

摘 要

随着信息安全领域对内网安全日益重视,在主板上引入可信平台模块作为可信根的可信计算技术成为目前解决终端安全的研究热点。本文主要围绕可信计算技术展开研究,针对目前大部分电子政务中正在应用的计算机不可能近期全部更换嵌入可信平台模块 TPM 的这一现实,提出了一种在 USBKey 上实现 TPM 功能的外挂 TPM 远景研究目标。

为了实现这一外挂式 TPM,一要对 BIOS 进行分析改造,二要弄清 TPM 的实现细节,为此本文对可信计算基本理论进行了研究,研究了可信平台模块的主要构成和功能,分析了可信计算技术的主要安全机制。

在对可信计算基本理论详细研究的基础上,本文对可信度量根 BIOS 的安全问题进行了研究与分析,可信度量根 BIOS 的安全问题一直被忽略, BIOS 安全研究属于信息安全研究的新领域,在这方面目前尚未出现成熟的理论和产品,本文在对 BIOS 详细分析的基础上,设计实现了 BIOS 安全检查系统,该系统实现了 BIOS 安全隐患检查和安全修补功能,确保可信度量根的安全可靠,本文还针对最新推出的 64 位 EFI BIOS 的安全问题进行了分析,并提出了 EFI BIOS 安全增强方案,提高了可信度量根的安全性。

为了在 USBKey 中实现 TPM 功能的远景目标,必须搞清楚 TPM 的实现细节,为此,本文研究了 IBM 的开源软件 TPM_emulator 0.5 仿真包的结构和功能,对仿真包进行了修改完善,并编译调试通过,对 TPM 命令进行了测试和试验。本文还特别针对 TPM_emulator 0.5 仿真包的可信平台认证部分实现进行了详细分析,描述了可信平台认证相关的主要函数的数据结构和实现流程,通过对这些仿真程序进行适当修改就可以移植到 USBKey 中实现 TPM 认证功能。

本文通过对 BIOS 的研究与分析,实现了 BIOS 安全检查系统,提高了可信度量根 BIOS 的安全性,掌握了可信度量根 BIOS 的修改方法;通过对 IBM 的开源软件 TPM_emulator 0.5 仿真包分析与调试,弄清了 TPM 的实现细节。课题的研究基本达到了预期的可信计算关键技术研究与仿真的目标。

关键词 可信计算; BIOS; EFI BIOS; TPM; TPM 仿真

Abstract

With the growing recognition of inner network security in the field of information security, the trusted computing technology which introduces trusted platform module (TPM) on the mainboard as the root of trust has become the hotspot of endpoint security. The paper studies the trusted computing technology, and depicts a prospect of peripheral TPM by realizing main TPM security function with USBKey, since most of the computers used in the e-government can not be replaced by the computers with TPM recently.

The peripheral TPM is achieved on the basis of analysis and reform of BIOS as well as careful study on the details of implementation of TPM. Therefore, this paper studies the fundamental theories of the trusted computing, the main components and functions of TPM and analyzes the main security mechanism of trusted computing technology.

On the basis of studies on the fundamental theories of the trusted computing, this paper studies and analyzes the security problems on the root of trust for measurement BIOS. These problems have been ignored all the time. The studies on BIOS security belong to a new field of information security. And the advent of relative theories and products has rarely been found. On the basis of detailed analysis on BIOS, this paper introduces the design and implementation of BIOS security examination system, which realizes the examination of hidden trouble in BIOS, and thus ensures the security and reliability of the root of trust for measurement. This paper also analyzes the security problems of the 64-bit EFI BIOS released recently, puts forward an EFI BIOS security enhancement solution, which reinforces the security of the root of trust for measurement.

To achieve the prospect of realizing main TPM security function with USBKey, the implementation details of TPM must be made clear. For this purpose, the structure and function of IBM's open software TPM_emulator 0.5 are studied here, which has been modified, improved and tested the TPM commands after compilation and debugging. The authentication of trusted platform in TPM_emulator 0.5 has been especially analyzed in detail. The data structure and implementation process of the primary function related with trusted platform authentication are introduced and the TPM authentication function may be attained by transplantation of these emulation program to USBKey after some appropriate modifications.

With the study and analysis of BIOS, BIOS security examination system has been achieved and the security of root of trust for measurement BIOS improved. In addition, the means of modification of the root of trust for measurement BIOS has been attained and the implementation details of TPM made clear after the analysis and debugging of IBM's open source TPM emulator 0.5. The research of the topic has

basically attained the expected goal of study and emulation of the key technology in trusted computing.

Keywords Trusted computing; BIOS; EFI BIOS; TPM; TPM emulation

独创性声明

本人声明所提交的论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得北京工业大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

签名: 徐子平 日期: 2007.9.30

关于论文使用授权的说明

本人完全了解北京工业大学有关保留、使用学位论文的规定，即：学校有权保留送交论文的复印件，允许论文被查阅和借阅；学校可以公布论文的全部或部分内 容，可以采用影印、缩印或其他复制手段保存论文。

(保密的论文在解密后应遵守此规定)

签名: 徐子平 导师签名: 王学军 日期: 2007.9.30

第1章 绪论

1.1 课题来源

本课题来源为国家自然科学基金资助项目《可信计算体系结构研究》(60573003)与北京电子科技学院工程科研基金项目《计算机 BIOS 安全检查》。《可信计算体系结构研究》项目主要根据 TCG 标准研究可信计算的体系结构和可信机制以及可信计算技术应用问题。《计算机 BIOS 安全检查系统》项目研究 BIOS 安全漏洞,设计开发一套计算机 BIOS 安全检查系统对主流计算机 BIOS 产品进行安全隐患检查并进行修补,保证计算机底层硬件驱动的安全,从而增强可信度量根的安全性。

1.2 课题研究背景

1.2.1 可信计算发展背景

信息系统产生安全问题的主要原因是主机软、硬件结构存在设计漏洞,对合法的用户没有进行严格的认证和授权控制,导致资源被滥用,恶意程序利用系统弱点肆意进行破坏。当前安全防范的重点还是放在对服务器和网络的保护上,大多数都忽略网络终端接入者本身的安全,事实上大多数的攻击事件恰恰就是由终端不安全而引发的。如果能够从网络终端系统平台建立起安全的防护体系,把不安全因素从源头控制好,那么有望从根本上解决大多数的安全问题。

鉴于此,1999年由 IBM、Intel、AMD、HP 和微软等许多业界巨头发起组成的可信计算联盟(Trusted Computing Group, TCG)就是专注于从计算平台体系结构上增强系统安全性的。它的主要思路是在计算机的硬件平台上引入安全芯片架构,通过安全芯片和相应软件提供的安全特性来从根本上提高系统的安全性。

1.2.2 可信计算主要研究领域与研究现状

目前可信计算的标准发展迅速,TCG 继 2001 年发布了基于硬件系统的“可信平台规范 1.0”以来,于 2003 年推出了“TPM 主规范 V1.2”,后来针对不同的终端类型和平台相继发布了一系列完整的规范,例如可信客户端、可信服务器、可信手机、可信网络连接等规范,将可信平台的外延不断扩大发展。

而 IBM、HP、Inter 等厂商都相继发布了具有 TPM 功能的 PC 机,Intel 和 AMD 计划在下一代产品中引入可信计算技术。而微软的 Vista 操作系统已全面

实现了可信计算功能。

我国在可信计算的研究与开发领域与国际上基本是同步的，武汉瑞达公司 2004 年推出了国内首款自主研发的具有 TPM 功能的可信安全计算机。联想 2005 年推出了基于可信技术的安全芯片，兆日、浪潮、卫士通、天融信等公司也在可信计算方面进行了大量研究。

国家对可信计算技术的研究与开发也高度重视，2005 年国家信息安全标准委员会专门成立了可信计算工作小组，开展了可信计算密码标准的研究工作。国家“十一五”期间将可信计算列入国家发改委信息安全专项，“863 计划”也启动了可信计算专项，给予重大资金支持。中国科学院的中国信息安全国家重点实验室等科研院所也展开了可信计算理论和应用的研究。

