第3章 可信度量技术

可信度量技术是可信计算的关键技术之一。可信计算平台通过可信度量，把信任关系从信任根扩展到整个计算机系统，以确保计算平台可信。本章首先介绍 TCG 的链形信任度量技术，然后介绍带有数据恢复功能的星形信任度量技术，以及基于 TPCM 的信任度量。

3.1 TCG 的链形信任度量技术

TCG 利用计算机启动序列建立信任链。PC 平台的启动序列如下：加电后首先 BIOS/EFI 取得控制权，BIOS/EFI 将初始化一些硬件设备。然后 BIOS/EFI 将控制权传递给系统安装的一些硬件板卡的固件，当这些硬件板卡完成自己的初始化工作后，BIOS/EFI 回收控制权。在此之后，BIOS/EFI 将控制权传递给系统的引导加载程序，引导加载程序加载操作系统内核，然后将控制权传递给操作系统内核。操作系统内核加载并安装各种设备驱动和服务。至此，系统启动完毕，等待执行用户程序。

计算机执行各种任务时，计算机的控制权将会在不同的实体之间传递。那么，用户如何才能知道计算机是否受到恶意攻击，以及计算机是否可信呢？显然，用户希望能够知道计算机的可信状态，以决定是否使用这台计算机或应当采用什么措施确保计算机可信。

TCG 采用度量、存储、报告机制解决这一问题，即对平台的启动序列的可信性进行度量，并对度量的可信值进行安全存储，当用户询问时提供报告。为此，首先要记录系统的启动序列和在启动序列中的可信度量结果，然后才能通过向用户报告启动序列和可信度量结果，向用户报告平台的可信状态。从这一观点看，所谓信任链技术，就是记录系统的启动序列和在启动序列中的可信度量结果的一种实现技术。

3.1.1 TCG 的信任链

TCG 给出的信任链定义如下：CRTM→BIOS/EFI→OS 装载器→OS→应用。其中，可信度量根核(CRTM)是 BIOS/EFI 里面的最先执行的一段代码，用于对后续启动部件进行完整性度量，因此 CRTM 是整个信任链度量的起点。

当系统加电启动时，CRTM 首先对 BIOS/EFI 的完整性进行度量。这种度量就是把 BIOS/EFI 当前代码的可信值计算出来，并把计算结果与预期的可信值进行比较。如果两者一致，则说明 BIOS/EFI 没有被篡改，BIOS/EFI 是可信的；如果两者不一致，则说明

BIOS/EFI 的完整性遭到了破坏，不可信了。如果 BIOS/EFI 是可信的，那么可信的边界将会从 CRTM 扩大到 CRTM＋BIOS/EFI，于是执行 BIOS/EFI。接下来，BIOS 对 OS 装载器进行度量。OS 装载器包括主引导扇区(Master Boot Record，MBR)、操作系统引导扇区等。如果 OS 装载器也是可信的，则信任的边界将会扩大到 CRTM＋BIOS/EFI＋OS 装载器，于是执行操作系统的加载程序。接下来，操作系统加载程序在加载操作系统之前，将首先度量操作系统。如果操作系统是可信的，则信任的边界将扩大到 CRTM＋BIOS/EFI＋OS 装载器＋OS，于是加载并执行操作系统。当操作系统启动以后，由操作系统对应用程序的完整性进行度量。如果应用程序是可信的，则信任的边界将扩大到 CRTM＋BIOS＋OS 装载器＋OS＋应用，于是操作系统加载并执行应用程序。上述过程看起来如同一根链条，环环相扣，因此称之为信任链。可信计算平台的启动和信任链流程如图 3-1 所示。

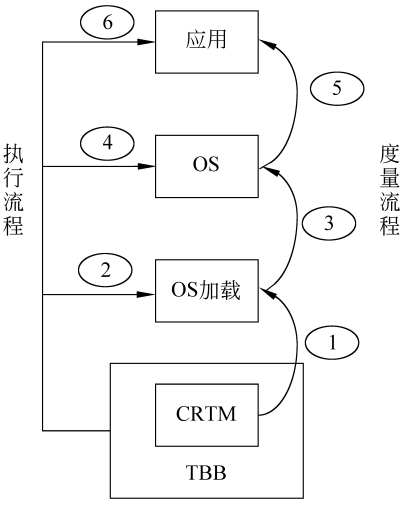

图 3-1　可信计算平台的启动和信任链流程

在信任链的执行过程中，度量的序列反映了系统启动的序列，度量的值就是反映系统可信状态的值，于是将这一过程中的度量值妥善地存储下来，就记录了系统的启动序列和在启动序列中的可信度量结果，为以后的报告机制提供了数据基础。

3.1.2　度量方法

这里需要解决一个问题：采用什么方法度量软件的可信性？这种方法既要能够准确反映软件是否被篡改、是否可信，度量的结果数据又比较短，以节省存储空间。TCG 采用哈希函数度量软件的完整性，以软件的完整性代表软件的可信性。哈希函数具有以下几个明显优点，特别适合这里的应用。

① 可以接受几乎任意长的输入数据，其输出长度固定，一般为几十字节。

② 输入数据的微小改变将引起输出的明显改变。

③ 安全性好，具有不可逆和抗碰撞攻击的能力。

接下来的一个问题是度量的结果存储在哪里？我们首先会想到硬盘有足够大的存储空间。但是，我们现在的数据是反映平台可信性的度量值，必须安全存储，而硬盘的安全性较低。可信计算的一个突出特点是采用了一个 TPM 芯片作为信任根，其安全性比硬盘高。因此，TCG 将可信度量值存储到 TPM 中。具体地，在 TPM 的存储器中专门开辟一片区域作为平台配置寄存器(PCR)，用于存储可信度量值。

为了使存储到 PCR 中的度量值能够反映系统的启动序列，TCG 采用了一种迭代计算哈希值的方式，称为“扩展”操作，即将 PCR 的现值与新值相连，再计算哈希值并作为新的完整性度量值存储到 PCR 中：

$$\text{New PCR}_i = \text{Hash}(\text{Old PCR}_i \parallel \text{New Value})$$，其中符号“‖”表示连接。

根据哈希函数的如下性质：

① 如果 $A \neq B$，则有 $\mathrm{Hash}(A) \neq \mathrm{Hash}(B)$。

② 如果 $A \neq B$，则有 $\mathrm{Hash}(A \parallel B) \neq \mathrm{Hash}(B \parallel A)$。

综上可知，PCR 中存储的度量值不仅能反映当前度量的软件完整性，也能反映系统的启动序列。当前软件的完整性或系统启动序列的任何改变都将引起存储到 PCR 的值的改变。

我们事先把系统启动过程中需要度量的部件的完整性值计算出来，并作为预期值存储起来。当系统实际启动时，信任链技术就会把当前度量的实际完整性值计算出来，并与预期的完整性值进行比较。如果两者一致，则说明被度量部件没有被篡改，数据是完整的，平台是可信的。如果两者不一致，则说明被度量部件被篡改，其数据完整性遭到了破坏，平台不可信了。这样，依靠信任链技术就可以在系统启动过程中检查，以发现系统资源的完整性是否得到确保，从而确保系统资源的完整性和系统的可信性。

3.1.3 度量日志

除了信任链的度量之外，TCG 还采用日志技术与之配合，对信任链过程中的事件记录相应的日志。日志将记录每个部件的度量内容、度量时序以及异常事件等内容。由于日志与 PCR 的内容是相关联的，因此攻击者篡改日志的行为将被发现，这就进一步增强了系统的安全性。

