Journal of Computer Applications
计算机应用,2018,38(2):327 -336
ISSN 1001-9081
CODEN JYIIDU
2018-02-10
http://www.joca.cn

文章编号:1001-9081(2018)02-0327-010
DOI:10.11772/j.issn.1001-9081.2017082159

具有瀑布特征的可信虚拟平台信任链模型

齐 能[1*],谭 良[1,2]
(1.四川师范大学 计算机科学学院,成都 610101; 2.中国科学院 计算技术研究所,北京 100190)
(*通信作者电子邮箱 qihuaneng@163.com)

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。为此,提出一种具有瀑布特征的信任链模型——TVP-QT,该模型以硬件可信平台模块(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可信虚拟平台模块(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。不仅从理论上证明了该模型的正确性,而且对实例系统的分析和讨论也表明了该模型的通用性与可行性;在Xen中对该模型进行了仿真实验,实验结果表明该信任链传递理论可以保证可信虚拟化环境在整个运行过程是安全可信的。

关键词:虚拟化;可信计算;可信虚拟平台;信任链;可信衔接点
中图分类号:TP309 **文献标志码**:A

Trust chain model with waterfall characteristic based on trusted virtualization platform

QI Neng[1*], TAN Liang[1,2]
(1. *College of Computer Science, Sichuan Normal University, Chengdu Sichuan* 610101, *China*;
2. *Institute of Computing Technology, Chinese Academy of Sciences, Beijing* 100190, *China*)

Abstract: The trusted virtual platform constructed by the combination of virtualization technology and trusted computing and its trust chain have become a research hot spot. But at present, most of the research 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, there are two separate trust chains, one starts from the underlying virtual platform, the other starts from the top-level user Virtual Machine (VM). In order to solve this problem, a trust chain model with waterfall characteristic called TVP-QT was proposed for the trusted virtual platform. This model starts with the physical Trusted Platform Module (TPM), and adds a Trusted-Joint Point (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 virtualization TPM (vTPM) for VM after the trusted chain is transmitted from the underlying virtual platform to the TJP, 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 the hierarchical and dynamic characteristics of the virtual platform, moreover guarantee the trust of the whole virtual platform. Finally, the correctness of the model was proved in theory, and the generality and feasibility of the proposed trust chain model in the instantiation system was analyzed and discussed. Simulation results on Xen show that the trust chain can ensure the trust and credibility of the trusted cloud platform in the whole running process.

Key words: virtualization; trusted computing; trusted virtual platform; trust chain; Trusted-Joint Point (TJP)

0 引言

随着云计算、大数据等大型计算应用环境在实际业务上更多的应用,具有提高效率、节约成本等优点的虚拟化技术得到工业界和学术界的高度关注,并得以快速应用和推广。如何保障虚拟化平台服务的可信[1-3],确认虚拟化平台向用户提供可信任的资源和服务[4-6],是目前亟待解决的问题。基于硬件信任根的可信计算技术作为保障信息系统安全的关键技术,通过构建从平台底层硬件到平台上层应用程序的信任链[7-8],并结合可信远程证明向平台外部实体提供可信证明[9-11]。因此,利用可信计算技术保证虚拟机平台可信、构建可信虚拟平台(Trusted Virtualization Platform, TVP)环境并构建其信任链模型成为目前的研究热点。

TVP的概念首先由Berger等[12]提出,随后一些学

收稿日期:2017-08-21;**修回日期**:2017-09-06。
基金项目:国家自然科学基金资助项目(61373162);四川省科技支撑项目(2014GZ007)。
作者简介:齐能(1993—),男,河南商丘人,硕士研究生,主要研究方向:可信计算、云计算; 谭良(1972—),男,四川泸州人,教授,博士,CCF高级会员,主要研究方向:可信计算、网络安全。

者[13-16]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，相关学者绝大多数都认为，在物理上，TVP 作为一个可以支持虚拟化技术的可信计算物理主机，与一般的可信计算的区别在于：拥有在物理硬件可信平台模块(Trusted Platform Module，TPM)构建起来的虚拟可信信任根；可以为多个用户虚拟机(Virtual Machine，VM)提供可信虚拟信任环境。这种 TVP 的运行架构如图 1 所示。从功能上看，TVP 架构主要分为四个层次：第一层为硬件信任根 TVP，为整个架构提供信任的物理保证；第二层主要包括虚拟机监视器(Virtual Machine Monitor，VMM)，及构建于 VMM 之上的管理域，它们通常被认为是 TVP 的可信计算基(Trusted Computing Base，TCB)；第三层是虚拟信任根(virtual Root of Trust，vRT)，由于实现方案不同(如图 1 中(a)、(b)所示)，可作为传统信任链的扩展，或者利用动态度量信任根(Dynamic Root of Trusted Measurement，DRTM)机制启动；第四层是与用户紧密相关的用户虚拟机。

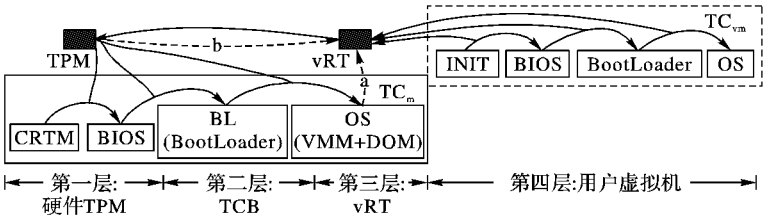

图 1 TVP 基本运行架构

Fig. 1 Basic running architecture of TVP

尽管如此，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，与云环境中虚拟化平台也不完全相符合。如图 1 所示，为了便于叙述，本文将图 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，显然是不精细的，且逻辑上也不完全合理的。第三层包括 VMM 以及 DOM 管理域(即宿主机 Domain0，为方便描述，后文称为 Dom0)，信任链为 CRTM(Core Root of Trust for Measurement)→BIOS→BootLoader→VMM→DOM OS→Apps，DOM 管理域包含 OS 及大量的应用程序，显然不能采用链式度量所有的应用程序并存储其平台配置寄存器(Platform Configuration Register，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在度量层次和度量时间上完全分隔的信任链，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。

针对上述问题，本文提出了一种具有瀑布特征的 TVP 架构 TVP-QT，并对 TVP-QT 信任链传递模型进行了构建，建立了拥有可信衔接点(Trusted-Joint Point，TJP)的 TVP-QT 及其完整的信任链模型。该模型以虚拟化硬件层物理 TPM 为起点，在可信虚拟化平台信任链和可信虚拟机信任链之间加入可信衔接点。当信任链从可信云平台传递到可信衔接点时，由可信衔接点负责对可信虚拟机的可信虚拟平台模块(virtualization Trusted Platform Module，vTPM)及相关组件进行度量，之后再将控制权交给 vTPM，由 vTPM 负责对虚拟机的 VBIOS(Virtual BIOS)、VMOS(OS of Virtual Machine)到 VM 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，能满足云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。

