第 35 卷 第 2 期 计 算 机 工 程 2009 年 1 月
Vol.35 No.2 Computer Engineering January 2009
·安全技术· 文章编号：1000—3428(2009)02—0129—04 文献标识码：A 中图分类号：TP309

TPM 接口命令标准符合性测试的设计与实现

崔 奇[1,2,3]，马 楠[1]，刘贤刚[1]
(1. 中国电子技术标准化研究所信息技术研究中心，北京 100007；2. 中国科学院软件研究所，北京 100080；
3. 中国科学院研究生院，北京 100080)

摘 要：标准符合性是衡量可信计算模块(TPM)芯片是否存在安全漏洞的重要因素。该文面向可信计算组织规范，给出一个针对 TPM 接口命令的标准符合性测试方案。运用有限状态机归纳测试集的方法实现了原型。对某款 TPM 芯片进行实测的结果表明，该方案是有效且切实可行的。

关键词：可信计算模块；接口命令；可信计算组织；标准符合性测试；有限状态机

Design and Implementation of Compliance Test on TPM Interface Command

CUI Qi[1,2,3], MA Nan[1], LIU Xian-gang[1]
(1. IT Research Center, China Electronics Standardization Institute, Beijing 100007; 2. Institution of Software, Chinese Academy of Sciences, Beijing 100080; 3. Graduate University of Chinese Academy of Sciences, Beijing 100080)

【Abstract】 Compliance is an important factor that decides whether TPM chip has security exploits. This paper presents a testing scheme on compliance of Trusted Platform Module(TPM) interface commands towards Trusted Computing Group(TCG) specifications, with methods on constructing testing sets with finite state machines. The prototype of the testing platform is implemented and used to test on the real TPM chips. The results show the effectiveness and feasibility of this scheme.

【Key words】 Trusted Platform Module(TPM); interface command; Trusted Computing Group(TCG); compliance test; Finite State Machine(FSM)

1 概述

可信计算组织(Trusted Computing Group, TCG)提出的可信计算技术为计算机安全提供了以硬件为基础的解决方案。它建立了信任链的概念，在信任根的基础上，逐级度量，逐级认证，确保计算机从平台启动、载入操作系统到运行应用程序都在用户预期的范围内正确执行。可信计算模块(Trusted Platform Module, TPM)作为储存信任根(RTS)和报告信任根(RTR)，是信任链的重要基础，其安全性必须得到保证。

近年来，国内外已经有多家厂商生产TPM芯片并投入使用，国内有兆日、联想、瑞达等，国外有Infineon, Atmel, ST Microelectronic, Winbond等，虽然各家厂商生产的芯片规格和技术指标有所差别，但均声称符合TCG标准。TCG尽管发布了相关的设计与实现规范[1-4]，却一直未公布测试标准，所以，缺乏一个行之有效的测试方法。如果TPM芯片的设计与实现未能完全符合规范，那么就可能存在漏洞，危及用户的安全。本文针对TPM接口命令的测试方法进行研究和设计，并在此基础上开发了一个测试平台。

2 相关工作

在对一个系统进行标准符合性测试时，往往首先根据设计规范归纳出具体的有限状态机，然后在此基础上设计测试流程[5]。但是由于规范本身通常不会明确给出相应的有限状态机，因此如何从规范中抽象出有限状态机的模型并归纳出测试序列成了急需解决的问题。

文献[6]通过监测主控总线的输入和受控总线的输出实现了对片上总线(On-Chip Bus, OCB)标准的一致性验证，其中包括从 OCB 规范中提取有限状态机以及在此基础上利用面向状态的语言(State-Oriented Language, SOL)提取状态转换测试序列的方法。尽管底层的总线有别于 TPM，但其相关思路依然具有参考价值。

