基于TrustZone的TEE设计与信息流形式化验证

孙海泳[1]，杨 霞[1*]，雷 航[1]，乔 磊[2]，杨 拯[1]

(1. 电子科技大学信息与软件工程学院 成都 610054；2. 北京控制工程研究所 北京 海淀区 100190)

【摘要】基于可信硬件构建安全关键应用的可信执行环境(TEE)，是嵌入式安全领域的一个研究热点。虽然底层硬件可信，但TEE软件仍可能因错误使用硬件指令或存在其他安全漏洞而导致机密信息泄露。该文基于ARM TrustZone技术提出了多层次的TEE架构，并建立了安全通信通道，用户层可信应用不能直接访问非可信环境的软硬件资源，只能通过内核层的安全通信通道API与外界通信，因此整个TEE的信息流不再受用户层影响。该文进一步提出了TEE形式化模型(TEEFM)，借助Coq辅助证明器验证了TEE信息流无干扰性并证明了TEE安全监控模块不存在整型溢出、程序返回地址异常等资源边界类软件漏洞，以此保证了TEE的自身安全性。

关 键 词 形式化方法； 信息流验证； 可信执行环境； TrustZone

中图分类号 TP311 **文献标志码** A doi:10.3969/j.issn.1001-0548.2019.02.016

The Design of TEE Based on TrustZone and Formal Verification of Information Flow

SUN Hai-yong[1], YANG Xia[1*], LEI Hang[1], QIAO Lei[2], and YANG Zheng[1]

(1. School of Information and Software Engineering, University of Electronic Science and Technology of China Chengdu 610054;
2. Beijing Institute of Control Engineering Haidian Beijing 100190)

Abstract It is a research hotspot to build a trusted execution environment(TEE) based on trusted hardware for security critical application in the security embedded area. Although the underlying hardware is trusted, vulnerabilities in the TEE software, such as the incorrect use of hardware instructions or other security vulnerabilities, can be exploited to divulge secrets. Based on ARM TrustZone, a multilayer architecture is proposed and a secure communication channel is built. The trusted applications from user layer can not access the memory of non-trusted environment directly, but communicate with external environment through the secure communication channel API. Thus, all of the information flow of TEE can hardly be influenced by the user layer. Furthermore, TEE formal module(TEEFM) is proposed and TEE information flow noninterference is verified by using Coq proof assistant. Besides, the absence of the software errors about resource boundary (e.g. integer overflow, function return address incorrect) in TEE secure monitor module is proved, which ensures the security of TEE itself.

Key words formal methods; information flow verification; trusted execution environment; TrustZone

ARM TrustZone[1]技术通过对处理器、内存、外设等硬件设备进行安全扩展，为上层软件提供相互隔离的可信执行环境TEE和通用执行环境(rich execution environment, REE)，从而将安全敏感应用与iOS、Android等通用操作系统分离。目前该技术已被广泛应用于电子银行、电子医疗等对安全性要求较高的移动应用领域。但在实际的系统设计中，为降低可信计算基(trusted computing base, TCB)的复杂度， TEE通常不实现文件系统和网络协议栈等复杂的软件模块，而借助REE相关功能进行数据存储和网络通信，但TEE与REE之间的通信模块也是目前被发现存在安全漏洞最多的地方。如高通QSEE内核的整型溢出漏洞[2]、华为RTOSck内核的TEE驱动提权漏洞[3]等，都是因TEE与REE通信模块的设计或实现不当而引发的。此外，有些安全应用未对敏感信息进行加密处理，如HTC指纹数据直接明文存储在Android侧[4]等，这些漏洞都可能被恶意利用从而对用户造成财产损失或隐私泄露。

因此，只有硬件安全隔离还远远不够，TEE软件的安全性更需加以重视。采用形式化方法验证

收稿日期：2017－10－21；修回日期：2018－03－15
基金项目：国家自然科学基金(61502031,61632005)
作者简介：孙海泳(1989－)，男，博士生，主要从事嵌入式操作系统设计与形式化验证方面的研究.
通信作者：杨霞，副教授，E-mail: xyang@uestc.edu.cn