对于 PC 而言，启动过程中的完整性度量事件日志应存放于系统的 ACPI 表中。图 3-2 定义了度量事件日志在 ACPI 表中的存放位置，并定义了该如何找到事件日志的起始位置。

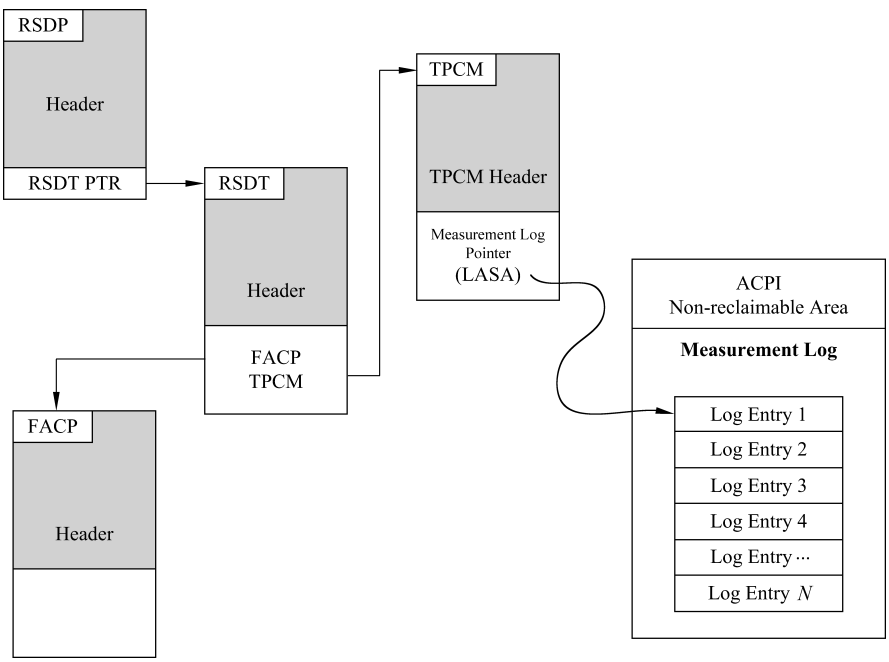

图 3-2 ACPI 表的事件日志结构

3.1.4　TCG 信任链技术的一些不足

根据上面的分析可以知道，TCG 的链式信任链技术的最大优点是实现了可信计算的基本思想：从信任根开始到硬件平台、到操作系统、再到应用，一级测量认证一级，一级信任一级，把这种信任扩展到整个计算机系统，从而确保整个计算机系统可信。其次，这种信任链技术与现有计算机有较好的兼容性，而且实现简单。

但是，TCG 的这种信任链技术具有下列几点不足。

1. 信任链中度量的是数据完整性，不是可信性

可信计算的主要目标是提高和确保计算平台的可信性。为此，应当度量并确保系统资源的可信性。但是，受可信性度量理论的限制，目前尚缺少简单易行的平台可信性度量理论与方法。相比之下，数据完整性的度量理论已经成熟，而且简单易行。TCG 采用数据完整性度量，代替可信性度量。理论和实践都表明，通过数据完整性的度量与检查，可以发现大多数对系统资源的完整性破坏，对确保系统资源的数据完整性和平台的可信性有重要贡献。因此我们说 TCG 的信任链技术是成功的。

但是，完整性并不等于可信性。目前，在学术界对可信性的理解尚没有统一的认识。我们的学术观点是："可信≈安全＋可靠"。人们普遍认为，数据安全性包括数据的秘密性、完整性和可用性。另外，确保数据完整性也是提高系统可靠性的一种措施。显然，数据完整性仅是安全性和可靠性中的一部分，而不是它们的全部。因此，TCG 的信任链度量是有一定局限性的。这一问题的解决依赖于可信度量理论的发展。

其次，由于 TCG 在信任链中采用的是度量数据完整性，因此它能确保数据的完整性，具体确保 BIOS、OS Loader、OS 的数据完整性。但是，数据完整性只能说明这些软件没有被修改，并不能说明这些软件中没有安全缺陷，更不能确保这些软件在运行时的安全性。例如，攻击者可以在一个可信系统中利用缓冲区溢出来替换可执行文件。基于数据完整性的度量是一种静态度量。静态度量可以确保软件加载时的可信性，而不能确保软件在执行时的可信性。因此，我们需要基于软件行为的动态度量。但是，软件的行为是复杂的，对其进行度量是比较困难的。

文献[45]对软件的行为进行了形式化的刻画和分析。国内外许多学者对软件的行为可信性进行了研究，并取得了一些研究成果。我国自然科学基金执行了"可信软件"重大研究计划，推动了对软件可信性的研究。

文献[47]提出一种基于软件行为的动态完整性度量方法。分析可执行文件或源代码的 API 函数调用关系得到软件的预期行为，建立软件预期行为描述集并发布；然后对软件进程的实际 API 函数调用行为进行监控；如果在软件执行过程中，软件行为符合软件预期行为描述集中的相关规则（软件行为认证码），则认为软件是可信的，否则说明软件不可信，此时应该对该软件进行控制。在实际科研项目中把静态度量与动态度量相结合，可得到较好的效果。这种基于行为的动态度量方法需要进一步提升之处是，如何提前感知攻击行为的发生，不能等到攻击行为发生了才感知到。

另外，TCG 的信任链还把信任值二值化，只考虑了可信和不可信两种极端状况，而且

认为在传递过程中没有信任损失，这显然是一种理想化的处理方法。

2. 信任链较长

根据信任理论可知：信任传递的路径越长，信任的损失可能越大。TCG 的信任链

CRTM→BIOS→OS Loader→OS→Applications

从 CRTM 到应用程序，中间经过了很多级的传递，可能产生信任的损失。

3. 信任链的维护麻烦

由于信任度量值的计算采用了一种迭代计算哈希值的“扩展”方式，这就使得在信任链中加入或删除一个部件，信任链中的软件部件更新（如 BIOS 升级、OS 打补丁等），相应的预期可信值都得重新计算，用户维护很麻烦。另外，为了确保系统可信，需要度量所有的可执行部件（如可执行程序、Shell 脚本、Pert 脚本等）。要度量这么多部件，十分麻烦。

4. CRTM 存储在 TPM 之外，容易受到恶意攻击

在实现技术上，CRTM 是一个软件模块，将它存储在 TPM 之外，容易受到恶意攻击。如果能把它存储到 TPM 内部，将会更安全。

3.2 星形信任度量模型与技术

3.2.1 星形信任度量模型

现有的计算机引导方式一般为链式结构，因此 TCG 采用 BIOS Boot Block→BIOS→OSLoader→操作系统→应用的信任度量。显然，这种依照引导方式进行度量的信任链较长。因为信任也是一种信息，因此在传输过程中会有损失，而且传输的路径越长，损失可能越大。由于链式信任度量方法的信任链很长，因此容易产生信任衰减。如图 3-3 所示，如果节点 A 对节点 B 的信任值为 $T(A,B)$，节点 B 对节点 C 的信任值为 $T(B,C)$，用 $T(A,C)$表示经由 B 点传递的 A、C 之间的信任关系。由传递性可以推断出 A 和 C 之间的信任值取两者之间的最小信任值，从而产生信任衰减。