在标准符合性测试中，基于“瀑布模型”的方法被广泛采用[7-8]，从需求分析、平台设计到实例分析，逐步实现。本文对测试平台的设计与实现也是建立在该流程之上。

文献[9]率先做了 TPM 标准符合性测试的尝试，分别在应用层和协议层从功能性、完整性和耐压性这 3 个方面对 TPM 进行了考察。然而该测试结构较为复杂，且其中的部分内容有重复，效率较低。同时，只提出了构建有限状态机的思路，没有给出对应实例。本文将描述具体的有限状态机设计，并对构建的方法进行改进。

3 测试需求与内容

3.1 TPM 结构与功能

TPM 是一个连接在 PC 主板上的微处理器芯片，可以生成并存储密钥，同时提供包括 RSA, SHA-1, HMAC 在内的加密算法，以实现数字签名、身份认证和完整性度量等安全功能。这些功能都是通过执行一系列 TPM 接口指令完成的。

3.1.1 TPM 命令结构

TPM 接口命令分为输入命令和输出响应 2 类。它从结构

基金项目：国家自然科学基金资助项目(60672112)
作者简介：崔 奇(1983－)，男，硕士研究生，主研方向：信息安全，可信计算，虚拟机；马 楠，工程师、博士；刘贤刚，博士研究生
收稿日期：2008-06-04 **E-mail：**cuiqi05@iscas.cn

上可以分为 4 个部分：请求头/响应头，特定变量，认证尾 1，认证尾 2，如图 1 所示。

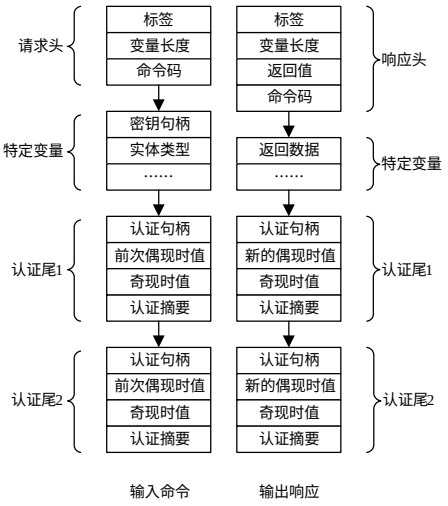

图 1　TPM命令结构

通常每个命令都有请求头/响应头，但其他3部分根据命令的不同而发生变化。如认证尾的存在与否即取决于该命令/响应的标签值，如标签为TPM_TAG_RQU_ COMMAND，则没有认证尾；如果是TPM_TAG_RQU_ AUTH1_ COMMAND或TPM_TAG_RQU_AUTH2_ COMMAND，就分别有1个和2个认证尾。TPM命令的执行结果通常都直接反映在输出响应中的返回值部分，规范中对此有较详细的　定义[2-3]。

3.1.2　TPM 基本功能

作为提供安全防护的密钥产生与存储设备，TPM 具有以下主要功能：

(1)加密解密相关，包括随机数产生、SHA-1 散列算法引擎、HMAC 计算引擎、RSA 非对称密钥引擎等。

(2)密钥存储相关，包括存储生成的密钥、对数据进行捆绑、封装、签名等防护。由于 TPM 芯片容量有限，因此直接受到保护的是密钥而不是机密数据本身。

(3)平台完整性相关，包括基于 PCR 寄存器的完整性度量及完整性报告。

(4)初始化与管理，包括获取所有权及各类认证协议，这些认证确保命令的执行者有执行它并访问相关数据的权限。

3.2　测试内容设计

根据以上分析，本文将接口命令的测试分成 3 个部分：可靠性测试，功能性测试和稳定性测试。

3.2.1　可靠性测试

当输入命令中的变量发生错误时，TPM将返回特定的错误值[2]。可靠性测试就是考察TPM的单个命令在输入参数错误的条件下，输出响应中响应头部分的返回值是否和规范所定义的一致。由于规范中没有详细定义TPM读取参数的顺序，当输入的变量中有多个发生错误时，将无从判断返回值是由哪个错误变量产生的，因此测试中每次输入只允许出现一个错误参数。

