doi:10.3969/j.issn.1000-436x.2013.05.004

基于扩展 LS^2 的可信虚拟平台信任链分析

常德显[1,2,3]，冯登国[1]，秦宇[1]，张倩颖[1,2]
（1. 中国科学院 软件研究所，北京 100190；2. 中国科学院 研究生院，北京 100049；
3. 解放军信息工程大学 三院，河南 郑州 450004）

摘 要：针对可信虚拟平台信任链的形式化分析问题，建立了包括虚拟机和虚拟信任根在内的可信虚拟平台完整的信任链模型，并详细定义其应满足的信任属性，通过扩展 LS^2，验证了可信虚拟平台信任链模型能够有条件地满足其正确性、唯一性。对实例系统分析表明本文所建立信任链模型的通用性及基于扩展 LS^2 分析方法的有效性。
关键词：可信计算；可信虚拟平台；安全系统逻辑；信任链；虚拟信任根
中图分类号：TP309 **文献标识码**：A **文章编号**：1000-436X(2013)05-0031-11

Analyzing the trust chain of trusted virtualization platform based on the extended LS^2

CHANG De-xian[1,2,3], FENG Deng-guo[1], QIN Yu[1], ZHANG Qian-ying[1,2]
（1. Institute of Software, Chinese Academy of Sciences, Beijing 100190, China;
2. Graduate University of Chinese Academy of Sciences, Beijing 100049, China;
3. 3rd Institute, PLA Information Engineering University, Zhengzhou 450004, China）

Abstract: Considering the effective formal analysis for the trust chain of the trusted virtualization platform, a trust chain model which includes the virtual machine and the virtual root of trust, was proposed firstly with the detail definition of the trusted properties. Through extending the LS^2, it verified the correctness and uniqueness of the trust chain formally under some conditions. Analysis for the photosystem shows the generality of the proposed trust chain model and the validity of the analysis method based on the extended LS^2.
Key words: trusted computing; trusted virtualization platform; logic of secure system; trust chain; virtual root of trust

1 引言

虚拟化技术因具有节省成本、提高效率等特有优势使其得以快速应用推广，比如，在云计算等大型计算应用环境中，虚拟化平台已经成为承担海量计算和应用服务的基础。但随之而来的一个关键问题就是如何为虚拟化平台提供服务可信的保障，用户在使用虚拟化平台提供的资源和服务时，亟需确认该服务平台是否可信任。可信计算技术基于硬件信任根，能够为平台构建从底层硬件到上层应用程序的信任链，并结合度量与远程证明机制为外部提供可信证明[1]，从而为平台提供可信运行环境保障，因此，利用可信计算技术构建可信虚拟平台（TVP, trusted virtualization platform）环境并对其信任进行验证成为目前的研究热点。

针对虚拟化平台多用户操作系统实例，即多个虚拟机（VM, virtual machine）并发运行于同一物理平台的需求，Stefan Berger[2]等人首先提出虚拟信任根（vRT, virtual root of trust）、虚拟可信平台模块（vTPM, virtual trusted platform module）的思想，通过为每个虚拟机提供独立的虚拟信任根来构建实现虚拟化可信平台的原型系统。HP 和 IBM 的研

收稿日期：2012-08-15；修回日期：2012-11-12
基金项目：国家自然科学基金资助项目(91118006, 61202414)；国家重点基础研究发展计划（“973”计划）基金资助项目(2013CB338003)
Foundation Items: The National Natural Science Foundation of China (91118006, 61202414); The National Basic Research Program of China (973 Program) (2013CB338003)

究人员在虚拟信任根的基础上，分别提出各自的 TVP 概念[3,4]，并针对不同应用需求建立用户可定制 TVP，他们的工作大大推动了 TVP，并使其在云计算环境中得到应用[5,6]。可信平台需要提供其信任证明，为此，需要对构建平台信任的基石——信任链进行形式化建模与分析，以确保平台信任的可验证。陈书义等人利用一阶逻辑对可信计算平台启动过程进行建模以分析其信任传递过程[7]，并提出长度受限的信任链模型。张兴等人基于无干扰模型对信任链进行了建模分析[8]，从系统信息流控制角度验证满足传递无干扰安全策略的信息流才能构建有效的信任链。为了验证本地的信任属性，需要利用远程证明协议对远程验证方提供验证。冯登国等[9,10]基于国际可信计算组织（TCG, Trusted Computing Group）的远程证明方案，提出改进的远程证明协议，并给出了基于可证明安全的形式化分析。这些针对平台信任进行的形式化的分析与验证工作，在一定程度上丰富了平台信任链理论模型，为平台信任证明提供有力支撑。

但是，上述方法主要针对普通可信计算平台，并不能直接适用于 TVP。目前，TVP 的信任链模型形式化分析主要存在以下问题。

1) 缺乏统一的 TVP 及其信任链的抽象模型。目前，考虑到不同的功能需求，已有的多个 TVP 研究方案分别侧重于不同的信任传递过程，导致它们在实现时也存在一定的差异，比如传统解决方案[2~4,6]以及 Jonathan M McCune 等人提出利用新的处理器机制为虚拟化平台提供动态信任链构建的方案[11]等，这些方案分别给出针对相应平台的特定信任链模型，不具有普遍性，而且各个已有模型并未给出从底层虚拟机系统到虚拟信任根，再到用户虚拟机的完整信任链。为了全面、有效地分析 TVP 信任链，需要建立一种统一的 TVP 及其完整信任链的抽象模型，以明确定义信任链传递过程应满足的信任属性。

2) 缺乏针对 TVP 信任链分析的形式化方法。一方面，已有的形式化分析方法主要关注通信协议的安全性分析，比如 BAN 逻辑[12]、应用 π 演算[13]等，它们不适合应用系统内部程序的安全性分析（比如 TVP 信任链传递属性等），而且现有针对可信平台信任链的形式化分析通常或者侧重于本地信任链的构建与验证[7,8]，或者侧重于信任属性的远程证明[9,10]，因此不够全面且不具有普适性；另一方面，由于虚拟化平台的特性（如多虚拟机并发、内存隔离、访问控制等），使得针对普通可信平台的分析方法不能直接应用于 TVP。为此，需要研究针对 TVP 信任链属性形式化分析的新方法。

本文针对上述问题，首先建立了 TVP 及其完整的信任链模型，并详细定义了 TVP 信任链的信任属性。与已有只针对特定 TVP 实现的模型相比，该模型更为抽象，摒弃了平台具体的实现方案（比如硬件信任根与虚拟信任根的映射关系建立等），涵盖了普通可信平台信任链及虚拟化平台特有的上层用户虚拟机的信任链。然后根据 TVP 信任链的分析需求，对安全系统逻辑（LS^2, logic of secure system）的语法、语义及证明规则进行针对性扩展，并对这种规则扩展给出严格的正确性证明。最后，基于扩展 LS^2 形式化验证了 TVP 信任链有条件地满足传递过程的正确性、唯一性，对实例系统信任链的分析表明本文提出的信任链模型的通用性及扩展 LS^2 方法的有效性。

2 TVP 及其信任链建模

本文主要针对 TVP 信任链传递的形式化分析，因此不考虑虚拟化平台自身的固有安全机制，比如虚拟机监控器（VMM，virtual machine monitor）的特权操作、虚拟机之间的隔离及内存操作控制等，可参考 Gilles Barthe 等给出的形式化描述与分析[14]。