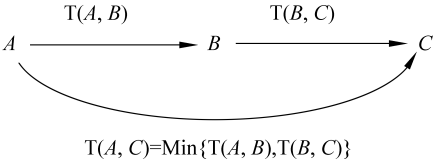

图 3-3 信任衰减

文献[14]提出一种星形信任度量模型。星形信任度量模型采用一级度量的方法，大大缩短了信任链长度。与链式结构一级度量一级的方案不同，星形信任度量模型是利用可信根分别对引导程序、操作系统内核、应用程序等进行度量。因为采用一级度量，从而解决了链式信任度量中的信任衰减问题。通过完整性度量后，TPM 才允许 CPU 启动平台。因为所有度量均由可信根的 CPU 进行，不是由平台的 CPU 度量，所以是可信根的

主动度量。星形信任度量模型如图 3-4 所示。

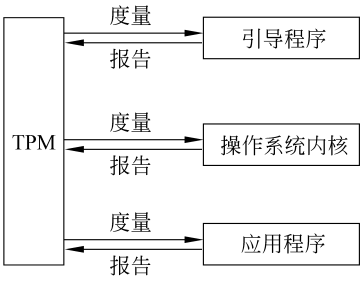

图 3-4　星形信任度量模型

在星形信任结构中，引导程序、操作系统内核和应用程序直接被可信根度量过，因此可以保证系统在启动过程中是安全的，软件没有被非法篡改。由于星形信任结构的信任值计算是根节点与每一个度量节点之间的独立计算结果，互相之间不会产生影响，因此添加和删除一个部件以及软件版本升级，根节点对该节点单独重新计算即可，不会影响到整个信任链。而且，星形信任关系不在信任代理间传递，度量部件均为一级度量，可以有效避免信任损失。

星形信任结构也有不足之处：在星形结构中，由于根节点处于中心位置，在平台的工作过程中需要不断地对各节点进行完整性度量和可信度的判断，因此可信根的计算量较大。由于可信根芯片 CPU 的效率一般低于平台主 CPU，因此有可能影响平台的工作效率。

3.2.2　具有数据恢复的星形信任度量

根据“可信≈可靠＋安全”的学术思想，可信计算机系统应该具备较好的可靠性。因此，信任根若检测到某个软件受到攻击并被篡改，除了阻止该软件运行之外，需要提供一种机制，以使该软件能够恢复到其正常的状态，保证平台的正常工作不受干扰。星形信任度量模型的备份恢复机制需要对 TCG 规范中的 TPM 结构进行扩展，将 RTM、RTS 和 RTR 集成在 TPM 芯片中，并要在平台中开辟一个受 TPM 直接管理的系统备份存储器，保存计算机系统引导程序和操作系统等希望被保护的内容代码。当 TPM 发现某个被度量的部件被篡改后，便直接调用系统备份存储器进行数据恢复，保证平台正常启动并运行。图 3-5 为具有数据恢复的星形信任度量模型。

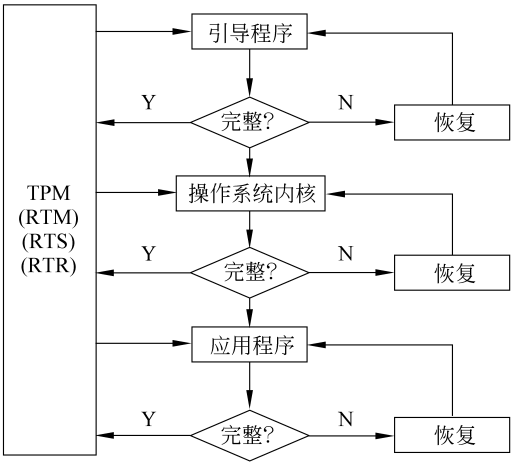

图 3-5　具有数据恢复的星形信任度量模型

如图 3-5 所示，TPM 在系统启动之前对引导程序、操作系统内核、应用程序分别进行完整性校验，若校验未通过，则认为以上内容被篡改，TPM 将从受保护的备份存储器中读取相应的备份代码，将其写入外部工作用存储器，并再次进行完整性校验，若通过校验，表明系统恢复成功，允许该软件启动。该模型为备份存储器提供受保护的安全存储环境，在进行完整性校验和数据恢复的过程中不会受到外部干扰，从而大大提高了系统的可靠性和可用性。

3.3 基于 TPCM 的信任度量

TPCM 是我国自主设计的一种可信平台模块方案。它在继承了 TCG 的 TPM 的一些优点外，也针对 TPM 的一些缺点进行了有效的改进，具有自己的一些突出特点。其主要特点是，平台启动时 TPCM 首先掌握对平台的控制权，并对平台关键部件进行完整性度量。本节介绍国家标准 GB/T 29827-2013《信息安全技术 可信计算规范 可信平台主板功能接口》里所定义的信任度量技术。

3.3.1 TPCM 优先启动与度量

通过主板电路设计保障实现 TPCM 和计算机主板其他通用部件之间的加电启动时序控制。确保用户启动 PC 后，TPCM 先于主板其他通用部件启动。RTM 认证并度量完扩展度量模块 1(Extended Measurement Module 1，EMM1)后，TPCM 发出控制信号启动主板的 CPU、芯片组和动态存储器等通用部件。其中 EMM1 为 Boot ROM 中的一段程序代码。如图 3-6 所示，TPCM 优先启动以及对 EMM1 的完整性度量流程为

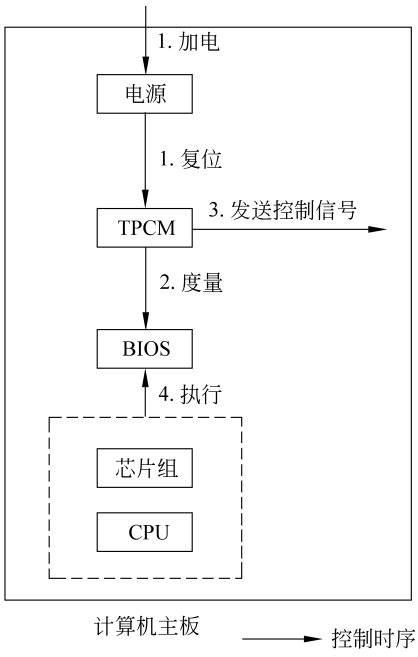

图 3-6　TPCM 优先启动以及对 EMM1 的完整性度量流程

① 用户启动 PC；

② TPCM 先于主板其他通用部件获得复位信号并执行 RTM；

③ TPCM 和主板其他通用部件设计电路，确保 RTM 能可靠地读取 Boot ROM 中的一段程序代码 EMM1，对其进行身份认证；

④ RTM 度量 EMM1 完整性；

⑤ RTM 度量 EMM1 完毕，TPCM 发出控制信号使主板的其他通用部件复位，开始正常开机引导过程。

3.3.2　扩展度量模块和信任传递

扩展度量模块(Extended Measurement Module，EMM)是指接受了完整性度量检查并被装载到当前执行环境中，度量后续代码装载的部件。作为 RTM 度量引擎的扩展度量模块，实现对执行部件的完整性度量，确保信任传递。基于 TPCM 的可信 PC 平台的 EMM 主要有 EMM1、EMM2 和 EMM3。

① EMM1：通过 RTM 对其进行完整性度量，并被装载到系统的一段 Boot ROM 初始引导程序，作为主板开机引导中装载执行部件对 Boot ROM 其他部件执行完整性度量的扩展度量模块。

② EMM2(BIOS/UEFI)：通过 EMM1 对其进行完整性度量，并被装载的扩展度量模块实体，负责对操作系统装载器进行完整性度量与可信装载的一段 Boot ROM 程序。