3.2.2　功能性测试

功能性测试主要考察 TPM 的基本功能是否符合规范定义。TPM 功能的执行是依靠输入指定的命令序列实现的，且一个功能的开始运行往往依赖于另一个功能的执行完成，例如加载密钥的实现必须依赖于生成密钥的完成。因此，从规范中提取出 TPM 工作的状态转换，并在此基础上构建有限状态机和相应的测试序列。这样，复杂的规范被抽象到若干个状态机中，大大简化了测试的设计与实现。

3.2.3　稳定性测试

稳定性测试主要是检验 TPM 在极限工作条件下的性能，比如重复输入同一指令后的工作状态以及承受词典攻击时采取的对策。由于规范对此尚未有明确定义，因此该部分是扩展内容，用作对标准符合性检测的补充。本文的设计将不涉及该内容。

4　测试平台设计

4.1　平台总体结构

根据测试内容，平台总体结构分为 3 个层次，见图 2。

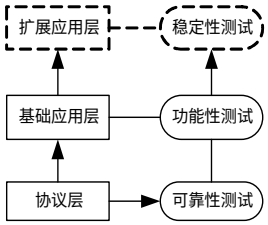

图 2　测试平台总体结构

协议层：测试的最底层，测试对象为单个 TPM 命令，验证其输入输出的设计是否符合规则，对应于可靠性测试。

基础应用层 :测试的中间层，测试对象为 TPM 命令组合，考察其能否实现规范所定义的基本功能，对应于功能性测试。

扩展应用层 :测试的最上层，测试对象为 TPM 扩展性能，包括对恶意攻击的防范等，对应于稳定性测试。

从程序本身来说，三者并不是紧密相连的，任意一层都可以作为一个模块分离出来独立测试；但从逻辑上看，3 部分是递进的关系 :协议层可靠性测试的主要对象是单个命令，基础应用层的功能性测试则是针对这些命令的有序组合，而扩展应用层的稳定性测试则只有在基本功能无误的前提下才有意义。同时，这样的平台结构满足可扩展性需求，如果 TCG 发布新的规范，在平台的对应部分添加新的测试内容即可。

4.2　协议层设计

TPM有3个各自独立的状态，分别是可用(enabled)/不可用(disabled)、激活(active)/未激活(inactive)、占有(owned)/未占有(unowned)。这3个状态的排列组合构成了TPM的8个操作模式，如图3所示。

图 3　TPM 操作模式

相同的命令在不同的操作模式下会返回不同的结果。同时，对于 TPM 输入的参数，其可能的错误类型也是不同的。在本文的设计中，变量的错误类型被定义为以下 2 种：(1)无效(invalid)：即输入参数的值是错误的；(2)非法(illegal)：即输入参数的类型或数据结构是错误的。

在此基础上，本文从规范[2-3]中提取错误输入所对应的输

出来构建测试集。值得注意的是，这些输出响应有些是在规范中明确定义的，有些则需要根据规范间接推导得出，所以，一种错误参数类型往往有几种可能结果，这些结果都作为合法输出列在测试集中。图 4 就是关于TPM_TakeOwnership命令的部分测试集。

TPM_TakeOwnership			
相关功能	Take Ownership		
前续功能	Read Capability		
后续功能	Clear Ownership, Create Keys		
前续命令	TPM_OIAP		
后续命令	TPM_TerminateHandle		
操作模式	变量名	错误类型	返回值
S1	*	*	TPM_OWNER_SET
S2	*	*	TPM_DISABLED
…			
S5	tag	Invalid	TPM_BADTAG
		Illegal	TPM_BAD_PARAMETER
	paramSize	Invalid	TPM_BAD_PARAM_SIZE
		Illegal	TPM_BAD_PARAMETER
	ordinal	Invalid	TPM_BAD_ORDINAL
		Illegal	TPM_BAD_PARAMETER
	…		
S6	*	*	TPM_DISABLED
…			