TEE软件是增强TEE安全性的有效方法。但TEE的设计比较灵活，如TEE既可以是一体化结构的安全服务库也可以是多层次的完整操作系统，设计方案的不同会直接影响形式化验证工作的可扩展性。如文献[5]的系统设计让运行于TEE中的可信应用直接与外界通信，所以在验证TEE信息流过程中必须验证可信应用代码，但这种验证方法会降低验证的可扩展性，因为一旦更新可信应用则需要重新验证整个TEE。针对此类问题，本文提出TEE多层次架构设计方案，并在此架构基础上，针对TEE、REE共存的多隔离域安全系统，提出TEEFM。该模型将函数输入输出参数纳入行为规范的定义，从而可检测出整型溢出、程序返回地址越界等资源边界类软件漏洞。此外，TEEFM引入了安全域、观察函数等概念用于支持隔离域间信息流的验证。基于TEEFM，在Coq定理证明器中验证了安全监控模块(secure monitor module, SMM)信息流无干扰性，增强了所构建TEE的安全性。

1 基于TrustZone的TEE设计

为降低可信应用对TEE机密性的影响，提高TEE信息流形式化验证的可扩展性，本文采用多层次的TEE架构设计，使得可信应用只能通过可信内核的安全通道API与外界通信。本文首先介绍多层次TEE架构设计，然后介绍安全通信通道设计，最后讨论本文所提TEE多层架构在形式化验证方面的优势。

1.1 TEE多层架构设计

整个TEE软件栈从上到下分为用户层、内核层、TrustZone层，如图1所示。

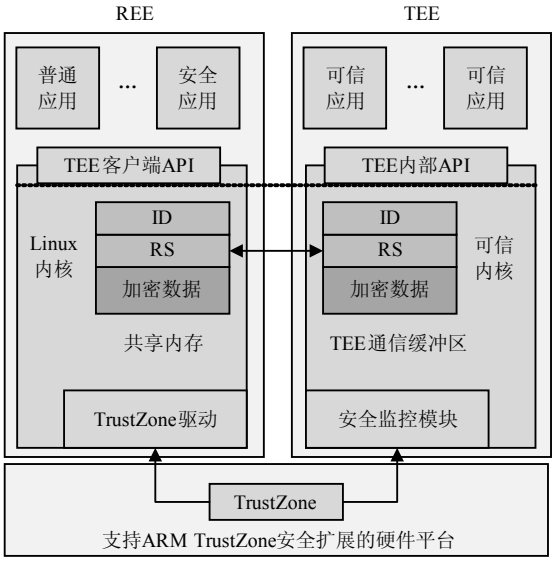

图1 TEE多层架构

在用户层，REE中能够请求安全服务的应用程序称作安全应用，其余应用程序称作普通应用，TEE中的应用程序称作可信应用。安全应用请求可信应用服务处理机密信息，由于两者位于不同的安全域，服务请求会涉及环境间的切换及通信，为保证交互细节对开发者透明，REE与TEE分别封装了TEE客户端API和TEE内部API。

在内核层，Linux内核为REE应用程序提供内存管理、进程管理、网络通信、文件系统管理等较完备的系统服务，可信内核仅为可信应用提供内存管理、进程管理和加解密服务。虽然选择Linux内核作为REE内核，但开发者仍可以为REE选择其他通用内核。

在TrustZone层，TrustZone驱动负责将REE安全应用的输入数据写入共享内存并调用SMC指令陷入SMM或将可信应用的处理结果从共享内存读出并返回给普通应用程序。安全监控模块SMM运行于TEE，用于TEE与REE间的执行环境切换。具体为，安全监控代码首先检查当前处理器的NS-bit位以确定中断请求方向，然后保存相应的环境上下文、重新设置处理器NS-bit位并清理CPU寄存器，最后恢复另外一个环境上下文。