1 相关工作

目前针对 TVP 及其抽象模型以及信任链传递模型的研究得到了国内外学者的广泛关注，本文就目前对 TVP、TVP 信任链模型的研究进行了以下总结和分析。

对于 TVP 的研究，早在 TVP 概念出现之前，就出现了利用可信计算技术解决虚拟系统平台安全的方案，为 TVP 的发展提供了一些理论和构建基础，这些平台包括 Terra[17]、PERSEUS[18]等，这些平台的主要思想是把底层计算平台分为两部分，包括运行着高安全性需求虚拟机的可信区域和其他不可信区域。随后，Berger 等[12]首先提出构建 TVP 的基本组件 vRT、vTPM 等基本思想，并且构建了具体的 TVP 架构。根据文献[12]的 vRT 等概念，HP、IBM 等研究机构分别提出并构建了相应的 TVP[13-14]，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 等[15]、王丽娜等[16]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。之后，常德显等[19]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。Zhang 等[20]提出一种具有可信域层次的 TVP，通过采用可信云平台和可信虚拟机进行分离的 TVP 构建机制，实现了对可信云平台以及可信虚拟机的安全保障。Yu 等[21]、池亚平等[22]、李海威等[23]提出的可信虚拟平台均为链式结构，存在信任链分离的问题。徐天琦等[24]、杨丽芳等[25]、蔡谊等[26]也分别利用可信计算技术构建 TVP 解决虚拟化平台服务器的安全性问题。总结起来，目前已有的研究成果把 TVP 的 VMM 和管理域都作为 TCB，一起作为虚拟机的 vRT，这显然存在过粗且逻辑上不完全合理的问题，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。

对于 TVP 信任链模型的研究，主要包括三个方面。其一是通过对可信计算组织(Trusted Computing Group，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 等[27]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；这种信任链模型是不完善的，无法适应比较复杂的 TVP 环境。Krautheim 等[28]对信任链扩展上提出了“Transitive Trust Chain”信任链模型，并且简要地指出了信任链传递过程为 TPM→VMM→TVEM manager→TVEM→VM OS→应用程序(APP)，但是此种信任链模型没有详细描述特权域操作系统以及虚拟机操作系统的可信度量。Shen 等[29]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 OS→APP，此种信任链模型也存在 Krautheim 等[28]所提模型中的问题。其二是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。常德显等[19]提出的 TVP 信任链从功能层次可表示为：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 等[20]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,此种信任链模型无法有效地在 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,文献[22-23]也存在此类问题。其三是树型或者星型的信任链模型。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[30]、曲文涛[31]基于星型信任度量结构,提出基于信任根(Root of Trust, RT)对各个节点进行度量的模型及其改进,但是此种信任链模型也存在负担重的管理域(Management Domain, MD)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,而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在度量层次和度量时间上分离的信任链,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。

综上所述,TVP 基本运行架构以及信任链传递模型均有待改进。特别地,由于 TVP 具有层次性和动态性,已有的研究成果不能精细地描述虚拟化环境中复杂的信任链传递机制,且存在两条分离的信任链。

2　具有瀑布特征的 TVP 及信任链模型

为解决引言中提出的问题,本文提出了具有瀑布特征的 TVP-QT 及信任链模型。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 的特权域操作、VM 之间的隔离性等安全性机制,可参考 Barthe 等[32]给出的形式化描述与分析。本章将对 TVP-QT 的功能组件以及 TVP-QT 信任链信任属性进行定义,并在下一章利用文献[33]提出的安全逻辑形式化方法对信任链进行形式化分析。本文主要对文献[19]给出的定义进行扩展,其基本定义不再叙述。

2.1　TVP-QT 信任模型

基于已有的 TVP 研究方案,本文提出的 TVP-QT 信任模型如图 2 所示。TVP-QT 分为五个层次:第一层是硬件信任根 TPM 构成的可信虚拟平台底层。第二层主要包括 VMM 以及管理域 Dom0 的相关组件,包括 VBOIS、VOSLoader、VMOS 等组件,作为 TVP-QT 的可信计算基。把 Dom0 Kernel 看成是可信基,这比已有的 TVP 更合理,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,也很容易受到攻击而改变[34-37]。第三层是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,最上层为用户虚拟机层次,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,本文依据文献[19]的基础定义进行扩展抽象定义。

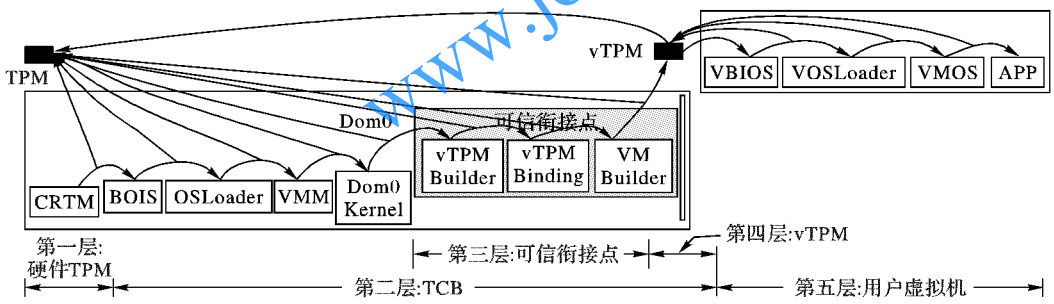

图 2　TVP-QT 运行架构

Fig. 2　Running architecture of TVP-QT

定义 1　TVP-QT 主要包括两类功能组件:TVP-QT:={M, RT}。其中:M 表示虚拟化平台所有主机类型集合,包括构成虚拟化平台的基本组件 VMM、管理域内核、可信衔接点及用户虚拟机等;信任根(RT)是构建 TVP 信任环境的基础,也是 TVP 的核心组件,对 TVP-QT 来说,它包括硬件 TPM、可信衔接点 TJP 和 vTPM。

其中:M = {m, vm},m:={VMM, Dom0 Kernel, TJP},vm:={vm_1, vm_2, …, vm_n};RT:={TPM, vRT} = {TPM, (TJP, vTPM)};并且 m 必须利用 TPM 进行可信度量来提供信任,vm 利用 TJP 和 vm 对应的 vTPM 实例进行可信度量。可信衔接点 TJP 可以划分为 TJP:={vTPM Builder, vTPM-VM Binding, VM Builder},其中 vTPM Builder 表示与创建和管理 vTPM 实例相关的组件,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 1 所示,其中 vTPM 管理组件可由 VMM 上的 vTPM 组件提供,比如目前存在于 Xen 上的 vTPM Manager。

表 1　TJP 功能组件来源

Tab. 1　Resource of TJP's components

TJP	主要组件	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	绑定组件	vTPM 管理组件,VMM
VM Builder	VM 启动、配置文件	VMM