图 4　TPM_TakeOwnership 测试集(部分)

注：* => 任意输入变量/任意错误类型

附：

(1)if (TPM_PERMANENT_FLAGS->ownership==FALSE) return TPM_INSTALL_DISABLED

(2)if (TPM_PERMANENT_DATA->endorsementKey=>invalid) return TPM_NO_ENDORSEMENT

(3)if (authHandle != OIAP)return TPM_AUTHFAIL

…

从图 4 可以看出，测试集主要分成 2 部分。前面的表格除了返回值列表外，还包括了相关功能、前/后续功能和前/后续命令，这些是为构建基础应用层的有限状态机所准备的。附录部分则是将规范[3]中详细定义的行为用逻辑表达式的方式列出来，包括一些无法反映在表格里但又需要进行测试的关系，以方便并完善测试的实现。

4.3　基础应用层设计

TPM 功能之间存在依赖性，基础应用层测试的直接对象是一系列功能组成的一个状态转换过程，而对于单个功能，通常也会涉及多个命令，例如，一个完整的签名(Sign)功能就需要依次执行 TPM_OIAP, TPM_Sign, TPM_Terminate Handle 这 3 个命令。因此，在协议层的测试集表格中列出了每个命令的相关功能、前/后续功能、前/后续命令。通过这几项，可以很方便地找到需要测试的状态转换过程以及过程中每个功能需要输入的命令序列。以图 5 为例，可以得出，Take Ownership 功能依赖于 Read Capability 功能的完成，其后续功能可以是 Clear Ownership 或 Create Keys；Take Ownership 功能需要执行 TPM_TakeOwnership 命令，在执行该命令之前，需要执行 TPM_OIAP 命令，之后则需要执行 TPM_TerminateHandle 命令。

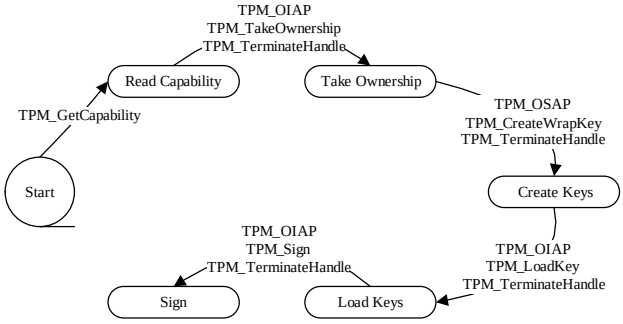

图 5　签名功能性测试的有限状态机

通过上述方法分析测试集，即以TPM基本功能为状态、以命令为输入，构建一个面向功能性测试的有限状态机。具体来说，是从规范[1]中提取出TPM的基本功能作为状态，然后从规范[3]中提取相关命令作为输入。事实上，这些都已经反映在本文定义的测试集中。图 5 就是以签名功能为例的测试有限状态机。需要指出的是，与一般的有限状态机[6-8]不同，在本文的设计中，输入接口只有一个，这个接口需要依次输入几个命令才能完成一次完整的状态转换。当输入的命令出错时，TPM不会切换到下一个功能状态，而是停留在原状态下。

5　测试实例实现

Libtpm 是 IBM 公司为 TPM 开发的一套函数库，与之配套的还有驱动程序与上层软件栈。本文的测试平台原型即是基于 Libtpm，在 Linux(内核版本 2.6.20-rc4-mm1)下实现的，测试对象为 Atmel at97sc3203 型号的 TPM 芯片。测试实例程序的流程如图 6 所示。