现阶段的可信计算热潮是从可信 PC 平台开始的，但是它涉及的研究和应用领域却要广泛得多。可信计算的理论、关键技术和应用是研究的重点。

可信计算涉及到的关键技术有：可信计算的系统结构、TPM 的系统结构、可信计算中的密码技术、信任链技术、信任的度量、可信软件、可信网络。

可信计算的理论基础包括：可信计算模型、可信性的度量理论、信任链理论、可信软件理论。

武汉大学张焕国教授在一篇文章指出：目前在可信计算技术领域理论研究相对技术开发是滞后的，至今没有公认的可信计算理论模型。同时可信度量是可信计算的基础，但是目前尚缺少软件的动态可信性的度量理论与方法^[1]。

同时可信计算的应用需要开拓，可信计算的应用是可信计算发展的根本目的。目前可信 PC 机、TPM 芯片都已经得到实际应用，但应用的规模和覆盖范围都还不够，有待大力拓展，特别是目前大部分用户计算机上都没有 TPM 模块，这大大地阻碍了可信计算应用的拓展。

1.3 课题远景研究目标与目前的主要任务

目前的可信系统都是基于主板上的可信平台模块，核心密码模块与计算机主板系统合一，存在着管理复杂、更新困难、维护服务不便等问题。由于密码模块嵌入在计算机主板上，计算机主板系统要当作密码设备进行管理，使设备的使用与安全管理都很复杂；由于核心密码模块嵌入在计算机主板上，使密钥的更新、密码算法的更替无法进行；同样造成维护服务的不便，系统的维护服务受到很大限制。而且 TPM 是可信计算平台的信任根，中国的可信计算机必须采用中国的根芯片，中国的根芯片必须采用中国的密码；另外，目前的电子政务系统的建设，用户终端计算机都不具备可信模块，而对于大的用户系统统一更新用户终端计算机更是很难实现的事情。

那么如何在现有计算机基础上实现我国自主可信的安全终端呢？目前还没

有相关产品,有些内网安全产品只是利用 USBkey 完成认证和签名等功能,但没有实现可信计算功能的产品,这是目前电子政务发展的瓶颈。

USBkey 是目前广泛应用的密码安全模块,USBkey 本身具有密码运算和保密存储功能,内置 CPU 智能芯片具有运算功能;芯片中置有对称和非对称、哈希等密码算法;加密、签名在模块中进行、不在计算机终端内存中进行;USBKey 中存放数字证书及对应私钥,是目前国内外最好的证书介质;尤其是其私钥不出模块、模块中数据不能被复制、不在网络中传播这一特性与 TPM 私钥不出芯片是完全一致的。课题远景目标是利用 USBkey 自身的密码运算和保密存储安全特性,按照 TPM1.2 规范在 USBkey 中实现 TPM 的主要功能,实现一种基于 USBkey 的外挂可信平台模块。

这种外挂式 TPM 模块,不但解决了在现有计算机配置条件下无需改动主板即可应用可信计算技术的问题,而且解决了核心密码模块与计算机主板系统合一所带来的密钥管理复杂、密钥更新困难、密码设备维护服务不便等问题。并且,TPM 是可信计算平台的信任根,中国的可信计算机必须采用中国的根芯片,在基于 USBKey 的外挂可信平台模块中,可以嵌入我国的自主密码算法,真正做到可信根的可信可控。

为了在 USBKey 上实现 TPM 主要功能的这一远景目标,课题目前的主要任务是进行三个方面的研究与实践:一、可信计算主要安全机制的研究;二、可信度量根 BIOS 安全问题的研究;三、可信平台模块仿真系统分析。

1.4 论文结构

全文共分 5 章,其结构如下:

第 1 章,绪论。介绍课题来源与研究内容;

第 2 章,可信计算平台主要安全机制研究。从理论研究角度对可信平台模块的结构、可信平台模块中的可信链度量机制、保密存储机制、可信认证机制进行系统研究与分析;

第 3 章,可信度量根的安全性研究。研究分析了可信链的传递原理以及可信度量根 BIOS 安全的重要性,介绍了课题在 BIOS 安全分析与改造、可信根 BIOS 安全增强方面的理论研究与实践;

第 4 章,可信平台模块仿真分析。介绍了课题对 TPM 仿真系统的研究实践,包括仿真系统环境的搭建与仿真系统的结构分析与可信平台认证模块分析;

第 5 章,结束语。对全文进行总结,对下一步工作的考虑。

第2章 可信计算平台主要安全机制研究

一个可信计算机系统由可信根、可信硬件平台、可信操作系统、可信应用系统构成^[1]（如图 2-1 所示）。可信计算的基本思想是：首先构建一个信任根，再建立一条信任链，从信任根开始到硬件平台，到操作系统，再到应用，一级认证一级，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的可信。

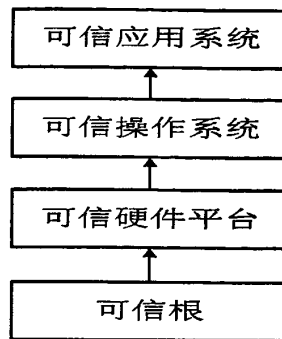


图 2-1 可信计算机系统构成

Figure 2-1 Architecture of Trusted Computer

TCG 的目标就是要在现有安全措施，如 X.509 数字证书、IPSec 系列协议、VPN、PKI、智能卡等的基础上再增补一个平台的信任根。TCG 标准规定，这个信任根就是可信计算技术的核心模块 TPM(Trusted Platform Module, 可信平台模块)。

2.1 可信计算平台的组成

可信计算平台中有两个重要部件，它们是可信平台模块(TPM)和可信软件协议栈(TSS)。

2.1.1 TPM 体系结构与功能

TPM 是可信计算技术的核心，是一个含有密码运算部件和存储部件的小型片上系统。在可信计算平台上执行大部分操作都需要授权，即使是 TPM 的所有者也不会例外，这样就提高了可信计算平台的可信度。TPM 的硬件结构组成如图 2-2 所示，由输入和输出、密码协处理器、散列消息认证码 HMAC 引擎等组件构成^[5]。

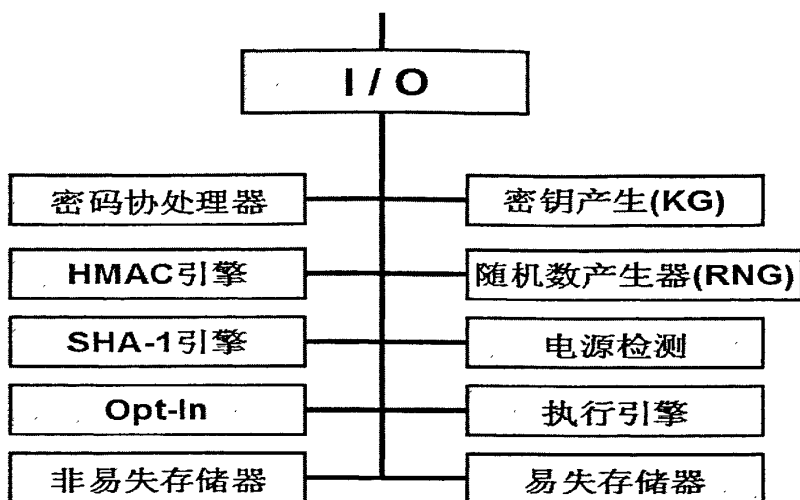


图 2-2 TPM 芯片硬件体系结构

Figure 2-2 TPM Component Architecture

TPM 的 I/O(输入/输出)部件, 负责管理通信总线, 执行内部总线和外部总线之间进行转换的通信协议, 完成协议的编码和译码, 将消息发送到相应的部件。

密码协处理器负责 RSA 运算的实现, 它内含一个执行运算的 RSA 引擎, 提供对内对外的数字签名功能, 内部存储和传输数据的加密解密功能, 以及密钥的产生、安全存储和使用等管理功能。TPM 使用公钥密码和对称密码, 实现加密、解密、签名和验证。

密钥生成器负责生成对称密码的密钥和非对称密码运算的密钥对, TPM 可以无限制地生成密钥。对于 RSA 算法而言, 它要完成大素数的测试, 密钥生成过程会使用到随机数发生器随机产生的数据。

HMAC 是基于 HASH 函数的消息认证码, 验证授权数据, 验证到达的请求在传输过程中未被修改。HMAC 引擎通过确认报文数据是否正确的方式为 TPM 提供信息, 它可以发现数据或者命令发生错误以及被篡改的情况。

