> w$：*iff*

定义 5.7 干扰源集 $interfsrcs : B \times D \to \mathrm{P}(D)$。

在系统中针对某个安全域存在干扰的有直接干扰关系和间接干扰关系的因素可以表示为一个集合，本文称作干扰源集，将其递归定义如下:

$interfsrcs(\Lambda, w) = \{w\}$，且

$interfsrcs(a \circ \alpha, w) =$

$$\begin{cases} \vartheta_1; & if \exists v.v \in interfsrcs(\alpha, w) \supset dom(a) \sim> v \\ \vartheta_2; & if \exists v.v \in interfsrcs(\alpha, w) \supset dom(a) \sim> v \end{cases}$$

。其中 $\vartheta_1 = \{dom(a)\} \cup interfsrcs(\alpha, w)$， $\vartheta_2 = interfsrcs(\alpha, w)$。

定义 5.8 弱预期函数 $wexpected : B \times D \to B$。

云计算环境下的弱预期函数 *wexpected* 可以被表述为：对安全域中无论是存在直接干扰关系还是间接干扰关系的动作都进行保留，并将不存在干扰关系的动作进行剔除，从而可以判断出在非传递无干扰安全策略下的系统的整体的预期行为。由于信任链的构建从链式构建的角度上看，前后进行度量的组件才会存在干扰关系，所以弱预期函数可以从信任链的构建机制上进行验证[58]。

$wexpected(\Lambda, w) = \Lambda$，且

$$\int B_1; \quad if : dom(a) \in interfsrcs(a \circ \alpha, w)$$ 。

其中， $B_1 = a \circ wexpected(\alpha, w)$ ，

$B_2 = wexpected(\alpha, w) = \Lambda \circ wexpected(\alpha, w)$ 。

定义 5.9 域集等价关系 $\overset{C}{\approx}$：$s \overset{C}{\approx} t$ _ iff$\forall u \in C.s \overset{u}{\sim} t$ 。

定理 5.1 云计算环境下系统需要满足的非传递无干扰策略的判定定理。

存在一个在视图角度上存在隔离的系统 M，如果 M 的安全域之间存在非传递的干扰关系，并且同时满足系统弱单步一致性、局部干扰性、输出一致性时，则该系统满足非传递无干扰策略[58]。

5.2 TVP-QT 信任链传递形式化描述

本文基于扩展后的无干扰理论，对 TVP-QT 信任链模型进行形式化描述。由于扩展后的无干扰理论从系统主图、域视图、主体行为视图由一个整体系统向系统内的主体进行细分，本文针对 TVP-QT 信任链传递模型，从以下方面对信任链进行形式化描述。

定义 5.10 信任关系可以用以下二元关系表示：$\rightarrow \in MD \times MD$ （针对于安全域 MD 级），$\rightarrow \in CD \times CD$ （针对于组合安全域级），$\rightarrow \in O \times O$ （针对于动作主体级）。$\exists u, v \in MD$ ，若 $u \rightarrow v$ ，则表示域 u 对域 v 进行了可信度量，并且度量成功，域 u 对域 v 是信任的，然后进行控制权的转移。

首先，按照 Min 安全域 MD 的信任传递方式，可以表示以下式子：

$MD_0 \rightarrow MD_1 \rightarrow \ldots \rightarrow u \rightarrow \ldots v \rightarrow \ldots MD_{n-1} \rightarrow MD_n . u, v, MD_i \in MD$

其中 MD_0 表示为可信虚拟平台的可信度量根：即图 3.1 中 TPM 下的 CRTM。

按照组合安全域 CD 的信任链传递方式，可以表示为以下公式：

$CD_1 \rightarrow CD_1 \rightarrow \ldots \rightarrow \ldots \rightarrow CD_{n-1} \rightarrow CD_n$

对基于加载前进行完整性度量的 TVP-QT 的信任链模型的形式化描述为：

$digest(MD_i, MD_j) = expect(MD_j) \Rightarrow MD_i \rightarrow MD_j \Rightarrow controlTrans(MD_i \rightarrow MD_j)$ ；