2.1 TVP 信任模型

HP、IBM 等研究机构都针对虚拟化环境提出并构建了相应的 TVP[2~4]，但这些研究主要侧重于具体应用场景的功能实现，缺乏一种抽象、通用的 TVP 定义。TVP 在物理上体现为一个支持虚拟化技术的可信主机，它与普通可信计算平台的区别主要在于：1）拥有构建于硬件可信芯片可信平台模块（TPM, trusted platform module）基础上的虚拟信任根；2）并发地为多个客户应用虚拟机提供信任环境。基于已有研究方案，本文给出如图 1 所示的 TVP 基本运行架构。

从功能上看，TVP 主要分为 4 个层次，硬件信任根 TPM 作为最底层，是平台信任的物理保障。第二层主要包括 VMM 及管理域（主要是其内核及相关域管理工具，简记为 $admindom_{ker}$），它们通常被认为是 TVP 的可信计算基（TCB, trusted computing

base)。第三层是 vRT，由于实现方案不同（如图 1 中 a、b 所示），其加载过程可能是传统信任链的一部分，或直接利用动态加载机制如动态度量信任根（DRTM，dynamic root of trusted measurement）机制启动，这使得它或者成为 TCB 的一部分，或者作为应用进程单独存在。最上层是用户虚拟机，是与用户应用密切相关的部分。基于上述分析，本文从功能角度给出以下 TVP 的抽象定义。

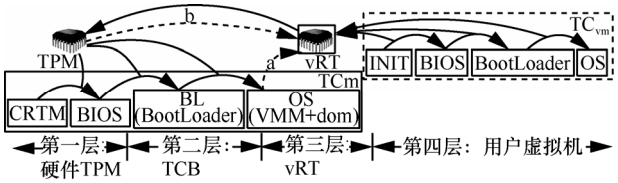

图 1　TVP 基本运行架构

定义 1　TVP 是具有可信功能的虚拟化计算平台，它主要包含 2 类功能组件：TVP:={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、管理域及用户虚拟机等，它们是利用虚拟化技术为用户提供资源与服务的主体；信任根（RT，root of trust）是构建 TVP 信任环境的基础，也是 TVP 的核心组件，对虚拟化平台来说，它包括硬件可信芯片 TPM 和 vRT。

对于 TVP 的主机 M，根据其类型进一步细化为 M:={m,vm}，其中，m:={vmm，$admindom_{ker}$}，特指底层的 VMM 及 $admindom_{ker}$，它们是 TVP 的 TCB 的主要组成部分。vm:={vm_1,⋯,vm_n}，表示虚拟化平台上层的用户虚拟机 vm 集合*。

相似地，TVP 的信任根也进一步分类为 RT:={TPM, vRT}，其中，TPM 是硬件信任根，主要用于为物理平台提供信任保障，它拥有非易失存储及密钥存储等固有特性；vRT 在功能实现上可表现为 m 中内核组件或独立的可信组件，这里将其抽象为一个独立功能组件，通过特定的映射关系与硬件信任根 TPM 关联以确保其可信性，即 vRT 依赖于 TPM，它以软件形式体现，用于为上层用户虚拟机提供信任保障。

因此，TVP 从功能角度可定义为 TVP:={(m, TPM), (vm_1, vRT_1),⋯,(vm_n, vRT_n)}，其中，m 必须使用 TPM 来构建信任，而虚拟机 vm 则是利用其相应的 vRT 来构建信任。

* 如无特别说明，本文所使用的 vm 均泛指任意用户 i 的虚拟机 vm_i。

2.2　TVP 信任链及其信任属性

TCG 组织从实体行为预期性角度给出可信的定义，并采用装载前度量的方案，给出了信任链传递和控制权转移的过程。与普通可信计算平台类似，TVP 的信任链同样需要保障平台能够基于信任根，通过逐级的信任传递，构建虚拟化平台的可信运行环境。由于虚拟化平台自身的特殊性，它要求并发地执行多个用户虚拟机实例，因此，其信任链与普通可信平台的信任链存在不同（如图 1 所示）。根据定义 1 可知，TVP 的信任链不仅包括普通可信平台从第一层到第二层的信任传递（主机 m），还要增加之后的第三层 vRT 与第四层用户虚拟机的信任传递，而且第四层中的用户虚拟机还需要多个实例并发执行，使得信任传递会出现多个不同分支，这与可信计算最初构建信任环境的思想并不一致。

为了确保这种信任传递的正确性，需要对 TVP 信任链进行形式化验证，证明在程序控制权传递过程中，各个进程的确能够按照预期执行，整个过程不存在信任缺失（比如存在其他程序执行、加载等情况），而且能够对外证明上述属性。本文将上述验证目标抽象为信任链的信任属性（TP，trusted property），这种信任属性的验证包括 2 个方面，一方面是信任链在本地平台构建过程中的唯一性、正确性验证，另一方面是平台向外部实体 R 证明自己确实构建了该信任链。其抽象定义如下。

定义 2　（TVP 信任链的信任属性 TP_{TVP}）TVP 的信任属性定义为一个二元组 TP_{TVP}:={TC_{TVP}, Ver_{TVP}}，其中，TC_{TVP} 表示 TVP 信任链构建时所包含的可信程序传递序列，Ver_{TVP} 表示对该信任链执行序列的远程验证。按照 2.1 节中对 TVP 中相应功能组件的定义，该信任属性可以进一步细化为 TP_{TVP}:={(TC_m, TC_{vRT}, TC_{vm}),(Ver_m, Ver_{vRT}, Ver_{vm})}。可见，该信任属性根据组件类型可分为 3 类：主机 m 的信任属性 TP_m、vRT 信任属性 TP_{vRT} 及用户虚拟机的信任属性 TP_{vm}。

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，即主机 m 在本地正确地完成从可信度量根（CRTM，core root of trust for measurement）到上层用户应用的可信启动过程：(CRTM → BL → OS → App)$_{TPM}$，且在信任传递过程中不存在其他程序代码加载。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，使远程验证者确信 TVP 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}}，表示 vRT 的本地可信加载及其对外的证明。需要注意的是，vRT 的信任属性与其实现方式密切相关（图 1 中的 a、b），它可能实现为一个微内核系统或一个应用进程，而且需要建立 vRT 与硬件信任根之间的强依赖关系，以硬件信任根保障软件 vRT 的可信。

3) vm 的信任属性与上述主机 m 的信任属性类似，表示为 TP_{vm}:={TC_{vm},Ver_{vm}}，其中，TC_{vm}:=(INIT $\rightarrow$ BL $\rightarrow$ OS $\rightarrow$ App)$_{vRT}$，表示 vm 的从初始启动程序 INIT 开始的本地信任链传递过程。与 TVP 平台主机 m 直接依赖于硬件信任根 TPM 有所不同,TC_{vm} 成立的关键是 vRT 的正确加载。Ver_{vm}:= Verify(vm,TC_{vm})表示 vm 信任链的外部验证。

上述可信属性 TP_{TVP} 的定义蕴含一个成立的前提，即 TVP 信任链构建过程一定是从主机 m $\rightarrow$ vRT $\rightarrow$ vm。此外，vm 的信任属性 TP_{vm} 必须依赖于 TP_{vRT}，即其信任属性建立于硬件信任根与 vRT 之上，而 vRT 的信任构建还与其实现方案密切相关。在本文建立的 TVP 模型中，对 vRT 不同实现方案进行了抽象：如果是图 1 中的 a，则需要依赖于 TP_m，如果是图 1 中的 b，则依赖于 DRTM 机制。