相对于已有的 TVP,本文提出的 TVP-QT 信任模型具有如下特点:

1)TVP-QT 更加精细。已有的 TVP 把整个管理域作为 TCB,TVP-QT 模型仅把 Dom0 Kernel、TJP 及 vTPM 作为 TCB。

2)TVP-QT 在逻辑上更合理。一方面,TVP-QT 可以按链式度量所有层次,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。

2.2　TVP-QT 信任链及属性

本文依据 TCG 对信任链的基本定义和要求构建了基于

TVP-QT 的信任链模型，如图 3 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。

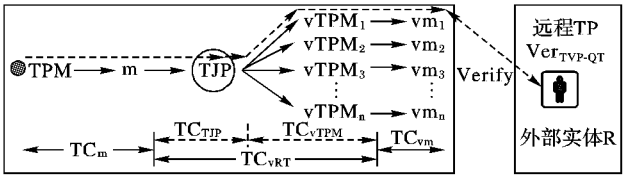

图 3 虚拟化平台可信环境信任链构建与验证

Fig. 3 Construction and validation of TVP-QT trust chain

本文将 2.1 节的可验证目标抽象为信任链的信任属性(Trusted Property, TP)，其抽象定义如下。

定义 2 TVP-QT 的信任属性为一个二元组：$TP_{TVP\text{-}QT} := \{TC_{TVP\text{-}QT}, Ver_{TVP\text{-}QT}\}$，其中 $TC_{TVP\text{-}QT}$ 表示 2.1 节对 TVP-QT 信任链模型具体构建过程描述的各个组件序列，$Ver_{TVP\text{-}QT}$ 表示为需要对 TVP-QT 信任链模型进行远程认证的执行序列。

按照定义 1 对 TVP-QT 中相应功能组件的定义，该 TVP-QT 信任属性可以进一步细分为：

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

由定义可知 TVP-QT 信任属性可以分为三类：m 的信任属性 TC_m，vRT 的信任属性 TC_{vRT}，以及 vm 的信任属性 TC_{vm}。其中 TC_{vRT} 包含 TC_{TJP} 和 TC_{vTPM} 两个属性。下面对 TVP-QT 三类组件的信任属性进行分别阐述。

1) 主机 m 的信任属性表示为 $TP_m := \{TC_m, Ver_m\}$，其中，TC_m 表示基于硬件信任根构建的信任链，即 CRTM→BIOS→OSLoader→VMM→Dom0 $Kernel_{TPM\_Static}$，此部分信任链可基于硬件可信芯片 TPM 的可信度量，在 TVP-QT 信任链可信传递过程中没有其他程序代码加载或者执行。$Ver_m := Verify(m, TC_m)$ 表示对外验证主机 m 所声称的信任属性 TC_m，使远程验证者 R 相信 TVP-QT 平台主机 m 拥有这样的信任链属性 TC_m。

2) vRT 的信任属性为 $TP_{vRT} := \{TC_{vRT}, Ver_{vRT}\}$。由定义 1 对 vRT 以及定义 2 对 TVP-QT 信任属性的定义：

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

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 $(\cdots\rightarrow Dom0\ Kernel\rightarrow TJP)_{TPM\_Static}$，完整地表示为：$(CRTM\rightarrow BIOS\rightarrow OSLoader\rightarrow VMM\rightarrow Dom0\ Kernel\rightarrow vTPM\ Builder\rightarrow vTPM\text{-}VM\ Binding\rightarrow VM\ Builder)_{TPM\_Static}$；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：$(vTPM\ Builder\rightarrow vTPM\text{-}VM\ Binding\rightarrow VM\ Builder)_{TPM\_Dynamic}$。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}；TC_{vTPM} 表示用户虚拟机信任根 vTPM。其中，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：$(TJP\rightarrow vTPM)_{TPM\_Static}$ 或 $(TJP\rightarrow vTPM)_{TPM\_Dynamic}$。$Ver_{vTPM} := Verify(vTPM, Ver_{vTPM})$ 表示对外验证 vTPM 所声称的信任属性 TC_{vTPM}。

3) vm 的信任属性表示为 $TP_{vm} := \{TC_{vm}, Ver_{vm}\}$，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为：$(TJP)_{TPM\_Dynamic}\rightarrow\{INIT\rightarrow VBIOS\rightarrow VOSLoader\rightarrow VMOS\rightarrow APP\}_{vTPM\_Static}$。$Ver_{VM} := Verify(vm, TC_{vm})$ 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链接起来，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 ms 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 ms 启动时采用的是静态度量，而在 vm 创建时，需要动态度量。层次性主要体现在 ms 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，保证底层信任链到顶层信任链的信任扩展。

3) TVP-QT 信任链模型利用构建的可信衔接点解决了虚拟平台信任链与虚拟机信任链的衔接问题。

3 基于扩展 LS^2 的 TVP-QT 信任链分析

本文将采用已有的形式化分析方法“扩展安全逻辑(Logic of Secure System, LS^2)”对 TVP-QT 信任链模型进行形式化分析。对 LS^2 的具体内容可参考文献[19,33]。

3.1 基本假定

在对 TVP-QT 信任链属性进行形式化分析前，本文假定以下条件是成立的：1) TVP 中的每个层次的镜像文件未受破坏，用户虚拟机可被可信度量；2) m 支持 DRTM 技术，能够动态度量 TJP 和 vTPM；3) vTPM 的平台身份密钥(Attestation Identity Key，AIK)已得到认证并颁发证书，具体实现方案参见 vTPM[14] 及 Trust Visor[38] 等；4) 远程挑战者 R 与本地 TVP 之间已经建立了安全信道[19]。

从 2.2 节的分析可知，本文对 TVP-QT 信任链的信任属性分析验证主要包括三部分：

1) 包括 TJP 在内的 m 信任链构建的验证及远程验证；

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。

在这三部分中，对 3) 的验证分析与文献[19]相同，具体过程可参见文献[19]，本文不再论述；下面本文只对 1)、2) 进行验证分析。

3.2 m 信任链的本地验证及远程证明

3.2.1 本地程序执行

根据 2.2 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下列程序 1(Code Segment 1)所示。

Code Segment 1：

```
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)       ≡…
```

程序执行流程:m 首先从 CRTM 启动执行,它从主机内存地址 m. bios_loc 中读取 BIOS 的代码 b,将其扩展到一个 PCR 中,之后执行指令 Jump b;然后 CRTM 将控制权传递给 m 的 BIOS,它从主机内存地址 m. os_loader_loc 中读取 OS_Loader 代码 o,将其扩展到一个 PCR 中,之后执行指令 Jump o,将控制权交给 OSLoader;OSLoader 继续按序从内存 m. vmm_loc 读取 VMM 的代码 v,将其扩展到 m. pcr. s[19],然后转换控制权给 VMM,VMM、Dom0 Kernel 执行相似流程,直到可信衔接点 TJP 的加载。