上述公式说明：安全域 MD_i 和 MD_j 进行控制权转移和信任传递的前提是，MD_i 将获得的 MD_j 摘要值（实际中可以存储在 PCR 中）与预期的摘要值值 $expect(MD_j)$ 进行相等的计算操作，如果是相等的则 MD_i 将信任传递给 MD_j ，同时将控制权转移给 MD_j 。其中，两个安全域 A 对 B 进行摘要值计算的函数表示为：digest(A,B)，对 A 的完整性摘要值进行预期值获取的函数可以表示为 expect(A)，$controlTrans(A \rightarrow B)$ 表示控制权从 A 转移到 B。

同理对应组合域有：

$digest(CD_i, CD_j) = expect(CD_j) \Rightarrow CD_i \rightarrow CD_j \Rightarrow controlTrans(CD_i \rightarrow CD_j)$ 。

5.3 扩展无干扰信任传递判定定理

可信虚拟平台在运行过程中，安全域之间信息流相互之间可以通过特定的安全机制和安全机制进行判断，并且在进行信任链构建时，只有预期的程序组件才会存在直接干扰关系，这样通过完整性度量的信任链机制才会是有效的[58]。本文通过给出云计算环境下的扩展非传递无干扰信任传递判定定理，判断在云计算环境下构建的信任链模型是否有效。

下面本文分别从系统、安全域（*MD* 和 *CD*）、动作主体三个角度对本文提出的信任链模型应该满足的安全策略进行描述和分析验证。其中，从系统角度描述系统符合的安全策略，则需要对安全域的输入和系统状态的论述。同理，从安全域角度描述安全策略，则需要对动作主体的输入和对系统状态的进行论述。

定理 5.2 TVP-QT 系统需要满足非传递无干扰关系的判定定理。

（1）系统的域满足输出一致性。即一个内部操作动作造成的输出影响只依赖于发出动作域的系统视图。且满足以下条件：

$$\forall n \in observe(u).contents(s,n) = contentso(s,n) \rightarrow s \overset{u}{\sim} t$$

（2）相对于（1），系统的动作主体满足输出一致性。即一个安全域内的操作动作造成的输出影响只依赖于发出动作动作主体的系统视图。

$$\forall n \in observeo(o_1).contentso(ds_s^u, n) = contentso(ds_t^u, n) \rightarrow ds_s^u \overset{u}{\sim} ds_t^u \rightarrow s \overset{u}{\sim} t;$$

$$\exists o_1, own(o_1) = u \subset MD$$

（3）系统中发生的一个动作造成的对系统状态影响只与发出该动作的域的上一状态系统视图相关联。

$$\exists u \in MD, dom(a) = u;$$

$$\exists o_1 \in O, own(a) = o_1.$$

（4）相对于（3），系统中发生的一个动作造成的对安全域状态影响只与发出该动作的动作主体的上一状态系统视图相关联。

$$ds_s^u \overset{own(a)}{\sim} ds_t^u \wedge contentsd(step(\frac{ds_s^u, a}{dom(a), own(a)})) \neq$$

$$contentsd(ds_s^u, n) \vee contentsd(step(\frac{ds_t^u, a}{dom(a), own(a)})) \neq contentsd(ds_t^u, n) \rightarrow$$

$$contentsd(step(\frac{ds_s^u, a}{dom(a), own(a)})) = contentsd(step(\frac{ds_t^u, a}{dom(a), own(a)}))$$

（5）系统中，如果一个动作改变了其动作主体的值，则发出该动作的主体一定可以写、访问该主体的系统视图，并且可以写、访问该动作所在的域的系统状态。

$\exists u \in MD, dom(a) = u;$

$\exists o_1 \in O, own(a) = o_1.$

$$contentso(step(\frac{ds_s^u, a}{u, own(a)})) \neq contentso(ds_s^u, n) \rightarrow$$

$$contentsd(step(\frac{ds_s^u, a}{u, own(a)})) \neq contentsd(ds_s^u, n) \rightarrow$$

$$contents(step(\frac{s, a}{u, own(a)})) \neq contents(s, n) \rightarrow$$

$$n \in altero(own(a), ds_s^u) \rightarrow n \in alterd(own(a), ds_s^u)$$

$$\rightarrow n \in alter(u, s) = alter(dom(a), s)$$

系统内对系统内的系统状态的操作有以下关系。

$\exists n \in N, n \in alter(u, s) \wedge n \in observe(v, s) \rightarrow u \sim > v.$