3 LS^2 及其扩展

3.1 LS^2 基本组成及特点

2009 年，CMU 大学的 Anupam Datta 等人提出一种可扩展的 LS^2，用于对安全应用系统的安全属性进行推理分析[15]。LS^2 基于协议组合逻辑（PCL，protocol composition logic），适合于对具有明确时序关系的复杂安全应用系统（比如 TVP）的安全属性进行分析。

LS^2 主要由 3 部分组成：编程模型、LS^2 语义语法和 LS^2 证明系统，其中，编程模型用于对安全应用系统中的行为、参与主体等进行建模，比如基本的密码学操作、内存控制及程序跳转等。LS^2 语义语法用于建立目标系统抽象的安全属性，通常以模态公式 $[P]_I^{t_b,t_e} A$ 表示，该式的含义为：线程 I 在时间段(t_b,t_e]顺序地执行程序 P 时，公式 A 成立。LS^2 的证明系统用于为上述抽象安全属性提供形式化推理证明，是 LS^2 的核心，需要对其正确性进行严格的证明。由于 LS^2 语义直观、便于扩展，而且不需要对敌手进行单独建模，并支持对未知代码加载执行的推理，因此适合对复杂应用系统安全属性的分析验证，具有广阔的应用前景。

目前，已有的形式化分析方法（比如 BAN 逻辑、应用 π 演算等）主要侧重于对网络系统通信协议的安全属性进行建模分析，而对于应用系统内部程序的安全性分析支持不够，尤其是无法分析那些影响系统安全的不可预知程序的加载行为（比如可信计算平台信任链传递过程中的程序控制权转移）。与这些方法相比，LS^2 的主要特点在于：1）语义简单且具有可扩展性，该逻辑利用已有进程演算的方法，对密码操作、并发进程之间的网络通信等进行抽象，便于对安全应用系统功能进行建模，而且可扩展新的安全操作，比如适用于可信计算环境中的平台配置寄存器（PCR，platform configuration register）的扩展操作等；2）不需要专门对敌手行为进行推理分析，它将攻击者作为程序执行时的一个额外线程，在遵守逻辑系统规则与约定的条件下，隐式地与正常参与的线程并发执行，证明系统确保违反安全属性的可能攻击者能够被检测；3）适合于对有时序规则的动态加载代码执行的行为分析，它利用新的不变量规则如程序跳转 Jump、重置 Reset 等来推理未知代码动态加载的行为。

上述特点使得 LS^2 非常适合用于对可信平台信任链传递过程中程序加载的不可预知性进行分析验证，因此，本文选择 LS^2 对 TVP 信任链进行分析验证。

3.2 针对 TVP 的 LS^2 扩展

由于已有 LS^2 是一种通用的系统安全属性分析方法，其基本编程语言、语义语法等无法充分描述 TVP 的程序行为及其信任链模型，进而不能直接应用该方法对相应的信任属性进行分析验证。因此，本文基于第 3 节 TVP 的抽象模型及其信任链的信任属性定义，对原始 LS^2 进行了如图 2 所示的扩展。

1) 编程语言及操作语义的扩展

语义扩展主要是根据第 2 节中 TVP 的定义，对原有 LS^2 中无法精确表示的主体对象进行进一步细化，主要包括主机、内存及与主机相关的线程等。对于操作语义来说，需要新增有关 vm 的操作定义(Reset vm)和(Jump vm)，分别表示 vm 的重置和程序跳转，其中，涉及到 vm 自身的初始启动程序 init

以及内存锁定处理，相应的推理规则在证明系统中进行扩展。本文将其作为一个特殊功能实体（单一程序）进行抽象，因此并未定义其内部的跳转操作，其 Reset 操作也是作为一个单独程序的重载来处理，因此不需要对操作语义进行新的扩展。

扩展的编程语言：

Machine m,vm,vRT

Location $l := \mathrm{m}.disk.k \mid \mathrm{m}.pcr.k \mid \mathrm{m}.dpcr.k \mid \mathrm{m}.vpcr.\mathrm{vm}.k$

Thread id $I, J ::= < \hat{X}, \eta, \mathrm{m} > \mid < \hat{X}, \eta, \mathrm{vm} >$

扩展的操作语义：

(Reset vm) $\iota, \sigma, T_1 \mid \cdots \mid T_n \to \iota[locs(\mathrm{vm}) \mapsto -],$
$\sigma[(locs(\mathrm{vm}) - locs(\mathrm{vm}, disk)) \mapsto init],$
$(T_1 \mid \cdots \mid T_n) - \{\mathrm{vm}\} \mid [\mathrm{Boot}(\mathrm{vm})]_l$
$(I = < \hat{\mathrm{vm}}, \eta, \mathrm{vm} >, \eta \quad fresh)$
(Jump vm) $[\mathrm{Jump}\ \ P]_l \to [P]_l$

扩展的 LS² 语法：

$M ::= \mathrm{Mem}(l, e) \mid \mathrm{IsLocked}(l, I) \mid \mathrm{Reset}(\mathrm{m}, I)$
$\mid \mathrm{Reset}(\mathrm{vm}, I) \mid \mathrm{Reset}(\mathrm{vRT}, I) \mid \mathrm{Jump}(I, e) \mid$
$\mathrm{Latelaunch}(\mathrm{m}, I) \mid \mathrm{Contains}(e, e') \mid e = e' \mid t \geq t' \mid \mathrm{Honest}(\hat{X}, \vec{P})$

扩展的证明规则：

$$\frac{(\forall Q \in IS(\mathrm{ININ}(\mathrm{vm})) \wedge (\neg \mathrm{Reset}(\mathrm{vRT}) on(t_b, t_e))). \vdash [Q]_I^{t_b, t_e} A(t_b, t_e)}{\vdash \mathrm{Reset}(\mathrm{vm}, I) @ t \supset \forall t'.(t' > t) \supset A(t, t')} \mathrm{Reset}_{\mathrm{vm}}$$

$$\frac{(\forall Q \in IS(P) \wedge (\neg \mathrm{Reset}(\mathrm{vRT}) on(t_b, t_e))). \vdash [Q]_I^{t_b, t_e} A(t_b, t_e)}{\vdash \mathrm{Jump}(I, P) @ t \supset \forall t'.(t' > t) \supset A(t, t')} \mathrm{Jump}_{\mathrm{vm}}$$

图 2　针对 TVP 平台的 LS² 扩展

2) LS² 语法扩展

由于在 TVP 中存在 3 种不同的主机实体，因此需要定义各自的 Reset 动作，需要注意的是，Reset(m)一定会导致 Reset(vRT)和 Reset(vm)，这也符合 TVP 的实际执行过程。由于 Reset(vRT)和 Reset(vm)在 TVP 中可以各自独立执行，为了构建 vm 的可信运行环境，必须满足 vRT 先于 vm 启动，且 vm 重启时 vRT 保持正常运行而不能重启。

3) 证明系统扩展

证明系统是 LS² 的核心，直接关系到安全属性验证过程的正确性、有效性。除了原有 LS² 中主机 m 的 Reset、Jump 规则，本文扩展定义了 vm 的 $\mathrm{Reset}_{\mathrm{vm}}$ 和 $\mathrm{Jump}_{\mathrm{vm}}$ 规则。

$\mathrm{Reset}_{\mathrm{vm}}$ 规则主要针对虚拟机的重启。该规则表示的是在 TVP 中 vm 重新启动后仍然保持其信任传递属性（公式 A）成立。该规则成立的前提条件是 vRT 在 vm 启动过程中未重新启动，即 $\neg$Reset(vRT)$on(t_b, t_e)$。