1.2 安全通道设计

为保证TEE与REE的物理内存隔离，本文使用TrustZone地址空间控制器将物理内存划分成安全域内存和非安全域内存，分别运行TEE和REE程序。为保证用户层程序不能直接访问或破坏内核层数据及代码，利用TrustZone提供的两个虚拟MMU，将安全域物理内存映射到TEE的虚拟内存空间，将非安全域物理内存映射到REE的虚拟地址空间。此外，一块固定大小的非安全域物理内存会被作为共享内存，分别映射到TEE和REE虚拟地址空间的固定位置，用于TEE与REE之间的通信通道。

由于共享内存属于TEE、REE均可访问的非安全域物理内存，所以必须对其中的敏感数据进行加密处理，否则REE内核层一旦被攻破就可导致信息泄露，如HTC指纹数据直接明文存储在Android侧而导致用户隐私的泄露。如图1所示，为保障数据机密性，将共享内存划分为非加密区与加密区。其中非加密区用于存储TEE安全服务Id、服务返回状态RS等非机密数据，加密区用于存储用户敏感数据如指纹信息等，并在TEE内核层实现了具有数据加密功能的安全通信通道API。具体功能可描述为：REE向TEE发起安全服务请求后，TEE用户层调用“sys_recv”API从共享内存读取加密数据到TEE通信

缓冲区然后解密发送给TEE用户层，TEE用户层完成数据处理后调用“sys_send”API将用户层数据先写入TEE通信缓冲区然后加密输出数据最后写回到共享内存。通过在内核层设置加密通道可防止直接REE明文泄露的情况发生。

在TrustZone硬件保护下，REE获取TEE信息的途径只有CPU寄存器和非安全域共享内存。而本设计方案使得TEE可信应用不能直接与REE通信，即一旦关闭安全通信通道API，只有负责环境上下文切换的SMM和可信内核中其他API的软件漏洞会导致TEE信息泄露。因此，验证SMM的信息流无干扰性可保证CPU寄存器不会成为TEE的通信通道，验证可信内核的功能正确性可保证除安全通道之外的其他API不会写信息到非安全域内存空间。接下来将建立TEE形式化模型来验证TEE的信息流无干扰性等安全属性。

2　TEE形式化模型

针对TEE经常遭遇整型溢出、程序返回地址越界等资源边界类软件攻击的问题，TEEFM将函数输入输出参数纳入系统行为规范的定义，参与系统整体逻辑推理，使TEEFM能够检测这类软件错误。此外，对TEE而言一个重要的安全属性是信息流无干扰性，该属性能够保证TEE与REE之间的调度模块不会成为通信通道。为便于建立信息流无干扰性模型，在TEEFM引入了安全域、观察函数[6-12]等相关概念。下面给出TEEFM的定义。

定义 1　TEEFM模型是一个五元组状态机：

$$M=(S,A,\Delta,\Omega,O)$$

式中，S 表示系统状态集，包括机器寄存器、具体内存数据等系统资源信息和安全域标识符等逻辑信息，表示系统的全局状态；A 表示系统行为，如执行一次系统调用，并用 $S\mapsto S'$ 表示系统状态从 S 迁移到 S'；Δ 表示安全域集，如安全域集可以是{TEE,REE}的集合或所有进程的集合；Ω 表示某安全域的可观察(observable)状态集或包含可观察状态的命题集；O 表示观察函数，$O:\Delta\times S\to\Omega$。

假设系统状态 S 是整型数据 a、b、c 的集合，即 $S=\{a,b,c\}$，则对安全域 δ 而言，如果安全域 δ 的可观察状态是 $\{a,b\}$，则 Ω 可以是状态集 $\{a,b\}$；也可以是命题集 $\{(a+b)/2\}$，表示 a 和 b 的平均值。