（6）系统内若发出某一动作，则该动作的动作主体，所属域唯一确定的。

$\forall a$，在系统状态 s，必然存在 $dom(a)$，$own(a)$，可唯一确定 a。

证明：

根据本文扩展无干扰给出的定理 5.1，系统需要满足输出一致性，单步一致性即可。

（1）证明输出一致性：

需要证明：

$$ds_s^u \overset{o}{\sim} ds_t^u \wedge ds_s^u \overset{own(a)}{\sim} ds_s^u \rightarrow step(\frac{ds_s^u, a}{dom(a), own(a)}) \overset{o}{\sim} step(\frac{ds_t^u, a}{dom(a), own(a)}) \quad (1)$$

$$s \overset{u}{\sim} t \wedge s \overset{dom(a)}{\sim} t \rightarrow step(\frac{s, a}{dom(a), own(a)}) \overset{u}{\sim} step(\frac{t, a}{dom(a), own(a)}) \quad (2)$$

式（2）是在域状态下的单步一致性，是在式（1）动作主体在域状态的延伸，本文在此只证明式（1）。

$\forall n \in OV$，存在下式：

$$ds_s^u \overset{o}{\sim} ds_t^u \wedge ds_s^u \overset{own(a)}{\sim} ds_s^u \rightarrow$$

$$contentso(step(\frac{ds_s^u, a}{dom(a), own(a)}), n) \overset{o}{\sim} contentso(step(\frac{ds_t^u, a}{dom(a), own(a)}), n)$$

若 $contentso(step(\frac{ds_s^u, a}{dom(a), own(a)}), n) \neq contentso(ds_s^u, a)$，由定理 5.2 条件（2）可得：

$$contentso(step(\frac{ds_s^u, a}{dom(a), own(a)}), n) \neq contentso(step(\frac{ds_t^u, a}{dom(a), own(a)}), n)$$；

若 $contentso(step(\frac{ds_t^u,a}{dom(a),own(a)}),n) \neq contentso(ds_t^u,a)$，由定理 5.2 条件（2）可得：

$$contentso(step(\frac{ds_s^u,a}{dom(a),own(a)}),n) \neq contentso(step(\frac{ds_t^u,a}{dom(a),own(a)}),n)$$，

其他情况下：$$\begin{array}{l} contentso(step(\frac{ds_s^u,a}{dom(a),own(a)}),n) = contentso(ds_s^u,n) \wedge \\ contentso(step(\frac{ds_t^u,a}{dom(a),own(a)}),n) = contentso(ds_t^u,n) \end{array}$$；

又由 $ds_s^u \overset{own(a)}{\sim} ds_t^u \rightarrow contentso(ds_s^u,n) = contentso(ds_t^u,n)$ 也可得

$$contentso(step(\frac{ds_s^u,a}{dom(a),own(a)}),n) \neq contentso(step(\frac{ds_t^u,a}{dom(a),own(a)}),n)$$

（2）证明单步一致性：

即证明：$dom(a) \not\sim u \rightarrow s \overset{u}{\sim} step(s,a)$

亦可从其逆否命题证明：

$\exists n \in V$，$contents(s,n) \neq contents(step(\frac{s,a}{dom(a),own(a)}),n) \rightarrow dom(a) \sim u$。

由定理 5.2 条件（3）可得

$$contents(step(\frac{s,a}{dom(a),own(a)}),n) \neq contents(s,n) \rightarrow n \in alter(dom(a),s)$$，

由定理 5.2，可得 $dom(a) \sim u$ 成立。

5.4 基于 Xen 的 TVP-QT 系统仿真及验证

本文基于开源的系统 Xen，结合本文第 3 章介绍的 TVP-QT 架构，对满足扩展无干扰的安全系统进行了仿真，具体的仿真实验可以结合本文第三章的第 3 节和第 4 节的基于 Xen 的实例系统和仿真实验，在此不再重复叙述。此实例系统利用虚拟机监视提供的资源隔离机制实现了满足扩展无干扰的安全系统。在该系统中，运行的安全域必须通过 VMM 才可以进行信息交流，所以每个安全域彼此之间是不能直接进行信息交流的。