3.2.2　本地可信属性描述

由上文信任链传递程序执行过程可知,体现主机 m 信任链的是主机进行可信度量后唯一对应的 PCR 值。因此,基于定义 2,可将 m 的本地信任传递属性归纳为:信任链加载的正确由 PCR 中度量值决定。即 m 的本地信任传递属性就是要求所有相应启动程序如 BIOS、OSLoader、VMM、Dom0_Kernel、vTPM Builder、vTPM-VM Binding、VM Builder 等都能按确定的先后顺序加载。以 LS^2 将这种顺序形式化表示为:

$$\begin{aligned}&\text{MeasuredBoot}_{SRTM}(m,t)=\\&\quad\exists t_S.\exists t_b.\exists t_o.\exists t_v.\exists t_d.\exists t_{vb}.\exists t_{vv}.\exists t_{vmb}.\exists t_{o\_app}\wedge\\&\quad\exists J.(t_S<t_b<t_o<t_v<t_d<t_{vb}<t_{vv}<t_{vvb}<t_{o\_app}<t)\wedge\\&\quad(\text{Reset}(m,J)@t_S)\wedge(\text{Jump}(J,\text{BIOS}(m))@t_b)\wedge\\&\quad(\text{Jump}(J,\text{OSLoader}(m))@t_o)\wedge\\&\quad(\text{Jump}(J,\text{VMM}(m))@t_v)\wedge\\&\quad(\text{Jump}(J,\text{Dom0\_Kernel}(m))@t_d)\wedge\\&\quad(\text{Jump}(J,\text{vTPM-Builder}(m))@t_{vb})\wedge\\&\quad(\text{Jump}(J,\text{vTPM-VMBinding}(m))@t_{vv})\wedge\\&\quad(\text{Jump}(J,\text{VM-Builder}(m))@t_{vvb})\wedge\\&\quad(\text{Jump}(J,\text{VM-Builder}(m))@t_{o\_app})\wedge\\&\quad(\neg\,\text{Reset}(m)\text{on}(t_S,t])\wedge(\neg\,\text{Jump}(J)\text{on}(t_S,t_b))\wedge\\&\quad(\neg\,\text{Jump}(J)\text{on}(t_B,t_o))\wedge(\neg\,\text{Jump}(J)\text{on}(t_o,t_v))\wedge\\&\quad(\neg\,\text{Jump}(J)\text{on}(t_v,t_d))\wedge(\neg\,\text{Jump}(J)\text{on}(t_d,t_{vb}))\wedge\\&\quad(\neg\,\text{Jump}(J)\text{on}(t_{vb},t_{vv}))\wedge(\neg\,\text{Jump}(J)\text{on}(t_{vv},t_{vvb}))\wedge\\&\quad(\neg\,\text{Jump}(J)\text{on}(t_{vvb},t_{o\_app}))\end{aligned}$$

上述公式表示:如果 TVP 的 m 基于信任链构建了本地信任环境,则其启动过程一定是从 BIOS 跳转到 OSLoader,从 OSLoader 到 VMM,从 VMM 到 Dom0_Kernel,然后从 Dom0_Kernel 到 TJP,而在此期间不会有其他程序执行。为保证程序逐级加载顺序与 PCR 值相对应,必须证明信任链本地属性是成立的。

定理 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)),那么该 m 的本地信任链传递过程就是唯一的、正确的,即确定地从 BIOS(m)到 OSLoader(m)再到 VMM(m)、Dom0 Kernel(m)、vTPM Builder(m)、vTPM-VM Binding(m)、VM Builde(m)。该信任属性形式化表示为:

$$\begin{aligned}&\text{ProtectedSRTM}(m)+\text{Mem}(m.\text{pcr}.s,\text{seq}(\text{BIOS}(m),\\&\quad\text{OSLoader}(m),\text{VMM}(m),\text{Dom0\_Kernel}(m),\\&\quad\text{vTPMBuilder}(m),\text{vTPM-VM Binding}(m),\\&\quad\text{VM Builder}(m)))\supset\text{MeasuredBoot}_{SRTM}(m,t)\end{aligned}$$

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 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,反复利用 PCR 公理即可直接得到在该序列中的所有子序列一定在时间 t 之前就出现在 m. pcr. s 中,即:

$$\begin{aligned}&\exists t_S,t_1,t_2,t_3,t_4,t_5,t_6,J.(t_S\leqslant t_1<t_2<t_3<t_4<t_5<t_6<t)\wedge\\&\quad(\text{Mem}(m.\text{pcr}.s,\text{seq}(\text{BIOS}(m),\text{OSLoader}(m),\\&\quad\text{VMM}(m),\text{Dom0\_Kernel}(m),\text{vTPM Builder}(m),\\&\quad\text{vTPM-VM Binding}(m),\text{VM Builder}(m)))@t)\wedge\\&\quad(\text{Mem}(m.\text{pcr}.s,\text{seq}(\text{BIOS}(m),\text{OSLoader}(m),\\&\quad\text{VMM}(m),\text{Dom0\_Kernel}(m),\text{vTPM Builder}(m),\\&\quad\text{vTPM-VM Binding}(m)))@t_6)\wedge(\text{Mem}(m.\text{pcr}.s,\\&\quad\text{seq}(\text{BIOS}(m),\text{OSLoader}(m),\text{VMM}(m),\\&\quad\text{Dom0\_Kernel}(m),\text{vTPM Builder}(m))),@t_5)\wedge\\&\quad(\text{Mem}(m.\text{pcr}.s,\text{seq}(\text{BIOS}(m),\text{OSLoader}(m),\\&\quad\text{VMM}(m),\text{Dom0\_Kernel}(m)))@t_4)\wedge(\text{Mem}(m.\text{pcr}.s,\\&\quad\text{seq}(\text{BIOS}(m),\text{OSLoader}(m),\text{VMM}(m)))@t_3)\wedge\\&\quad(\text{Mem}(m.\text{pcr}.s,\text{seq}(\text{BIOS}(m),\text{OSLoader}(m)))@t_2)\wedge\\&\quad(\text{Mem}(m.\text{pcr}.s,\text{seq}(\text{BIOS}(m)))@t_1)\wedge\\&\quad\text{Reset}(m,J)@t_S\wedge(\neg\,\text{Reset}(m)\text{on}(t_S,t))\end{aligned}\tag{1}$$