随机数产生器 RNG 为 TPM 产生密码运算所需的随机数, 由噪声源、收集器、状态寄存器和混淆函数组成。

SHA1 引擎负责完成一种基本的 Hash 运算, 输出为 160 位二进制数。它用于系统资源的完整性校验, 数字签名的数字摘要。SHA1 已被王小云教授攻破, 但新的 Hash 函数还没有得到公认之前, 我们目前仍采用 SHA1。

选项控制 (Opt-in) 提供了对 TPM 功能开启与关闭的机制, 通过改变一些永久性的可变标志位设置 TPM 的功能选项, 但这种设置必须在 TPM 的所有者或者经所有者授权的情况下才能进行, 原则上不允许远程进行设置和改变。TCG 建议使用专门的硬件代表 TPM 的所有者。

执行引擎部件负责运行程序代码, 同时执行经过 I/O 传送给 TPM 的命令,

在执行命令之前应确信命令执行环境是隔离的和安全的。

非易失存储器用于存储固定标识, 持久信息和 TPM 状态。非易失存储器其中已经设置了一些存储内容, 如凭证密钥 EK (Endorsement Key) 等。EK 是 TPM 的主密钥, 平台出厂前, TPM 初始化时产生 EK, 在产生 EK 的同时产生 TPM 的证书。其余空间, 经 TPM 的所有者授权的实体可以使用。易失存储器是工作存储器, 速度快, 容量小; 存储一些不太重要的数据信息。

以上若干部件构成一个有机统一的安全执行环境, TPM 芯片首先验证当前底层固件的完整性, 如正确则完成正常的系统初始化, 然后由底层固件依次验证 BIOS 和 OS 的正确性, 如果验证正确则正常运行操作系统, 否则停止运行。之后利用 TPM 内置的密钥生成器生成的各种密钥对应用模块执行加解密, 向上提供安全通信接口, 以保证上层应用的安全。

2.1.2 TSS 系统结构与功能

TSS 是对可信计算平台提供支持的软件, 它的设计目标是对使用 TPM 功能的应用程序提供一个唯一入口: 提供对 TPM 的同步访问; 管理 TPM 的资源; 适当的时候释放 TPM 的资源等。TSS 有两部分组成, 分别是 TCS(TSS core service) 和 TSP(TSS service provider)。TCS 驻留在用户态, 通常以系统服务形式存在, 它通过 TDDL 和 TPM 进行通信。TCS 提供了几乎所有的基本功能和复杂功能, 像上下文管理、密钥管理、事件管理和审计管理等, 它为 TSP 提供接口。TSP 通过 TCS 来使用 TPM 的功能。TSP 也提供了丰富的面向对象的应用接口, 包括上下文管理、密钥管理和安全操作等。TSS 平台软件从结构上可以分为三层, 自下至上分别为 TDDL、TCS 和 TSP。各部分功能如图 2-3 所示^[4]:

TDDL (TPM 驱动程序库) 提供两个功能: 一是通过提供标准接口, 屏蔽各种不同安全芯片的差别。另一个功能是在用户模式和内核模式之间提供一个通信通道。

TCS(TSS 核心服务)是用户模式的系统进程, 通常以系统服务形式存在, 它通过 TDDL 与安全芯片进行通信。除提供安全芯片所具有的所有原始功能外, 还提供如密钥管理等功能。通过 TCS 的接口, 上层应用可以非常直接、简便地使用安全芯片提供的功能。

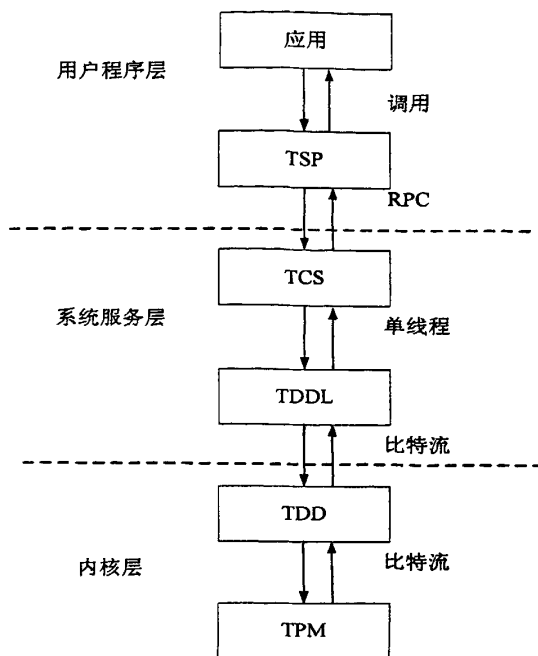


图 2-3 TSS 结构

Figure 2-3 TSS Achitecture Diagram

TSP(TSS 服务提供者)是用户模式的用户进程，位于 TSS 的最上层，它为应用程序提供了丰富的、面向对象的接口，使应用程序可以更加方便地利用安全芯片提供的功能构建所需要的安全特性。

可信计算技术的核心是基于 TPM 的安全芯片，由 TSS 配合 TPM 对可信计算平台提供支持，在它们共同作用下，可信计算平台提供基于硬件保护的安全存储和各种密码运算等功能。以 TPM 为基础，可信计算平台的可信机制主要通过三个方面来体现：

1)可信度量是获取影响平台完整性的平台特性序列的过程。任何将要获得控制权的实体，都需要先对该实体进行度量。

2)可信存储是可信度量和可信报告的一个中间步骤，所有度量值形成一个序列，然后在日志中保存，并在 PCR (Platform Configuration Register) 中保存这些序列的摘要。

3)可信报告是证明可信存储内容的过程。通过报告机制来完成对平台可信性的查询，如果平台的可信环境被破坏，询问者可以拒绝与该平台交互或向该平台提供服务。

2.2 可信平台的可信链度量机制

信任根是系统可信的基点。TCG 认为一个可信计算平台必须包含 3 个信任

根：可信度量根 RTM(root of trust for measurement)、可信存储根 RTS(root of trust for storage)和可信报告根 RTR(root of trust for reporting)。可信度量根是指计算机 BIOS 中的启动模块，可信存储根和可信报告根是可信计算核心模块可信平台模块 TPM (Trusted Platform Module)，信任根的可信性由物理安全和管理安全确保。信任链把信任关系从信任根扩展到整个计算机系统。在 TCG 的可信 PC 技术规范中，具体给出了可信平台的信任链度量机制，如图 2-3 所示。^[7]

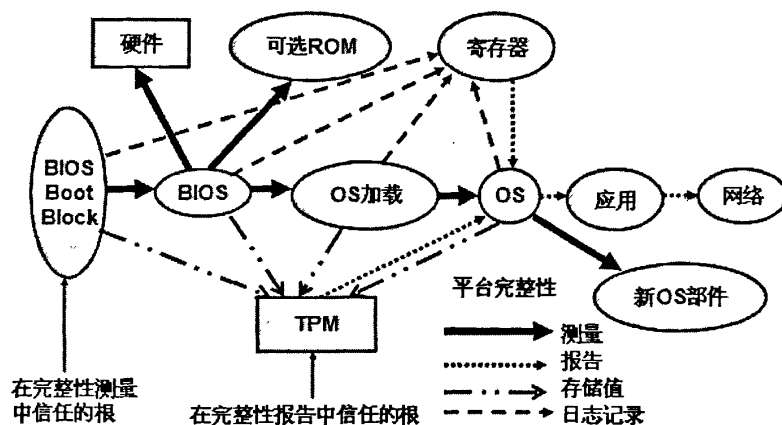


图 2-4 信任链的度量机制

Figure 2-4 Measurement Mechanism of Trusted Chain

从图中可以看出：这个信任链以 BIOS Boot Block 作为可信度量根，以 TPM 芯片为可信存储根和可信报告根，BIOS Boot Block 在可信链中又被称为可信度量根核 CRTM(core root of trust for measurement)。可信链首先检查 CRTM 的完整性后，将控制权交给 CRTM；CRTM 检查 BIOS 的完整性，计算正确后将控制权交给 BIOS；BIOS 通过 TPM 对系统组件、外围设备选项 ROM 进行度量和计算，与 OS Loader(操作系统加载程序)对比一致后将控制权传给 OS 加载程序；OS 加载程序对 OS（操作系统）及带各种应用的 OS 进行同样的操作，这种从 CRTM 到 BIOS，再从 OS Loader 到 OS 的度量步骤，我们称为信任链的传递，即从下往上一级一级认证。

