第 3 章　系统安全硬件基础

本章导读

系统的安全机制主要是由软件实现的，尤其在大众化的主流系统中更是如此，不管是访问控制机制，还是安全检查机制，亦或是加密支撑机制。但是，就算软件的实现完全没有缺陷，纯软件方法实现的安全机制也是缺乏根基的。本章在分析此类不足的基础上，介绍基于硬件的可信平台的思想，重点介绍可信平台模块的相关技术及其应用。

3.1　问题与发展背景

我们需要从系统实现的角度了解安全机制的纯软件实现所存在的问题。这个问题的关键是看软件本身是否有能力对其自身实施强有力的保护，如果软件自身受到被破坏的威胁，它就很难正常提供安全功能。不幸的是，不借助外在因素，软件很难实现自我保护，或者说，它们会面临自身难保的局面。借助硬件建立可信环境是摆脱困局的一道曙光。

3.1.1　纯软件安全机制的不足

安全机制软件包含程序和配置数据两方面的内容，它们都需要受到保护。程序和配置数据的完整性都会直接影响安全机制的正常工作，有些配置数据的机密性也会影响安全机制正常发挥作用。可问题是，仅凭软件自身，很难掌握和保障程序及配置数据的完整性或机密性。

把安全机制程序篡改为后门程序或木马程序是破坏安全机制程序完整性的突出示例。篡改配置数据为本无权限的攻击者提供特权是破坏配置数据完整性的常发案例。如果安全机制是加密机制，那么，该机制工作时使用的密钥可视同为配置数据，在此情形下，窃取密钥属于严重破坏了配置数据的机密性。

在常规系统中，程序和配置数据都存放在硬盘等存储介质中，只要获得对系统的掌控权，就能从相应介质中得到它们，如果它们没被加密，那么根据需要进行修改并非难事。程序通常都是不加密的。

我们可以假设安全机制软件的实现没有任何缺陷，但无法保证其他软件也都没有缺陷，因为软件如此繁多，而且出自众多开发者之手。因此，攻击者劫取系统控制权的可能性总是存在的。例如，第 1 章附录案例中的卡尔利用一个备份程序的漏洞便获取了操作系统的完全控制权。可见，程序和配置数据的完整性被破坏是不可避免的。

假如程序或配置数据采取了加密措施进行保护，就引出了密钥的管理问题。显然，在有众多对象需要保护的情况下，不可能对所有密钥都进行加密，系统中必然有未受加密保护的密钥存在，而它们只能存放在硬盘等存储介质中，没有其他选择。根据前面的

分析，获取这些密钥并不困难，而一旦密钥被窃取，加密保护就形同虚设。当然，篡改加密程序也能破解加密保护措施。

篡改硬盘等介质中的程序和配置数据不一定会影响已经运行的安全机制，但在下一次重新运行之后，发挥作用的就是已被篡改的安全机制了。另外，获得了系统掌控权的攻击者也可以篡改已经调入内存中的程序和配置数据，这样，被篡改的安全机制很快就能发挥作用，不用等到下一次重新运行。

特别糟糕的是，程序或配置数据被篡改之后，系统还没有办法发现这些问题，因为执行完整性检查任务的程序也可能被篡改。例如，基于 MD5 算法的 md5 程序常用于检查程序或配置数据是否被修改，如果攻击者修改了 md5 程序或该程序用于进行对比的基准值，再用该程序进行检查就不可能得出正确的结果。

归纳起来说，由于难以应对遭篡改的厄运，纯软件实现的安全机制所能建立的安全性是有限的，最根本的原因是硬盘等存储介质中的程序和配置数据暴露在众目睽睽之下，缺乏有效的保护。最起码，系统需要一些有保障的基本功能，例如，对完整性进行检查的功能；也需要一种能防止关键数据外泄的存储空间，例如，用于保存基本密钥的空间。

学术界和工业界为了弥补纯软件方法的不足进行过很多探索，由国际各大软硬件厂商和研究机构共同倡导的可信计算技术就属于其中之一。该技术的基本出发点是借助低成本的硬件芯片建立可信的计算环境，而这类硬件芯片恰好提供基本的完整性度量功能和密钥管理功能。下面先了解一下该技术的发展背景。

3.1.2 可信计算技术的形成

早在 20 世纪 70 年代末，尼巴尔第（G. H. Nibaldi）就对可信计算的概念进行了探讨，建立了可信计算基（TCB，Trusted Computing Base）的思想。该思想为美国国防部的 TCSEC 标准的制定奠定了重要的基础。可信计算基思想的重要启示之一是，通过硬件、固件和软件的合作来构筑系统平台的安全性和可信性。

可信计算要研究的根本问题是信任问题。信任问题的本质是实体行为的可预测性和可控制性，即实体的完整性。因此，如何度量和维护实体的完整性自然是可信计算的关键使命。

软件与硬件相结合是解决完整性度量问题的正确方向，以可信平台模块（TPM，Trusted Platform Module）为基础的可信计算技术是沿着该方向开拓的一种解决问题的途径。

1999 年，可信计算平台联盟（TCPA，Trusted Computing Platform Alliance）的创立，是该技术发展的重要推动因素。起初，TCPA 由微软、英特尔（Intel）、IBM 等 190 家公司组成，它致力于数据安全的可信计算，包括研制密码芯片、特殊的 CPU、主板或操作系统安全内核。

2003 年 4 月，TCPA 演变为可信计算组织（TCG，Trusted Computing Group）。TCG 在 TCPA 强调安全硬件平台构建的宗旨之外，进一步融入了软件安全性的要素，旨在从

跨平台和操作环境的硬件组件与软件接口两个方面促进不依赖特定厂商的可信平台规范的制定。

2003 年年底，TCG 推出 TPM 1.2 规范（最终版定格于 2009 年），该规范得到了业界的广泛采纳，符合该规范的 TPM 产品纷纷推向市场。2013 年年初，TCG 推出 TPM 2.0 规范。与 TPM 1.2 相比，TPM 2.0 有了很多改进。本章主要基于 TPM 2.0 进行介绍。

3.1.3 可信计算的前期基础

TCPA 和 TCG 把可信计算向实用化推进了一大步，而业界在它们成立之前开展的很多研究工作，则为 TCPA 和 TCG 技术体系的建立打下了重要的基础。以下几项工作具有典型的意义。

1991 年，卡内基梅隆大学的泰格（J. D. Tygar）等人提出了基于安全协处理器的 Dyad 系统模型。该模型的安全协处理器硬件为系统提供私密性和完整性支持。它通过数字签名检验操作系统和其他系统软件的完整性，为进入运行状态的操作系统提供完整性验证和加解密等服务，并支持操作系统验证其他组件的完整性、实现信息的加/解密、建立与远程系统的加密连接。

1994 年，美国可信信息系统公司的克拉克（P. C. Clark）和乔治华盛顿大学的霍夫曼（L. J. Hoffman）给出了基于智能卡的引导完整性令牌系统（BITS，Boot Integrity Token System）模型。该模型把系统的主引导程序存放在智能卡中，以保护主引导程序的完整性。存放在智能卡中的还有其他引导文件的哈希值以及用户口令和主机标识。系统启动时，首先验证用户使用智能卡的合法性和智能卡与主机的匹配关系，然后从智能卡中取出主引导程序，开始引导过程。主引导程序从主机中读取其他引导文件，完成引导过程。存放在主机中的文件的完整性借助智能卡中的哈希值进行验证。

1997 年，宾夕法尼亚大学的阿玻（W. A. Arbaugh）等人提出了 AEGIS 安全引导体系结构模型。该模型修改了主机系统的 BIOS，并增加了一个 AEGIS ROM，以实现对可执行代码的完整性检查，如果完整性检查失败，则提供系统恢复支持。该模型把引导过程涉及的系统组件划分为 6 层：第 0 层是基础 BIOS 和 AEGIS ROM，包含验证代码、公钥证书和系统恢复代码；第 1 层是其余 BIOS；第 2 层是扩充的只读存储器；第 3 层是操作系统的引导块；第 4 层是操作系统；第 5 层是应用软件。第 0 层中的软件是可信软件，用作完整性检查链的根。其余各层的可执行代码在执行之前，由低层进行完整性检查。完整性检查通过哈希值和数字签名实现。

2001 年，IBM 沃森研究中心的代尔（J. G. Dyer）、达特茅斯学院的史密斯（S. W. Smith）和美国 Cryptographic Appliances 公司的魏因加特（S. Weingart）等人研制了 IBM 4758 安全协处理器。它在一个物理装置中封装了三组成分：硬件、固件和软件。硬件通过 PCI 接口与主机系统连接。固件包含 POST（上电自检）和微引导程序。软件包含操作系统装载程序、操作系统和应用软件。IBM 4758 是一个独立的缩微计算机系统，支持应用软件在其内部运行，安全应用软件可以部署在其内部，并通过安全协议与主机系统中的软件通信，构成大的应用系统。IBM 4758 内部把组件划分为若干层，借助数字签名实现内部

系统的安全引导。

实际上，IBM 4758 的研究工作远在 1999 年 TCPA 成立以前就已经启动了。以上这些，都是可信计算相关的具有代表性的系统方面的研究工作，它们对 TCG 的可信计算技术体系的建立具有重要的意义。

3.1.4 可信计算的研究热潮

TCPA 和 TCG 可信计算体系规范的推出，引起了工业界和学术界的广泛关注，掀起了一股研究热潮，出现了大量研究项目，产生了大批研究成果。

2004 年，IBM 公司的丸山（H. Maruyama）等人在他们设计实现的 TPod 体系结构中利用可信平台实现了系统的可信引导。在 TPod 实现的可信引导中，基础 BIOS 作为信任根首先执行并度量其余 BIOS 的完整性，其余 BIOS 执行并度量操作系统装载程序 GRUB 的完整性，GRUB 度量操作系统（这里包括 SELinux 内核和/etc/init 脚本等）的完整性。