接下来对程序 1 中信任链的执行过程进行说明,最先执行的操作是以 CRTM 为起点启动 m,即 Reset(m,J),然后 m 执行第一个信任程序 BIOS(m)。利用 LS^2 规则,在某个时间 t_B,程序会跳转到 b,此时内存位置被锁定,防止其他程序跳转[19]。即有以下属性(2)成立。

$$\begin{aligned}&\forall t',b,o\\&(((\text{Mem}(m.\text{pcr}.s,\text{seq}(\text{BIOS},b,o))@t')\wedge(t_s<t'<t))\supset\\&\exists t_b\cdot((t_s<t_b<t)\wedge(\text{Jump}(J,b)@t_b)))\wedge\\&(\text{IsLocked}(m.\text{pcr}.s,J)@t_b)\end{aligned}\tag{2}$$

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)和 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,且其他时间不会有程序跳转,相应的内存位置(即 PCR 值)被该线程锁定,即有 o、v 等对应的类似式(2)的属性成立,限于篇幅,这些属性略。

根据式(2)及 o、v、d 等对应的属性可知,如果前提条件满足,那么 m 上执行程序的顺序一定是从 BIOS(m)到 OSLoader(m)再到 VMM(m)、Dom0Kernel(m)、TJP(m)。

$$\begin{aligned}&\exists t_S,t_b,t_o,t_v,t_d,t_{vb},t_{vv},t_{vmb},t_{o\_app}\\&\quad(t_S<t_b<t_o<t_v<t_d<t_{vb}<t_{vv}<t_{vvb}<t_{o\_app}<t)\wedge\\&\quad(\neg\,\text{Reset}(m,I)\text{on}(t_S,t])\wedge(\text{Reset}(m,J)@t_S)\wedge\\&\quad(\text{Jump}(J,\text{BIOS}(m))@t_b)(\neg\,\text{Jump}(J,\text{BIOS}(m))\\&\quad\text{on}(t_S,t_b))\wedge(\text{Jump}(J,\text{OSLoader}(m))@t_o)\wedge\end{aligned}$$

$(\neg\ Jump(J,OSLoader(m))on(t_b,t_o))\wedge$
$(Jump(J,VMM(m))@t_v)\wedge$
$(\neg\ Jump(J,VMM(m))on(t_o,t_v))\wedge$
$(Jump(J,Dom0\_Kernel(m))@t_d)\wedge$
$(\neg\ Jump(J,Dom0\_Kernel(m))on(t_v,t_d))\wedge$
$(Jump(J,vTPM\text{-}Builder(m))@t_{vb})\wedge$
$(\neg\ Jump(J,vTPM\text{-}Builder(m))on(t_d,t_{vb}))\wedge$
$(Jump(J,vTPM\text{-}VM\ Binding(m))@t_{vv})\wedge$
$(\neg\ Jump(J,vTPM\text{-}Builder(m))on(t_{vb},t_{vv}))\wedge$
$(Jump(J,VM\ Binding(m))@t_{vmb})\wedge$
$(\neg\ Jump(J,VM\ Builder(m))on(t_{vv},t_{vmb}))\wedge$
$(Jump(J,VM\ Binding(m))@t_{o\_app})\wedge$
$(\neg\ Jump(J,VM\ Builder(m))on(t_{vmb},t_{o\_app}))$ (3)

定理 1 即得证。

可信计算技术提供的度量扩展机制能够保证只有程序得到正确的内存值时才能继续运行下一个程序，可确保程序是在无攻击者存在的情况下运行的。

3.2.3 信任链远程验证

TVP-QT 的 m 需要对外部挑战者证明自身的信任属性，需要证明 $MeasuredBoot_{SRTM}(m,t)$ 成立，证明方法可参照上文以及文献[33]，本文在此仅对验证程序进行说明。m 信任传递的远程验证过程中涉及到的程序，如程序 2（Code Segment 2）所示。

Code Segment 2：

```
TPM_SRTM(m)   ≡w = read m. pcr. s;
              r = sign(PCR(s),w),AIK^-1(m);
              send r
Verifier(m)   ≡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)))
```

首先，m 读取存储的 PCR 值，用 AIK 签名（$AIK^{-1}(m)$）并将其发送给远程验证者 R。然后，R 验证该签名，如果 PCR 值是匹配的，则说明 m 的可信属性是值得信任的。在此过程中 R 与 m 应是不同的实体，以保证该验证过程的有效性，避免验证过程的伪造。

这些前提条件形式化表示为：

$$\Gamma_{SRTM}=\{\hat{V}\neq AIK(m),Honest(AIK(m),\{TPM_{SRTM}(m),TPM_{DRTM}(m)\})\} \quad (4)$$

3.3 可信衔接点 TJP 的本地验证及远程证明

本节根据 2.2 节对 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。

3.3.1 本地程序执行

根据 2.2 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，其信任链本地执行过程中涉及到的程序如程序 3（Code Segment 3）所示。

Code Segment 3：

```
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)                        ≡…
```

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[19]；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：

$$Honest(TPM_{SRTM}(m)\succ TJP_{DRTM}(m)\wedge TJP_{DRTM}(m)\succ vTPM_{SRTM/DRTM}(vm))$$

此外，TVP 在启动 m 时，相应的线程 K 对必须能够对当前 m 对应的 PCR 值有锁控制，这种控制对潜在的攻击者也成立，表示为：

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

由于 TJP 的 vTPM Builder 被抽象为一个单独的应用程序，利用 Latelaunch（vTPM Builder）动态加载机制确保其可信执行，即 K_{DRTM} 成立[27]。

3.3.2 本地可信属性描述

基于定义 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，可将 TJP 的本地信任传递属性归纳为：TJP 信任链加载程序执行顺序的正确性由最终产生的 PCR 值决定。即 TJP 的本地信任传递属性就是要求所有相应启动程序如 vTPM Builder、vTPM-VM Binding、VM Builder 等都能按确定的先后顺序加载。以 LS^2 将这种顺序形式化表示为：

$MeasuredBoot_{DRTM}(TJP,t)=$
$\forall I\in m.\ \exists t_{vtb}.\ \exists t_{vvb}.\ \exists t_{vvmb}.\ \exists t_{vtpmb}.$
$\exists K.(t_{vtb}<t_{vvb}<t_{vvmb}<t_{vtpmb}<t)\wedge$
$(Jump(K,vTPM\ Builder(TJP))@t_{vtb})\wedge$
$(Jump(K,vTPM\text{-}VM\ Binding(TJP))@t_{vvb})\wedge$
$(Jump(K,VM\ Builder(TJP))@t_{vvmb})\wedge$
$(Jump(K,vTPM(TJP))@t_{vtpmb})\wedge$
$(\neg\ Reset(m)on(t_{vtb},t])(\neg\ Jump(K)on(t_{vtb},t_{vvb}))\wedge$
$(\neg\ Jump(K)on(t_{vvb},t_{vvmb}))\wedge$
$(\neg\ Jump(K)on(t_{vvmb},t_{vtpmb}))$ (5)