如图 3.1 所示，从 TVP-QT 第一层到系统最上层存在着很多不同的组合安全域，比如 TPM 域、VMM 域等，并且针对系统中的可信衔接点 CJP 的不同组件是存在于不同的组合安全域中。由 CJP 存在域不同的安全域中可知本文对安全域 *MD* 和组合安全域 *CD* 的划分是正确的。并且 CJP 中的某一组件若发出动作，可能存在与

Dom0 域也有可能存在 vTPM Manager 域中。并且 Dom0 中的组件主体发出的某一动作可能会与信任链信息流传递时与 CJP 中发出的某一动作相同，但是在对信任链是无干扰的，验证了在系统上对动作发出主体的定义的有效性。

由于 TVP-QT 系统之上可以并行运行虚拟机，根据信息流的不同情况，本文对实际系统的无干扰验证分为两种情况进行验证。

（1）存在一个用户虚拟机

在此种情况下虚拟机运行在 VMM 之上，VMM 系统上的设备驱动模型由运行于虚拟机上的前端驱动和运行于 VMM 之上的后端驱动组成，实现对硬件功能的使用。该用户虚拟机在运行过程中产生的信息流可以从图中的顺序进行传递。若使该虚拟机信任链有效，必须确保系统中存在非预期的干扰。

根据定理 5.2，基于 Xen 的 TVP-QT 系统的 I/O 设备是满足定理 5.1 的要求的；验证描述如下：

由输出一致性的定义可知，该虚拟机是在云计算环境下进行资源隔离的，并且其使用的资源都是经过 VMM 分配的，无其他虚拟机对该虚拟机产生影响，虚拟机的各个组件在信任传递过程中，经过多个组件之后，系统的状态都是一致的。由局部干扰下可知，只存在一个虚拟机的情况下，只有 VMM 才可以对虚拟机的信息流产生影响，除虚拟机监视器之外的其他系统组件是对虚拟机不可见的。由弱单步一致性可知，该虚拟机在运行过程中的动作只会对此虚拟机系统进行状态的改变。

（2）存在多个用户虚拟机

该情况下，在存在一个用户虚拟机之上在运行多个虚拟机，多个虚拟机需要共享设备资源，仅仅对在存在一个用户虚拟机下增加了虚拟机之间的信息流的无干扰情况。因此本文对多个虚拟机在系统中共享资源的同时进行无干扰的验证。

由输出一致性可知，VMM 在资源分配时，都是会对每个虚拟机进行属性标识的，在虚拟机使用资源时，不存在资源混乱的情况，比如某个虚拟机会使用其他虚拟机的虚拟存储，这样可以保证系统中的输出信息可以定位到某一个虚拟机，此虚拟机的不同状态下产生的操作不干扰其他虚拟机的运行。由局部干扰性定义可知，存在多个虚拟机的情况下，VMM 能够对虚拟机之间彼此的访问进行信息的处理，虚拟机之间不可能不通过 VMM 而进行资源的访问，这是有 VMM 的资源隔离特性所决定的。因此虚拟机彼此之间是无法直接产生干扰关系的，所以虚拟机内部的组件进行构建时，只有内部存在直接干扰关系的动作，而间接干扰关系可以通过 VMM 进行隔离，保证了单个虚拟机组件的信任传递。由弱单步一致性定义可知，对于若干虚拟机共享的客体对象，VMM 系统必须具有同步保护机制以防止不同虚拟机对该资源的竞争[58]。

综上，依据本文给出的 TVP-QT 信任传递模型，经过完整性验证，系统运行能够达到可信目标。

5.5 本章小结

本章按照云计算环境运行特征，对原有无干扰理论中的安全域、动作等定义进行扩充，并将动作主体和动作对安全域以及系统状态的影响等扩展到无干扰理论中；其次，针对本文提出的TVP-QT，利用扩展无干扰理论进行形式化分析；然后给出了云计算环境下信任链构建过程中需要满足的扩展无干扰信任判定定理，只有系统满足了这些判定定理，才可以有效的进行信任链的构建。最后基于Xen的TVP-QT的分析和验证证明了，TVP-QT是满足扩展无干扰安全判定定理的。