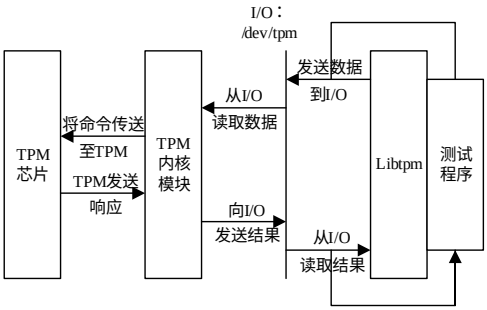

图 6　TPM 测试实例流程

在可靠性测试的实现中，程序调用 TPM_Transmit 函数，将预设好的输入数组依次写入设备文件，TPM 芯片读入这些数据并处理完毕后，将响应写入输出缓存中，同样通过设备文件传递给上层应用程序，测试程序就在此时截取响应中的返回值加以分析，如果与预期值一致，则测试通过，反之则失败。

功能性测试方面则通过直接调用 Libtpm 中提供的函数，向 TPM 芯片发送指令。如果一个完整的状态转换流程成功完成，则测试通过，否则，将在没有顺利执行的功能处终止并报告测试失败。

图 7 和图 8 示出了测试结果的一部分。

FUNC_TPM_SIGN	
Read Capability	PASS
Take Ownership	PASS
Create Keys	PASS
Load Keys	PASS
Sign	PASS
PASS	

图 7　TPM 签名功能测试报告

CMD_TPM_GetCapability (S1 Mode)				
变量	错误	预期返回值	实际返回值	结果
Tag	invalid	TPM_BADTAG	TPM_BADTAG	PASS
	illegal	TPM_BAD_PARAMETER	TPM_BAD_PARAMETER	PASS
paramSize	invalid	TPM_BAD_PARAM_SIZE	TPM_BAD_PARAM_SIZE	PASS
	illegal	TPM_BAD_PARAMETER	TPM_BAD_PARAMETER	PASS
ordinal	invalid	TPM_BAD_ORDINAL	TPM_BAD_ORDINAL	PASS
	illegal	TPM_BAD_PARAMETER	TPM_BAD_PARAMETER	PASS
capArea	invalid	TPM_BAD_MODE	TPM_BAD_MODE	PASS
	illegal	TPM_BAD_PARAMETER	TPM_BAD_PARAMETER	PASS
subCapSize	invalid	TPM_BAD_MODE TPM_BAD_PARAMSIZE	TPM_BAD_PARAMSIZE	PASS
	illegal	TPM_BAD_PARAMETER	TPM_BAD_PARAMETER	PASS
subCap	invalid	TPM_BAD_MODE	TPM_BAD_MODE	PASS
	illegal	TPM_BAD_PARAMETER	TPM_BAD_PARAMETER	PASS

图 8　TPM_GetCapability 命令可靠性测试报告

从图 7 和图 8 可以看出，Atmel 该款芯片的这 2 项设计基本都遵循了 TCG 规范。

6　结束语

本文针对 TPM 接口命令的标准符合性测试提出了一个设计方案，并构建了平台原型进行实物芯片测试。结果表明，本方案具备可行性，并且能较好地反映出 TPM 的接口命令是否按照规范设计实现。在未来的工作中，将在逐步完善测试集的基础上，研究自动生成有限状态机与测试序列的方法。

参考文献

[1] Trusted Computing Group. TCG TPM Main Specification, Version 1.2 Revision 94, Part 1, Design Principles[Z]. (2006-03-29). https://www.trustedcomputinggroup.org.

[2] Trusted Computing Group. TCG TPM Main Specification, Version 1.2 Revision 94, Part 2, TPM Structures[Z]. (2006-03-29). https://www.trustedcomputinggroup.org.

[3] Trusted Computing Group. TCG TPM Main Specification, Version 1.2 Revision 94, Part 3, TPM Commands[Z]. (2006-03-29). https://www.trustedcomputinggroup.org.

[4] Trusted Computing Group. TCG PC Client Specific TPM Interface Specification(TIS), Version 1.2[Z]. (2006-03-29). https://www.Trustedcomputinggroup.org.

[5] Chow T S. Testing Software Design Modeled by Finite-state Machines[J]. IEEE Trans. on Software Engineering, 1978, SE-4(3): 178-187.