在完整性度量中，把操作系统等作为简单组件对待属于粗粒度的问题处理方法。从操作系统层开始，直到应用软件层，如果把度量对象的粒度细化到与实际应用系统比较一致的程度，则需要解决更多的问题。细粒度的完整性度量是需要深入研究解决的问题。

2004 年，IBM 沃森研究中心的塞勒（R. Sailer）等人提出了完整性度量的 IMA（Integrity Measurement Architecture）体系结构，设计实现了 Linux 系统的组件细化完整性度量原型系统。IMA 原型以可信平台为可信硬件，给出了系统 BIOS、操作系统装载程序 GRUB、Linux 内核、Linux 模块、应用软件等各层组件执行前的完整性度量方法，重点给出了 Linux 内核、Linux 可装载内核模块、动态可装载库、结构化数据和可执行用户程序等的度量方法。

2005 年，卡内基梅隆大学的史（E. Shi）和 IBM 沃森研究中心的范 • 多恩（L. Van Doorn）等人提出了为分布式系统建立可信环境的 BIND（Binding Instructions and Data）框架。BIND 把代码的完整性证明细化为关键代码段的完整性证明，并为关键代码段产生的每组数据均生成一个认证器。认证器附着到相应数据上，从而实现关键代码段的完整性证明与其所产生的输出数据的绑定。因此，BIND 可以通过关键代码段及其输入数据的完整性证明来达到系统完整性证明的目的。

2006 年，宾夕法尼亚州立大学的耶格（T. Jaeger）、IBM 沃森研究中心的塞勒和加州大学伯克利分校的山克（U. Shankar）提出了基于信息流的 PRIMA（Policy-Reduced Integrity Measurement Architecture）完整性度量体系结构，并研究了以 SELinux 为基础的原型系统。PRIMA 项目的研究工作在 IMA 研究成果的基础上引入了 CW-Lite 信息流模型来处理组件依赖关系，为基于信息流的系统完整性动态度量进行了卓有成效的尝试。

随着网络的深入渗透和新应用的不断兴起，建立信任关系的现实需求显得更加迫切。以上只是可信计算方面的若干典型工作，2006 年之后还有很多成果不断涌现在我们面前。然而，在可信信息系统建设的道路上，依然是“路漫漫其修远兮”，大家尚需“上下而求索”。

3.2 可信平台基本思想

可信平台的主体是常规计算机，可以是普通 PC 机、服务器、智能手机、其他移动设备等，通过添加低成本硬件组件后扩展而成，当然，必须配备相应的软件。可信平台的新增核心功能是检查系统是否出现被篡改的状况，并提供正确的检查结果。新增硬件组件旨在提供可信赖的关键功能和存储保障，有望弥补纯软件措施的不足，支持可信环境的建立。

3.2.1 基本概念

计算机病毒恐怕是人们最熟悉的安全威胁之一，几乎会困扰到每位计算机用户。病毒感染也是系统被篡改的最典型方式之一。知道计算机感染了病毒并不可怕，可怕的是不知道计算机是否受到了病毒感染，这是一种用户不知道是否该相信自己的计算机的状态。建立可信平台就是要增强用户对计算机系统的信心，其中的核心概念是信任。

定义 3.1 信任（Trust）指的是对行为符合预期的认同感。

一个系统是否可信反映的是它值得拥有的用户赋予它的信任的程度，拥有的信任程度较高表示系统比较可信，反之，表示系统不太可信，这取决于系统的行为与用户对它的预期的符合程度。

定义 3.2 信任根（Root of Trust）指的是系统关键的基本元素的集合。它拥有描述平台信任相关特性的最小功能集。它是默认的信任基础。

构成信任根的元素包含提供保护能力的硬件组件，描述平台的信任相关特性意味着要提供对平台的完整性进行度量的基本功能。信任根是平台的组成部分，具有对平台的行为进行度量并提供正确度量结果的基本能力。当然，信任根无法对它自己的行为进行度量，实际上，依靠一个平台是无法检查其中的信任根的行为的，因此，信任根的可信性只能作为假设前提被接受。一个平台的信任根是为该平台建立信任的源头。

定义 3.3 度量核心信任根（CRTM，Core Root of Trust for Measurement）指的是执行完整性度量任务的最基础的指令集合。它是信任根的组成部分。

度量核心信任根是信任根的一部分，它的任务是对平台进行完整性度量，其中的指令是整个平台的完整性度量过程中最早被执行的，它在系统上电的早期阶段就开始工作，可以在 BIOS 开始工作之前就执行完整性度量任务，可以度量 BIOS 的完整性。

定义 3.4 信任传递（Transitive Trust）指的是信任根为可执行的功能建立信任的过程，一个功能的信任建立后，可用于为下一个可执行的功能建立信任。

信任是通过完整性度量建立的，经过度量表明完整性完好的功能就是可信的功能，就会获得相应的信任。平台的完整性度量是一步一步展开的，每步度量得到的结果都可

以作为后续度量的依据，每个环节建立的信任都可以为后续信任的建立提供支持，信任是随着度量工作的展开不断传递的。

定义 3.5 信任链（Chain of Trust）指的是信任根从初始完整性度量起建立的一系列信任组成的序列。

由于信任的传递性效应，从平台上电时的初始度量开始，平台中信任的传递轨迹呈现为一个信任链，信任一环扣一环。

定义 3.6 可信平台模块（TPM）指的是由 TCG 定义的、通常以单芯片形式实现的硬件组件。它拥有独立的处理器、RAM、ROM 和闪存，具有独立于宿主系统的状态，通过专用接口与宿主系统进行交互。它提供的核心功能是存储和报告完整性度量结果。

一般而言，可信平台模块是一个物理上密封的自成体系的低成本硬件芯片，芯片内有程序以及执行这些程序的处理器，还有比较有限的存储空间。TPM 插接在计算机主板上，与主板上的 CPU 并肩工作，芯片内的程序、程序的执行、存储空间都受到物理外壳的保护，提供纯软件方法所缺乏的篡改检测和秘密保持能力。

定义 3.7 由 TPM 实现并以命令形式提供的操作称为受保护功能（Protected Capability），由 TPM 提供的、与外界隔离的、只能通过受保护功能访问的存储空间称为受保护存储区（Shielded Location）。

借助物理外壳屏障，TPM 禁止外壳以外的实体直接操作外壳以内的程序和存储空间，再加上 TPM 内的程序是由 TPM 内的处理器执行的，使用的也是 TPM 内的存储空间，因此，TPM 中的程序提供的功能是得到有效保护的，称为受保护功能。

定义 3.8 可信构造块（TBB，Trusted Building Block）指的是计算机中用于构造信任根的组件的集合，属于信任根的一部分。

可借助图 3.1 以普通 PC 机为例说明 TBB 的含义。

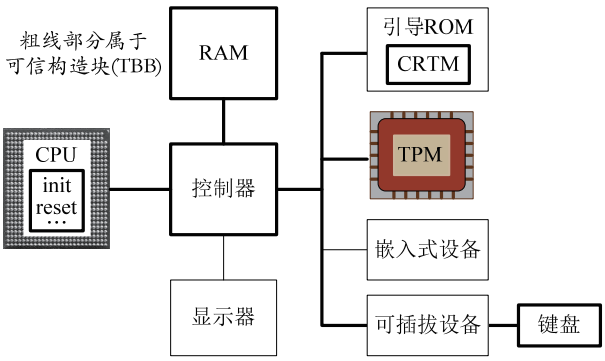


图 3.1 把普通 PC 机扩展成可信平台

图 3.1 中给出的是把普通 PC 机扩展成可信平台的概念框架。其中，主要是添加了 TPM

硬件组件；同时，对存放系统引导初始程序的 ROM 进行了扩展，增加了 CRTM 程序；另外，计算机中的相应关键组件也需要考虑到，尤其要关注的是组件之间的连接途径，参见图中的粗线部分。

注意，图 3.1 中的粗线部分组成了 TBB，它包括 CRTM、CPU 中的 init 和 reset 等指令、控制器、RAM、键盘，以及相应的连接通道。TPM 不属于 TBB 之列。可以说，TBB 和 TPM 构成了信任根，TBB 是信任根中位于受保护存储区之外的部分。

简而言之，把普通 PC 机扩展成可信平台主要包含两大类工作，一是把 TPM 装配到 PC 机中，二是在 PC 机中提供 TBB。两类工作的综合结果便是为 PC 机装备了信任根。

定义 3.9 可信计算基（TCB）指的是系统中负责实现系统安全策略的软硬件资源的集合。它的重要特性之一是能够防止它之外的软硬件对它造成破坏。

如前所述，TCB 是一个历史悠久的概念，在 20 世纪 70 年代被提出来，是安全系统中的基本概念，不是专门针对可信平台的概念。一个 TCB 包含系统中用于实现安全策略的所有元素，是这些元素构成的整体。开发安全系统，关键在于实现 TCB。

3.2.2 信任根的构成

把一台计算机扩展成一个可信平台，相比之下，可信平台具有了信任根，而原来的计算机缺乏信任根。信任根的核心作用是检查平台配置（或称状态）的可信性。

平台状态可信性的检查主要通过完整性度量工作来落实。平台的信任根在某个时刻对平台的状态进行了完整性度量之后，必须把度量结果保存好，以供需要时使用。每当外部实体需要了解平台状态的可信性时，信任根就取出相应的度量结果并提供给它。