6 总结与展望

6.1 工作总结

本文对具有瀑布特征的可信虚拟平台及其信任链模型、信任链形式化分析方法进行研究。针对目前可信虚拟平台逻辑不合理、设计粒度过粗的问题，提出了一种具有瀑布特征的可信虚拟平台架构，该可信虚拟平台架构在层次上添加了可信衔接点层次，主要由虚拟机构建模块、虚拟可信平台模块构建模块、虚拟机和其虚拟可信平台模块绑定模块组成。当可信虚拟平台启动时，不仅可以以静态度量方式参与底层虚拟化平台的启动，也可以和虚拟可信模块共同作为虚拟机启动动态度量的虚拟可信根。并构建了其上的信任链模型。该模型以硬件 TPM 为起点，在底层虚拟化平台和顶层用户虚拟机信任链之间加入可信衔接点。当信任链从底层虚拟化平台传递到可信衔接点时，由可信衔接点负责对用户虚拟机的 vTPM 进行度量，之后将控制权交给 vTPM，由 vTPM 负责对用户虚拟机启动的组件及应用进行度量。该模型中可信衔接点具有承上启下的瀑布特征，能满足虚拟化环境的层次性和动态性特征，保证了整个可信虚拟平台的可信性。基于 Xen 的信任链构建的实验结果表明本信任链传递方法可以保证可信虚拟化环境在整个运行过程是安全可信的，基于安全系统逻辑形式化方法进行该信任链进行形式化分析，证明了该信任链的安全性。

此外基于扩展的无干扰理论形式化方法进行信任链形式化分析，针对目前的非传递无干扰理论均没有考虑到云计算运行中时的安全域、动作所属主体以及动作对安全域和系统状态的影响进行详细的说明，对无干扰理论在安全域动作所属主体等进行详细的扩展，并定义了云计算环境下的非传递无干扰安全定理，结合上述可信虚拟平台和信任链对扩展进行了实例验证。

6.2 研究展望

本文对云计算环境下的可信虚拟平台及其信任链模型、形式化分析方法进行了略微的工作。但是本文的可信虚拟平台架构仅考虑到虚拟机作为服务的方式，目前的云计算出现了很多新型虚拟化技术，比如：容器技术、Unikernel等新型的虚拟化技术。如何针对这些新型的虚拟化技术进行可信平台构建也是下一步的研究方向。并且本文在信任链构建过程中也是针对单个或少数虚拟机同时启动的情况下，没有对云计算环境中存在很多虚拟机的情况进行信任链构建进行分析；并且没有对在信任链构建过程中如果存在虚拟机迁移的情况进行分析。因此针对虚拟机迁移及分布式虚拟机信任链构建机制也是下一步的研究方向。

此外，在基于扩展无干扰的形式分析过程中，本文选择了本文设计的可信虚拟平台及其信任链进行分析，如何针对其他及传统的信任链进行形式化分析也是可以研究的工作。

参考文献

[1]National Institute of Standards and Technology | NIST [EB/OL]. NIST. [2018-03-10].https://www.nist.gov/.

[2]柯文浚, 董碧丹, 高洋. 基于 Xen 的虚拟化访问控制研究综述[J]. 计算机科学, 2017, 44 (s1):24-28.

[3]石源, 张焕国, 赵波, 等. 基于SGX的虚拟机动态迁移安全增强方法[J]. 通信学报, 2017,38(9):65-75.

[4]林闯, 苏文博, 孟坤, 等. 云计算安全: 架构、机制与模型评价[J]. 计算机学报, 2013, 09:1765-1784.

[5]俞能海, 郝卓, 徐甲甲, 等. 云安全研究进展综述[J]. 电子学报, 2013, 02:371-381.

[6]Ali M, Khan S U, Vasilakos A V.Security in cloud computing :opportunities and challenges[J]. Information Science, 2015, 305:357-383.

[7]Zhao D, Mohamed M, Ludwig H. Locality-aware Scheduling for Containers in Cloud Computing[J]. IEEE Transactions on Cloud Computing, 2018,PP(99): 1-1.

[8]Kumar P R, Raj P H, Jelciana P. Exploring Data Security Issues and Solutions in Cloud Computing[J]. Procedia Computer Science, 2018,125:691-697.