式(5)表示：如果 TVP 基于信任链构建 TJP 信任环境，则其启动过程一定是从 vTPM Builder 跳转到 vTPM-VM Binding，然后到 VM Builder，在程序执行期间，程序启动序列都有唯一的 PCR 值，并且不会有其他应用程序或者组件执行。需要证明的信任链本地信任属性如下。

定理 2 如果 TJP 加载成功，且与该 TJP 加载过程对应的 PCR 值为 seq（vTPM Builder（TJP），vTPM-VM Binding（TJP），VM Builder（TJP）），那么该 TJP 的本地信任链传递过程就是

唯一的、正确的,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:

$$\begin{aligned}&\text{ProtectedDRTM(TJP)} +\\&\text{Mem(m.dpcr.d,seq(vTPM Builder(TJP),}\\&\text{vTPM-VM Binding(TJP),VM Builder(TJP)))}\supset\\&\text{MeasuredBoot}_{\text{DRTM}}\text{(TJP,t)}\end{aligned}\tag{6}$$

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。

3.3.3　信任链远程验证

TVP-QT 的 TJP 需要向 R 证明自己在信任链传递过程中进行可信度量的组件的信任属性及构建的可信执行环境,需要证明 $\text{MeasuredBoot}_{\text{DRTM}}\text{(TJP,t)}$ 成立。

1)远程验证程序执行。

首先,根据 TCG 远程证明协议规范及在虚拟化平台中的实现,给出 TJP 信任传递的远程验证过程中涉及到的程序,如程序 4(Code Segment 4)所示。

Code Segment 4:

$$\begin{aligned}&\text{TPM}_{\text{DRTM}}\text{(TJP)} \equiv \text{w} = \text{read m.pcr.d;}\\&\qquad\qquad\text{r} = \text{sign(PCR(s),w),AIK}^{-1}\text{(m);}\\&\qquad\qquad\text{send r}\\&\text{Verifier(TJP)}\quad \equiv \text{sig} = \text{recieve;}\\&\qquad\qquad\text{v} = \text{verify sig, AIK(m);}\\&\qquad\qquad\text{match v, (PCR(s),}\\&\qquad\qquad\text{seq(vTPM Builder(TJP),vTPM-VM}\\&\qquad\qquad\text{Binding(TJP),VM Builder(TJP)))}\end{aligned}$$

首先,m 读取本地 TJP 的 PCR 值,用 AIK 签名($\text{AIK}^{-1}\text{(m)}$)并将其发送给挑战者。然后,由远程验证者 R 验证签名,并用预期的度量值与收到的 PCR 值进行匹配,验证 TJP 是否拥有可信属性。在此过程中远程验证者与 TJP 所属的主机 m 应是不同实体。

这些前提条件形式化表示为:

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

2)信任链属性的远程验证。

根据远程证明协议执行流程,给出以下信任传递属性的远程证明目标。

定理 3　如果远程验证者确认 TJP 提供的度量值是唯一的、正确的,那么该 TJP 对应的 PCR 值一定是如下的确定序列 seq(vTPM Builder(TJP),vTPM-VM Binding(TJP),VM Builder(TJP)),因为根据定理 2 可知,该序列表明该虚拟机的确执行了相应的信任链传递过程。

形式化表示为:

$$\begin{aligned}&\Gamma_{\text{DRTM}} \vdash [\text{Verifier(m)}]_{\text{V}}^{t_B:t_E}\exists t.(t<t_e)\wedge\\&\quad(\text{Mem(m.pcr.d,seq(vTPM Builder(TJP),}\\&\quad\text{vTPM-VMBinding(TJP),VM Builder(TJP)))@t)}\end{aligned}\tag{7}$$

$$\begin{aligned}&\Gamma_{\text{DRTM}},\text{Protected}_{\text{SRTM}}\text{(m)}\\&\quad\vdash[\text{Verifier(m)}]_{\text{V}}^{t_B:t_E}\exists t.(t<t_E)\wedge\\&\quad\text{MeasureBoot}_{\text{DRTM}}\text{(m,t)}\end{aligned}\tag{8}$$

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　实例系统分析与讨论

本文基于 Xen 虚拟化平台对 TVP-QT 信任链进行实际的验证和分析,如图 4 所示。其中,vRT 被分散在 Dom0、vTPMmanager 域和 vTPM 域。本章根据第 2 章中对 TVP-QT 信任链的描述,将 TVP-QT 信任链分为三部分,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,trampoline.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 Oxl0000 后读入 Xen 的镜像,并对此镜像和_startxen()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。

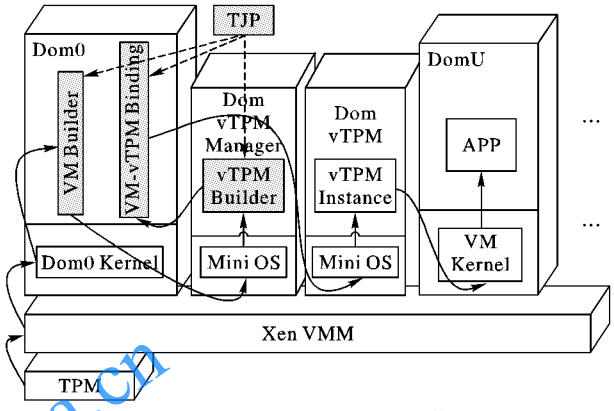

图 4　基于 Xen 的 TVP-QT 系统

Fig. 4　TVP-QT system based on Xen

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、vtpm-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。由于 DRTM 技术受硬件厂商所限制,利用静态度量方式对 vTPM 实例域进行可信度量。VM 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，还涉及 Dom0、vTPMManager 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。

5 实验及结果分析

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。

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[40-41]，实验物理平台的配置为 Intel Core i3 @3.4 GHz 处理器，内存为 8 GB，物理存储为 1 TB。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 4 GB，并且部署不同的应用作为仿真实验的测试对比。

表 2 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 2 物理平台（Dom0）和用户虚拟机（DomU-Ubuntu）配置

Tab. 2 Configure infomation of Dom0 and (DomU-Ubuntu)

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

本文实验按照云计算环境实际建立了 Dom0，DomU（Ubuntu LTS14.04）以及实验所需的 vTPMManager 域和 vTPM 实例域，它们的配置文件和启动方式本文不再作详细叙述。

5.1 TVP-QT 信任链构建

TVP-QT 信任链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，计算 Hash 值作为新的完整性度量值存储到 PCR 中[29]，描述如下：

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

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，按照表 3 的顺序对 PCR 进行存储。其中 PCR[0] ~ PCR[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。具体的存储情况如表 3 所示。

按照 TVP-QT 信任链顺序存储的信任链信息结果如图 5 所示。一旦程序或文件内容发生变化，下次执行该程序或者打开该文件时，用户虚拟机就可以根据 PCR 值的信息判断平台状态是否可信。