显然，信任根需要执行完整性度量、保存度量结果和呈现度量结果三类操作，简称为度量、存储和报告操作。这三类操作都必须正确、可靠地完成，外部实体才可能了解到平台可信性的真实状况。与此相对应，信任根包含度量信任根、存储信任根和报告信任根。

定义 3.10 信任根中，负责完整性度量的部分称为度量信任根（RTM，Root of Trust for Measurement），负责存储度量结果的部分称为存储信任根（RTS，Root of Trust for Storage），负责报告度量结果的部分称为报告信任根（RTR，Root of Trust for Reporting）。

定义 3.3 中定义的度量核心信任根（CRTM）属于度量信任根（RTM）的组成部分，它拥有在建立信任链的过程中最先被执行的代码，参见图 3.2。典型的 RTM 是由 CRTM 控制的 CPU，包括在这种控制之下在该 CPU 上运行的代码。

CRTM 是平台完整性度量的起点，它对平台进行初始度量。每当给平台接通电源时，上电复位功能将平台置于一个预知的初始状态，此时，信任根中以固件形式存在的 CRTM 代码可以获得 CPU 的控制权，由 CPU 执行，对平台的初始配置进行度量，开始建立一条新的信任链。

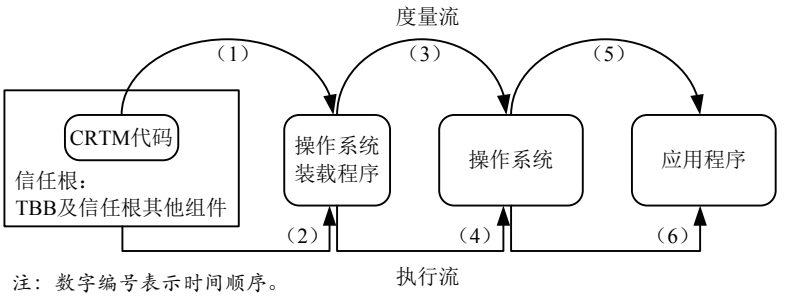


图 3.2　简单系统引导过程中的信任传递

信任传递从信任根开始，信任根提供的功能默认是可信的。首先，信任根度量和描述第二组功能的信任特征，借助该描述，相关实体可以确定该组功能是否值得信任，如果该组功能的可信程度是可以接受的，那么，该组功能的信任便得以建立。这样的过程可以不断重复下去，第二组功能可以为第三组功能建立信任，如此等等。

图 3.2 中由编号（1）、（3）、（5）标注的度量流刻画了操作系统装载程序、操作系统和应用程序的信任建立过程。CRTM 度量操作系统装载程序并为之建立信任，操作系统装载程序度量操作系统并为之建立信任，操作系统度量应用程序并为之建立信任。

图 3.2 中由编号（2）、（4）、（6）标注的执行流描述了信任根、操作系统装载程序、操作系统和应用程序的执行顺序。除信任根外，其他的每个组件都是在其前序组件完成了对它的度量之后才执行的。

图 3.2 展示的是平台组件信任建立过程的简化情形，在实际系统中，信任根为平台组件建立信任的过程会复杂很多，但原理是相同的。

信任根中的 RTM 实施完整性度量时，把度量形成的信息传递给信任根中的 RTS，由 RTS 保存。RTS 由 TPM 提供的受保护存储区和受保护功能构成，用硬件手段为信息提供强有力的存储和保护。TPM 行使 RTS 的职能。

信任根中 RTR 的作用是根据需要为特定实体提供由 RTS 保存的内容。由 RTR 提供的信息称为 RTR 报告。一份典型的 RTR 报告常常是某些保存在 TPM 中的值的摘要，且附有数字签名。RTR 报告所针对的 TPM 中的值的典型类型有：

① 平台配置的相关证据；

② 审计日志；

③ 密钥属性。

RTR 由 TPM 提供的受保护功能构成，它通过与 RTS 的交互完成报告任务。这种交互需要具备抵御软件攻击和物理攻击的能力，以便准确地提供有待报告的真实内容。

信息是否可信与信息的出处很有关系。RTR 提供的报告必须能够反映该报告的来源，以便外部实体能够结合该来源判断该报告是否值得信赖。TPM 以密钥作为 RTR 的标识，相应地，外部实体凭借密钥辨别 RTR 报告的真实性。标识 RTR 的目的是让外部实体知道它收到的报告来自某个 TPM 的 RTR。

用于标识 TPM 身份的密钥称为背书密钥（EK，Endorsement Key），TPM 就是用 EK 来标识 RTR 的。EK 是非对称密钥，基于某个种子生成。每个种子及由其生成的密钥能

唯一地标识一个 TPM。由同一个种子生成的所有非对称密钥都代表同一个 TPM 和 RTR。

RTR 可针对平台的状态给出报告，为使外部实体确信它收到的报告能够准确地反映目标平台的状态，必须证实 RTR 位于目标平台上，或者说，RTR 与目标平台是绑定在一起的。这是靠平台证书来落实的，平台证书证明了 TPM 与平台的物理关联性，从而证明了 TPM 中的 RTR 与平台的绑定关系。

对 EK 的直接使用存在隐私泄露问题。EK 具有唯一性和密码学意义上的可证明性，它能唯一地标识一个确定的 TPM 以及一个确定的平台。过多地使用 EK 会产生行为踪迹的聚合结果，对行为聚合结果的分析很有可能泄露平台用户不愿透露的个人信息。出于保护隐私的考虑，TCG 提倡使用因域而定的签名密钥，限制使用 EK。

3.2.3 对外证明

从构造上看，可信平台是装备了信任根的平台。从功能上看，可信平台有能力让外部实体对平台的可信性进行验证。

平台可信性验证的理念是，由信任根对平台进行度量，并提供度量结果，外部实体根据该结果验证平台是否处于可信状态。这里有一个非常关键的问题，那就是，对于获得的度量结果，外部实体必须能够肯定该结果反映的确实是目标平台的真实状态。

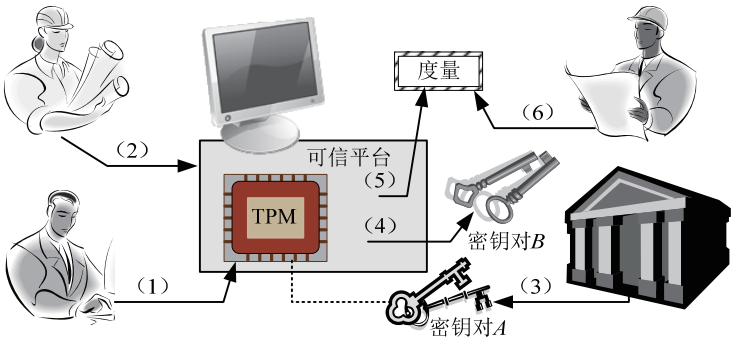
上述外部实体指的是平台以外的实体。为了向这样的实体证明度量结果的真实性，可信平台需要采取一系列的证明措施，此类证明称为对外证明（Attestation）。借助它们可以向外部实体证明平台所拥有的可信相关性质。

我们知道，信任根是可信的，这是预设前提。只要能够证明一个度量是由某平台的信任根提供的，那就等同于证明了该度量反映的就是该平台的真实状态。首当其冲，必须证明提供度量的平台拥有信任根。由于可信平台模块（TPM）是信任根的决定性组件，所以，首先要证明平台中存在一个正宗的 TPM（不是假冒的 TPM）。

可信平台以非对称密钥为纽带实现对外证明。可信平台涉及的对外证明可用图 3.3 加以说明，从最基础层到最贴近应用层，有待证明的内容主要包括：存在正宗的 TPM、平台拥有信任根、基本签名密钥与 TPM 相关联、签名密钥与 TPM 相关联、度量与平台状态相关联、被度量对象的可信性。这些内容的证明依次由图中的（1）～（6）标示。

第（1）层面，证明有一个符合 TCG 规范的真实 TPM 存在，这通常由 TPM 制造商完成。出厂前，制造商为 TPM 生成一套内嵌的背书密钥（EK），并为它颁发证书，称为背书证书。出厂后，TPM 的内嵌 EK 及其配套的证书能够证明该 TPM 的真实性。EK 是用背书种子生成的，TPM 交付使用后，其管理者可用该种子为它重新生成 EK，原证书仍有效。

第（2）层面，证明平台拥有一个度量信任根（RTM）、一个正宗的 TPM 以及一条 RTM 与 TPM 之间的可信路径，这项工作可以由平台制造商承担。证明的方法是提供一个凭证，把安装在平台上的 TPM 的 EK 的公钥信息记载在凭证中，以此作为 TPM 与平台相关联的证据。这样的凭证就是平台证书。RTM、TPM 及两者间的可信路径的组合代表着一个信任根。这个层面的工作证明平台拥有信任根。



(1) 外部实体证明TPM的真实性　(4) 平台证明密钥对受TPM保护
(2) 外部实体证明平台拥有信任根　(5) 平台证明那是平台度量的状态
(3) 证明机构证明密钥对属于TPM　(6) 外部实体证明被度量对象的可信性

图 3.3　可信平台涉及的对外证明

第（3）层面，第三方证书机构（CA）为 TPM 的一套非对称密钥颁发凭证，该凭证载明该套密钥的公钥信息，能证明该套密钥归一个未透露身份的正宗 TPM 所有，并具有特殊属性。这样的凭证称为对外证明密钥证书，相应的 CA 称为对外证明 CA。这里所说的密钥实际上是 TPM 用于签名的基本密钥，如图 3.3 中的密钥对 *A*。