定义 2　系统行为规范

假定 V 表示行为局部状态，S 表示系统行为对应的函数输入输出状态，则定义系统行为规范 $\mu(s_0,v_0,s_1,v_1)$ 是关于 $(S\times V)\times(S\times V)$ 的谓词：系统在初始状态 s_0 下，系统行为接收输入 v_0 并输出 v_1 的同时状态由 s_0 迁移到 s_1。

定义 3　系统状态安全性

使用逻辑公式来表示系统安全属性，如“安全域标识符取值一定大于等于0小于最大安全域的个数”等，并用I表示系统安全属性集。如果系统任意行为的初始状态关于I是安全的且迁移后的状态关于I仍是安全的，则系统关于I是安全的。系统安全性可形式化表示为:

$$\forall s,s'\in S\quad s\models I\ \wedge s\mapsto s'\Rightarrow\ s'\models I$$

式中，符号⊨表示成真指派，如 $s\models I$ 表示在状态 s 下 I 为真。

定义 4　信息流无干扰性

给定安全域 δ，则基于单步状态的“展开条件”[13] (unwinding condition)可解释为系统行为引发的单步状态迁移不会破坏两状态间的不可分辨性(indistinguishability)，这里的不可分辨性是指安全域 δ 对两状态的观察等价性。其更直观的含义是如果单步状态迁移保持不可分辨性，那么改变初始状态中的 δ 不可观察数据不会影响 δ 对结果状态的观察。

下面给出形式化的描述：对任意安全域 δ 和状态机 M，且 $\mapsto_M$ 表示单步状态迁移，即 M 关于 δ 满足信息流无干扰性的条件是：

$$\forall s_1,s_2\in S\quad \exists s_1',s_2'\in S$$
$$s_1\mapsto_M s_1'\wedge s_2\mapsto_M s_2'\wedge O(\delta,s_1)=O(\delta,s_2)$$
$$\Rightarrow O(\delta,s_1')=O(\delta,s_2')$$

3　SMM形式化验证

本节以SMM为例对TEEFM进行语义分析。鉴于篇幅关系，省略了状态集、行为规范的相关定义及状态安全性的相关证明，只给出信息流无干扰性定理及证明。

3.1　SMM信息流无干扰性验证

假设安全域集为{TEE,REE}，在关闭TEE与REE间授权通信通道的前提下，TEE与REE间信息流无干扰性包含以下两种情况：

1) REE请求切换到TEE再回到REE的整个操作过程前后REE可观察状态(CPU寄存器的值)不变，即REE数据不受TEE影响；

2) TEE请求切换到REE再回到TEE的整个操作过程前后TEE可观察状态(CPU寄存器的值)不变，即TEE数据不受REE影响。

以上两种情况是安全域间严格意义上的信息流

无干扰性，但在Trustzone框架下本文更关心不同安全域间是否存在隐蔽通道导致TEE机密信息流向REE，因此在实际验证过程中只需对情况1)进行验证。其更直观含义是：TEE对REE而言就像一个“黑盒子”，REE不能观察到TEE任意的中间执行过程，而TEE的执行结果只能由加密通道输出到REE。

但是REE请求切换到TEE再回到REE是一个大步语义(包含多步状态迁移)，无法保证信息流路径的确定性。为此，将这一大步语义进行展开并将定义4的信息流无干扰性证明拆分成以下3个引理。

1) 状态存储引理：如果两个活动状态(active states)具有不可分辨性，并且经一步状态迁移分别到达两个非活动状态(inactive states)，那么这两个非活动状态也具有不可分辨性。