表 3 仿真实验 PCR 存储情况

Tab. 3 Storage of PCR in simulation experiment

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 VMM 内核代码 Dom0OS 启动相关信息 Dom0 OS Kernel	第一层：TPM 第二层：TCB
PCR[8]	MiniOs 及 vTPMMgr 配置	第三层：TJP 第四层：vTPM
PCR[9]	负责 vTPM-VM 相关组件	
PCR[10]	VM 配置文件	
PCR[11]	MiniOs 及 vTPM 实例域	
PCR[12]	VBIOS 配置信息	
PCR[13]	VOSLoader	第五层：VM 部分
PCR[14]	VM 启动的其他信息	
PCR[15]	VM 中的应用程序	

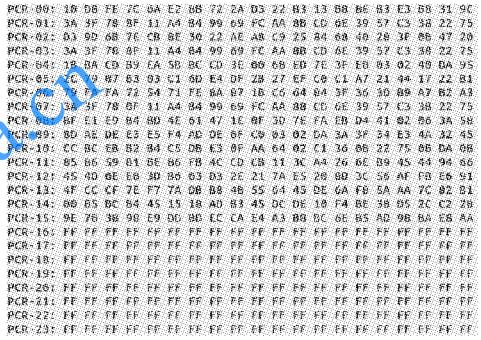

图 5 信任链 PCR 信息

Fig. 5 PCR information of trust chain

5.2 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，需要对其独立度量并且带来额外的开销。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 1 所示）进行对比分析。

本节性能测试的实验环境采用表 1 所描述实验环境，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[41]、WPS for Linux[42]、Wine[43]、Eclipse[44]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。

5.2.1 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 的信任链构建过程为：

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

vTPM Builder →vTPM-VM Binding→VM Builder→ other_app

对以上两条信任链进行 10 次实验,并记录每次的完成时间,结果如图 6 所示。由图 6 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。

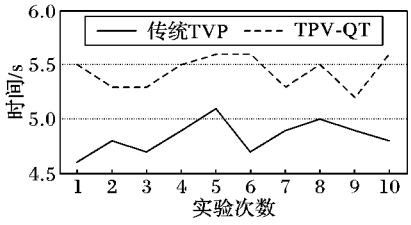

图 6　m 信任链构建时间

Fig. 6　Construction time of m trust chain

5.2.2　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 的信任链构建过程为:

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

对以上两条信任链进行 10 次实验,并记录每次的完成时间,结果如图 7 所示。由图 7 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。

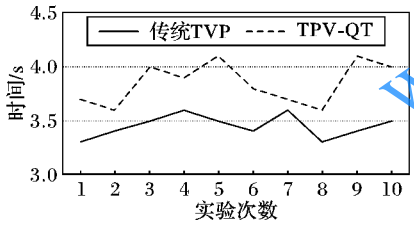

图 7　vm 信任链构建时间

Fig. 7　Construction time of vm trust chain

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开销的情况下实现对平台的可信度量,保证了虚拟化环境的安全可信。

6　结语

利用可信计算技术构建可信虚拟平台并且构建信任链模型是目前解决云计算安全一个重要的研究方向。本文为了解决已有 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,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。最后,本文基于 Xen VMM 实现了 TVP-QT 原型系统,并对 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,验证了该信任链的正确性和有效性。

参考文献:

[1]　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, 36(9): 1765 – 1784. (LIN C, SU W B, MENG K, et al. Cloud computing security: architecture, mechanism and modeling [J]. Chinese Journal of Computers, 2013, 36(9): 1765 – 1784.)

[2]　俞能海, 郝卓, 徐甲甲, 等. 云安全研究进展综述[J]. 电子学报, 2013, 41(2): 371 – 381. (YU N H, HAO Z, XU J J, et al. Review of cloud computing security [J]. Acta Electronica Sinica, 2013, 41(2): 371 – 381.)

[3]　ALI M, KHAN S U, VASILAKOS A V. Security in cloud computing: opportunities and challenges [J]. Information Science, 2015, 305: 357 – 383.)

[4]　XU P, CHEN H, ZOU D, et al. Fine-grained and heterogeneous proxy re-encryption for secure cloud storage [J]. Chinese Science Bulletin, 2014, 59(32): 4201 – 4209.

[5]　XIANG S, ZHAO B, YANG A, et al. Dynamic measurement protocol in infrastructure as a service [J]. Tsinghua Science and Technology, 2014, 19(5): 470 – 477.

[6]　YU F, ZHANG H, ZHAO B, et al. A formal analysis of trusted platform module 2.0 hash-based message authentication code authorization under digital rights management scenario [J]. Security and Communication Networks, 2015, 9(15): 2802 – 2815.

[7]　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, 35(10): 15 – 18. (TAN L, XU Z W. Development of the transitive trusted chain based on TPM [J]. Computer Science, 2008, 35(10): 15 – 18.)

[8]　徐明迪, 张焕国, 张帆, 等. 可信系统信任链研究综述[J]. 电子学报, 2014, 42(10): 2024 – 2031. (XU M D, ZHANG H G, ZHANG F, et al. Survey on chain of trust of trusted system [J]. Acta Electronica Sinica, 2014, 42(10): 2024 – 2031.)

[9]　谭良, 陈菊, 周明天. 可信终端动态运行环境的可信证据收集机制[J]. 电子学报, 2013, 41(1): 77 – 85. (TAN L, CHEN J, ZHOU M T. Trustworthiness evidence collection mechanism of running dynamic environment of trusted terminal[J]. Acta Electronica Sinica, 2013, 41(1): 77 – 85.)

[10]　于爱民, 冯登国, 汪丹. 基于属性的远程证明模型[J]. 通信学报, 2010, 31(8): 1 – 8. (YU A M, FENG D G, WANG D. Property-based remote attestation model [J]. Journal on Communications, 2010, 31(8): 1 – 8.)

[11]　谭良, 陈菊. 一种可信终端运行环境远程证明方案[J]. 软件学报, 2014, 25(6): 1273 – 1290. (TAN L, CHEN J. Remote attestation project of the running environment of the trusted terminal [J]. Journal of Software, 2014, 25(6): 1273 – 1290.)

[12]　BERGER S, CÁCERES R, GOLDMAN K A, et al. VTPM: virtualizing the trusted platform module [C]// USENIX-SS'06: Proceedings of the 15th USENIX Security Symposium. Berkeley, CA: USENIX Association, 2006, 15: 305 – 320.

[13]　DALTON C I, PLAQUIN D, WEIDNER W, et al. Trusted virtual platforms: a key enabler for converged client devices [J]. ACM SIGOPS Operating Systems Review, 2009, 43(1): 36 – 43.

[14]　BERGER S, CÁCERES R, PENDARAKIS D, et al. TVDc: managing security in the trusted virtual datacenter [J]. ACM SIGOPS Operating Systems Review, 2008, 42(1): 40 – 47.