第（4）层面，可信平台要证明一套非对称密钥受到某个未透露身份的正宗 TPM 保护，并具有特殊属性。证明方法是用平台中的 TPM 的签名密钥对该套密钥的相关信息进行签名。例如，图 3.3 中的密钥对 *A* 是 TPM 的基本签名密钥，已得到对外证明 CA 的证明，平台可用它对密钥对 *B* 的相关信息进行签名，以证明密钥对 *B* 受到该 TPM 的保护。密钥对 *B* 指的也是签名密钥，它得到证明之后，也可用于为其他密钥提供证明。

第（3）和（4）层面都是证明相应的非对称密钥得到了 TPM 的保护，这些密钥都属于签名密钥，通常仅用于对受保护存储区中的内容进行签名，换言之，它们仅用于对存储在 TPM 中的内容进行签名。这类受到 TPM 保护、具有专门用途的密钥称为对外证明密钥（AK，Attestation Key）。

第（5）层面，可信平台为一个度量结果提供证明，证明它反映的是软件组件或固件组件在平台中所处的状态。当然，该度量结果是对相应软件或固件进行度量得出的，这是前提。证明方法是，用对外证明密钥（AK）对该度量结果进行签名。例如，图 3.3 中的密钥对 *B* 属于 AK，可用于进行此类签名。支撑该证明方法的道理是，若度量是平台所进行的，则它反映平台的状态。

归纳一下，第（1）层面证明 TPM 是正宗的，第（2）层面证明基于正宗的 TPM 构建了信任根，第（3）层面证明某密钥归平台所有，第（4）层面证明某密钥受到平台的保护，第（5）层面证明度量反映了平台的状态。各个层面的证明是环环相扣的。

第（6）层面，在度量结果信得过的前提下，外部实体验证被度量的软件或固件是否可信，并提供具有证明效力的结论。该证明结论也以凭证的形式呈现，凭证中载明相应度量结果及其所代表的状态。假如外部实体事先掌握了被度量对象的可信状态信息，那么，结合度量结果判断被度量对象是否可信并不困难。

对外证明思想的启示是，可信平台任意状态受信任的程度是依靠多个方面的证明建

立起来的，证明环节不但涉及平台中的软件和硬件因素，还涉及平台以外的其他因素，例如，第三方机构。显然，局限于平台本身很难达到此目的，单纯依靠其中的软件更是力不从心。

3.3 可信平台模块（TPM）

可信平台模块（TPM）为可信平台提供由硬件保护的功能和由硬件保护的存储空间，提供纯软件方法所缺乏的能力，同时，借助非对称密钥，提供平台身份的标识能力。本节介绍 TPM 的基本组成、基本功能和存储区域。

3.3.1 基本组成

TPM 之所以能克服纯软件安全机制的不足，最重要的一点是它提供硬件保护能力，为敏感信息提供保护。需要保护的敏感信息包括数据和密钥等，称为受保护对象。

借助特殊的物理封装，TPM 能够对位于物理外壳以内的内容进行强有力的保护。但是，TPM 的存储能力非常有限，无法容纳所有的受保护对象。视具体应用情况，可能有很多受保护对象存在于 TPM 之外。TPM 采用加密方式来保护必须移至物理边界外的受保护对象。

TPM 的硬件保护能力为受保护对象提供机密性和完整性支持。对于存放在 TPM 内的受保护对象，只有受保护功能才能对它们进行操作，它们的机密性和完整性得到了很好的保护。

对于必须移到 TPM 外的受保护对象，TPM 对它们进行了加密，能防止它们的内容外泄，从而保护了机密性。当受保护对象从 TPM 外装入 TPM 时，TPM 用安全哈希功能验证它们的完整性，如果发现完整性被破坏，TPM 将拒绝装入相应对象，以此保护完整性。

有些受保护对象必须永远驻留在 TPM 内，永不离开 TPM：例如，上下文密钥，是用来对即将离开 TPM 的受保护对象进行加密的密钥，是对称密钥；再如，原始种子，是一种随机数，用来生成密钥。这些密钥用于保护其他受保护对象，而这些受保护对象也可以是密钥，进一步用于保护别的受保护对象。

显然，TPM 主要是通过隔离的存储区域、专用的操作功能和加密技术等实现硬件保护功能的。TPM 的组成结构可以用图 3.4 进行原理性描述。

由图 3.4 可知，TPM 由执行引擎、供电检测单元、管理单元、授权单元、随机数生成单元、密钥生成单元、哈希引擎、非对称引擎、对称引擎、易失性存储器、非易失性存储器、I/O 缓冲区等单元组成。图中粗线条的外边框表示特殊的物理外壳，它从物理上把 TPM 的组成成分与外界隔离开来。TPM 的组成单元通过内部数据通信路径相连，并透过唯一的专门 I/O 接口与外部主机进行连接。

图 3.4 中的 I/O 缓冲区表示 TPM 的 I/O 接口以及 TPM 与外部主机进行交互的缓冲区。缓冲区不属于 TPM 的受保护存储区，它是 TPM 与主机系统共享的区域。TPM 从缓冲区获取功能请求数据，并把响应结果数据送入缓冲区。相反，主机系统把功能请求数据送

入缓冲区，并从缓冲区获取响应结果数据。

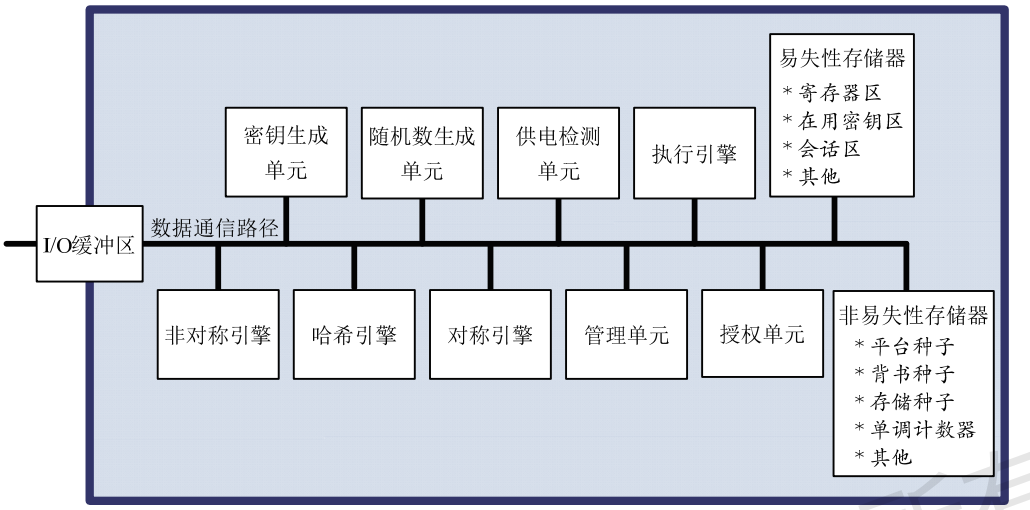


图 3.4　TPM 组成结构

执行引擎是 TPM 中的处理器，属于 TPM 的大脑，它负责执行实现 TPM 各种功能的程序，或者说，受保护功能是由它执行的。

供电检测单元管理 TPM 的供电状态。可信平台的制造商应确保平台的所有供电状态变化都要通知 TPM。TPM 只支持“通电”和“断电”两种供电状态。任何要求 RTM 复位的供电状态转换都会促使 TPM 复位，任何促使 TPM 复位的供电状态转换也都会促使 RTM 复位。

授权单元提供授权检查功能。每执行一条命令，TPM 都要检查授权，验证请求执行命令的实体是否拥有访问相应受保护存储区的必备授权。受保护存储区中的某些内容可能无须授权就可访问，有些内容可能只要符合简单的授权条件就可访问，另一些内容可能需要满足复杂的授权策略才能访问。

管理单元提供对 TPM 的管理和维护功能。其他单元的作用在后续两节中介绍。

3.3.2　基本功能

TPM 提供的受保护功能具有很丰富的内容，其中的大部分都与密码技术密切相关，这从图 3.4 给出的 TPM 组成单元的名称就能看出来。本节简要介绍这方面的基本功能，包括哈希运算、密钥生成、非对称加密与解密、对称加密与解密、非对称签名与验证、对称签名与验证。

TPM 的哈希引擎提供一系列哈希运算功能。在 TPM 内部，哈希功能用于进行完整性检查、授权验证或作为 TPM 的其他功能的基础。哈希功能也可供 TPM 外部的宿主计算机中的软件直接调用。

在 TPM 的众多功能中，完整性度量时使用最频繁的功能之一是扩展（Extend）操作，该操作是以哈希功能为基础实现的，可描述如下：

$$digestnew = H(digestold \| data)$$

扩展操作是针对受保护存储区中的某项内容执行的，其中，H 是哈希运算，digestold 是该项内容原来的值，data 是作为参数的数据，digestnew 是扩展操作完成后得到的该项内容的结果值。扩展操作的作用就是把给定的数据扩展为某项内容中后更新该项内容。

TPM 内部经常需要用到随机数，例如，在生成密钥时，或在签名时。随机数生成单元的作用就是根据需要随时生成随机数。除 TPM 内部使用外，随机数生成功能也可供外部调用，内部使用和外部调用在生成随机数方面效果是一样的。

TPM 的密钥生成单元可以为密码运算生成所需的密钥。TPM 可生成两类密钥，一类是由随机数生成单元直接生成后作为秘密保存在受保护存储区中的密钥，另一类是根据种子生成的密钥，种子通常由随机数生成单元生成并永久保存在 TPM 中。

非对称引擎提供非对称密码运算功能。非对称密码运算使用成对的密钥，其中一个是必须保密的私钥，另一个是对外公开的公钥，公钥可用于加密，私钥可用于解密。TPM 利用非对称密码算法实现对外证明、身份标识和秘密共享等功能。TPM 实现的常见非对称密码算法有 RSA 算法和 ECC 算法等。