规则 $\mathrm{Jump}_{\mathrm{vm}}$ 针对未知程序加载。该规则表示在 TVP 的 vm 中程序 P 在跳转后仍然保持之前的可信属性（公式 A）成立，该规则成立的前提条件与 $\mathrm{Reset}_{\mathrm{vm}}$ 相同：$\neg$Reset(vRT)$on(t_b, t_e)$，即 vRT 未被重置。

本文对 LS² 的扩展主要包括 2 个方面，一方面是对 LS² 编程语言、操作语义及语法进行的扩展，另一方面是对其证明系统的扩展。对编程语言、操作语义及语法的扩展主要是从应用需求出发，对其能够表述的语义进行丰富，这种扩展主要用于对 TVP 信任链的安全属性进行定义与描述，因此不需要对其进行正确性证明。由于安全属性的分析验证依赖于 LS² 证明系统，证明系统的正确性与安全属性验证结论是否有效直接相关，因此，必须对上述扩展的 LS² 证明系统（规则）提供严格的正确性、有效性证明，其详细证明过程参见附录。

4　TVP 信任链分析

4.1　基本假定

在对 TVP 信任链属性分析之前本文假定以下条件是满足的：1）TVP 中涉及到的所有系统镜像文件（包括主机 m 及各个用户虚拟机）的完整性未受破坏，且用户虚拟机已预先植入所需的可信度量及证明代理；2）主机 m 支持动态加载 DRTM 技术，能够为 vRT 提供动态的可信运行环境；3）vRT 的平台身份密钥（AIK，attestation identity key）已得到可信第三方的认证并颁发证书，这里不考虑其具体实现方案（参见 vTPM[2]及 TrustVisor[11]等）；4）远程验证方案基于 TCG 组织给出的完整性报告协议，且在远程挑战者 R 与本地 TVP 之间已经建立了安全信道。

根据定义 1 和定义 2，本文对 TVP 信任链的信任属性分析验证主要包括 2 部分：本地信任链构建的验证及该信任链的远程验证，如图 3 所示。其中，对主机 m 和 vRT（将其作为一个单独的软件组件扩展信任传递）的信任属性证明可参考 Anupam Datta 等人[15]基于 LS² 对普通可信计算平台的可信属性进行的分析过程。在此基础上，本文重点对 vm 的信任链进行分析，一方面验证在 TVP 中，利用 vRT 所构建的本地 vm 的信任链是否唯一、正确；另一方面确保远程挑战者 R 能够验证该 vm 的确基于该信任链构建了信任环境。

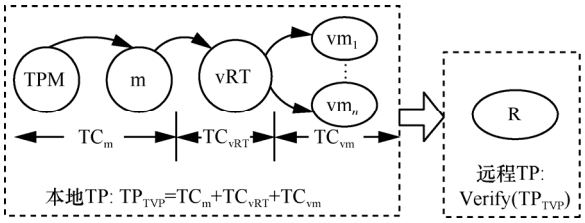

图 3 TVP 的信任传递证明

4.2 信任链的本地验证

1) 本地程序执行

根据 2.2 节中 TVP 用户虚拟机信任属性 TP_{vm} 定义，其信任链本地执行过程中涉及到的程序如图 4 所示。

```
Latelaunch(m) ≡ vRT = read m.SLB;
                extend m.dpcr.k, vRT;
                Jump vRT;
...
INIT(vm) ≡ b = read vm.bl_loc;
           extend m.vpcr.vm, b;
           Jump b
BL(vm) ≡ o = read vm.os_loc;
           extend m.vpcr.vm, o;
           Jump o
OS(vm) ≡ a = read vm.app_loc;
           extend m.vpcr.vm, a;
           Jump a
APP(vm) ≡ ⋯
```

图 4 TVP 中 vm 信任链传递

程序执行流程：首先确保 vRT 的正确加载并运行；然后 vRT 将控制权传递给初始程序 INIT(vm)，它从虚拟机内存地址 vm.*bl_loc* 中读取 Bootloader 中的代码 *b*，将其扩展到一个虚拟 PCR 中（其中，m.*vpcr*.vm 表示该虚拟机在这里存储所有相关度量值，且该虚拟机的度量值存储于主机 m 而不是 vm）；之后执行指令 Jump *b* 将控制权交给 Bootloader；Bootloader 继续按序从内存 vm.*os_loc* 读取 OS 的代码 *o*，将其扩展到 m.*vpcr*.vm，然后转换控制权给 OS；OS 执行相似的过程将控制权交给虚拟机的应用代码 *a*。

在此过程中，要求 TVP 的用户虚拟机必须在 vRT 成功加载之后启动，否则会导致在 vRT 启动之前的 vm 无法使用该 vRT，将其表示为 $\text{Honest}(\text{TPM}_{\text{DRTM}}(\text{m}) \succ \text{vRT}_{\text{SRTM}}(\text{vm}))$。此外，TVP 在启动 vm 时，相应的线程 *J* 对必须能够对当前 vm 对应的虚拟 PCR 值有锁控制，这种控制对潜在的攻击者也成立，表示为

$$\begin{aligned}&\text{ProtectedSRTM}(\text{vm}) = \forall t, J.(\text{Reset}(\text{vm}, J)@t)\\&\quad \supset (\text{IsLocked}(\text{m}.vpcr.\text{vm}, J)@t)\end{aligned}$$

由于 vRT 被抽象为一个单独的软件程序（无论其实现形式是独立的轻量级可信执行环境或仅是提供可信功能的应用进程），利用 Latelaunch(m)动态加载机制确保其可信执行，即 J_{DRTM} 成立[15]。

2) 本地可信属性描述及证明

根据上述信任链传递中程序执行过程可知，最终体现 vm 信任链的是虚拟平台的 *vPCR* 值，它与执行程序之间存在唯一性、确定性映射。因此，基于定义 2 及上述映射关系，可将 vm 的本地信任传递属性归纳为：如果最终的 *vPCR* 中度量值序列是正确的值，那么在该虚拟机上信任链所加载的程序顺序就是正确的。即 vm 的本地信任传递属性就是要求所有相应启动程序如 Bootloader、OS、APP 等都能按确定的先后顺序加载。以扩展 LS^2 将这种顺序形式化表示为

$$\begin{aligned}&\text{MeasuredBoot}_{\text{SRTM}}(\text{vm}, t)\\&\forall I \in m.\exists t_T.\exists t_B.\exists t_O.\exists J \in \text{vm}.(t_T < t_B < t_O < t)\\&\wedge(\neg\text{Reset}(\text{vRT}, I)on(t_T, t]) \wedge (\text{Reset}(\text{vm}, J)@t_T)\\&\wedge(\text{Jump}(J, \text{BL}(\text{vm})@t_B)) \wedge (\text{Jump}(J, \text{OS}(\text{vm})@t_O))\\&\wedge(\neg\text{Reset}(\text{vm}, J)on(t_T, t]) \wedge (\neg\text{Jump}(J)on(t_T, t_B))\\&\wedge(\neg\text{Jump}(J)on(t_B, t_O))\end{aligned}$$

上述公式表示：如果 TVP 的 vm 基于信任链构建了本地信任环境，则其启动过程一定是从 BL（Bootloader）跳转到 OS，而在此期间不会有其他程序执行。这就需要证明上述程序启动序列与 *vPCR* 值之间的一一映射关系。基于前文的假定前提，要证明的信任链本地信任属性如下。