2) 数据完整性引理：如果一个非活动状态经一步状态迁移到达另外一个非活动状态，那么这两个非活动状态之间具有不可分辨性。

3) 状态恢复引理：如果两个非活动状态具有不可分辨性，并且经一步状态迁移分别到达两个活动状态，那么这两个活动状态也具有不可分辨性。

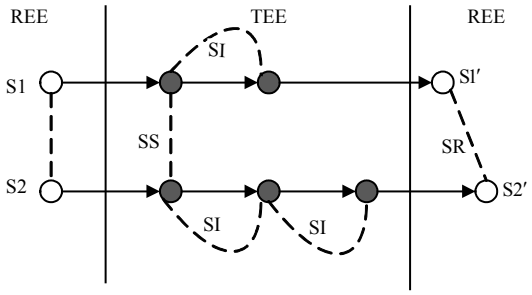

图2 应用3个引理证明SMM信息流无干扰性

如图2所示，箭头表示单步状态迁移，虚线表示状态间的不可分辨性，空心圆表示活动状态，实心圆表示非活动状态。具体的，状态存储引理保证了REE切换后的两个非活动初始状态(保存到安全内存中的REE上下文)具有不可分辨性，数据完整性引理保证了TEE中任意的进程调度执行没有破坏非活动状态间的不可分辨性(保存到安全内存中的REE上下文没有被TEE修改)，状态恢复引理保证了TEE切换回REE后的两个活动状态(REE恢复后的CPU寄存器)具有不可分辨性。本文主要证明义务是使用状态存储和状态恢复这两个引理来证明SMM模块的调度功能不会产生TEE到REE的隐蔽通道，而数据完整性引理涉及TEE可信内核中进程的调度执行，是下一步的研究工作。

SMM信息流无干扰性在抽象层的形式化证明如下(不存在TEE机密信息流向REE的隐蔽通道)：

引理 1 SMM抽象层信息流无干扰性

Lemma SMM_abs_NI:

$$\forall i\ \mathrm{abs}_1, \mathrm{abs}_2, \mathrm{abs}_1', \mathrm{abs}_2',\ \mathrm{abs}_1 \mapsto^{+} \mathrm{abs}_1' \wedge \mathrm{abs}_2 \mapsto^{+} \mathrm{abs}_2' \wedge \mathrm{obseq}\ i\ \mathrm{abs}_1\ \mathrm{abs}_2 \Rightarrow \mathrm{obseq}\ i\ \mathrm{abs}_1'\ \mathrm{abs}_2'$$

式中，obseq 表示安全域i可观察的任意两个状态之间的等价关系(不可分辨性)，对i取值1，表示安全域为REE；$\mapsto^{+}$表示REE请求切换到TEE再回到REE的一个大步语义，可展开为：REE调用 smc_from_ree 切换到TEE，TEE再调用 smc_from_tee 回到REE。

证明：1) 展开 smc_from_ree 的定义分别从两个初始的等价活动状态 abs_1 abs_2 (初始REE环境上下文)得到两个等价非活动状态 abs_1'' abs_2'' (保存到安全内存中的REE环境上下文)，当前安全域变为TEE状态。存储引理得证。

2) 展开 smc_from_tee 的定义分别从两个等价非活动状态 abs_1'' abs_2'' (保存到安全内存中的REE环境上下文)得到两个等价活动状态 abs_1' abs_2' (恢复后的REE环境上下文)，当前安全域变为REE，状态恢复引理得证。证毕。

给定抽象层状态集 absState 、实现层状态集 conState 以及一致性关系 R: $\mathrm{absState} \times \mathrm{conState}$，若抽象层与实现层满足互模拟的一致性关系，当且仅当满足以下条件：

引理 2 SMM抽象层与实现层的一致性证明

1) 任一抽象层行为都有一实现层行为与之对应

$$\forall \mathrm{abs}, \mathrm{abs}' \in \mathrm{absState},\ \mathrm{con} \in \mathrm{conState},\ \mathrm{rs}_1,\ \mathrm{rs}_1',\ \mathrm{rs}_2,\ \mathrm{rs}_2'$$
$$\mathrm{world\_yield\_abs\ abs\ rs}_1\ \mathrm{abs}'\ \mathrm{rs}_1' \wedge R(\mathrm{abs},\ \mathrm{con}) \Rightarrow \exists \mathrm{con}' \in \mathrm{conState}\ \mathrm{world\_yield\_con\ con\ rs}_2\ \mathrm{con}'\ \mathrm{rs}_2' \wedge R(\mathrm{abs}',\ \mathrm{con}')$$