③ EMM3：存储于外部存储器中用来装载操作系统内核的执行部件，通过 EMM2 对其进行完整性度量并被完整性装载到主板系统中，对被装载的操作系统内核执行完整性度量的扩展度量模块。

对于 PC 而言，信任链包括从开机到操作系统内核装载之前的可信启动过程，基于部件完整性度量与扩展度量部件，保障部件代码的装载和执行环境的可信。由 TPCM 初始信任状态发起，以扩展度量模块为节点的信任传递与扩展，实现传递部件之间的信任关系。信任链上主要部件之间的相互协作关系是：RTM 度量 EMM1，EMM1 度量 EMM2，EMM2 度量 EMM3，EMM3 度量操作系统内核；可信计算主板以 RTM、EMM1、EMM2 和 EMM3 为节点搭建信任链传递，如图 3-7 所示。信任链的建立过程满足如下要求。

① 信任链的建立过程以 RTM 为起点。

② 当需要装载一个部件到当前环境上运行时，首先对该部件进行完整性度量，然后再将其加载。

③ 当主板系统启动完成后，可信根 TPCM 中的 PCR 寄存器记录了按照系统事件启动顺序依次迭代计算的哈希值。

④ PCR 中的哈希值与启动过程中的部件启动事件相对应。

⑤ PCR 中的哈希值与系统引导过程的度量日志记录启动顺序相对应。

⑥ PCR 中的哈希值和度量事件日志，在每次开机过程中重新计算生成。

PC 从开机引导到 OS 内核装载之前，以信任链建立为核心功能，基于 TPCM 信任根支撑的主板系统实现部件完整性度量与被度量，度量执行部件与被度量部件角色变化的信任传递关系和一般流程如下。

① TPCM 先于计算机主板其他部件(包括主 CPU)启动并确保 RTM 作为信任起点的度量执行部件,可靠地读取 Boot ROM 中的 EMM1 并对其执行完整性度量,生成的哈希值和日志临时存储于 TPCM 中;

② TPCM 发送控制信号,使主板其他部件(如 CPU、芯片组和动态存储器等)复位,开始执行 Boot ROM 中的 EMM1 模块,构建以 EMM1 为当前系统控制和执行部件的可信执行环境;

③ EMM1 获得系统执行控制权,作为 RTM 的扩展度量模块执行部件;

④ EMM1 对 EMM2 部件进行完整性度量,并临时寄存当前哈希值和日志;

⑤ EMM2 获得当前系统控制权,对 EMM3 执行完整性度量,并将哈希值存储到 TPCM 的 PCR 中,度量事件日志保存到 ACPI 表中;

⑥ EMM3 获得当前度量执行部件控制权,作为可信度量执行部件;

⑦ EMM3 对操作系统内核执行完整性度量,将哈希值存储到 TPCM 的 PCR 中,并把日志保存到 ACPI 表中。

经过上述步骤后,主板系统以 TPCM 中的 RTM 为度量起点,以 EMM1、EMM2 和 EMM3 部件为度量代理的信任链建构完成,保障 OS 内核获得一个可信的执行环境。

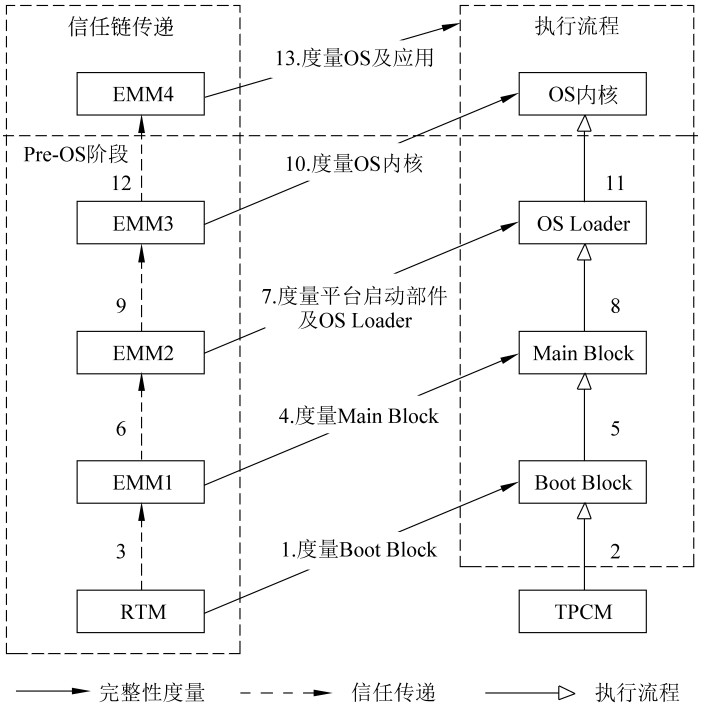

图 3-7　主板引导过程部件信任传递关系网络和信任链建立流程

3.3.3　一种 TPCM 度量 Boot ROM 的实现方法

基于 TPCM 的可信计算平台主板电路设计确保 TPCM 先于主板通用部件获得复位

信号并可信引导，完成对 Boot ROM 的 EMM1 度量；度量完毕，TPCM 发送控制信号使主机板的其他通用部件加电复位，实现开机可信引导。TPCM 和通用主机平台的电路组成结构及双机开机时序关系如图 3-8 所示。

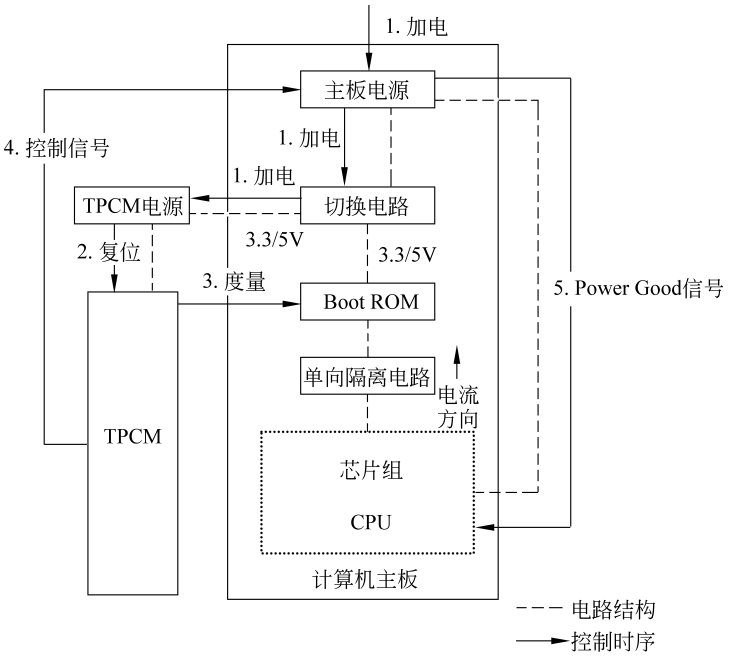

图 3-8　双系统开机时序关系

TPCM 先于主板启动的设计特点如下。

① 平台主板设计上满足 TPCM 和其他通用部件之间基于双电源模块加电。

② TPCM 外围电源与信号等采用独立电路系统设计。

③ 通过主板的通用部件电源控制功能截获其加电和复位信号，实现双系统在用户开机过程的分时控制。平台主板的其他通用部件电源设计实现切换与控制电路功能，在冷启动和主板复位过程，截获主机电源 Power Supply 的“Power Good”信号，隔离除 TPCM 外的其他通用部件的电压信号、复位信号和时钟信号等；在获得 TPCM 发出的控制信号时，切换与控制电路给通用部件（如 CPU、芯片组等）分发电压信号、时钟信号和复位信号等，主板通用部件进入正常开机引导。