对称引擎提供对称密码运算功能。对称密码运算使用单一的密钥，它既是加密密钥，也是解密密钥。TPM 的对称密码运算功能既支持分组密码算法，也支持流密码算法。流密码算法仅用于在传递机密参数时对参数进行加密。分组密码算法可用于加密命令参数，如加密认证信息；也可用于加密要存放到 TPM 之外的受保护对象。

不同的 TPM 产品实现的具体密码算法可以有所不同，可能是对称密码算法，也可能是非对称密码算法。在中国销售的 TPM 产品需要提供中国国家密码主管部门批准使用的密码算法。

TPM 提供签名功能，签名操作既可以基于非对称密码算法，也可以基于对称密码算法。基于非对称密码算法时，签名的方法取决于所采用的算法（RSA 算法或者 ECC 算法）。可用于签名的密钥具有签名属性。签名密钥还可以分为受限密钥和非受限密钥，前者不会被 TPM 用于对消息摘要签名，除非该摘要是由该 TPM 计算得到的。

签名使用私钥，验证签名使用公钥。验证签名的命令接收的参数有公钥的句柄、消息摘要、含有对摘要的签名的数据块。其中，公钥的句柄相当于公钥的标识符。TPM 将检查签名方法与相应的密钥是否一致。如果签名被验证为有效，TPM 将为此生成一个证明。

对于对称密码算法，目前只定义了基于哈希的消息认证码（HMAC，Hash Message Authentication Code）算法的签名方法。它的操作可描述为：

$$HMAC(K, m) = H((K \oplus opad) \| H((K \oplus ipad) \| m))$$

其中，K 是密钥，m 是待认证的消息，H 是哈希算法，$\oplus$ 是异或运算，||是连接运算，opad 和 ipad 分别是外填充数和内填充数，它们的值分别是：

$$opad = 0x5C5C5C\cdots5C5C（长度等于一个块的大小）$$

$$ipad = 0x363636\cdots3636（长度等于一个块的大小）$$

采用 HMAC 算法的签名操作是计算 HMAC(K, m)的值，验证签名操作也是计算该值，然后将两次计算得到的结果进行对比，若相等则验证成功。

3.3.3 存储空间

TPM 内提供两类存储器，即易失性存储器和非易失性存储器。易失性存储器用于存放临时数据，断电时这些数据可能会丢失。非易失性存储器用于长久保存数据，断电时不会引起其中的数据丢失。在功能上，TPM 的易失性存储器类似于 PC 机的内存，非易失性存储器类似于 PC 机的硬盘。

TPM 的易失性存储器的主要用途包括：用作内部寄存器、存放被装载对象、存放会话结构、用作 I/O 缓冲区等。

TPM 最典型的内部寄存器是平台配置寄存器（PCR，Platform Configuration Register），它们属于受保护存储区，主要用于存储与度量日志相关的内容。可信平台的功能之一是维护度量日志，该日志记录平台中从引导开始的一系列影响平台安全状态的事件。日志数据在 TPM 中的保存方法是借助扩展命令追加到 PCR 中。一个 PCR 可以保存一个日志的数据。

存储在 TPM 之外的密钥或数据必须装载到 TPM 中，才能被 TPM 使用或处理，此类密钥或数据统称为被装载对象，在 TPM 中被存放在易失性存储器中。TPM 为每个被装载对象均分配一个标识号，称为句柄。随后的 TPM 命令将通过句柄引用被装载对象。

TPM 利用会话来对操作序列进行控制，会话的作用可以是对操作进行审计、为操作提供授权或对在命令中传递的参数进行加密。会话由 TPM 创建，存放在 TPM 的易失性存储器中，通过句柄引用。

TPM 的 I/O 缓冲区由 TPM 的易失性存储器构成，这部分存储器不属于受保护存储区。

TPM 的非易失性存储器可以保存 TPM 相关的持久状态。生成平台密钥、背书密钥和存储密钥所需的种子都保存在非易失性存储器中。部分非易失性存储器也可分配给平台或经 TPM 属主授权的实体使用。非易失性存储器也用于实现单调计数器。

非易失性存储器包含受保护存储区，受保护存储区只能由受保护功能访问。

有些被装载对象可以长久保存在非易失性存储器中。当 TPM 命令要引用保存在非易失性存储器中的对象时，为了提高访问效率，TPM 可以先把相应对象移到易失性存储器上的对象槽中，然后当作易失性存储器中的对象一样访问。

3.4 TPM 的基本用法

本节介绍 TPM 与外界交互的基本方式，在主机中与 TPM 交互的基本方式，为了给应用提供支持所需要的软件体系结构，以及在应用中使用 TPM 功能的主要方式。

3.4.1 交互数据包

TPM 通过传递数据包的方式与外界进行交互。用于交互的数据包有两种类型，一种是命令包，另一种是响应包。命令包用于由外部程序向 TPM 发送执行命令的请求，响应包用于由 TPM 向外界反馈对命令执行请求的处理结果。

TPM 的命令包和响应包有明确的格式。外部程序按照命令包的格式构造需要由 TPM

执行的命令，根据响应包的格式解读由 TPM 提供的响应结果。命令包和响应包的组成框架如图 3.5 所示。

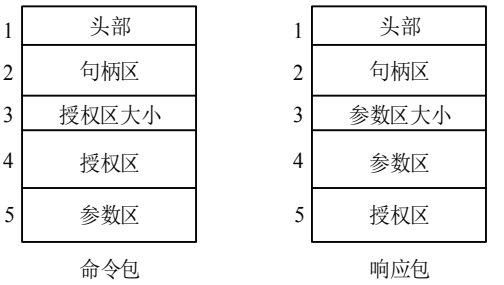


图 3.5　TPM 的命令包和响应包

TPM 命令包由以下 5 个部分组成：

第 1 部分是命令包的头部，描述命令包的总长度和命令码，并说明命令包中是否含有授权区。

第 2 部分是句柄区，给出命令执行时需要用到的句柄。句柄的数量由命令确定，最小值是 0，最大值是 3。

第 3 部分是一个 32 位的值，描述授权区的大小。

第 4 部分是授权区，描述 1～3 个会话结构。这些会话不限于授权型会话，可以是审计型会话或加密型会话。

第 5 部分是参数区，描述向命令提供的参数。

有些命令的执行涉及授权问题，授权由实施授权操作的会话完成。当第 1 部分中的标记表明命令包中不包含授权区时，命令包将没有第 3 和 4 部分的内容。

图 3.6 给出一个 TPM 命令包示例，命令包的内容在粗线方框中，TPM 的数据大小以 8 位为一个单位。图中命令包总长度为 211 个单位，其中，授权区占 61 个单位，参数区占 128 个单位。该命令包涉及两个句柄。

偏移量	大小	成分名称	成分值	
0	2	标记	TPM_ST_SESSIONS	
2	4	命令包长度	211	头部
6	4	命令码	TPM_CC_Example	
10	4	句柄A	相应值	句柄区
14	4	句柄B	相应值	
18	4	授权区大小	61	授权区大小
22	4	authHandle	相应值	
26	2	nonceCallerSize	20	
28	20	nonceCaller	相应值	授权区
48	1	sessionAttributes	相应值	
49	2	hmacSize	32	
51	32	HMAC	相应值	
83	4	dataSize	124	参数区
87	124	data[dataSize]	相应缓冲区	
211			命令包	

注：大小以8位为一个单位。

图 3.6　一个 TPM 命令包示例

TPM 响应包由以下 5 个部分组成：

第 1 部分是响应包的头部，描述响应包的总长度和响应码，并说明响应包中是否含有授权区。

第 2 部分是句柄区，说明命令执行时用到的句柄。句柄的数量由命令确定，最小值是 0，最大值是 3。

第 3 部分是一个 32 位的值，描述参数区的大小。

第 4 部分是参数区，给出由 TPM 产生的值。

第 5 部分是授权区，可以包含 1～3 个会话结构。

图 3.7 给出一个 TPM 响应包示例，响应包的内容在粗线方框中。该响应包总长度为 203 个单位，其中，参数区占 128 个单位，授权区占 57 个单位。该响应包涉及一个句柄。

偏移量	大小	成分名称	成分值	
0	2	标记	TPM_ST_SESSIONS	
2	4	响应包长度	203	头部
6	4	响应码	0（表示成功）	
10	4	句柄	相应值	句柄区
14	4	参数区大小	128	参数区大小
18	4	dataSize	124	参数区
22	124	data[dataSize]	相应缓冲区	
146	2	nonceTpmSize	20	
148	20	nonceTPM	相应值	
168	1	sessionAttributes	相应值	授权区
169	2	hmacSize	32	
171	32	HMAC	相应值	
203			响应包	

注：大小以8位为一个单位。

图 3.7　一个 TPM 响应包示例

3.4.2　原始交互方法

远在操作系统被装载之前，TPM 就要开始工作。在这个阶段，不能指望操作系统中的驱动程序能提供任何帮助，外部程序必须采取最原始的方式，直接与 TPM 打交道，实现与 TPM 进行交互。以 PC 机为例，TPM 通过 LPC 总线接入主板，这为 TPM 在无驱动程序的情况下工作创造了条件。