定理 1 如果 vRT 加载（即 J_{DRTM}）成功，而且与该 vm 启动过程对应的 *vPCR* 值为 seq(INIT(vm), BL(vm),OS(vm),APP(vm))，那么该 vm 的本地信任链传递过程就是唯一的、正确的，即确定地从 INIT(vm)到 BL(vm)再到 OS(vm)。该信任属性形式化表示为

$$\begin{aligned}&J_{\text{DRTM}} + \text{ProtectedSRTM}(\text{vm}) +\\&\text{Mem}(\text{m}.vpcr.\text{vm}, \text{seq}(\text{INIT}(\text{vm}),\\&\text{BL}(\text{vm}),\text{OS}(\text{vm}),\text{APP}(\text{vm})))\\&\supset \text{MeasuredBoot}_{\text{SRTM}}(\text{vm}, t)\end{aligned}$$

证明 首先，根据前提条件可知在时间点 *t* 有

$$\begin{aligned}&\text{Mem}(\text{m}.vpcr.\text{vm},\ \text{seq}(\text{INIT}(\text{vm}),\text{BL}(\text{vm}),\\&\text{OS}(\text{vm}),\text{APP}(\text{vm})))\end{aligned}$$

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.*vpcr*.vm 中，即

$$\begin{aligned}&\exists t_T,t_1,t_2,t_3,J.(t_T\le t_1<t_2<t_3<t)\\&\wedge(\mathrm{Mem}(\mathrm{m}.vpcr.\mathrm{vm},\mathrm{seq}(\mathrm{INIT(vm)},\mathrm{BL(vm)},\\&\qquad\qquad\mathrm{OS(vm)},\mathrm{APP(vm)}))@t)\\&\wedge(\mathrm{Mem}(\mathrm{m}.vpcr.\mathrm{vm},\mathrm{seq}(\mathrm{INIT(vm)},\mathrm{BL(vm)},\mathrm{OS(vm)}))@t_3)\\&\wedge(\mathrm{Mem}(m.vpcr.\mathrm{vm},\mathrm{seq}(\mathrm{INIT(vm)},\mathrm{BL(vm)}))@t_2)\\&\wedge(\mathrm{Mem}(m.vpcr.\mathrm{vm},\mathrm{seq}(\mathrm{INIT(vm)}))@t_1)\\&\wedge(\mathrm{Reset}(\mathrm{vm},J)@t_T)\\&\wedge(\neg\mathrm{Reset}(\mathrm{vm})\mathrm{on}(t_T,t])\end{aligned}$$

接下来考虑图 4 中程序执行过程，最先执行的操作是启动 vm, 即 Reset(vm, *J*), 然后 vm 利用 vRT 执行第一个信任程序 INIT(vm)。利用图 2 中规则 $\mathrm{Reset}_{\mathrm{vm}}$，建立并证明程序 INIT(vm) 的不变量 $A_{\mathrm{INIT(vm)}}(t_b,t_e)$：在某个时间 t_B，程序会跳转到 *b* 且在其他时间不会有程序跳转，内存位置（即 *vPCR* 值）被该线程锁定。即有以下属性(1)成立

$$\begin{aligned}A_{\mathrm{INIT(vm)}}(t_b,t_e)=\forall t',b,o.(((&\mathrm{Mem}(\mathrm{m}.vpcr.\mathrm{vm},\\&\mathrm{seq}(init,b,o))@t'\wedge(t_b<t_T<t'\le t_e))\\&\supset\exists t_B.((t_T<t_B<t')\wedge(\mathrm{Jump}(J,b)@t_B)))\\&\wedge(\mathrm{IsLocked}(\mathrm{m}.vpcr.\mathrm{vm},J)@t_B))\end{aligned}\tag{1}$$

类似地，当程序跳转并执行 BL(vm)时，利用 $\mathrm{Jump}_{\mathrm{vm}}$ 规则建立并证明 BL(vm) 的不变量 $A_{\mathrm{BL(vm)}}(t_b,t_e)$：在 BL(vm)执行后的某个时间点，程序必然跳转到 OS(vm)且期间不会有其他程序跳转，即有以下属性(2)成立

$$\begin{aligned}A_{\mathrm{BL(vm)}}(t_B,t_e)=\forall t',o,a.(((&\mathrm{Mem}(\mathrm{m}.vpcr.\mathrm{vm},\\&\mathrm{seq}(init,b,o,a)))@t'\wedge(t_B<t'\le t_e))\\&\supset\exists t_O.((t_B<t_O<t')\wedge(\mathrm{Jump}(J,o)@t_O)))\\&\wedge(\neg\mathrm{Jump}(J)on(t_B,t_O)))\end{aligned}\tag{2}$$

根据式(1)、式(2)可知，如果前提条件满足，那么 vm 上执行程序的顺序一定是从 INIT(vm)到 BL(vm)再到 OS(vm)

$$\begin{aligned}&\exists t_T,t_B,t_O,J.(t_T<t_B<t_O<t)\wedge\\&(\neg\mathrm{Reset}(\mathrm{vm},I)on(t_T,t])\wedge(\mathrm{Reset}(\mathrm{vm},J)@t_T)\\&\wedge(\mathrm{Jump}(J,\mathrm{BL(vm)}@t_B))\wedge(\neg\mathrm{Jump}(J)on(t_T,t_B))\\&\wedge(\mathrm{Jump}(J,\mathrm{OS(vm)}@t_o))\wedge(\neg\mathrm{Jump}(J)on(t_B,t_O))\end{aligned}$$

定理 1 得证。

虽然上述证明过程未显式地描述攻击者的存在，但已经蕴含着攻击场景。比如，在 INIT(vm)之后跳转到 *b* 的过程中，由于 *b* 是从内存 vm.*bl_loc* 读取的，而该位置可能在之前已被攻击者线程写入其他程序，但可信计算技术提供的度量扩展机制使得能够推理只有得到正确的内存值时才能继续运行下一个程序。同样的，需要证明从 *b* 跳转到 *o* 的正确性。利用 LS^2 提供的 $\mathrm{Reset}_{\mathrm{vm}}$ 规则和 $\mathrm{Jump}_{\mathrm{vm}}$ 规则，证明 TVP 上本地信任链传递的唯一性、正确性成立。

4.3　信任链的远程验证

TVP 的 vm 需要向外部挑战者证明自己所声称信任属性，即其信任链传递过程中所执行程序的确定序列，使外部挑战者相信它的确按上述信任链构建了可信执行环境，需要证明 $\mathrm{MeasuredBoot}_{\mathrm{SRTM}}$(vm,*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[16,17]，给出 vm 信任传递的远程验证过程中涉及到的程序，如图 5 所示。

$$\begin{aligned}&\mathrm{vRT}_{\mathrm{SRTM}}(\mathrm{vm})\equiv\\&\qquad w=\mathrm{read\ m}.vpcr.\mathrm{vm};\\&\qquad r_{\mathrm{vm}}=\mathrm{sign}(vPCR(\mathrm{vm}),w),AIK^{-1}(\mathrm{vm});\\&\qquad \mathrm{send}\ r_{\mathrm{vm}}\\&\mathrm{Verifier(vm)}\equiv\\&\qquad sig=\mathrm{receive};\\&\qquad v=\mathrm{verify}\ sig,AIK(\mathrm{vm});\\&\qquad \mathrm{match}\ v,(vPCR(\mathrm{vm}),\\&\qquad \mathrm{seq(INIT(vm),BL(vm),OS(vm),APP(vm))})\end{aligned}$$

图 5　TVP 中虚拟机信任传递的远程验证程序

首先, vm 读取本地虚拟 *PCR* 值, 用自己的 AIK 签名（$AIK^{-1}(\mathrm{vm})$）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 vm 拥有所声称的可信属性，否则验证失败。在此过程中，vRT 必须先于 vm 启动以确保 vm 信任链建立，且远程验证者与 vm 应是不同实体，以保证该验证过程的有效性。