[9]沈昌祥, 张大伟, 刘吉强, 等. 可信3.0战略: 可信计算的革命性演变[J]. 中国工程科学, 2016, 18(6):53-57.

[10]余发江, 陈列, 张焕国. 虚拟可信平台模块动态信任扩展方法[J]. 软件学报, 2017, 28(10):2782-2796.

[11]谭良, 徐志伟. 基于可信计算平台的信任链传递研究进展[J]. 计算机科学, 2008, 10:15-18.

[12]中国信通院-研究成果-权威发布-专题报告[EB/OL]. [2017-07-10]. 中国信息通信研究院. http://www.caict.ac.cn/kxyj/qwfb/ztbg/201709/t20170919_2208939.htm.

[13]工业和信息化部关于印发《云计算发展三年行动计划（2017-2019 年）》的通知[EB/OL].[2017-04-10]. 中华人民共和国工业和信息化部. http://www.miit.gov.cn/n1146295/n1146592/n3917132/n4062056/c5570298/content.html.

[14]McAfee: 2017年全球云计算安全报告[EB/OL]. [2017]. 中国云计算社区. http://www.chinacloud.cn/show.aspx?id=25993&cid=29.

[15]徐明迪, 张焕国, 张帆, 杨连嘉. 可信系统信任链研究综述[J]. 电子学报, 2014, 10:2024-2031.

[16]BERGER S, CACERES R, GOLDMAN K A, et al. VTPM: virtualiz-ing the trusted

platform module[C]// Proc of the 15th USENIX Security Symposium. Berkeley, USA, 2006. 305-320.

[17]Xensource, Xen Open-Source Hypervisor[EB/OL]. [2018-03-12]. Xensource. https://www.citrix.com/downloads/xenserver/.

[18]Data Storage, Converged, Cloud Computing, Data Protection | Dell EMC US[2018-03-10]. Dell Inc. https://www.dellemc.com/en-us/index.htm..

[19]Microsoft - Official Home Page [EB/OL]. [2018-03-10]. Microsoft 2018. https://www.microsoft.com/zh-cn/.

[20]Garfinkel T, Pfaff B, Chow J, et al. Terra: a virtual machine-based platform for trusted computing[C]// Nineteenth ACM Symposium on Operating Systems Principles. ACM, 2003:193-206.

[21]B. PFITZMANN, J. RIORDAN, C. STUBLE,et al. "The PERSEUS system architecture", Technical Report RZ 3335 (#93381), IBM Research Division, Zurich Laboratory, 2001.

[22]CHRIS I D, DAVID P, WOLFGANG W, et al. Trusted virtual platforms: a key enabler for converged client devices[C]// Proc of the ACM SIGOPS Operating Systems Review. New York, USA, 2009.36-43.

[23]BERGER S, RAMON C, DIMITRIOS P, et al. TVDc: managing security in the trusted virtual datacenter[C]. Proc of ACM SIGOPS Operating Systems Review. New York, USA, 2008.40-47.

[24]KRAUTHEIM F J, DHANANJAV S P, ALAN T S. Introducing the trusted virtual environment module: a new mechanism for rooting trust in cloud computing[C]// Proc of the 3rd International Conference on Trust and Trustworthy Computing. 2010:211-227.

[25]Zhang Lei, Chen Xingshu, Liu Liang, Jin Xin.Trusted domain hierarchical model based on noninterference theory[J]. The Journal of China Universities of Posts and Telecommunications. August 2015, 22(4):7-16.

[26]王丽娜, 高汉军, 余荣威, 等. 基于信任扩展的可信虚拟执行环境构建方法研究[J]. 通信学报, 2011, 32(9):1-8.

[27]常德显, 冯登国, 秦宇, 张倩颖. 基于扩展 LS2 的可信虚拟平台信任链分析[J]. 通信学报, 2013, 34(5):31-41.

[28]Yu, Z., Zhang, W. & Dai, H. J A Trusted Architecture for Virtual Machines on Cloud Servers with Trusted Platform Module and Certificate Authority[J]. Journal of Signal Processing Systems, 2017, 86(2-3):327-336.

[29]池亚平, 李欣, 王艳, 王慧丽. 基于 KVM 的可信虚拟化平台设计与实现[J]. 计算机工程与设计, 2016, (06):1451-1455.