2) 任一实现层行为都有一抽象层行为与之对应

$$\forall \mathrm{con}, \mathrm{con}' \in \mathrm{conState}, \mathrm{abs} \in \mathrm{absState},\ \mathrm{rs}_1,\ \mathrm{rs}_1',\ \mathrm{rs}_2,\ \mathrm{rs}_2'$$
$$\mathrm{world\_yield\_con\_spec\ con\ rs}_1\ \mathrm{con}'\ \mathrm{rs}_1' \wedge R(\mathrm{abs}, \mathrm{con}) \Rightarrow \exists \mathrm{abs}' \in \mathrm{absState}\ \mathrm{world\_yield\_abs\_spec\ abs\ rs}_2\ \mathrm{abs}'\ \mathrm{rs}_2' \wedge R(\mathrm{abs}', \mathrm{con}')$$

式中，world_yield函数作为SMM的核心功能，用于不同安全域的调度：首先根据处理器安全状态位(NS-bit)判断当前安全域并调用相应的子函数进行环境上下文的保存与恢复，然后更新NS-bit位，最后跳转到相应安全域的入口函数执行。引理2中 world_yield_abs 和 world_yield_con 分别是 world_

yield函数的抽象层和实现层定义，均包含初始状态、函数输入参数以及返回状态、函数返回结果。

引理2的证明比较简单直观，只需要对条件规范的定义和一致性关系R的定义展开，然后引入实例化状态，即可完成证明。

引理2中没有SMMI的定义，这是因为引理2一旦得证，关系R就会将抽象层满足的SMMI属性引入到实现层，因此再加入SMMI就会成为冗余证明义务。

3.2　SMM实现层的信息流无干扰性验证

根据引理2抽象层与实现层的一致性关系，将引理1的安全属性引入到实现层。下面给出SMM在实现层的信息流无干扰性定理。

定理 1　SMM实现层信息流无干扰性

Lemma SMM_con_NI:

$\forall\, i\ \mathrm{con}_1,\ \mathrm{con}_2,\ \mathrm{con}_1',\ \mathrm{con}_2'$

$\mathrm{con}_1 \mapsto^{+} \mathrm{con}_1' \wedge \mathrm{con}_2 \mapsto^{+} \mathrm{con}_2'$

$\wedge \mathrm{obseq}\ i\ \mathrm{con}_1\ \mathrm{con}_2 \Rightarrow \mathrm{obseq}\ i\ \mathrm{con}_1'\ \mathrm{con}_2'$

证明：由层次间的一致性关系R可知，对任意con_1和con_2一定存在abs_1和abs_2，使得$R(\mathrm{abs}_1,\ \mathrm{abs}_2)$和$R(\mathrm{abs}_2,\ \mathrm{con}_2)$为真。由条件“$\mathrm{obseq}\ i\ \mathrm{con}_1\ \mathrm{con}_2$”和关系$R$的定义可得出新条件“$\mathrm{obseq}\ i\ \mathrm{abs}_1\ \mathrm{abs}_2$”。引理2第二条规则可知，对$j$=1,2，一定存在$\mathrm{abs}_j'$使得$\mathrm{abs}_j \mapsto \mathrm{abs}_j'$且$R(\mathrm{abs}_j',\mathrm{con}_j')$，因此“$\mathrm{obseq}\ i\ \mathrm{con}_j'\ \mathrm{abs}_j'$”为真，证明目标变为“$\mathrm{obseq}\ i\ \mathrm{abs}_1'\ \mathrm{abs}_2'$”，直接带入引理1结束证明，证毕。