④ 在有选择性实现 TPCM 与主板其他通用部件加电工作过程中，如 Boot ROM 芯片加电，TPCM 及外围部件电路系统的电源和主板其他不需要加电的部件采用单向隔离电路保护，确保电流不能倒灌进入平台主板的其他通用部件。TPCM 及外围部件电路系统的电源，在其他通用部件加电复位后，其单向隔离电路被正向电流导通，TPCM 重新复位；TPCM 电源设计实现智能检测到其他部件的“倒灌”电流而出现断路保护，停止输出电压信号。

TPCM 先于主板启动的平台可信引导过程如下。

① Power On：用户启动开机按钮动作，Power Supply 发出“Power Good”信号。

② 计算机主板通用部件电源截获“Power Good”信号，切断给通用部件加电；将电源转发给 TPCM 电源。

③ 在隔离保护电路支撑下，TPCM 电源模块给 TPCM 加电和给主板 Boot ROM 电路供电。

④ TPCM 嵌入式系统启动运行。

⑤ TPCM 自我可靠性启动与部件完整性检验。

⑥ TPCM 检测物理防护电路和主板 Boot ROM 连接电路 SPI 总线的可靠性。

⑦ TPCM 可靠地读取 Boot ROM 中的 EMM1 代码。

⑧ 完整性度量 EMM1 并扩展 PCR 值。

⑨ 度量 EMM1 执行完毕，TPCM 给主板通用部件电源发送控制指令，切换电路使主板电源给通用部件加电、复位，使主板正常启动。

3.4 TCG 动态可信度量根技术

信任链是可信计算的关键技术，通过信任链的作用确保系统资源的数据完整性，从而提高系统的可信性。但是，早期的信任链只在平台启动时才进行一次完整性校验。这对于频繁开机和关机的 PC 来说，是基本可以的。但是，对于服务器这种一旦开机后就长时间不关机的计算平台来说，远远不够。用户很难相信开机时几分钟的数据完整性度量，很难确保服务器几年的数据完整性。因此，需要能够反复进行系统完整性度量的信任链机制。

在工业界，Intel、AMD 等 CPU 厂商都致力于通过改进 CPU 的体系结构增强计算平台的安全性。最著名的是 Intel 的 LaGrande 计划和 AMD 的 Presidio 计划。这两个计划都是通过改进 CPU 和芯片组的安全功能来增强平台的安全性。在这些技术的支持下，可以实现信任链的多次完整性度量，因而比较适合服务器。TCG 称这种可多次度量的技术为动态可信度量根（Dynamic Root of Trust Measurement，DRTM）技术，而称原来的可信度量技术为静态可信度量根（Static Root of Trust Measurement，SRTM）技术。

LaGrande 计划的体系结构包含了很多安全特性，如它具有可信执行技术（Trusted eXecution Technology，TXT）特性的 CPU 和 TCG TPM。为了能够反复进行系统完整性度量，Intel 的 CPU 中增加了一条新指令 SENTER。这条指令能够创建可控和可认证的执行环境。该环境不受系统中任何部件的影响，因此可以确保执行这条指令所加载的程序的执行不会被恶意篡改。

LaGrande 体系结构的保护目标主要如下。

① 保护执行：在 CPU 芯片中提供一个安全的区域来运行一些敏感的应用程序，以使得用户即使处于可能被攻击的环境中，仍然能够相信这个区域的安全性。

② 密封存储：采用密码保护用户的数据，使用户即使处于可能被攻击的环境中，仍能够相信被保护数据的机密性和完整性。

③ 远程证明：提供一种机制，使用户能够相信一个远程平台是可信的。

④ I/O 保护：对平台的 I/O 进行保护，使用户和应用程序之间的交互路径是可信路径。

2005 年，AMD 宣布了包含虚拟机和安全扩展的下一代 CPU 规范。在其扩展指令中有一条 SKINIT 指令，用以支持反复进行系统完整性度量。SKINIT 指令对 CPU 进行初始化，为程序建立一个安全执行环境。在这个环境里，屏蔽了所有中断，关闭了虚拟内存，禁止 DMA，除正在执行的这条指令的处理器外，其他处理器都不工作。用于加载虚拟机管理器(VMM)的程序被封装到一个安全的加载程序块中。在 CPU 执行安全加载程序前，这个程序要被度量，其度量值被写进特定 PCR 中。这个 PCR 只能通过特殊的 LPC 总线周期才能读取，而软件是无法模拟这个 LPC 总线周期的，这就保证了只有 CPU 能够读取这个特定的 PCR。这个特定 PCR 的值就是信任根的度量值，CPU 的 SKINIT 指令就是可信度量根。为了防止 SKINIT 指令被攻击，前面所述的所有步骤和条件都是以原子形式执行的。

有了 Intel 和 AMD 的上述技术支持，TCG 提出了 DRTM 的概念。DRTM 是与 SRTM 相对而言的，之所以称之为动态，是因为其信任度量不仅可以在平台启动的时候进行一次，还可以在平台启动以后的任何时刻再次进行。

TCG 的 DRTM 是 CPU 的一条新指令，在 Intel 的 CPU 中这条新指令是 SENTER，在 AMD 的 CPU 中这条新指令是 SKINIT。这条新指令的执行告诉 TPM 开始信任度量，并创建一个可信的计算环境。信任链由这条新指令开始信任度量，并重置 PCR 寄存器。这种信任链可以在任何时候执行信任链、创建可信计算环境，既可以在平台启动时，也可以在平台启动后的任何时候。

由于这种动态信任度量起始于 CPU 的一条新指令，而且不需要重启平台就能进行，所以信任链排除了 BIOS 及其配置、可选 ROM 及其配置、Boot Loader，因此在静态度量情况下针对 BIOS 和 Boot Loader 的攻击也就失效了，从而提高了平台的安全性。

动态可信度量的目的是引导一个可信的操作系统，建立一个可信的环境。但是，这个可信的操作系统会创建多个互相隔离的安全区域，以允许用户使用自己的操作系统或应用程序。当可信操作系统、用户普通操作系统或应用程序都使用同一个 TPM 时，就会发生一个问题。这个问题就是 TPM 如何区分它们呢？换句话说，就是 TPM 如何区分度量的命令和度量值属于可信操作系统、用户操作系统，还是属于应用程序呢？

为了解决这个问题，TCG 在 TPM v1.2 中引入了 Locality 的概念，并用 Locality 与不同的度量请求相关联，从而达到能够区分度量请求者的目的。共定义了 5 个不同的 Locality：Locality 0 与 TPM v1.1b 的设备关联；Locality 1～Locality 3 没有特别规定，一般 Locality 1 与可信操作系统关联，Locality 2 与用户操作系统关联，Locality 3 与用户应用程序关联；Locality 4 与 CPU 关联，用于认证 DRTM。

对于静态可信度量，TCG 在 TPM v1.1 中规定 PCR0～PCR15 仅在平台重启时才被允许重置成初始值。为了支持动态可信度量，TCG 在 TPM v1.2 中新增了 PCR16～PCR23，并允许在动态度量时把 PCR16～PCR23 重置成初始值。