[15]　KRAUTHEIM F J, PHATAK D S, SHERMAN A T. Introducing the trusted virtual environment module: a new mechanism for rooting trust in cloud computing [C]// TRUST'10: Proceedings of the 3rd International Conference on Trust and Trustworthy Computing, LNCS 6101. Berlin: Springer, 2010: 211 – 227.

[16]　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. (WANG L

N, GAO H J, YU R W, et al. Research of constructing trusted virtual execution environment based on trust extension[J]. Journal on Communications, 2011, 32(9): 1-8.)

[17] GARFINKEL T, PFAFF B, CHOW J, et al. Terra: a virtual machine-based platform for trusted computing [C]// SOSP '03: Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles. New York: ACM, 2003: 193-206.

[18] PFITZMANN B, RIORDAN J, STÜBLE C, et al. The PERSEUS system architecture, IBM Research Report RZ 3335 (#93381) [R]. Zurich: IBM Research, 2001.

[19] 常德显,冯登国,秦宇,等.基于扩展 LS^2 的可信虚拟平台信任链分析[J].通信学报,2013,34(5):31-41.(CHANG D X, FENG D G, QIN Y, et al. Analyzing the trust chain of trusted virtualization platform based on the extended LS^2 [J]. Journal on Communications, 2013, 34(5): 31-41.)

[20] ZHANG L, CHEN X, LIU L, et al. Trusted domain hierarchical model based on noninterference theory [J]. The Journal of China Universities of Posts and Telecommunications, 2015, 22(4): 7-16.

[21] YU Z, ZHANG W, DAI H, et al. 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.

[22] 池亚平,李欣,王艳,等.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,2016,37(6):1451-1455.(CHI Y P, LI X, WANG Y, et al. KVM-based trusted virtualization platform design and implementation [J]. Computer Engineering and Design, 2016, 37(6): 1451-1455.)

[23] 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(1):1-5.(LI H W, FAN B, LI W F. Research and improvement on constructing method of a trusted virtualization platform [J]. Netinfo Security, 2015(1): 1-5.)

[24] 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),2014,52(3):531-534.(XU T Q, LIU S F, HAN L. KVM-based trusted virtualization architecture model [J]. Journal of Jilin University (Science Edition), 2014, 52(3): 531-534.)

[25] 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,2014,33(2):170-172.(YANG L F, LIU L. Design of trusted computing security platform architecture based on virtual machine [J]. Coal Technology, 2014, 33(2): 170-172.)

[26] 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(6):77-79.(CAI Y, ZUO X D. Trusted computing platform for virtualization technology [J]. Information Security and Communications Privacy, 2013(6): 77-79.)

[27] SCARLATA V, ROZAS C, WISEMAN M, et al. TPM virtualization: building a general framework [M]// Trusted Computing. [S. l.]: Vieweg+Teubner, 2007, 2007: 43-56.

[28] KRAUTHEIM F J, PHATAK D S, SHERMAN A T. Introducing the trusted virtual environment module: a new mechanism for rooting trust in cloud computing [C]// TRUST 2010: Proceedings of the 3rd International Conference on Trust and Trustworthy Computing, LNCS 6101. Berlin: Springer, 2010: 211-227.

[29] SHEN C, ZHANG H, WANG H, et al. Research on trusted computing and its development [J]. Science China Information Sciences, 2010, 53(3): 405-433.

[30] 朱智强.混合云服务安全若干理论与关键技术研究[D].武汉:武汉大学,2011:91-117.(ZHU Z Q. The research on some theories and key technologes of hybrid cloud computing security [D]. Wuhan: Wuhan University, 2011: 91-117.)

[31] 曲文涛.虚拟机系统的可信检测与度量[D].上海:上海交通大学,2010.(QU W T. Trusted detect and measure for virtual machine system [D]. Shanghai: Shanghai Jiao Tong University, 2010.)

[32] BARTHE G, BETARTE G, CAMPO J D, et al. Formally verifying isolation and availability in an idealized model of virtualization [C]// FM 2011: Proceedings of the 17th International Symposium on Formal Methods, LNCS 6664. Berlin: Springer, 2011: 231-245.

[33] DATTA A, FRANKLIN J, GARG D, et al. A logic of secure systems and its application to trusted computing [C]// SP '09: Proceedings of the 2009 30th IEEE Symposium on Security and Privacy. Washington, DC: IEEE Computer Society, 2009: 221-236.

[34] CHEN G, JIN H, ZOU D, et al. SafeStack: automatically patching stack-based buffer overflow vulnerabilities [J]. IEEE Transactions on Dependable and Secure Computing, 2013, 10(6): 368-379.

[35] VERMEULEN S. SELinux Cookbook [M]// Birmingham, UK: Packet Publishing, 2014: 2-9.

[36] VARMA P D K, RADHA V. Prevention of buffer overflow attacks using advanced stackguard [C]// Proceedings of 2010 International Conference on Advances in Communication, Network, and Computing. Washington, DC: IEEE Computer Society, 2010: 357-359.

[37] WANG Z, JIANG X. HyperSafe: a lightweight approach to provide lifetime hypervisor control-flow integrity [C]// SP '10: Proceedings of the 2010 IEEE Symposium on Security and Privacy. Washington, DC: IEEE Computer Society, 2010: 380-395.

[38] McCUNE J M, LI Y, QU N, et al. TrustVisor: efficient TCB reduction and attestation [C]// SP '10: Proceedings of the 2010 IEEE Symposium on Security and Privacy. Washington, DC: IEEE Computer Society, 2010: 143-158.

[39] TAKEMURA C, CRAWFORD L S. The Book of Xen: A Practical Guide for the System Administrator [M]. San Francisco, CA: No Starch Press, 2009: 2-15.

[40] Xen Project. The Xen Project, the powerful open source industry standard for virtualization [EB/OL]. [2017-03-22]. http://www.xenproject.org.

[41] Mozilla Firefox Ltd. The new, fast browser for Mac, PC and Linux | Firefox [EB/OL]. [2017-04-12]. https://www.mozilla.org/en-US/firefox/#.

[42] Kingsoft Office Software. Best office run on Linux platform, WPS Office for Linux [EB/OL]. [2017-05-05]. https://www.wps.com/linux?from=download_page.

[43] CodeWeavers Inc. WineHQ — Run Windows applications on Linux, BSD, Solaris and macOS [EB/OL]. [2017-04-12]. https://www.winehq.org/.

[44] The Eclipse Foundation. Eclipse [EB/OL]. [2017-04-02]. https://www.eclipse.org/downloads/.

This work is partially supported by the National Natural Science Foundation of China (61373162), the Science and Technology Support Project of Sichuan Province (2014GZ007).

QI Neng, born in 1993, M. S. candidate. His research interests include trusted computing, cloud computing.

TAN Liang, born in 1972, Ph. D., professor. His research interests include trusted computing, network security.