和其他设备类似，TPM 配有专用的工作寄存器，主机上的程序通过读、写这些寄存器可以启动并指挥 TPM 工作。

TPM 中配有多种工作寄存器，其中包括访问寄存器（AR）、状态寄存器（SR）和数据寄存器（DR）等，如图 3.8 所示。

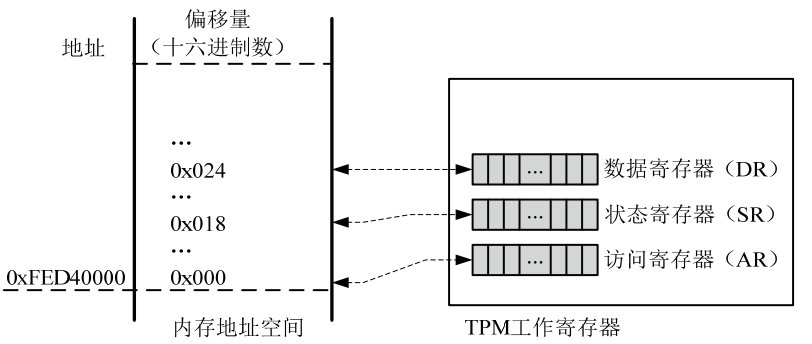


图 3.8　TPM 工作寄存器的内存映射

借助 AR，可发出使用 TPM 的请求，或放弃对 TPM 的控制。

借助 SR，可了解 TPM 是否处于接收命令的就绪状态、TPM 是否已提供了可用的响应信息、TPM 是否正在期待接收更多的数据等，也可以指示 TPM 执行最近写入的命令。

借助 DR，可向 TPM 发送数据，或从 TPM 接收数据。外部程序通过循环多次读或写 DR 完成与 TPM 之间的数据传输。向 TPM 发送命令包时，需要循环多次写 DR。从 TPM 提取响应包时，需要循环多次读 DR。

可信平台的硬件系统支持为 TPM 的工作寄存器实现内存映射，把工作寄存器映射到预留的内存空间中。在图 3.8 中，TPM 的工作寄存器被映射到起始地址为 0xFED40000 的一片内存地址空间中，寄存器 AR、SR 和 DR 在该片空间中的偏移量分别为 0x000、0x018 和 0x024。

实现内存映射后，在主机系统中不需要专门的 I/O 指令，只需使用读、写内存的常规指令就可以访问 TPM 的工作寄存器。例如，使用写内存的指令把数据写入地址 0xFED40024 中，等同于把数据写入 TPM 的 DR 之中。这样，通过读、写指定的内存地址空间，就可以实现与 TPM 的交互。

3.4.3　软件体系结构

在操作系统进入正常工作状态之后，应用程序可以在操作系统的支持下使用 TPM 的功能。为了使 TPM 的功能能够在应用程序中充分发挥作用，需要一个良好的软件体系结构把 TPM 的功能传递给应用程序。

TCG 采用结构层次划分思想描述软件体系结构，即软件栈（TSS，TCG Software Stack），为可信平台定义了三层软件接口，它们分别是设备驱动程序库接口（TDDLI，TPM Device Driver Library Interface）、软件核心服务接口（TCSI，TSS Core Service Interface）和服务提供方接口（TSPI，TCG Service Provider Interface），如图 3.9 所示。

在图 3.9 给出的 TSS 层次结构中，从低到高，共有 6 层内容：底层是 TPM；顶层是应用程序；介于两者之间，由低到高的其他各层依次是 TPM 设备驱动程序、TPM 设备驱动程序库、软件核心服务和服务提供方。其中，TPM 及其设备驱动程序是内核模式（即内核态）的内容，其他都是用户模式（即用户态）的内容。

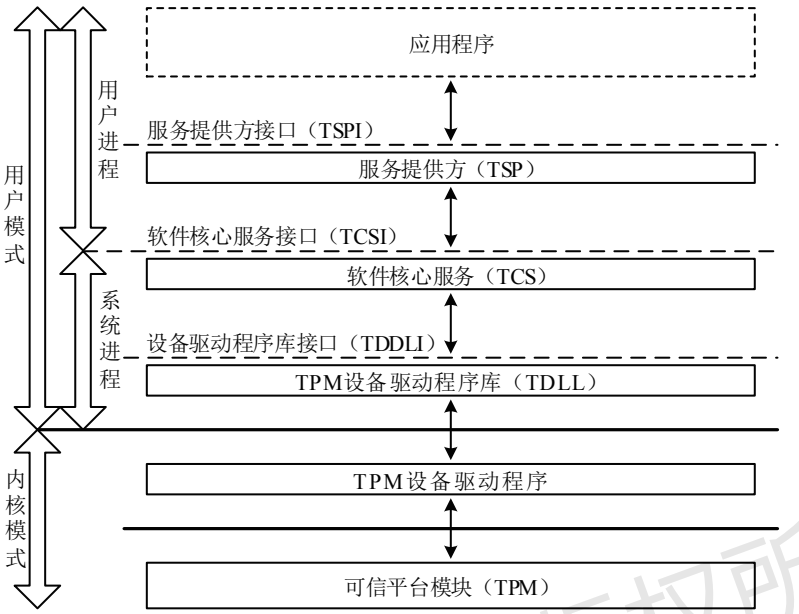


图 3.9　TSS 层次结构

在三层接口中，最下层的接口是 TDDLI，这是上层软件访问 TPM 设备驱动程序库的接口，属于用户模式的接口，与直接访问内核模式的 TPM 设备驱动程序的接口相比，这样的接口有以下优点：

（1）它使得 TSS 的任何一种实现都能够与任何一种 TPM 进行正确的通信；

（2）它能够为 TPM 应用程序提供一种独立于操作系统的接口；

（3）它使得 TPM 的销售商可以向用户提供用户模式的 TPM 软件仿真程序。

TPM 设备驱动程序库（TDDL）实现用户模式与内核模式之间的切换。不过，TDDL 不处理线程级别的软件与 TPM 的交互，也不对 TPM 命令进行串行化处理，这些任务由软件栈中的高层组件去完成。另外，TPM 不支持多线程的访问，只接受单线程的访问，所以，对于每个平台，在任何确定的时刻，只能有一个 TDDL 实例存在。

中间层的接口是 TCSI，它是上层软件访问一组通用的平台服务的接口。虽然，在一个平台上可以存在多个服务提供方（TSP），但是，TCSI 为它们提供共同的软件核心服务（TCS），使它们可以呈现共同的行为。TCS 提供以下 4 个核心服务：

（1）环境管理：实现线程级的 TPM 访问，支持线程访问 TPM；

（2）凭证与密钥管理：存储与平台关联的凭证和密钥；

（3）度量事件管理：管理事件日志记录，管理对关联的 PCR 的访问；

（4）参数块生成：负责对 TPM 命令的串行化、同步和处理。

软件核心服务层以系统进程的形式在用户模式中工作，可以相信它能够管理用户向 TPM 提供的授权信息。

最上层的接口是 TSPI，这是应用程序访问 TPM 的 C 语言接口。TSP 与应用程序在

相同的进程地址空间中工作，有关用户授权方面的工作在这一层中进行。授权操作可以通过本层中的代码提供的用户接口实现，也可以通过 TCS 的回调（Callback）机制实现（如果是远程调用的话）。为了给最终用户提供一个一致的授权接口，本地应用程序不必提供授权服务，它们可以依靠平台中固有的服务来完成授权操作。

TSP 提供环境管理和加密处理等两种服务。环境管理器通过动态的句柄来为应用程序和 TSP 资源的有效利用提供支持，每个句柄为一组相关的可信平台操作提供一个环境，应用程序中的不同线程可以共享同一个环境，每个线程也可以拥有自己独立的环境。

为了全面利用 TPM 提供的保护功能，系统应该提供加密支持功能，不过，TSP 只提供平台操作所必需的相应支持，典型的支持功能包括消息摘要计算和字节流生成等。TSP 没有提供诸如大宗数据加密等方面的接口。

TDDLI、TCSI 和 TSPI 确定了 TCG 的可信平台的软件接口规范。根据这些接口规范，可以方便地建立可信平台的软件系统以及基于可信平台的应用系统。

3.4.4 应用方案

在 TSS 软件体系结构的框架下，利用 TPM 可以建立多种类型的应用。根据不同的需求，以 TPM 为基础的可信平台可以在新的应用基础架构上为应用系统提供支持，也可以在现有应用基础架构上为应用系统提供支持。图 3.10 给出基于 TPM 的三种典型的可信平台应用支持方案。

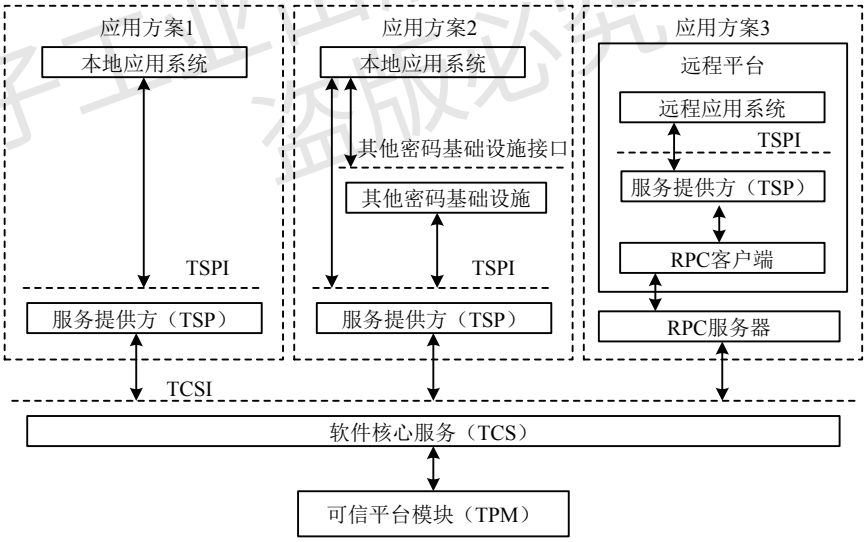


图 3.10　基于 TPM 的三种典型的可信平台应用支持方案

在图 3.10 的三种方案中，两种针对本地应用，一种针对远程应用。在应用方案 1 中，本地应用系统直接建立在可信平台之上。在应用方案 2 中，本地应用系统建立在可信平台与一个现有的加密及安全服务的基础架构之上，该现有的基础架构也可利用可信平台提供的支持功能。在应用方案 3 中，远程应用系统建立在可信平台与远程过程调用（RPC，