在信任链的理论中，无论是最底层的 BIOS 启动模块还是到最上层的应用，在得到信任以运行之前，都需要经过度量，即一个度量或者认证的过程。数据完整性度量技术通过消息认证码 MAC (Message Authentication Code) 的一致性校验来实现度量的功能，这是现代主流的度量技术。对于 MAC 的产生，可以利用强的分组密码或者利用 Hash 函数来实现。TPM 中提供了满足单向性、抗碰撞性的安全 Hash 函数 SHA-1 引擎，可以为度量的实现提供保证。

同时 TPM 还提供安全存储以及内部的平台配置寄存器 PCR(Platform Configuration Register)，这样配合 TPM 的内部 TPM_Extend 命令就可以把信任链

传递下去。TPM_Extend 命令把当前的 PCR 值和新计算的值合并一起再调用 SHA1 运算,即使用 $SHA1(SHA1(SHA1(0+m1)+m2)+m_i)$ 的方式,计算出新的 PCR 值,更新上次计算出的 PCR。这样每个更新的 PCR 值都是在前面已经度量过的 PCR 值以及新一轮度量的新计算值的基础之上产生,保证了信任链是从源头传递上来的。

可信链度量与传递具体实现过程如下:

1) 对 BIOS 的度量

BIOS 启动模块经过 TCG 认证,确保 PC 被加电之后处于信任链的源头,之后对 BIOS 实行度量。在此之前应该对 BIOS 进行采样,与标准的 BIOS 样品库进行比较,并在 BIOS 漏洞库的基础上对 BIOS 进行安全性分析,根据最终生成的分析报告确定 BIOS 是否安全。对于安全的 BIOS 代码进行 SHA1 运算,得出的 MAC 作为此后完整性度量的依据。如果度量通过,PCR 的状态更新为 BIOS 度量的记录。

2) 对 OS Loader 的度量

BIOS 运行完成硬件自检之后,在启动系统之前,首先对 OS Loader 进行完整性度量。真正的 OS Loader 是主引导记录(MBR)中的主引导程序,所以对 OS Loader 的度量是通过 MBR 进行 HASH 函数校验保护来实现的,其 HASH 值还可以存储在主引导记录的备用存储区内。

3) OS loader 对操作系统进行度量

Windows 分区中的第一个扇区为引导扇区 BOOT,引导扇区由磁盘参数表和引导记录(Boot Record)构成,系统的度量需要考虑引导记录、程序文件和存档性数据文件等方面的度量。但是 windows 是不开放源代码的系统,对其进行度量的难以实现。Linux 是开放源代码的操作系统,采用 Linux 或自主知识产权的操作系统才能实现操作系统的可信。

4) 操作系统对应用程序的度量

软件数据完整性是信息安全的重要部分。但是,软件数据完整性还不能保证动态的安全性。所以上层应用程序的度量除了完整性检测度量之外,还需要借鉴其他一些度量方法。

随着现代软件的功能越来越完善,软件代码的规模也越来越庞大,需要度量的数据量也超乎寻常,需要对软件程序有所选取地进行度量过程。通过层次分析法,尽可能多地列举多种可能的度量方面,并计算出每个方面的权重值,以确定最重要的度量点。

其次对应用程序进行可信等级划分,对于基本不可信以及可信程度低的软件,则不能通过度量。另外因为软件的模糊性,需要引入专家评估系统,对应用程序的等级进行认定。只有通过完整性检验并且等级被认定为可信的软件才能得到操作系统的授权运行。

2.3 可信平台的保密存储机制

2.3.1 密钥类型

TCG 定义了七种密钥，每种类型都有相应的功能限制。TCG 密钥可以从广义上分为签名密钥和存储密钥，可以进一步分为身份密钥、授权密钥、绑定密钥、移植密钥和遗赠密钥。^[25]

TPM 的密钥和 TPM 的保护存储功能相关，TPM 的 RTS 保护存放在 TPM 里的密钥和数据。TPM 只在易失存储器中管理一小部分当前正在使用的密钥，而不活跃的密钥是保存在 TPM 外面的，只有需要使用的時候才调用 TPM 内部命令 TPM_LoadKey()加载到 TPM。

签名密钥 (Signing key) 一般是用来对数据和消息进行签名的非对称密钥；存储密钥 (Storage key) 是用于加密其他密钥和数据；身份密钥 (Identity Key) 专门用来对 TPM 产生的数据进行签名 (例如对 TPM 记录的 PCR 值进行签名)；凭证密钥 (Endorsement Key) 用来解密 AIK 证书和解密建立平台所有者时的所有者授权数据，是一个非可移植的平台解密密钥，不能用来加密普通数据和签名；绑定密钥 (Bind Key) 用来在一个平台上加密小块数据 (例如一对对称密钥)，而在另外一个平台上解密；遗赠密钥只是为了兼容老的应用程序而设置的，不推荐使用。授权密钥 (Authentication Key) 是用来保护传输会话的对称密钥。

在 TPM 中最重要的三个密钥是：凭证密钥 EK(Endorsement Key)，身份密钥 AIK (Attestation identity Key) 和存储根密钥 SRK (Stored Root Key)。

EK 是一个模长为 2048 比特的 RSA 公私钥对，私钥在 TPM 内部产生，永远不会暴露在 TPM 的外部。对一个 TPM 以及一个平台而言，EK 是唯一的，并且一直在 TPM 内部被保护，因此它是信任的基础。

AIK 用来向服务提供者提供平台的身份证明。EK 的主要功能是生成身份证明密钥 AIK 和建立 TPM Owner，由 TPM 的 Owner 来生成存储根密钥 SRK(Stored Root Key)，使用 SRK 来加密、存储其它密钥。

SRK 在建立 TPM 的拥有者时产生，该密钥的私钥保存在 TPM 的保护区域中，需要提供授权数据才可以使用。SRK 的私钥不能被导出，后续生成的密钥都直接或间接的由该密钥加密，它是 TPM 密钥存储体系的根。当清除 TPM 的拥有者时，SRK 也同时被清除，所有被 SRK 直接或间接加密的数据都将无法使用，包括密钥和数据。

2.3.2 树型密钥保护层次

基于 TPM 的安全芯片还有一个重要功能是密钥或关键安全参数的安全存

储, 有效地克服了传统安全解决方案的致命缺陷, 即存储在硬盘的密钥数据易被盗取或破坏。TPM 是采用硬件保护存储通过专门的硬件存储块来存储用户的秘密信息, 如文件加密密钥、鉴别密钥等。硬件保护使得通过加密的秘密信息只能在拥有相应密钥的专有存储块中才能被解密。

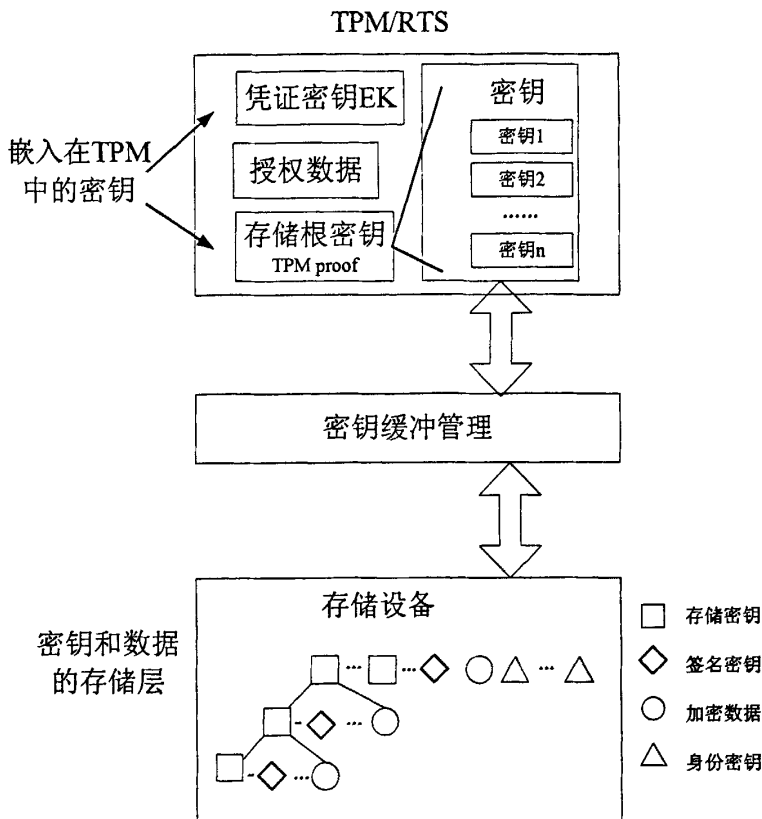


图 2-5 可信存储根层次结构

Figure 2-5 Root of Trust for Storage (RTS) Architecture

TPM 中以树型结构来保护密钥数据和关系, 由父密钥全程负责其子密钥的生命周期管理, 包括存储加密保护和使用授权, 实现了一个用于保护所有密钥的树型层次(如图 2-5 所示), 其根是 SRK(storage Root Key), 每层中的密钥都被其上一层的密钥加密^[8]。在每个可信平台有一个 SRK 用来保护其它所有的密钥, 它处于密钥保护树形结构的顶层。SRK 是这个树型结构的根, 是 TPM 直接保护的唯一密钥。