[30]李海威, 范博, 李文锋. 一种可信虚拟平台构建方法的研究和改进[J]. 信息网络安全, 2015, (01):1-5.

[31]蔡谊, 左晓栋. 面向虚拟化技术的可信计算平台研究[J]. 信息安全与通信保密, 2013, (06):77-79.

[32]徐天琦, 刘淑芬, 韩璐. 基于 KVM 的可信虚拟化架构模型[J]. 吉林大学学报(理学版), 2014, (03):531-534.

[33]杨丽芳, 刘琳. 基于虚拟机的可信计算安全平台架构设计[J]. 煤炭技术, 2014, (02):170-172.

[34]陈亮, 曾荣仁, 李峰,等. 基于无干扰理论的信任链传递模型[J]. 计算机科学, 2016, 43(10):141-144.

[35]F. John Krautheim*, Dhananjay S. Phatak, and Alan T. Sherman,Introducing the Trusted Virtual Environment Module:A New Mechanism for Rooting Trust in Cloud Computing[C]// TRUST 2010, LNCS 6101, 2010:211–227.

[36]朱智强. 混合云服务安全若干理论与关键技术研究[D].武汉大学,2011:07-31.

[37]曲文涛. 虚拟机系统的可信检测与度量[D].上海交通大学,2010:90-117.

[38]CHEN S Y, WEN Y Y,ZHAO H. Formal analysis of secure bootstrap in trusted computing[C]// Proc of the 4th International Conference on Autonomic and Trusted Computing Berlin, Springer, 2007:352-360.

[39]张兴, 黄强, 沈昌祥. 一种基于无干扰模型的信任链传递分析方法[J]. 计算机学报, 2010, 33(1):74-81.

[40]Rushby J. Noninterference, transitivity, and channel-control security policies[M]// SRI International, Computer Science Laboratory, 1992:01-50.

[41]Kai E, Meyden R V D, Zhang C. Intransitive noninterference in nondeterministic systems[C]// ACM Conference on Computer and Communications Security. ACM, 2012:869-880.

[42]Paolo Baldan, Alessandro Beggiato. Multilevel Transitive and Intransitive Non-interference, Causally[J]. Theoretical Computer Science, 2018, 706:54-82.

[43]张兴, 陈幼雷, 沈昌祥. 基于进程的无干扰可信模型[J]. 通信学报, 2009, 30(3):6-11.

[44]赵佳, 沈昌祥, 刘吉强, 等. 基于无干扰理论的可信链模型[J]. 计算机研究与发展, 2008, 45(6):974-980.

[45]刘威鹏, 张兴. 基于非传递元干扰理论的二元多级安全模型研究[J]. 通信学报, 2009, 30(2):52-58.

[46]陈菊, 谭良. 一个基于进程保护的可信终端模型[J]. 计算机科学, 2011, 38(4):115-117.

[47]徐甫. 支持进程代码修改的非传递元干扰可信模型[J]. 计算机工程, 2013, 39(11):150-

153, 168.

[48]秦晰, 常朝稳, 沈昌样, 等. 容忍非信任组件的可信终端模型研究[J]. 电子学报, 2011, 39(4):934-939.

[49]Smith J, Nair R. Virtual Machines: Versatile Platforms for Systems and Processes (The Morgan Kaufmann Series in Computer Architecture and Design)[M]. Morgan Kaufmann Publishers Inc. 2005.

[50]Adams K, Agesen O. A comparison of software and hardware techniques for x86 virtualization[J]. Acm Sigops Operating Systems Review, 2006, 40(5):2-13.

[51]Get Docker [EB/OL]. [2018-03-10]. Docker. https://www.docker.com/get-docker.

[52]KVM project[EB/OL], [2018-03-10]. KVM. http://www.linux-kvm.org/.

[53]Goguen J A, Meseguer J. Security Policies and Security Models[C]// IEEE Symposium on Security & Privacy. DBLP, 1982:11-20.

[54]Datta A, Franklin J, Garg D, et al. A Logic of Secure Systems and its Application to Trusted Computing[J]. 2009:221-236.

[55]GILLES B, GUSTAVO B, JUAN D C, et al. Formally verifying isolation and availability in an idealized model of virtualization[C]// Proc of the 17th International Conference on Formal Methods. Berlin, Springer, 2011:231-245.