表1　SMM验证的工程量统计

统计分类	Coq代码行数
行为规范	298
安全状态验证	427
一致性证明	125
机密性验证	638
总计	1 488

实际证明过程中发现寄存器r8～r12不能满足定理1中的可观察等价状态，通过分析实现代码发现这些寄存器没有被正常的保存与恢复，修复该错误后最终完成了定理1所有子义务的证明，保证了SMM的信息流无干扰性，这也表明了采用形式化方法对信息流进行验证的有效性。整个SMM包含187行C代码86行汇编代码，表1对SMM验证的工程量进行了统计，包括行为规范定义、安全状态验证、一致性证明及信息流验证。

4　结 束 语

本文针对多隔离域的安全系统架构提出TEE形式化模型TEEFM。采用TEEFM可方便地对安全域间的信息流无干扰性进行验证。此外，基于TEEFM验证过的软件系统可防止整型溢出、函数返回地址异常等资源边界类错误的发生。本文借助Coq定理证明器，采用两层结构对TEEFM进行语义分析，借助层次间的一致性关系将TEE抽象层安全属性如SMM信息流无干扰性引入到实现层，从而保证了SMM的安全性。

未来工作将对可信内核的功能正确性进行验证，以进一步分析TEE的资源边界问题，如可信应用不能直接访问可信内核的资源、除指定的共享内存外可信内核不能写数据到其他非安全域内存等，最终实现对整个TEE的安全性验证。

参 考 文 献

[1] ARM. ARM security technology building a secure system using trustzone technology[EB/OL]. (2009-04-01). http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.prd29-genc-009492c/index.html.

[2] ROSENBERG D. Qsee trustzone kernel integer over flow vulnerability[EB/OL]. (2014-07-01). https://forum.xda-developers.com/galaxy-s4-att/general/qsee-trustzone-kernel-integer-overflow-t2840415.

[3] HUA Wei. Huawei's mate7 mobile phone has vulnerabilities[EB/OL]. (2015-06-09). http://www.huawei.com/cn/psirt/security-advisories/hw-432807.

[4] HTC. HTC caught storing fingerprint data in unencrypted plaintext[EB/OL]. (2015-08-10). https://www.extremetech.com/mobile/211985-htc-caught-storing-fingerprint-data-in-unencrypted-plain-text.

[5] SINHA R, RAJAMANI S, SESHIA S, et al. Moat: Verifying confidentiality of enclave programs[C]//Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. US: ACM Press, 2015: 1169-1184.

[6] DAM M, GUANCIALE R, KHAKPOUR N, et al. Formal verification of information flow security for a simple ARM-based separation kernel[C]//Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security. [S.l.]: ACM Press, 2013: 223-234.

[7] SANGIORGI D. An introduction to bi-simulation and coinduction[M]. Cambridge: Cambridge University Press, 2012: 824-833.

[8] KLEIN G, ANDRONICK J, ELPHINSTONE K, et al. Comprehensive formal verification of an OS microkernel[J]. ACM Transactions on Computer Systems (TOCS), 2014, 32(1): 2.

[9] LEROY X. Formal certification of a compiler back-end or: Programming a compiler with a proof assistant[C]//ACM SIGPLAN Notices. [S.l.]: ACM Press, 2006, 41(1): 42-54.

[10] Global Platform. Globalplatform device technology "TEE client API specification"[EB/OL]. [2017-01-01]. http://www.globalplatform.org/specificationsdevice.asp.

[11] YANG Bo, FENG Deng-guo, QIN Yu, et al. Secure access scheme of cloud services for trusted mobile terminals using trustzone[J]. Journal of Software, 2016, 27(6): 1366-1383.

[12] COSTANZO D, SHAO Z, GU R. End-to-end verification of information-flow security for C and assembly programs [J]. ACM SIGPLAN Notices, 2016, 51(6): 648-664.

[13] GOGUEN J A, MESEGUER J. Unwinding and inference control[C]//Security and Privacy. [S.l.]: IEEE, 1984: 75.

编　辑　蒋　晓