2.4 可信平台认证机制

可信计算平台中很重要的一个概念就是 TPM 的身份, 包括平台的身份、平台拥有者的身份和平台使用者的身份。不同的身份有不同的权限, 而且这些身份

之间也有一定关系。

在 TPM 规范中 TCG 定义了 5 种类型的证书, 每种类型只提供必需的信息来完成一个明确的操作。这 5 种类型证书包括: 凭证证书、平台身份证书、一致性证书、平台证书、确认证书^[8]。

凭证证书中存放了 EK 的公钥部分, 其用途是通过保护 EK 来提供平台真实性的一种证明; 平台证书由平台的生产商提供, 用来证明平台的安全是真实的; 一致性证书由平台的生产商或评估实验室提供, 它通过一个授权的一方为平台的安全特性提供证明; 确认证书由一个确认实体发布, 它用来证明平台的组件是可信的^[4]。

AIK 是一个签名密钥, TPM 使用 AIK 来证明自己的身份, 凡是经过 AIK 签名的实体, 都表明已经经过了 TPM 的处理。AIK 的生成虽然使用了 EK, 但是生成的 AIK 中却不包含任何有关平台或 EK 的隐私信息。这就使得 AIK 可以证明 TPM 的身份但不会泄露任何隐私信息, 提高了系统的安全性。因此 AIK 证书的产生过程是平台认证的关键。在第 4 章可信平台仿真分析中, 将对 AIK 证书的生成过程结合仿真程序进行详细的分析。

2.5 本章小结

本章主要阐述可信计算的基本理论, 包括可信平台的基本构成、可信平台模块 TPM 和可信软件栈 TSS 的结构与功能, 以及可信平台的主要可信机制——可信度量机制、可信存储机制和可信认证机制的研究与分析, 本章内容是论文后续研究与实践的基础。在研究可信计算平台可信机制的过程中, 本人发现可信计算技术与密钥恢复的矛盾, 因此本人提出了在可信计算中引入密钥托管技术的可信应用方案, 并在 2006 年中国信息保密专业委员会论文集第十六卷上发表题为《可信计算中的密钥托管问题研究》的论文。

第3章 可信度量根 BIOS 的安全性研究

3.1 BIOS 安全研究的意义

BIOS (Basic Input/Output System) 是计算机系统的一个重要组成部分,它是固化在计算机主板上 ROM 芯片里的软件系统。一般而言,传统信息安全威胁较多集中在软件系统上,而忽略了计算机 BIOS 固件的安全风险。随着 BIOS 芯片容积的扩大和 BIOS 功能的扩展, BIOS 在信息安全方面得到越来越多的重视,其存在的安全隐患也逐渐增多。BIOS 中提供的安全功能或其存在的安全隐患,相对于存在于硬盘上运行于操作系统中的其它安全软件、病毒和安全漏洞来讲,具有不易清除、不易监控和隐蔽性强的特点。

在可信平台系统中, BIOS 是信任链的度量根,是可信度量的源头,因此 BIOS 的安全可控对可信平台的应用是至关重要的。本文通过对 BIOS 模块结构的分析和 BIOS 标准代码样本的提取,实现了 BIOS 完整性度量,提高了可信计算度量根的安全性。

3.2 BIOS 安全检查原理

系统对 BIOS 进行安全检查的原理是基于 BIOS 安全隐患的特征码匹配扫描和基于 BIOS 标准代码的完整性度量,即通过与 BIOS 安全隐患库特征码匹配,发现 BIOS 的安全隐患;通过与 BIOS 标准代码样本库的代码模块的消息摘要相比较,判定代码的完整性。图 3-1 是系统的 BIOS 安全检查原理示意图。

安全检查系统的 BIOS 文件解析模块首先对 BIOS 影像文件进行解析,得到 BIOS 基本信息和 BIOS 结构信息, BIOS 结构信息包括组成 BIOS 的各个模块的信息和模块文件。安全检查系统的 BIOS 安全隐患扫描模块对于 BIOS 安全隐患库中的每一个安全隐患,取得其特征码,在 BIOS 所有模块中进行特征码扫描匹配,确定 BIOS 存在的安全隐患;安全分析引擎的 BIOS 完整性度量模块识别 BIOS 的可执行代码模块,并生成该代码模块的消息摘要,从标准代码样本库中查找该模块对应的标准模块并对二者的消息摘要进行比较,确定模块代码的完整性。

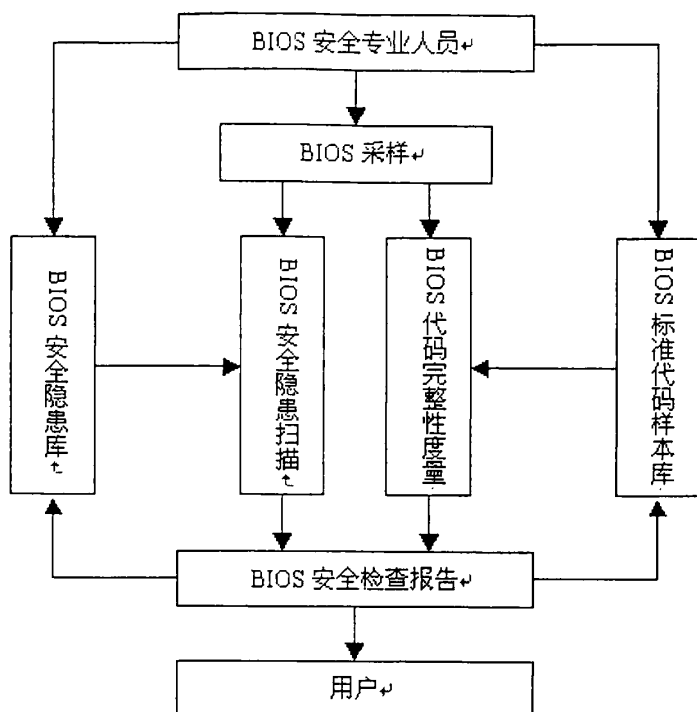


图 3-1 BIOS 安全检查原理示意图

Figure 3-1 Principle of BIOS Secure Examination

3.3 BIOS 安全检查系统组成

计算机 BIOS 安全检查系统采用 C/S 架构，由 4 个子系统构成：采样子系统，存储服务子系统，安全分析子系统，安全管理子系统（如图 3-2 所示）。

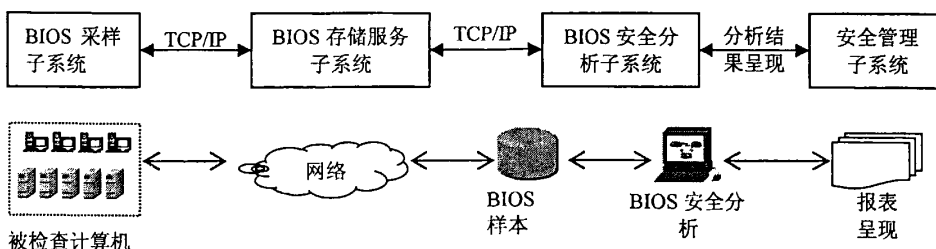


图 3-2 计算机 BIOS 安全检查系统架构图

Figure 3-2 Computer's BIOS Secure Examination System Framework

采样子系统在被采样计算机上运行，可以在 Win98/Win2000/WinXP 系统下运行。获取被采样计算机的 BIOS 影像文件后，通过 TCP/IP 网络上载样本到存