将这些前提条件形式化表示为

$$\begin{aligned}&\Gamma_{\mathrm{SRTM}}=\{\mathrm{Honest}(\hat{AIK}(\mathrm{vm})),\\&(\mathrm{TPM}_{\mathrm{DRTM}}(\mathrm{m})\succ vRT_{\mathrm{SRTM}}(\mathrm{vm})),V\ne\hat{AIK}(\mathrm{vm})\}\end{aligned}$$

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 2 如果远程验证者确认 vm 提供的度量值是唯一的、正确的，那么该 vm 对应的 PCR 值一定是如下的确定序列 seq(INIT(vm),BL(vm),OS(vm),APP(vm))，因为根据定理 1 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

$$\Gamma_{\mathrm{SRTM}} \vdash [\mathrm{Verifier(vm)}]_V^{t_b,t_e} \exists t.(t < t_e) \wedge (\mathrm{Mem}(m.pcr.\mathrm{vm}, \mathrm{seq}(\mathrm{INIT(vm)}, \mathrm{BL(vm)}, \mathrm{OS(vm)}, \mathrm{APP(vm)})@t) \quad (3)$$

$$\Gamma_{\mathrm{SRTM}}, \mathrm{Protected}_{\mathrm{SRTM}}(\mathrm{vm}) \vdash [\mathrm{Verifier(vm)}]_V^{t_b,t_e} \exists t.(t < t_e) \wedge \mathit{MeasuredBoot}_{\mathrm{SRTM}}(\mathrm{vm}, t) \quad (4)$$

这两个属性有递进关系，即如果属性(3)证明成立，则属性(3)利用定理 1 的结论直接可证。因此，这里需要明属性(4)。

证明 首先根据前提假定及 $[\mathrm{Verifier(vm)}]_V^{t_b,t_e}$，利用公理 VER 可直接得到

$$[\mathrm{Verifier(vm)}]_V^{t_b,t_e} \exists t_S, e, I.(t_S < t_e) \wedge \hat{I} = AI\hat{K}(\mathrm{vm}) \wedge \mathrm{Contain}(e, SIG_{AIK(\mathrm{vm})^{-1}}\{|PCR(s), \mathrm{seq(INIT(vm),BL(vm),OS(vm),APP(vm))}|\}) \wedge ((\mathrm{Sent}(I,e)@t_S) \vee \exists l.(\mathrm{Write}(I,l,e)@t_S))$$

根据图 5 中的远程验证程序，建立并证明以下程序不变量：对于程序前缀 $Q \in IS(\mathrm{vRT}_{\mathrm{SRTM}}(\mathrm{vm}))$，有以下属性成立。

$$[Q]_I^{t_b,t_e}(\forall l,e,t.(t \in (t_b,t_e]) \supset \neg\mathrm{Write}(I,l,e)@t) \wedge (\forall t',e'.((t' \in (t_b,t_e]) \wedge \mathrm{Send}(I,e')@t') \supset (\exists e'', t_R.(t_R < t') \wedge (\mathrm{Read}(I, \mathrm{m}.vpcr.\mathrm{vm}, e'')@t_R) \wedge e' = SIG_{AIK(\mathrm{vm})^{-1}}\{|PCR(s), e''|\}))$$

该属性表明在验证过程中如果本地没有写入内存的操作且发送了数据 e'，则在之前的某时刻本地一定读取了值 e''，且 e' 是一个签名值。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

$$[\mathrm{Verifier(vm)}]_V^{t_b,t_e} \exists t_R, e, e'', I.(t_R < t_e) \wedge \hat{I} = AI\hat{K}(\mathrm{vm}) \wedge \mathrm{Contain}(e, SIG_{AIK(\mathrm{vm})^{-1}}\{|PCR(s), \mathrm{seq(INIT(vm),BL(vm),OS(vm),APP(vm))}|\}) \wedge (\mathrm{Read}(I, \mathrm{m}.vpcr.\mathrm{vm}, e'')@t_R) \wedge e = SIG_{AIK(\mathrm{vm})^{-1}}\{|PCR(s), e''|\}$$

分别利用等值公理 Eq 和 Read 公理，有

$$[\mathrm{Verifier(vm)}]_V^{t_b,t_e} \exists t_R, e'', I.(t_R < t_e) \wedge \mathrm{Contain}(e, SIG_{AIK(\mathrm{vm})^{-1}}\{|SIG_{AIK(\mathrm{vm})^{-1}}\{|PCR(s), e''|\}, \mathrm{seq(INIT(vm),BL(vm),OS(vm),APP(vm))}|\}) \wedge (\mathrm{Mem}(\mathrm{m}.vpcr.\mathrm{vm}, e'')@t_R) \quad (5)$$

此时需要判定 e'' 的值，根据上述推理过程可知有 2 种可能：

$$(e'' = \mathrm{seq(INIT(vm),BL(vm),OS(vm),APP(vm))}) \vee \mathrm{Contain}(e'', SIG_{AIK^{-1}(\mathrm{vm})}\{|PCR(s), \mathrm{seq(INIT(vm),BL(vm),OS(vm),APP(vm))}|\}) \quad (6)$$

根据公理 PCRC

$$\vdash (\mathrm{Mem}(\mathrm{m}.vpcr.\mathrm{vm}, e'')@t) \supset \neg\mathrm{Contains}(e, SIG_K\{|e'|\})$$

以及 $\mathrm{Mem}(\mathrm{m}.vpcr.\mathrm{vm}, e'')$ 存在的事实，可知式(6)中第 2 种可能不成立，故只有 $e'' = \mathrm{seq(INIT(vm),BL(vm),OS(vm),APP(vm))}$ 成立。

利用等值公理 Eq 对式(5)进行变换可得

$$[\mathrm{Verifier(vm)}]_V^{t_b,t_e} \exists t_R.(t_R < t_e) \wedge (\mathrm{Mem}(\mathrm{m}.pcr.\mathrm{vm}, \mathrm{seq(INIT(vm),BL(vm),OS(vm),APP(vm))}@t_R)$$

即定理 2 属性式(3)得证。利用属性式(4)结论及定义 1，可直接证明属性式(4)成立。

根据上述证明可知，在 TVP 信任链构建过程中，能够有条件地保持其信任属性，即构建信任链所需要执行的不同程序在跳转过程中，不会被其他恶意代码所控制或插入，从而不存在信任缺失的情况，且这种信任属性能够向远程验证者提供证明。

5 实例系统分析与讨论

上述分析验证过程是针对第 2 节提出的 TVP 抽象模型进行的，为了在实际系统中应用本文的分析方法，选择课题组已经构建的 TVP 实例系统[18]（如图 6 所示）并对其信任链进行分析。该实例系统基于 XEN 半虚拟化平台，其中，vRT 被实现为一个独立的可信服务域（TSD, trusted service domain），以减轻虚拟化平台管理域的运行负载。

根据第 2 节的通用 TVP 抽象模型，该实例系统信任链由 3 部分组成：第一部分如图 6 中 1)所示，即 m:={vmm, $\mathrm{admindom}_{\mathrm{ker}}$}的加载运行过程，主要包括 CRTM→BL→(VMM+OS)，由 TPM 为其提供信任保障。第二部分如图 6 中 2)所示，是由微内

核及其中的可信功能所构成的 TSD，由于 TSD 中只有内核空间代码，因此可将其看作独立运行的可信软件。第三部分是用户虚拟机的运行(图 6 中 3))，其信任链与第 2 节中定义一致。