Remote Procedure Call）机制之上，其中的 RPC 机制工作在 TCS 与 TSP 之间，把本地可信平台提供的功能传递给远程平台。

3.5 TPM 应用案例

微软在 Windows Vista 操作系统中实现的 BitLocker 机制把 TPM 应用于提供数据保护功能，是 TPM 应用的一个重要案例。本节主要介绍 TPM 在 BitLocker 中的应用方法，而不是对 BitLocker 进行全面介绍。

3.5.1 BitLocker 简介

BitLocker 针对的主要是离线攻击问题，特别是，在计算机丢失或失窃后，它将防止计算机中的数据被泄露。同时，它关心系统的可信引导，着力确保在操作系统之前执行的所有引导相关代码都不存在被篡改的问题，防止它们被植入病毒或 Rootkit 等恶意程序，使得操作系统运行起来并开始接受用户登录时系统处于可信的状态。

概括而言，为了达到数据保护的目的，BitLocker 实现以下两方面的功能：

① 整卷加密：对操作系统所在的硬盘分区进行整分区加密；

② 完整性检查：在引导过程中验证引导组件和引导配置数据的完整性。

为了实现正常引导，BitLocker 要求把系统部署在两个分区中，这两个分区分别是：

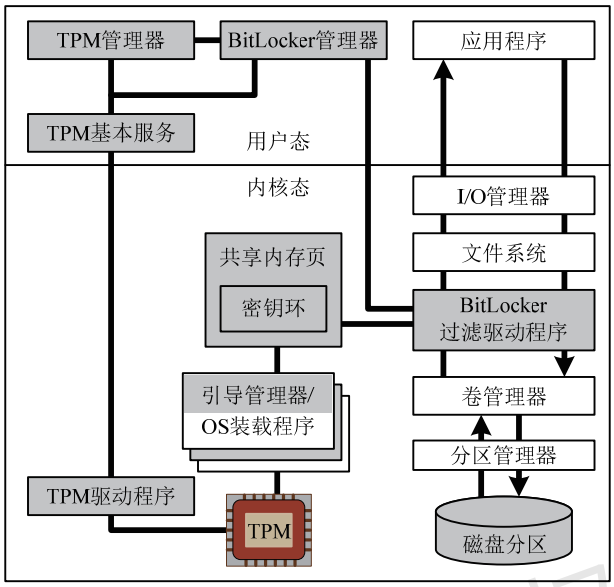
① 系统分区：该分区必须是活动分区，用于存放引导系统所需的代码和数据。它的功能是引导计算机，大小至少是 1.5GB。

② Windows 分区：该分区用于安装操作系统和存放用户数据，包括操作系统、页面文件、休眠文件、临时文件和敏感数据等。BitLocker 对它进行整分区加密。

操作系统安装在 Windows 分区中。按常理，该分区应该被设为活动分区，系统从该分区完成引导过程。但是，由于该分区是加密的，常规的引导程序无法识别其中的内容，无法从中找到操作系统代码并把它装载到内存中。因此，需要借助系统分区进行引导。该分区没有被加密，其中含有引导管理器和引导实用工具，可以帮助对 Windows 分区中的内容进行解密，进而装载安装在 Windows 分区中的操作系统。

BitLocker 的体系结构如图 3.11 所示，其中的阴影部分属于 BitLocker 的组成成分。系统正常工作时，对磁盘分区的加密、解密操作由 BitLocker 过滤驱动程序（fvevol.sys）完成。该驱动程序位于操作系统的文件系统与卷管理器之间，为系统提供透明的加密保护功能。

BitLocker 运用 TPM 硬件的功能为加密保护和完整性检查提供支持。操作系统进入工作状态后，内核中的 TPM 驱动程序（tpm.sys）负责指挥 TPM 工作。该驱动程序把 TPM 的功能传递给用户层的 TPM 基本服务，以便上层程序能方便地使用 TPM 的功能。



注：阴影部分属于BitLocker的组成成分。

图 3.11　BitLocker 的体系结构

3.5.2　BitLocker 整卷加密

BitLocker 以磁盘扇区为单位，对硬盘的 Windows 分区进行加密。除引导扇区、元数据区和坏扇区外，该分区中的其他所有扇区都将被加密。

引导扇区是分区中的第一个扇区，按国际通行惯例，该扇区中存放的是引导计算机所需的基本程序代码。

元数据区是 BitLocker 预留的紧随引导扇区之后的若干扇区，主要用于存放解密用的密钥和对卷进行描述的统计与引用等信息，其中的密钥已被加密。元数据一式三份，存储在元数据区，以防万一某扇区受损后影响 Windows 分区的正常使用。

坏扇区是那些已被标记为不可用的扇区。

简单地说，BitLocker 用数据加密密钥（FVEK）对磁盘卷进行加密，用主密钥（VMK）对 FVEK 进行加密，用 TPM 的存储根密钥（SRK）对 VMK 进行加密，如图 3.12 所示。FVEK 和 VMK 可由 TPM 生成。加密过的 FVEK 和 VMK 都保存在元数据区中。

使用 TPM 对 VMK 进行加密时，可以指明要启用授权功能，启用该功能意味着让用户提供授权信息。图 3.12 中的 PIN 码表示此类授权信息，这是口令类型的信息。如果加密时启用授权功能，则在解密时，TPM 将要求用户提供相同的授权信息。当授权信息不一致时，TPM 将不执行解密操作。

让 TPM 执行对 VMK 进行加密的命令时，也可不启用授权功能，这样，对已被加密的 VMK 进行解密时，就不需要用户提供授权信息。

实际上，为了增强保护强度，BitLocker 并不是简单地用 FVEK 对所有扇区实施相同的加密，而是采用不同的密钥对不同的扇区进行处理。它把 FVEK 分解成两部分，一部

分用于对扇区做预处理，另一部分用于对预处理后的扇区做加密。

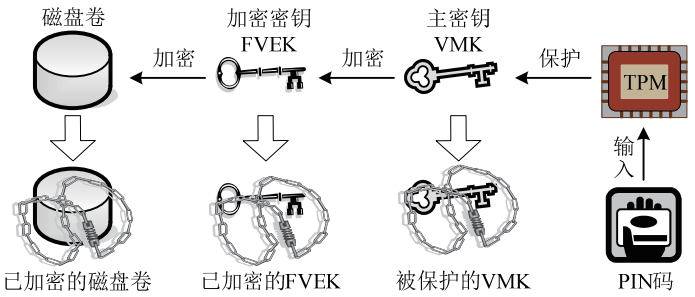


图 3.12　基于 TPM 的 BitLocker 加密

对每个扇区的预处理方法是，把 FVEK 的相应部分与该扇区的扇区号结合起来，导出一个扇区密钥，然后用该扇区密钥对该扇区进行变换。

显然，由于扇区号的不同，不同扇区对应的扇区密钥是不同的。两个不同的扇区，就算里面存放的内容是相同的，经过变换后得到的结果也是不同的。综合起来，相同的信息，存放在不同的扇区中，经过 BitLocker 加密后得到的密文是不同的。这可以增大利用密文推测明文的难度，有助于抵御借助密文的攻击。

虽然在加密处理过程中，除 FVEK 外，BitLocker 还使用了扇区密钥，但那是可以根据 FVEK 和扇区号计算出来的，因此，解密的时候，关键是获得明文的 FVEK。

在计算机引导的过程中，系统分区中的引导组件先运行。引导组件可从 Windows 分区的元数据区中取出密文形式的 VMK 和 FVEK。在启用加密授权的情况下，如果用户能提供正确的 PIN 码，引导组件中的解密代码可请求 TPM 对加密过的 VMK 进行解密，得到明文的 VMK。有了 VMK，就可以对密文的 FVEK 进行解密。

可见，在装载操作系统之前，系统分区中的引导组件就可以解密出 FVEK，此后，用该密钥便能对加密过的磁盘卷进行解密。解密过程如图 3.13 所示。如果对 VMK 加密时没提出授权要求，则系统引导时无须用户提供 PIN 码，引导组件可自动完成解密工作。

顺便提一下，除用 TPM 加密外，BitLocker 也提供用恢复口令对 VMK 进行加密得到的密文副本，万一 TPM 出故障或 PIN 码遗失，可利用恢复口令解密 VMK。

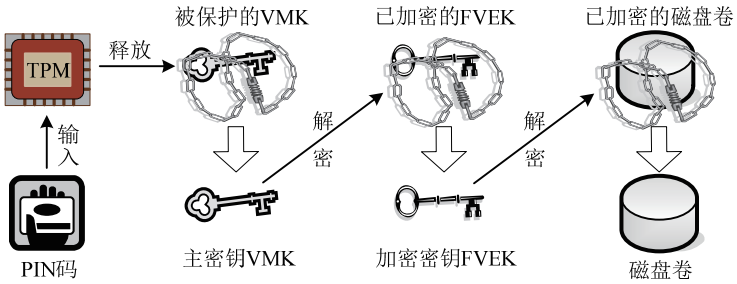


图 3.13　基于 TPM 的 BitLocker 解密

3.5.3 BitLocker 引导检查

本节考察按 BitLocker 的要求划分的两个分区。BitLocker 对 Windows 分区实施了整分区加密的措施，该分区中的信息得到了保护。但是，加密措施没有惠及系统分区，该分区中的信息原样呈现给用户，存在被篡改的威胁。