储服务子系统。

存储服务子系统通过 TCP/IP 网络接收所有被采样计算机的 BIOS 样本，对所有样本集中存储。

安全分析子系统被包装成 DLL 库，由管理子系统负责调用；管理子系统为安全分析子系统指定 BIOS 样本，由安全分析子系统实施安全分析并生成分析报告。

安全管理子系统负责为安全分析任务的管理、查询操作提供用户交互界面。安全管理子系统通过 TCP/IP 网络从存储服务子系统获取将要被分析的 BIOS 样本，并将该样本提交给安全分析子系统进行分析，然后将分析结果呈现给用户，并打印分析报告。

3.4 采样子系统的设计与实现

采样子系统在被采样计算机上运行，获取被采样计算机的基本信息和 BIOS 影像文件，通过 TCP/IP 网络将样本上传到存储服务子系统。采样子系统使用 C++ 开发，可以在 in98/Win2000/WinXP 系统下运行。采样子系统的关键技术包括 BIOS 基本信息收集技术和 BIOS 采样技术。

对于符合 SMBIOS (System Management BIOS, BIOS 与操作系统的接口规范) 的计算机，可以通过访问 SMBIOS 的结构获得系统信息，共有两种办法可以访问：一是通过即插即用功能接口访问 SMBIOS 结构，这个在 SMBIOS2.0 标准里定义了，从 SMBIOS 2.1 开始这个访问方法不再被推荐使用；二是基于表结构的方法，表内容是 Table Entry Point 的数据，这个访问方法从 SMBIOS 2.1 以后开始被使用，鉴于市场上计算机已经均支持 SMBIOS2.3 标准，所以采用基于表结构的访问方式。

为了得到 BIOS 的参数，首先要访问 SMBIOS EPS 表，从物理内存 0xF0000-0xFFFFF 间寻找关键字“_SM_”；找到后再向后 16 个字节，看后面 5 个 BYTE 是否是关键字“_DMI_”，如果是，EPS 表即找到。

从 EPS 表 18h 处可以得到 SMBIOS 数据表的真实内存位置，采样程序需要包含有 BIOS 基本信息的 TYPE0 数据表中的部分信息。程序通过定位 EPS 表，遍历寻找 TYPE0 数据表，最终得到 BIOS 的基本信息。目前主要使用的是 TYPE0 表中 09h 的 BIOS ROM 大小值。

3.5 安全分析子系统的设计与实现

安全分析子系统对 BIOS 影像文件进行分解，然后根据 BIOS 安全隐患库的特征码在模块中进行特征码扫描和匹配，确定 BIOS 存在的安全隐患；最后生成

技术性检查报告供安全管理子系统使用。

BIOS 影像文件是由多个 BIOS 功能代码模块或数据模块按照一定的结构组合形成的。每一个功能代码模块或数据模块都按照固定头部结构封装。头部结构中的信息包括：头部特征字、模块类型、模块压缩长度、模块实际长度、模块压缩算法。系统对 BIOS 影像文件的解析过程，就是在 BIOS 影像文件中顺序查找这些头部特征字，然后根据头部结构中的模块长度、压缩算法等信息，将模块内容读出并存储成独立的二进制文件。

除少数模块外，大部分 BIOS 模块都被压缩存储。要对 BIOS 进行安全隐患扫描，必须把这些压缩存储的模块解压缩后才能进行隐患特征码的扫描匹配。BIOS 模块压缩一般采用公认的压缩算法，主要是 LZSS 和 LZINT 算法。

BIOS 安全隐患库中存储已发现的所有的 BIOS 安全隐患。系统将所有的 BIOS 安全隐患归结为 4 种类型，即：BIOS 功能障碍隐患、BIOS 配置漏洞隐患、BIOS 物理攻击隐患、BIOS 木马隐患。

BIOS 代码的完整性度量，就是要检查 BIOS 模块中的代码是否被修改而改变或增加了模块的功能。系统中 BIOS 代码的完整性度量，通过比较被检查代码模块和对应标准代码模块的消息摘要来实现的。

BIOS 安全分析引擎是系统的核心执行代码。图 3-3 是 BIOS 安全分析引擎工作机理示意图。

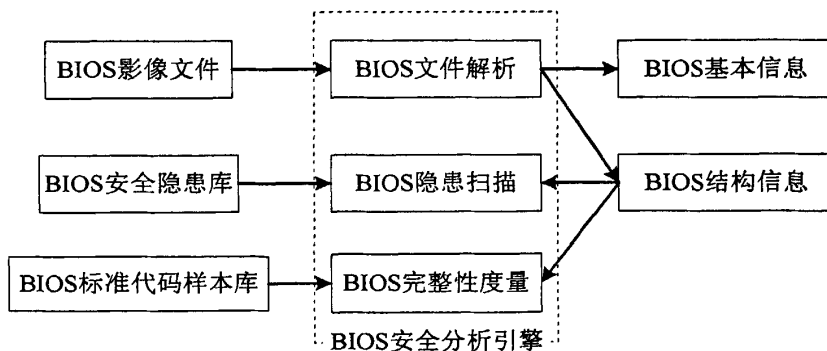


图 3-3 BIOS 安全分析引擎工作机理示意图

Figure 3-3 Principle of BIOS Secure Analysis Engine

安全分析引擎的执行步骤为：

步骤一 读取 BIOS 影像文件，判定 BIOS 类型。

步骤二 根据 BIOS 类型，按照模块头部特征字和结构，从 BIOS 影像文件中解析并解压缩各个组成模块并存储成独立的二进制文件。

步骤三 对安全隐患库中的每一个安全隐患记录，在生成的所有模块中查找匹配隐患特征码，确定 BIOS 存在的安全隐患。

步骤四 对生成的可执行代码模块，计算模块的消息摘要，并与标准代码样本库中的对应模块的消息摘要进行比较，度量 BIOS 代码模块的完整性。

步骤五 根据步骤三和步骤四的结果,生成 BIOS 安全风险检查技术报告。

3.6 系统运行测试结果分析

3.6.1 BIOS 样本管理

BIOS 安全检查程序启动后,通信和存储模块负责与存储服务子系统连接,使用 FTP 协议获取服务器上存储的所有 BIOS 检查样本,并按照未被检查和已被检查分类列表显示,并为用户提供对存储服务子系统上所有样本的浏览检索服务。

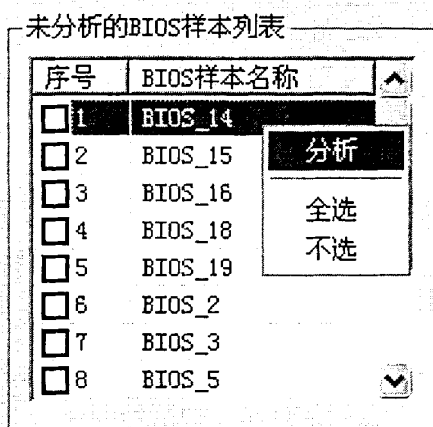


图 3-4 未分析的 BIOS 样本列表

Figure 3-4 List of unanalyzed BIOS Samples

对于未分析的 BIOS 样本,用户可以选择对该样本进行分析。

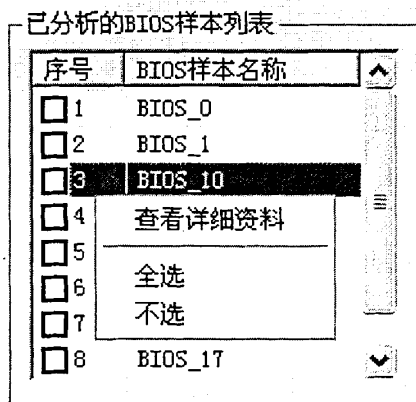


图 3-5 已分析 BIOS 样本列表

Figure 3-5 List of analyzed BIOS Samples

对于已被分析的 BIOS 样本可以查看该 BIOS 的详细分析数据。

3.6.2 BIOS 安全检查结果

BIOS 安全检查结果显示内容包括：BIOS 基本信息、BIOS 结构信息、BIOS 安全隐患检查结果。

BIOS 基本信息记载被检查 BIOS 的基本信息，包括以下信息项：BIOS 厂商、BIOS 文件长度、BIOS 序列号、BIOS 发布时间、BIOS 版本号、BIOS 版权信息。