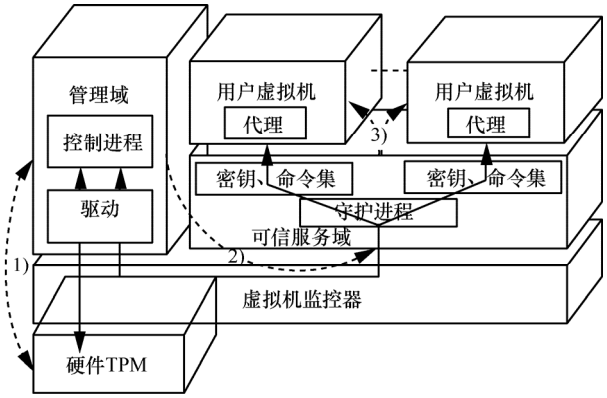

图 6　基于 XEN 的 TVP 实例系统

本文建立的通用抽象模型不关注具体系统实现细节，以上述实例系统为例，由于目前较新的 DRTM 机制对其保护的应用有诸多限制，比如要求受保护的代码自包含等，因此上述实例系统在实现时并未采用 DRTM 机制保障 TSD 的安全，而是将 TSD 作为 $\text{admindom}_{\text{ker}}$ 之后加载的第一个应用进程。这种实现上的调整不影响本文所建立的抽象模型，只需在推理时，将底层 m 原有信任链增加一个可信进程的控制传递节点即可，即将图 4 中的 latelaunch(m)更改为 $\text{TPM}_{\text{SRTM}}(\text{m})$[15]，相应地在分析过程中增加从(VMM+OS)到 TSD 的不变量证明。可见，具体的系统实现仅是对推理细节的改变，本文所建立的抽象信任链模型及其信任属性能够满足具体系统实现建模需求，不同的实现只需调整相应推理细节。

在分析过程中，验证了第 4 节中为确保 TVP 信任链的正确性所必须满足的前提条件。结合上述实例系统讨论如下。

1) 必须保障 vRT 自身的可信。在实例系统中，TSD 作为 vRT，其安全性直接关系到用户虚拟机的信任链构建。TSD 作为特殊功能域，必须利用底层硬件信任根保证其可信性，即建立硬件信任根与 TSD 之间的强绑定关系。孤立的 TSD 会使从(VMM+OS)到 TSD 的不变量不满足，从而后续获得运行权限的程序 INIT(vm)不可信，导致 vm 信任链构建失败的结论。

为此，在实例系统中除了在 TSD 与 TPM 之间构建了密钥关联关系之外，对主机 m 的信任链作了如下扩展：CRTM → BL → (VMM+OS) → TSD，以确保其 VRT 的 TSD 的可信加载。

2) 必须确保 vRT（即实例系统中的 TSD）先于用户虚拟机加载运行。在分析过程中，如果有普通用户虚拟机先于 TSD 启动，则会导致推理过程中由于内存锁操作的失败而无法证明程序跳转时的不变量成立，这是由于产生了信任环路。此外，vm 运行后的 TSD 重启动操作（即 Reset）会导致已有锁定的 PCR 值不可信（比如丢失或被破坏），从而得出信任缺失的证明结论。

在实例系统中通过对 TSD 加载运行的调度监控来解决该问题。监控进程位于实例系统的管理域内核中，由于初始信任链能够保证管理域内核可信，从而保障监控进程的可信，确保 TSD 正确的加载顺序。

6　结束语

信任链是构建 TVP 的基础，但目前缺乏针对其信任传递正确性的形式化分析。为此，本文首次建立了抽象度更高的 TVP 完整信任链的模型，并详细定义其应满足的信任属性。扩展已有 LS² 的编程语言、语义语法和证明系统，并对这种扩展提供严格的正确性证明。基于扩展 LS² 对 TVP 链传递过程的正确性、唯一性进行了形式化分析验证。对实例系统信任链的分析表明本文所建立模型的通用性及扩展 LS² 分析方法的有效性，并讨论了 TVP 信任链为保持其信任传递属性所必须满足的前提条件。

TCG 组织于 2011 年 9 月发布了虚拟可信平台架构规范[19]，为 TVP 的设计与实现提供架构性指导，但并未给出其信任链构建及远程证明的详细方案，本文对此做出有意义的探索，满足了 TVP 信任链形式化分析的需求，并为可信计算平台的建模分析提供参考。未来将在此基础上，研究基于扩展 LS² 的安全属性自动化分析方法，进一步提高其分析效率。

附录：LS² 扩展规则的正确性证明

为了证明 TVP 信任链的信任属性，本文对 LS² 的证明系统规则进行如下的扩展，2 个规则各自含义及其正确性证明如下。

$$\frac{(\forall Q \in IS(\text{INIT}(\text{vm})) \wedge (\neg \text{Reset}(\text{vRT}) on(t_b,t_e))). \vdash [Q]_I^{t_b,t_e} A(t_b,t_e)}{\vdash \text{Reset}(\text{vm},I)@t \supset \forall t'.(t' > t) \supset A(t,t')}$$

$$\frac{(\forall Q \in IS(P) \wedge (\neg \text{Reset(vRT)} on(t_b,t_e))). \vdash [Q]_I^{t_b,t_e} A(t_b,t_e)}{\vdash \text{Jump}(I,P)@t \supset \forall t'.(t' > t) \supset A(t,t')} \quad \text{Jump(vm)} \tag{8}$$

1) Reset(vm) 规则

该规则的含义是：对于 TVP 的虚拟机来说，在其重新启动过程中（INIT(vm)）如果没有 vRT 重置事件(Reset(vRT))发生，则在虚拟机重新启动之后，之前的属性 A 依然成立。

证明

①与硬件信任根为单一平台提供信任保障不同，TVP 的 vRT 需要为多个不同用户虚拟机提供信任保障，用户虚拟机的错误执行可能导致信任的断裂或环路。虚拟机 vm 的 INIT(vm)表示其启动过程中的程序序列，保持其信任属性的前提是 vRT 的正常运行，如果 vRT 在此过程中发生重置 Reset，就会导致相应的信任传递断裂，新运行的 vRT 就无法保证当前 INIT(vm)的正确性，进而无法确保安全属性即不变量 A 的成立。因此，必须要求 $\neg$Reset(vRT) on (t_b,t_e) 这一前提条件满足。

②在①基础上，分析需要证明的目标：对于任何时间点 t_0，都存在一个时间迹 $\mathcal{I}$ 满足

$$\mathcal{I} \vDash^{t_0} \text{Reset}(\text{vm},I)@t \supset \forall t'.(t' > t) \supset A(t,t')$$

根据 LS^2 的正确性推理假定，假设存在 $\Gamma \vdash \varphi$ 和满足 $\mathcal{I} \vDash \Gamma$ 的任意时间迹 $\mathcal{I}$，则需要证明的事实为 $\mathcal{I} \vDash \varphi$（这里 φ 即是要证明的属性 A）。根据前面的证明目标，可得该时间迹满足 $\mathcal{I} \vDash^{t_0} \text{Reset}(\text{vm},I)@t$，因此，选择任意时间 t'使得 $t'>t$，则证明目标可变换为验证属性 $\mathcal{I} \vDash^{t_0} A(t,t')$ 成立。