[6] Su Man-Yun, Shih Che-Hua, Huang Juinn-Dar, et al, FSM-based Transaction-level Functional Coverage for Interface Compliance Verification[C]//Proceedings of the 2006 Conference on Asia South Pacific Design Automation. Yokohama, Japan: [s. n.], 2006: 448-453.

[7] 邓　昳, 刘又诚. 软件标准符合性测试[J]. 北京航空航天大学学报, 1997, 23(1): 68-73.

[8] 亓　伟, 叶晓俊, 王建民. ODBC 标准符合性测试框架[J]. 计算机工程, 2005, 31(23): 101-103.

[9] Ahmad-Reza S, Selhorst M, Stüble C, et al. TCG Inside: A Note on TPM Specification Compliance[C]//Proceedings of the 1st ACM Workshop on Scalable Trusted Computing. Alexandria, Virginia, USA: [s. n.], 2006: 47-56.

(上接第 122 页)

4　性能分析

改进后的PQ_SIP协议栈与原SIP 协议栈的不同之处在于：在消息头域中，PQ_SIP仅增加 1 个包含 3 bit表示优先级的CSipPriority头域类和 1 个包含 1 bit表示复位位的CSipRecover头域类；在消息收发器Transceiver功能方面，PQ_SIP在消息收发类CSipTSC中增加了成员函数CString PRI()和SendToTSP ()用于判断优先级；在协议栈的结构方面，增加 2 个定时器T_2和T_3以及 2 个缓存区WAITING DB用于存放优先级低的数据业务，。

通过在 SIP 请求消息头域中添加 3 bit 的标识符来划分优先级，能使网络区别对待不同用户和数据业务以及实现运营商对呼叫或会话的间接控制，增加消息和用户的权限，提高了整个网络和用户 PC 资源的利用率，可在一定程度上节约开销，避免由于无限制为不同业务和用户开线程而导致用户系统资源大量被占用的问题。

5　结束语

本文设计一种基于 SIP 协议栈的支持 QoS 服务质量的 PQ_SIP 协议栈，重点阐述增加优先等级后的拓展型 PQ_SIP 协议栈的工作原理，给出 PQ_SIP 协议栈结构设计流程，并分析其性能，为今后对 SIP 协议的改进与设计提供参考。实践证明，该协议栈能有效提高网络的资源利用率，体现数据和用户业务等级，有利于特殊情况下频繁的平战转换，特别是在某些授权访问的限权过程中能发挥较大优势，可做到零成本，而无须添加任何硬件设备。

优先级策略最终目的是针对某些对用户的权限和请求的优先级要求很强烈的网络，对不同用户和业务划分等级或限权。在性能方面，在实际应用中发现，为达到这一目的，往往要牺牲网络中具有较低优先级的用户和业务的一部分传统性能，如时延、视频图像的质量、回声等。因此，进一步的研究方向是继续将优先级措施与其他 QoS 需求相结合，根据不同的应用环境，设计最适合网络综合性能指标要求的协议栈。

参考文献

[1] 蔡闻怡, 陈一民. 基于代理的 SIP 穿越 NAT 和防火墙方案[J]. 计算机工程, 2007, 33(22): 148-150.

[2] 蒋　艳, 郭　伟, 刘　伟. 一种基于 SIP 的升空平台通信系统的优化切换[J]. 计算机应用, 2008, 28(2): 374-381.

[3] 万晓喻, 樊自甫, 张　洪, 等. 下一代网络的业务生成技术[M]. 北京: 人民邮电出版社, 2005.

[4] Rosenberg J, Schulzrinne H, Camarillo G, et al. Session Initiation Protocol[S]. RFC3261, 2002.

[5] 王开西, 杨放春. NGN 中基于策略控制的过载控制体系结构[J]. 计算机工程, 2007, 33(24): 102-104.

[6] Resnick P. Internet Message Format[S]. RFC2882, 2001.