BIOS 结构信息记载被检查 BIOS 的模块解析结果，列出所有解析出的模块信息。每个模块包括以下信息项：模块类型代码、模块类型名称、模块存储文件、模块实际长度、模块压缩长度、模块压缩比率、模块功能描述。

图 3-6 显示对 BIOS 进行采样检查后对其基本信息和结构信息的分析结果。

图 3-7 显示 BIOS 安全隐患，记载被检查 BIOS 存在的安全隐患。每个被报告的安全隐患包括安全隐患名称、安全隐患类型、隐患对应模块、建议解决方案、安全隐患描述。

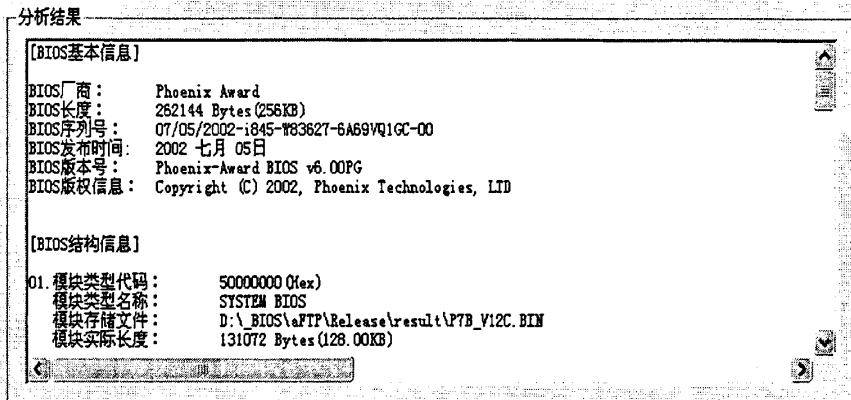


图 3-6 BIOS 基本信息和结构信息分析结果

Figure 3-6 Output of Analyzed BIOS Basic Information and Structure

序号	安全隐患名称	安全隐患类型	隐患对应模块	建议解决方案	安全隐患描述
1	BIOS网卡远程开机安全隐患	BIOS固有安全隐患		为BIOS打补丁	计算机主板BIOS提供了通过网卡远程开机功能。在计算机关闭电源的
2	BIOS定时开机隐患	BIOS固有安全隐患		为BIOS打补丁	计算机主板BIOS提供了定时开机功能。如该计算机设置了预定开机时

图 3-7 BIOS 安全隐患检查结果

Figure 3-7 Output of Scanning BIOS Secure Hidden Trouble

3.7 BIOS 安全隐患库和标准代码样本库管理

管理子系统提供在线连接独立的服务器, 从该服务器上为系统更新 BIOS 安全隐患库和标准代码样本库的功能。

管理子系统提供独立的程序, 使 BIOS 安全研究专业人员能够对 BIOS 安全隐患库和标准代码样本库进行增加、删除、编辑等管理功能, 并及时存储到独立服务器上, 供使用计算机 BIOS 安全检查系统的用户下载和更新。

计算机 BIOS 安全检查系统的更新, 主要是作为系统基础数据的 BIOS 安全隐患库和 BIOS 标准代码样本库的更新。BIOS 安全隐患库和 BIOS 标准代码样本库的数据更新, 需要 BIOS 安全专业人员的长期的数据跟踪收集、分析和研究。另外, 针对 BIOS 安全检查结果的反馈进行进一步的深入分析, 也能够补充和完善 BIOS 安全隐患库和 BIOS 标准代码样本库; 反过来, 两个库的补充和完善又进一步促进了 BIOS 安全检查结果的覆盖范围和准确性。

课题针对新出现的 BIOS 安全问题, 将不断补充和完善 BIOS 安全隐患库和 BIOS 标准代码样本库, 对两个库的数据进行维护, 并通过网络为计算机 BIOS 安全检查系统的用户单位或个人提供两个库的网络在线更新和下载, 保证可信计算根的安全。

3.8 EFI BIOS 安全性增强研究与设计

BIOS 主要分为两种, 一种是用于老机器的 16 位实模式传统 BIOS, 另一种适用于新机器的全新的 EFI (Extensible Firmware Interface, 可扩展的固件接口) BIOS, 传统 BIOS 不仅具有很多安全漏洞, 而且其 16 位实模式运行方式已经严重阻碍新型高性能机器的功效发挥。全新的 EFI BIOS 技术不仅有效地解决了传统 BIOS 实模式问题, 同时拥有诸多自身优越特性。EFI BIOS 中的分层结构技术, 能够屏蔽计算机的硬件层信息, 使操作系统的开发和固件的开发相对独立; EFI BIOS 采用高级语言开发, 方便开发新的应用功能; EFI BIOS 独有的显卡驱动能力, 提供彩色显示方式, 界面美观且易于操作; EFI BIOS 提供对网络的支持, 其 TCP/IP 协议栈相对于传统 BIOS 无盘工作站, 提供更强大的远程交互、控制和诊断等功能。

EFI BIOS 技术在为个人计算机提供便利的同时, 却从底层为计算机平台带来不安全因素。根据可信平台规范, EFI BIOS 环境下存在两个问题急需解决: 1、系统资源访问, 即在 EFI BIOS 中通过调用 EFI BIOS Shell 程序, 硬盘中的资料信息完全暴露出来, 使用者无需进入操作系统就可以得到硬盘中的资料; 2、启动项管理, 即在 EFI BIOS 中可以随意添加更改启动项, 使用者通过向启动项中添加入侵性的 Agent 驱动程序, 即可实现从底层的信息盗取或是底层对系统

的监控。

本人提出利用 USBKey 提供的唯一标识符和高强度密码算法及独立于机器的安全运算环境, 增强 EFI BIOS 底层安全, 实现了开机时的强身份认证和安全控制管理, 增强了可信度量根的安全性。

3.9 本章小结

计算机 BIOS 作为可信度量根是可信的源头, BIOS 安全研究与实践具有重要意义。本章主要阐述本人在 BIOS 安全方面所做的研究与实践。BIOS 安全检查系统实现了对 BIOS 安全隐患检查和安全修补的功能, 本章主要介绍 BIOS 安全检查原理和 BIOS 安全检查系统主要模块的设计思想和实现的关键技术, 并且针对最新发布的 64 位 EFI BIOS 的安全问题, 提出了安全增强方案, 并在《信息安全与保密通信》2007 年 11 月发表名为《一种基于 USBKEY 的 EFI 安全增强设计方案》的论文。

第4章 可信平台模块仿真分析

为了在 USBkey 中实现 TPM 主要功能的远景目标,本人对 IBM 发布的 TPM 仿真软件包 TPM-simulator0.5 进行编译调试,分析 TPM-simulator0.5 软件包结构,并对其主要功能模块的数据结构体和程序流程进行分析,弄清 TPM 功能实现的细节,为将其移植到 USBKey 中实现 TPM 功能打下基础。

TPM-simulator0.5 是 IBM 发布的一个开源代码 TPM 仿真软件包,仿真包中的数据结构和 API 函数完全按照 TPM1.2 规范编写。TPM-simulator0.5 经过编译后可以加载到内核中运行,可以仿真实现大部分 TPM 的功能,是一个功能比较完善的 TPM 仿真包。

4.1 仿真包的编译调试

TPM_emulator-0.5 仿真包基于 Linux 操作系统环境并支持大多数 Linux 内核及版本,考虑到各种所需系统软件和工具包下载的便捷性和可操作性,这里使用的仿真环境为 Linux Debian 系列的 Ubuntu Dapper (6.06)版。

通过两个多月的分析调试,本人基本弄清了仿真包的编译调试方法,编译安装 TPM_emulator-0.5 需要以下几个主要步骤:

1. 建立 TPM_emulator-0.5 编译环境

为了能对系统进行升级和下载有用软件包,首先需要配置软件包下载源地址以便今后更好地维护和升级软件包。然后安装编译所需的 gcc 和 make 等编译工具包,下载并安装 fakeroot(伪装根用户)工具包以便使用后续 fakeroot 工具进行内核编译。

2. 编译操作系统内核

在 Ubuntu Dapper (6.06)操作系统上编译 tpm-emulator0.5,首先必须建立内核树(内核文件路径及节点)结构,课题采用支持 TPM_emulator-0.5 仿真包的内核压缩包“linux_kernel_v2.6.18_rc5.tar.tar”重新编译内核。