[56]WANG Zhi, JIANG Xu- xian. HyperSafe:a lightweight approach to provide lifetime hypervisor control- flow integrity[C]// Proc of IEEE Sym-posium on Security and Privacy. Washington DC:IEEE Computer Society, 2010:380-395.

[57]JONATHAN M M, NING Q, LI Y L, et al. TrustVisor: efficient TCB reduction and attestation[C]// Proc of the IEEE Symposium on Security and Privacy. Oakland, USA, 2010: 143-158.

[58]张帆, 张聪, 陈伟, 等. 基于无干扰的云计算环境行为可信性分析[J/OL]. 计算机学报, 2017:1-15[2018-03-23]. http://kns.cnki.net/kcms/detail/11.1826.TP.20170728.1254.024.html.

[59]Mozilla Firefox Ltd.[EB/OL]. [2018-03-10]. http://www.firefox.com.cn/download/.

[60]CodeWeavers Inc.[EB/OL]. [2018-03-10]. https://www.winehq.org/.

[61]Kingsoft Office Corporation.[EB/OL]. [2018-03-10]. http://linux.wps.cn/.

[62]The Eclipse Foundation. [EB/OL]. [2018-03-10]. https://www.eclipse.org/downloads/.

.

致谢

首先我要感谢我的导师谭良老师，在我从一个非计算机专业转到计算机领域中信息安全专业的过程中，谭老师一步步的指导我进行科研，从开始的专业技术学习、研究方向选择、硕士论文的写作和完成，谭老师都非常仔细严谨。在我的研究生生涯，谭老师无论自己的工作多么忙碌，都会抽时间来指导我进行科研工作。谭老师对科研的热爱、最新研究方向的洞察力及严谨的科研态度都是我以后无论在工作上还是生活中值得学习的。并且，在科研指导之外，谭老师经常教导我做人的道理，在生活和学习出现阻力时，我都会想到谭老师的一句话“作为我的研究生，技术上需要有一技之长，毕业以后需要有独当一面的能力”。在我的研究生生涯即将结束时，我十分感谢谭老师对我的指导和帮助，感谢他为我以及我的师兄弟们付出的努力和心血，谢谢您。

感谢四川师范大学，感谢为四川师范大学建设付出努力的领导和老师们。四川师范大学的图书馆资源、校园环境、实验室、研究生宿舍，丰富多彩的学术活动、知识讲座，都为我的科研和生活增添了独特的色彩，使我能够把更多的学习精力放在科研上。

感谢在研究生期间对我进行过指导的计算机科学学院的老师们，感谢李晓宁老师、黎静辅导员，感谢在研究生期间的任课老师，是你们的用心指导和辛苦付出才能让我一点点积累专业技能和生活经验。感谢同样跟随谭老师进行学习的师兄师弟，感谢研究生实验室的同学们，是你们的帮助才能让我在平时的科研多出了更多时间来阅读文献和科研实验。感谢参考文献的作者们，是你们的富有深度的科研成果让我对可信计算、云计算、信息安全领域更加深入。

同时感谢我的家人，感谢我的爱人孟胜杰女士，是你们的支持和默默鼓励才能让我在研究生期间能够专注科研和学习。

感谢在百忙中抽出时间对我的论文进行审阅的各位老师，是你们的宝贵意见让我的论文更加完善。

再次感谢所有人，在以后的生活中，我会用更多的时间去帮助其他需要帮助的人。

硕士期间科研成果和参加的科研项目

论文录用情况：

[1]. 齐能,谭良. 具有瀑布特征的可信虚拟平台信任链模型[J]. 计算机应用, 2018, 38(2):327-336.

[2]. 齐能,谭良. 一种具有瀑布特征的可信虚拟平台信任链模型[C]// CTCIS2017. 长沙.（会议论文）

[3]. 谭良, 齐能, 胡玲碧. 虚拟平台环境中一种新的可信证书链扩展方法[J]. 通信学报.（已录用）

参与的科研项目：

[1]. 国家自然科学基金项目：可信平台模块虚拟化问题研究. 编号：61373162.

[2]. 国家自然科学基金项目：低碰撞区跳频序列在高动态无线通信网络中的应用研究. 编号：61701331.