最后我们指出，TCG 的 DRTM 技术可以实现信任链的多次度量，而且克服了 SRTM 的一些缺点。它进一步提高了平台系统的安全性，而且比原来的静态可信度量更适合服

务器应用。这些都是有积极意义的。Intel 已经将支持 DRTM 的 TXT 技术升级为 SGX (Software Guard Extensions)技术。

但是应当指出,DRTM 技术只是实现信任链的多次度量,其度量的内容还是软件的数据完整性,并不是软件的行为可信性,因此仍然不能确保软件的行为可信。确保软件的行为可信是一件困难的事情,在这方面目前还缺少完善的理论和技术,需要我们进一步开展研究。

3.5 本章小结

可信计算平台通过可信度量技术,把信任关系从信任根扩展到整个计算机系统。TCG 的信任链技术总体上是成功的。它的最大优点是实现了可信计算的基本思想:从信任根开始到硬件平台,到操作系统,再到应用,一级测量认证一级,一级信任一级,把这种信任扩展到整个计算机系统,从而确保整个计算机系统可信。其次,这种信任链技术与现有计算机有较好的兼容性,而且实现简单,但是它也有一些明显的不足之处。

具有数据恢复功能的星形信任结构减少了信任传递,基于 TPCM 的信任度量确保 TPCM 首先启动,完成对 Boot ROM 程序的可信度量。DRTM 技术从一条安全指令开始信任度量,可以在平台运行时创建可信计算环境,不需要重启平台。

我们完全相信,在世界各国的共同努力下,可信度量技术将会越来越完善,将在可信计算中发挥越来越大的作用。

习题

(1) TCG 链形信任度量为何将启动过程中的完整性度量事件日志存放于系统的 ACPI 表中?

(2) TCG 链形信任度量技术有哪些不足之处?

(3) 星形信任度量技术有哪些优势?

(4) 基于 TPCM 的信任度量如何实现 TPCM 先于主板通用部件启动,完成对 Boot ROM 关键代码的度量?

实验

Trusted Grub 实验。

① 请到 https://github.com/Rohde-Schwarz/TrustedGRUB2 处下载;

② 编译、安装、试运行 Trusted Grub2,若无 TPM 支持,请用软件模拟;

③ 试用其支持的可信相关命令工具;

④ 分析其实现原理。

第4章 可信软件栈

4.1 可信软件栈的概念

可信软件栈是可信计算平台的重要组成部分。它基于可信计算平台的硬件资源，特别是TPM，提供支撑可信计算目标的可信服务，并保证系统软件与环境的可信性。可信软件栈位于应用软件与TPM之间，是应用软件方便使用TPM的桥梁，应用程序通过调用可信软件栈的接口使用TPM提供的安全功能。

4.2 TCG TPM 1.2 软件栈

TCG的软件栈(TCG Software Stack，TSS)，是TCG定义的一种为上层的可信计算应用提供访问TPM接口的软件系统，是可信计算平台体系中必不可少的组成部分。在整个体系中，TPM是整个平台的信任根；信任链将信任从信任根依次传递给BIOS、操作系统内核和应用程序。TSS为应用程序访问TPM提供支持，并对TPM进行管理。

国际TCG为可信软件栈制定了一系列的技术规范，2003年10月推出TSS Specification Version 1.1规范，2007年4月在前一规范基础上做了进一步修改，制定出TSS Specification Version 1.2规范，该规范中指出："TSS的设计目标包括：为应用程序调用TPM安全保护功能提供入口点，提供对TPM的同步访问，向应用程序隐藏TPM所建立的功能命令以及管理TPM资源"。

在整个可信计算平台体系中，可信软件栈(TSS)是必不可少的系统组件。首先，为了应用软件能够方便地使用TPM的可信计算功能，应该提供相关的软件支持，如设备驱动、设备应用接口等。其次，由于TPM的计算能力和存储资源有限，TPM不可能独立完成可信计算的所有功能，必须借助TSS的参与。

4.2.1 TCG TPM 1.2 软件栈体系结构

可信软件栈具有多层次的体系结构，如图4-1所示，可分为TSS服务提供(TCG Service Provider，TSP)层、TSS核心服务(TCG Core Service，TCS)层和TSS设备驱动库(TCG Device Driver Library，TDDL)，各个层次都定义了规范化的函数接口。TSP主要作为本地和远程应用的可信代理，TCS用于提供公共服务的集合，而TDDL负责与TPM

的交互。

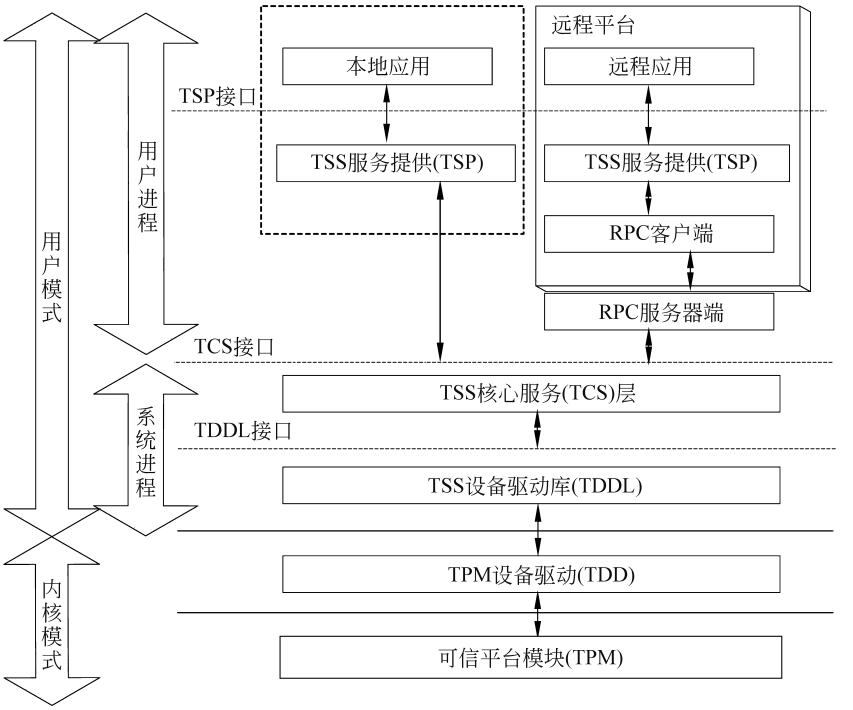

图 4-1　TCG TPM 1.2 TSS 体系结构图

TSP 层以共享对象或动态链接库的方式直接被应用程序调用。TSP 接口(即 TSPI)对外提供了 TPM 的所有功能和它自身的一些功能,比如密钥存储和弹出关于授权数据的对话框。

TCS 层有以下几个任务:管理 TPM 的资源,比如,授权会话和密钥上下文的交换;提供一个 TPM 命令数据块产生器,这个命令数据块产生器能够把 TCS API 请求转换成 TPM 能够识别的比特流;提供一个全局的密钥存储设备;同步来自 TSP 层的应用程序访问。如果操作系统支持,TCS 层必须以系统服务方式实现,也应该是 TDDL 的唯一使用者。

TDDL 是一个提供与 TPM 设备驱动进行交互的 API 的库。一般来说,TPM 生产厂商会随 TPM 驱动一起附带 TDDL 库,以便 TSS 实现者能够和 TPM 进行交互。TDDL 提供一个小的 API 集合来打开和关闭设备驱动,发送和接收数据块,查询设备驱动的属性和取消已经提交的 TPM 命令。