由于前提条件中有 $\mathcal{I} \vDash^{t_0} \text{Reset}(\text{vm},I)@t$ 及 $\mathcal{I} \vDash {}' A@t'$ *if* $\mathcal{I} \vDash^{t'} A$ 成立，可知 $\mathcal{I} \vDash^{t} \text{Reset}(\text{vm},I)$，根据 $\vDash$ 的定义（语义判定），在主机 m 的时间点 t 上发生 Reset 归约，因此，在时间点 t，时间迹 $\mathcal{I}$ 包含 INIT(vm)中的程序（比如第一个执行程序 INIT(vm)$_l$）。根据 LS^2 定理 1：假定 $\mathcal{I}$ 是一个迹，该迹包含时间 t_b 时的线程 $[P\theta]_l$。那么，对于任何的 $t_e \geq t_b$，如果有程序前缀 $Q \in IS(P)$ 使得 $\mathcal{I} \gg [Q]_l^{t_b,t_e} \mid \theta.$ 成立，则必然存在一个程序 $Q \in IS(\text{INIT(vm)})$ 使得 $\mathcal{I} \gg [Q]_l^{t,t'} \mid .$，同时利用前缀匹配定义：$\mathcal{I} \gg [Q]_l^{t,t'} \mid .$ 蕴含着 $\mathcal{I} \vDash^{t_0} (A(t_b,t_e))(t/t_b)(t'/t_e)$，因此，通过变量替换后即可知 $\mathcal{I} \vDash^{t_0} A(t,t')$ 得证，即 Reset(vm)规则成立。

2) Jump(vm) 规则

该规则的含义是：对于虚拟化平台中的虚拟机来说，如果某程序执行时属性 A 成立，在该程序跳转后，如果 vRT 未重启，则程序跳转之前的属性 A 依然成立。

证明

①与硬件信任根为普通可信平台提供的可信功能一样，vm 中每一次程序跳转过程都需要 vRT 对目标程序进行度量和扩展，因此，在 vm 中程序跳转过程中，vRT 必须能够正确加载并提供信任功能，否则就无法保证 vPRC 中度量值的正确性，进而无法确保不变量公式 A 的成立。因此，在 Jump 规则的假设中，必须包含前提条件 $\neg$Reset(vRT) on(t_b,t_e)。

②需要证明对于任何基础时间点 t_0 来说，有以下推论成立

$$\mathcal{I} \vDash^{t_0} \text{Jump}(I,P)@t \supset \forall t'.(t' > t) \supset A(t,t')$$

同 1），假定 $\mathcal{I} \vDash^{t_0} \text{Jump}(I,P)@t$，并选择任意时间点 t' 使得 $t'>t$，则需要的证明目标是 $\mathcal{I} \vDash^{t_0} A(t,t')$。

根据假设 $\mathcal{I} \vDash^{t_0} \text{Jump}(I,P)@t$ 及 $\mathcal{I} \vDash {}' A@t'$ *if* $\mathcal{I} \vDash^{t'} A$，可知 $\mathcal{I} \vDash^{t} \text{Jump}(I,P)$。因此，利用 $\vDash$ 的定义，线程 I 在时间点 t 进行了行为 Jump P 的规约，因此在时间点 t 时间迹 $\mathcal{I}$ 包含程序 P。利用定理 1，必然有一个程序 $Q \in IS(P)$ 使得 $\mathcal{I} \gg [Q]_I^{t,t'} \mid .$。利用此，再根据前缀匹配定义，得到 $\mathcal{I} \vDash^{t_0} (A(t_b,t_e))(t/t_b)(t'/t_e)$。从而 $\mathcal{I} \vDash^{t_0} A(t,t')$ 得证。

参考文献：

[1] REINER S, XIAOLAN Z. Design and implementation of a TCG-based integrity measurement architecture[A]. Proc of the 13th USENIX Security Symposium[C]. Berkeley, USA, 2004. 223-238.

[2] BERGER S, CACERES R, GOLDMAN K A, *et al*. VTPM: virtualizing the trusted platform module[A]. Proc of the 15th USENIX Security Symposium[C]. Berkeley, USA, 2006. 305-320.

[3] CHRIS I D, DAVID P, WOLFGANG W, *et al*. Trusted virtual platforms: a key enabler for converged client devices[A]. Proc of the ACM SIGOPS Operating Systems Review[C]. New York, USA, 2009. 36-43.

[4] BERGER S, RAMON C, DIMITRIOS P, *et al*. TVDc: managing security in the trusted virtual datacenter[A]. Proc of ACM SIGOPS Operating Systems Review[C]. New York, USA, 2008. 40-47.

[5] KRAUTHEIM F J, DHANANJAV S P, ALAN T S. Introducing the trusted virtual environment module: a new mechanism for rooting trust in cloud computing[A]. Proc of the 3rd International Conference on Trust and Trustworthy Computing[C]. 2010.211-227.

[6] 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.

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

[8] 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.
ZHANG X, HUANG Q, SHEN C X. A formal method based on non-interference for analyzing trust chain of trusted computing platform[J]. Chinese Journal of Computers, 2010, 33(1):74-81.

[9] 冯登国，秦宇. 可信计算环境证明方法研究[J].计算机学报，2008, 31(9):1640-1652.

FENG D G, QIN Y. Research on attestation method for trust computing environment[J]. Chinese Journal of Computers, 2008, 31(9):1640-1652.

[10] 冯登国，秦宇. 一种基于 TCM 的属性证明协议[J]. 中国科学, 2010,53(3):454-464.
FENG D G, QIN Y. A property-based attestation protocol for TCM[J]. Science China, 2010,53(3):454-464.

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

[12] BURROWS M, ABADI M, NEEDHAM M R.A logic of authentication[A]. Proc of the Royal Society[C]. London, UK, 1989. 233-271.

[13] ABADI M, FOURNET C. Mobile values, new names, and secure communication[A]. Proc of the 28th Symposium on Principles of Programming Languages[C]. London, ACM, 2001. 104-115.

[14] GILLES B, GUSTAVO B, JUAN D C, *et al*. Formally verifying isolation and availability in an idealized model of virtualization[A]. Proc of the 17th International Conference on Formal Methods[C]. Berlin, Springer, 2011.231-245.

[15] DATTA A, FRANKLIN J, GARG D, *et al*. A logic of secure systems and its application to trusted computing[A]. Proc of the 30th IEEE Symposium on Security and Privacy[C]. Los Alamitos, USA, 2009. 221-236.

[16] Trusted Computing Group. TCG infrastructure working group architecture part II-integrity management version 1.0[EB/OL]. http://www.trustedcomputinggroup.org, 2006.

[17] CABUK S, CHEN L Q, PLAQUIN D, *et al*. Trusted integrity measurement and reporting for virtualized platforms[A]. Proc of the International Conference on Trusted Systems[C]. Berlin, Springer, 2010. 180-196.

[18] CHANG D X, CHU X B, QIN Y, *et al*. TSD: a flexible root of trust for the cloud[A]. Proc of the IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications[C]. Liverpool. Springer, 2012, 119-126.

[19] Trusted Computing Group. Virtualized trusted platform architecture specification version 1.0 [EB/OL]. http://www.trustedcomputinggroup.org.

作者简介：

常德显（1977-），男，河南邓州人，中国科学院软件所博士生，解放军信息工程大学讲师，主要研究方向为信息安全、可信计算。

冯登国（1965-），男，陕西靖边人，中国科学院软件所研究员、博士生导师，主要研究方向为网络和信息安全。

秦宇（1979-），男，重庆人，博士，中国科学院软件所助理研究员，主要研究方向为网络与系统安全、可信计算。

张倩颖（1986-），女，河北三河人，中国科学院软件所博士生，主要研究方向为网络与系统安全、可信计算。