系统分区中的引导组件肩负着解密 FVEK 和引导计算机的重任，如果它们被植入病毒或木马之类的恶意程序，那么，这些恶意程序轻易就可获得解密后的 FVEK，而 FVEK 是 BitLocker 实施对 Windows 分区进行保护的关键。一旦该密钥泄露，该保护措施自然便被破解。

为了应对系统分区中的引导组件可能遭受篡改和植入恶意程序的威胁，BitLocker 在引导阶段借助 TPM 硬件提供的功能对引导组件进行完整性检查，仅当完整性检查顺利通过时，才相继解密 VMK 和 FVEK，并继续引导过程。一旦发现完整性受到破坏，立刻终止引导过程并锁住计算机。

完整性检查的基本思路是，计算待检查对象的哈希值，并与已知的正确哈希值进行对比，当且仅当前后的值相同时检查通过。基于 TPM 的实施方法是，从上电开始，按照一定次序计算待检查对象的哈希值，并用 TPM 的扩展命令把该值存入 TPM 的 PCR（平台配置寄存器）中，然后检查相应 PCR 中的值与预先掌握的值是否一致。

TPM 提供的封装（Seal）功能可以在一组 PCR 中的值与一个加密操作之间建立映射，具体地说，可以选定一组 PCR，把执行加密操作时其中的值与该加密操作对应起来，使得解密时，只有当该组 PCR 中的值与加密时的值相同时解密操作才能成功。

在 BitLocker 进行完整性检查这个问题上，利用 TPM 封装功能的方法是，选定一组 PCR，在 TPM 用存储根密钥（SRK）对 VMK 进行加密时，建立该组 PCR 中当时的值与 VMK 之间的对应关系，以后，当要对 VMK 进行解密时，该组 PCR 中的值必须与加密 VMK 时的值相同，解密才能成功。

封装命令的逆命令是解封装。封装命令包含加密操作和建立映射操作。解封装命令包含验证映射操作和解密操作，映射验证成功后，解密才能成功。这里所说的映射就是指一组 PCR 中的值与加密操作的映射。

3.5.2 节介绍 BitLocker 的加密过程时，我们简单地说，让 TPM 对 VMK 进行加密，实际上，是让 TPM 对 VMK 进行封装。在图 3.12 中，我们用保护表达封装的意思，用被保护的 VMK 表示封装后的 VMK。相应地，3.5.2 节所说的对加密后的 VMK 进行解密，实际上是对封装后的 VMK 进行解封装。在图 3.13 中，我们用释放表达解封装的意思。

在 BitLocker 初次对 Windows 分区进行整卷加密的过程中，它计算系统分区中指定组件的哈希值，并把它们扩展到指定的 PCR 中，这组寄存器随后用于封装 VMK。完全有理由认为，在初次进行整卷加密时，系统分区中的指定组件都是可信赖的，因此，指定 PCR 中此时的值可以作为以后完整性检查的对比基准。

此后，每次引导计算机时，借助 TPM 对完整性度量的支持，存储在系统分区中的 BitLocker 代码按照预定次序对指定组件进行哈希值计算，并把计算结果扩展到指定的 PCR 中。然后，让 TPM 对 VMK 进行解封装。如果解封装成功，便得到明文的 VMK，

同时，表明完整性检查顺利通过，可以断定指定组件没有出现被篡改现象。

概括地说，BitLocker 利用 TPM 提供的封装功能和对完整性度量的支持，在计算机引导过程中，完成对存储在系统分区中的引导组件的完整性检查，并同时获得主密钥 VMK。

3.6 本章小结

本章针对纯软件安全机制的不足，介绍硬件为安全机制提供的基本支撑，目的是强化一种理念，那就是，硬件和软件相结合才能从根本上应对安全问题。抛开软件的实现普遍不可避免地存在缺陷不说，纯软件安全机制无论在机密性还是完整性方面都存在先天的不足，这些不足需要硬件来弥补。

实现硬件安全机制并不难，难的是推出低成本的硬件解决方案。历史表明，这是硬件安全机制得以普及的决定性因素。在众多的硬件安全方案中，本章选择介绍国际可信计算组织（TCG）推动的以可信平台模块（TPM）为基础的可信平台技术，虽然 TCG 的历史还不长，但 TPM 和可信计算是技术长期发展与积累的结晶。

可信平台技术在安全要素方面的切入点是完整性，也就是可信性。它倡导用硬件建立信任根，从计算机引导阶段开始检查系统的完整性，通过建立信任链，把信任根的支持传递到应用系统，以期确定整个系统的可信性。同时，它强调，不能局限于机制内部，要借助外部力量，依靠对外证明，用全局的视角看待系统的可信性。

TPM 是可信平台的核心组件，但仅仅在计算机中插上一个 TPM 芯片还不足以构成一个可信平台，主板和固件必须相应地提供必要的支持，特别地，BIOS 之类的需要提供支持。TPM 提供由硬件加以保护的安全功能和存储空间，建立防篡改和防泄露的基本保障。

TPM 在计算机引导前期就要开始工作，要掌握它的使用方法，需要同时考虑操作系统工作前和工作后两种情况。在操作系统工作后，程序可以在 TPM 驱动程序的支持下工作，甚至可以调用更高层次的 TPM 服务。在操作系统工作前，程序只能以最原始的方式与 TPM 打交道，相当于也要承担一部分本该驱动程序承担的工作。

Windows Vista 操作系统实现的 BitLocker 机制是一个很好的 TPM 应用案例，它是以 TPM 1.2 为基础实现的。该机制主要实现硬盘的整卷加密功能，也称整盘加密或整分区加密功能，目的是保护硬盘中的数据就算在计算机被盗时也不泄露。同时，该机制也实现引导过程中的完整性检查，这也是为了给整卷加密功能提供保障。该机制利用 TPM 的封装功能巧妙地处理加密和完整性检查的双重需求。该机制提供的保护作用主要在计算机关机时和引导时体现。

3.7 习题

1．为什么说纯软件安全机制在机密性和完整性方面都存在致命的不足？

2．分析并判断对错：可信计算是 1999 年 TCPA 成立之后出现的新技术。

3．从可信计算相关的发展情况看，该领域主要立足于解决哪些方面的问题？

4．把普通计算机改造成可信平台的最重要工作是给计算机配上信任根，具体而言，配置信任根是要给计算机添加什么和改造什么？其目的分别是什么？

5．根据度量、存储和报告的需求，信任根可细分成三种类型，分析并判断对错：三类信任根都位于 TPM 的受保护存储区，得到 TPM 的保护。

6．已知 TPM 用自己的背书密钥（EK）作为报告信任根（RTR）的标识，而 RTR 在生成报告时注明自己的标识，说明外部实体如何确定收到的报告是否来自某个可信平台。

7．验证一个 TPM 是否正宗的方法是，检查它的背书密钥（EK）与背书证书是否匹配，匹配则正宗，不匹配则不正宗。请问以下验证需要检查什么匹配关系？

（1）验证一个平台是否拥有信任根；

（2）验证一对签名密钥是否属于某 TPM；

（3）验证关于某被度量组件的可信性的结论是否真实。

8．验证 TPM 是否正宗时需要假定 TPM 制造商是可信的，请问进行上题中的各项验证时分别需要什么假定？

9．在一个可信平台上，当需要把受保护数据存放到宿主计算机硬盘中时，如何利用 TPM 保护这些数据的机密性和完整性？

10．度量信任根（RTM）复位意味着 PCR 寄存器清零，进而意味着具备建立新信任链的条件，试说说宿主计算机中怎样的供电状态变化可为新信任链的建立创造条件？

11．完整性度量时常用的 TPM 扩展命令可以简要地表示为：

```
TPM2_PCR_Extend(n, data[m])
```

其中，n 是 PCR 寄存器标记，data[m]是待扩展的数据列表，n 和 m 由实际情况确定。命令的功效是：

```
for i=1 to m do PCR[n] = H(PCR[n] || data[i])
```

即把数据列表中的值依次扩展到寄存器 PCR[n]中。

试分析：用一个 PCR 寄存器能否记录一个度量流的度量轨迹？为什么？

12．TPM 的非易失性存储器中存有多种密钥种子，这些种子是哪个单元生成的？哪个单元会使用这些种子？

13．签名都是用密钥标识签名者的身份，无论是基于非对称密码算法还是基于对称密码算法的签名。TPM 同时支持这两种类型的签名，请问这两种签名类型在签名和验证签名方面有什么不同？

14．宿主计算机中的程序可以通过哪几种方式使用 TPM 的存储空间？

15．假设 TPM 的 AR、SR 和 DR 都是 8 位寄存器，请设计一个程序，使用原始交互方式，向 TPM 发送图 3.6 所示的命令包，先说明设计方案，再给出用伪代码表示的程序。

16．假设 TPM 的 AR、SR 和 DR 都是 8 位寄存器，请设计一个程序，使用原始交互方式，从 TPM 接收图 3.7 所示的响应包，先说明设计方案，再给出用伪代码表示的程序。

17．在图 3.9 所示的 TSS 体系结构中，TCS 和 TSP 的软件都提供环境管理功能，这两层环境管理的作用分别是什么？

18．在图 3.10 给出的应用方案中，应用方案 3 所涉及的远程平台是否必须是可信平

台？为什么？

19．图 3.11 所示的 BitLocker 体系结构中的 TPM 基本服务应该对应到 TSS 体系结构中的哪一层？为什么？

20．假如没有引导阶段的完整性检查，为什么向系统分区植入恶意代码就有可能破解 BitLocker 的数据保护机制？

21．在启用了 BitLocker 机制的 Windows 操作系统中，只要在计算机引导过程中能够顺利对主密钥 VMK 进行解封装，就可以断定引导组件的完整性没有被破坏，为什么？