以上三个层次的关系如下:最上层的 TSP 向用户的应用程序提供接口,它把来自应用程序的参数打包传给 TCS 模块,由 TCS 模块提供具体的功能函数(如密钥管理);TCS 模块对来自 TSP 模块的参数进行分析和操作后写成一个 TPM 可以识别的字节流,通过 TDDL 传到 TPM 里,TPM 接收到字节流后进行相应的操作,把结果以字节流的形式通过 TDDL 返回到 TCS,TCS 对字节流进行分析后把结果传给 TSP,最后由 TSP 把正式的

结果返回给应用程序。

TSS 的主要设计目标如下。

(1) 提供应用程序到 TPM 的正确入口点；

(2) 提供应用程序到 TPM 的层次化访问模式；

(3) 提供对 TPM 的同步访问；

(4) 通过合适的字节分类和排列，隐藏指令流的构建；

(5) 正确管理 TPM 资源。

实体可以在 TSS 和 TPM 上执行某些功能，通常拥有下面的一种角色。

(1) TPM 所有者(TPM Owner)：该角色是平台的拥有者，一个平台只有一个 TPM 所有者。

(2) TPM 用户(TPM User)：该角色可以装载并使用 TPM 对象，例如 TPM 的密钥。TPM 用户可以是能为一个对象提供授权数据的任意实体，第一个 TPM 使用者由 TPM 所有者创建。

(3) 平台管理员(Platform Administrator)：是控制平台中操作系统、文件系统或数据的实体，有可能是 TPM 所有者。

(4) 平台用户(Platform User)：是在平台管理员允许下使用平台数据或资源的实体，有可能是 TPM 用户。

(5) 操作者(Operator)：能直接对平台进行操作，有可能是 TPM 所有者或 TPM 用户，但不大可能是平台管理员或平台用户。

(6) 公共使用者(Public)：该角色可以执行平台中那些没有身份标识和认证约束的操作系统、文件系统或数据的任意功能，同样也能执行那些没有授权约束的 TPM 功能。

根据对象划分模块的思想，TSS 各层次的功能模块划分如图 4-2 所示。TSP 层的模

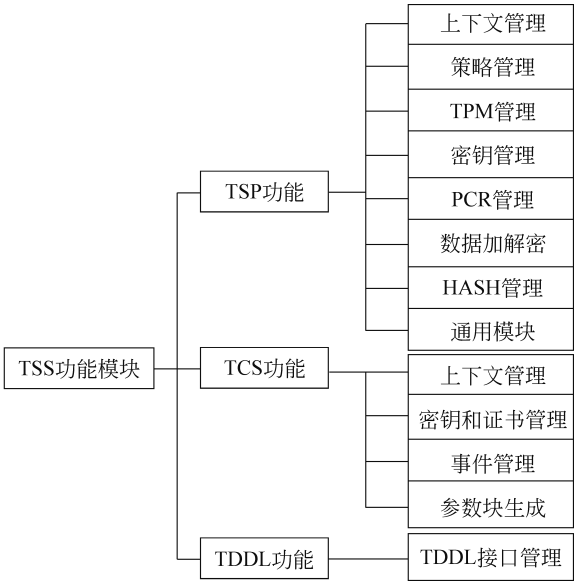

图 4-2　TCG TPM 1.2 TSS 功能模块

块包括：上下文管理、策略管理、TPM 管理、密钥管理、PCR 管理、数据加解密、HASH 管理和通用模块。TCS 层的模块包括：上下文管理、密钥证书管理、事件管理和参数块生成。TDDL 模块包括 TDDL 接口管理。

下面分别介绍 TSS 各层 TSP、TCS 和 TDDL 的功能。

1）TSP 层功能

TSP 层位于 TSS 的最上层，为应用程序提供访问 TPM 的服务。TSP 层和应用程序都位于用户层，并且在用户进程空间内，为应用程序与 TPM 之间的数据传输提供保护，可以为应用程序提供 TPM 安全平台的所有功能。

TSPI 采用了面向对象的思想，所有的 TSPI 函数通过对一个或多个对象句柄参数的处理实现类的特定实例。应用程序需要用到的工作对象（Working Object）可以分为非授权对象和授权对象。非授权对象包括 PCR 合成对象、Hash 对象、可迁移数据对象、代理簇对象和 DAA 对象；授权对象包括 TPM 对象、密钥对象、非易失性数据对象和加密数据对象。TSP 层中另外两类对象是上下文对象和策略对象。图 4-3 描述了对象的总体关系。

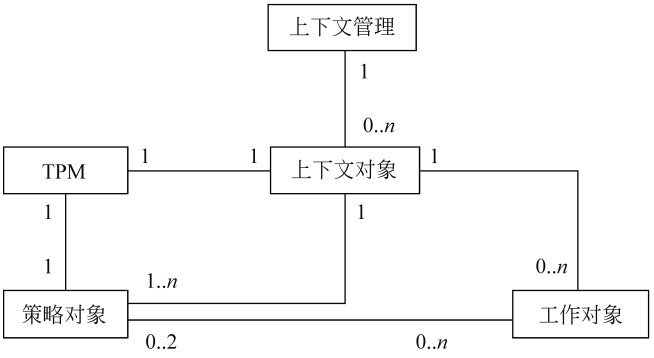

图 4-3　对象的总体关系

下面对图 4-3 中的对象关系进行说明。

上下文管理与上下文对象的关系：一个上下文管理对应 0 个或多个上下文对象。因此，上下文管理与上下文对象是 1：(0,1,…,n)的关系；

上下文对象与 TPM 的关系：如果 TPM 是工作对象，则上下文对象和 TPM 对象是 1∶1 的关系；

上下文对象与工作对象的关系：一个上下文对象可以对应多个工作对象。如果工作对象没有创建，则工作对象与上下文对象没有对应关系。因此，上下文对象与工作对象是 1：(0,2,…,n)的关系；

上下文对象与策略对象的关系：一个上下文对象产生时，其就对应了一个默认策略（与授权有关的信息），上下文对象还可以根据需要另外再创建策略对象。因此，上下文对象与策略对象是 1：(1,2,…,n)的关系；

策略对象与工作对象的关系：多个工作对象可能对应 1 个或 2 个策略，如果需要用到一个具体的策略，就需要再重新创建一个策略，因此，策略对象与工作对象是(0,1,…,

2)：(0,1,…,n)的关系。

图 4-4 描述了非授权工作对象的关系。

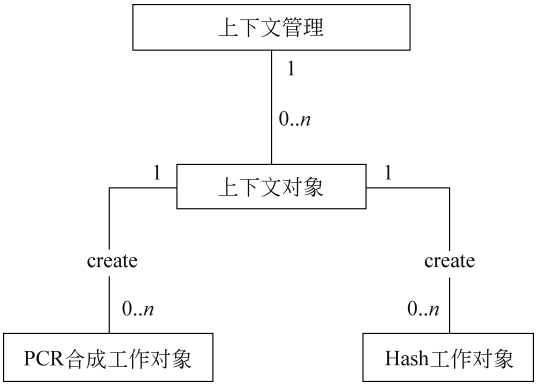

图 4-4　非授权工作对象的关系

下面对图 4-4 中的非授权工作对象关系进行说明。

上下文对象与 PCR 合成工作对象的关系：一个上下文对象可以创建多个 PCR 合成工作对象，也可以不创建 PCR 合成工作对象。因此，上下文对象与 PCR 合成工作对象是 1：(0,1,…,n)的关系；

上下文对象与 Hash 工作对象的关系：一个上下文对象可以创建多个 Hash 工作对象，也可以不创建 Hash 工作对象。因此，上下文对象与 Hash 工作对象是 1：(0,1,…,n)的关系；

因为 PCR 合成工作对象和 Hash 工作对象无须授权信息，与策略对象无关，故用上下文对象创建它们时，不用考虑它们与策略之间的联系。

图 4-5 描述了授权工作对象的关系。

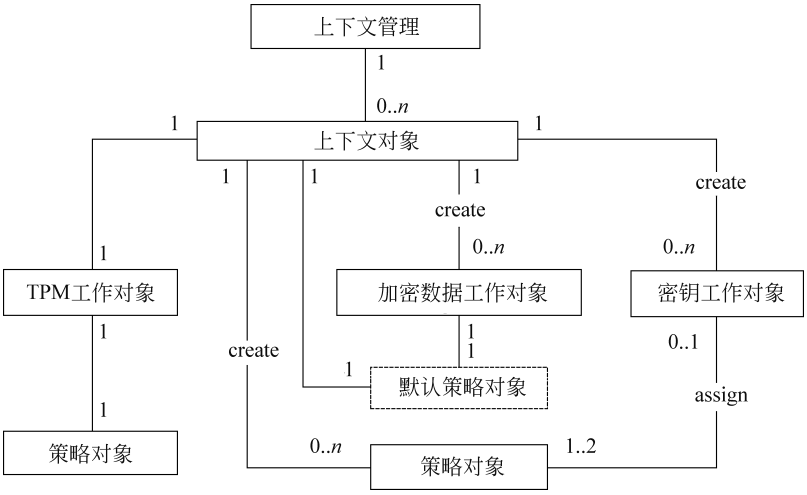

图 4-5　授权工作对象的关系

下面对图 4-5 中的授权工作对象的关系进行说明。

用户需要为其要使用的策略提供授权数据，通过 TSPI 提供的 Tspi_Policy_AssignToObject()方法某一策略可被分配给一些工作对象，例如 TPM 工作对象、加密数据工作对象和密钥工作对象。每个工作对象利用其获得的策略对象的内部功能处理授权的 TPM 命令。

上下文对象与 TPM 工作对象的关系：一个上下文对象创建一个 TPM 工作对象，此工作对象与 TPM 所有者的策略对象一一对应；

上下文对象与加密数据工作对象的关系：一个上下文对象对应 0 个或多个加密数据工作对象，因此，上下文对象与加密数据工作对象是 1：(0,1,…,n)的关系。一个上下文对象同时与一个默认策略相对应，加密数据工作对象直接与这个默认策略对象一一对应；

上下文对象与密钥工作对象的关系：一个上下文对象可以创建多个密钥工作对象，也可以不创建 PCR 合成工作对象。因此，上下文对象与密钥工作对象是 1：(0,1,…,n)的关系。每个密钥工作对象的策略对象中的授权信息不同，上下文对象须根据需要创建多个不同的策略与密钥工作对象相对应。

下面简要分析 TSP 各功能模块。

① 上下文管理

上下文管理是 TSS 中使用的一个特定概念，用于为 TSS 这个多模块多对象的耦合软件栈提供各模块间资源使用的动态管理，并为 TSP 层和 TCS 层之间的信息交互提供支持。因此，TSP 层的上下文管理的作用，一方面是通过对内部数据对象的管理来协调各功能模块间的资源使用，另一方面包含一些与对象执行环境相关的信息，如用户的身份、TSS 的应用环境信息，用于在和 TSS 其他层次(如 TCS)交互时使用。

② 策略管理

TSP 策略管理功能用于为不同应用程序配置相应的安全策略，以及为应用程序提供特定的授权秘密信息的处理操作(如回调和生命周期等)。策略对象与加密数据对象、密钥对象和 TPM 对象之间存在对应关系。

③ TPM 管理

TSP 的每个上下文环境中都包括一个 TPM 对象，并且该 TPM 对象自动与一个策略对象相关联，用于处理 TPM 所有者的授权数据。此外，TPM 管理模块还提供了一些基本控制和报告的功能。

④ 密钥管理

在 TSS 中，每个密钥对象代表密钥树中的一个具体的密钥节点。每个需授权的密钥对象，需要对应分配一个策略对象，用于管理授权的秘密信息。在 TCG 规范中一共定义了 7 种密钥类型，每种密钥类型都赋予一种特定的功能，所有密钥可以笼统地划分为存储密钥和签名密钥。它们可以进一步细化为：平台密钥、身份密钥、封装密钥、凭证密钥和遗赠密钥。对称密钥被单独划分为鉴定密钥。

⑤ 数据加解密

加密数据对象用于将外部(如用户、应用程序)产生的数据与系统关联起来(与平台或 PCR 绑定)，或者用于为外部提供数据加密/解密服务。

⑥ PCR 管理

PCR 操作用于建立系统平台的信任级别，并且通过 PCR 对象提供对 TPM 内部 PCR 的选择、读、写等简单操作。所有需要使用 PCR 信息的 API 函数，都需要使用某个 PCR 对象。

⑦ HASH 管理

HASH 管理用于为数字签名操作提供密码学安全功能，哈希值代表与特定的字符集合对应的唯一值。

除以上 7 类根据 API 操作对象划分的功能模块外，TSP 还提供一类特定的函数用于直接完成对象属性与关联关系的管理，在此简称为通用模块。

2）TCS 层功能

TCS 层为本地和远程使用者提供唯一资源入口，它对 TPM 的有限资源进行了抽象，可以提供更高程度的虚拟化。TCS 提供了一个公共的服务集合，可以为多个 TSP 使用。TCS 接口（TCSI）用来提供一种直接而简便的方法来请求和控制 TPM 服务。可把 TCS 当作一个软件 TPM，通过对 TPM 软件层进行抽象很大程度上弥补了 TPM 的以下缺陷：

① 一次只有一个操作可以执行；

② 只能通过一个驱动程序与其进行串行通信；

③ 本地软件与它的通信很受限制；

④ 可使用的资源有限；

⑤ 速度慢。

为了保证一些并发应用程序可同时使用本地 TPM 资源，将 TSS 核心服务提供给 TSP，这种服务还能提供给其他远程系统上的应用程序。通过软件验证和对特定状态下 TPM 功能的控制，TPM 远程证明能实现两个远程计算机之间的验证，从而保证不同系统上软件间通信的安全性。在这样的现实需求背景下，TCS 提供本地和远程服务。

我们使用的 TCS 接口定义在 TSS1.2 规范一起发布的 TCS.WSDL 文件中，用 Web 服务描述语言（WSDL）定义 TCS 通用接口，可以使 TSS 具有更好的平台兼容性，让更多的开发者使用 TSS。

定义 TCS 接口需要考虑如下目标。

① 对各平台的兼容性；

② 对各种工具良好的支持；

③ 工业界接受认可；

④ 基于 TCP/IP；

⑤ 向下兼容旧版本。

TCS 主要分为以下几个功能模块。

① TCS 上下文管理

TCS 上下文管理包括上下文抽象数据对象内存管理、权限管理等。TCS 上下文管理提供动态管理来有效使用服务提供者和 TCS 的资源，每个处理都为一组 TCG 的相关操作集提供上下文。服务提供者内的不同线程共享同一个上下文，每个服务提供者将获得一个独立的上下文。TCS 层的上下文管理也提供动态的句柄，它主要是管理内存和密钥