3. 安装密码运算库

在编译 tpm_emultor0.5 仿真模块之前,还需要确保已经在系统中安装了 GNU MP library,这个库中包含了仿真包进行密码运算时所需的数据。

4. 编译安装 TPM-simulator0.5

在上述环境准备好之后,将 tpm_emulator-0.5.tar.tar 仿真包解压到任意路径下,执行 make 文件进行编译安装。

上述四个步骤是在实践中不断摸索尝试的结果,在编译调试过程中需要注意以下两点:

(1) 安装好操作系统后,并不能直接对 TPM_emulator-0.5 仿真包进行安装编译。因为某些操作系统的内核不支持 TPM,没有为仿真包建立系统的内核树(即建立内核相应路径下文件夹和文件结构),只有采用支持 TPM 的内核压缩包对系统重新编译以后才能形成运行仿真包软件所需的系统内核树结构。之后才能安装编译 TPM_emulator-0.5 仿真包,否则安装编译 TPM_emulator-0.5 仿真包将无法找到所需系统文件,出现错误。

(2) 通过上层 tddl (编译运行可执行文件 test_tddl) 来给命令运行监视终端窗口程序传送命令及参数。在安装好 TPM_emulator-0.5 仿真包后,应首先打开一个终端窗口启动命令运行监视终端窗口后,同时再打开一个终端窗口执行已经编译好的可执行文件 test_tddl 来给命令运行监视终端窗口发送相应的 tpm 命令,命令执行结果可在命令运行监视终端窗口中查看。

4.2 仿真包的结构分析

TPM_emulator-0.5 压缩包中有六个文件目录和四个辅助文件,六个文件目录分别为 crypto, tddl, tpm, tpm_dev, tpm_dev 和 tpm_dev,四个辅助文件分别为 AUTHORS, README, Makefile 和 ChangeLog。如下图所示:

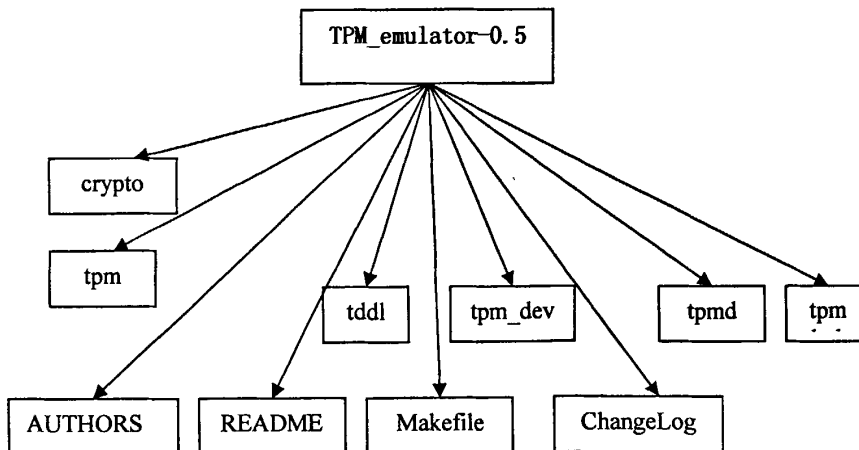


图 4-1 仿真包的目录组成

Figure 4-1 Structure of Directory in Emulator software

每个目录的功能和作用如下:

crypto

由于 TPM 必须支持最低限度的密码算法和操作,因此 crypto 中提供了 SHA1 Hash 函数, HMAC, rc4, RSA 密钥对的生成及签名。

tddl

为 TPM 提供了一个简略的 TCG Device Drive Library (TDDL, TCG 设备驱动库), 主要提供函数接口的功能。

`tpm`

包含了一系列实现 TPM 功能的函数。

`tpm_dev`

已经废弃的内核空间的 TPM emulator,

`tpmd`

一个用户态的后台程序, 用来执行 TPM 仿真, 通过 unix socket 来通信。

`tpmd_dev`

一个内核模块, 这是另一种 TPM 仿真的方式, 是通过设备驱动来实现的, 也就是通过硬件设备/dev/tpm 来进入 TPM, 并通过此设备来提供前向兼容性。它是运行在内核空间中, 所以不能使用标准库和系统库。

TPM_emulator-0.5 压缩包的目录之间的关系如图 4-2 所示, `tpmd` 是命令运行监视终端窗口, 通过它可以监视内核 TPM 命令的执行结果情况, `test_tddl` 是 TPM 命令输入终端窗口, 从该窗口可以输入要测试的命令, 两个窗口之间通过 socket 进行通信。

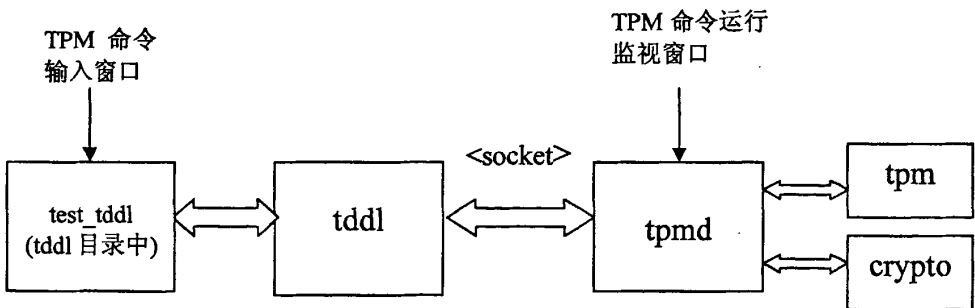


图 4-2 仿真包目录之间的关系

Figure 4-2 Relationship of Emulator's Directories

对 TPM 仿真子程序的说明主要集中在 `tddl`, `tpmd`, `tpm` 和 `crypto` 这几个目录中。通过对 `tddl`, `tpmd` 及整个仿真子程序的 Makefile 进行分析, 经过编译程序装载后:

整个仿真子程序的 Makefile 把以上六个目录中的所有 .c 和 .h 都进行编译, 产生相应的内核程序和用户程序。

`tddl` 中的 Makefile 编译 `tddl.c` 和 `tddl.h` 进入内核, 编译后, .c 文件装载在 `/usr/lib/` 目录下, .h 文件装载在 `/usr/include/` 目录下。而 `test_tddl.c` 文件编译为测试 `tddl` 的应用程序。

`tpmd` 中的 Makefile 编译 `tpm` 和 `crypto` 中的文件, 编译得到的 `tpmd` 存储在

/usr/sbin 目录下。

4.3 仿真包程序工作原理分析

4.3.1 Tpmdd 目录程序工作原理分析

tpmd 中包含如图 4-3 所示的三个文件：

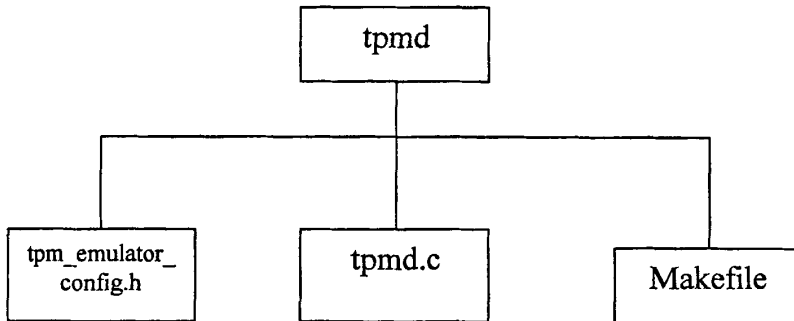


图 4-3 tpmdd 目录组成

Figure 4-3 Struct of Diretory in Tpmdd

tpmd.c 是用户界面的程序，它可以通过 unix socket 与 tddl 通信，由此接收要处理的命令，然后转入编译进来的 tpm 模块进行处理，得到的响应又通过 socket 传回 tddl。

通过对程序的分析，tpmd.c 的流程图如图 4-4 所示：

当在这个程序界面输入 `tpmd -f clear` 时候，程序开始启动。首先进入 `parse_options()` 函数，根据传进的参数 (-f) 设置 `opt_foreground` 的值 (-f 表示使程序在前台运行) 并且设置启动模式。然后打开随机设备 `/dev/urandom`，初始化信号句柄，根据变量 `opt_foreground` 进行判断后，进入命令的循环处理 (`main_loop()`)。 `main_loop()` 是该程序的关键之处，图 4-5 是对 `main_loop()` 函数流程的分析：

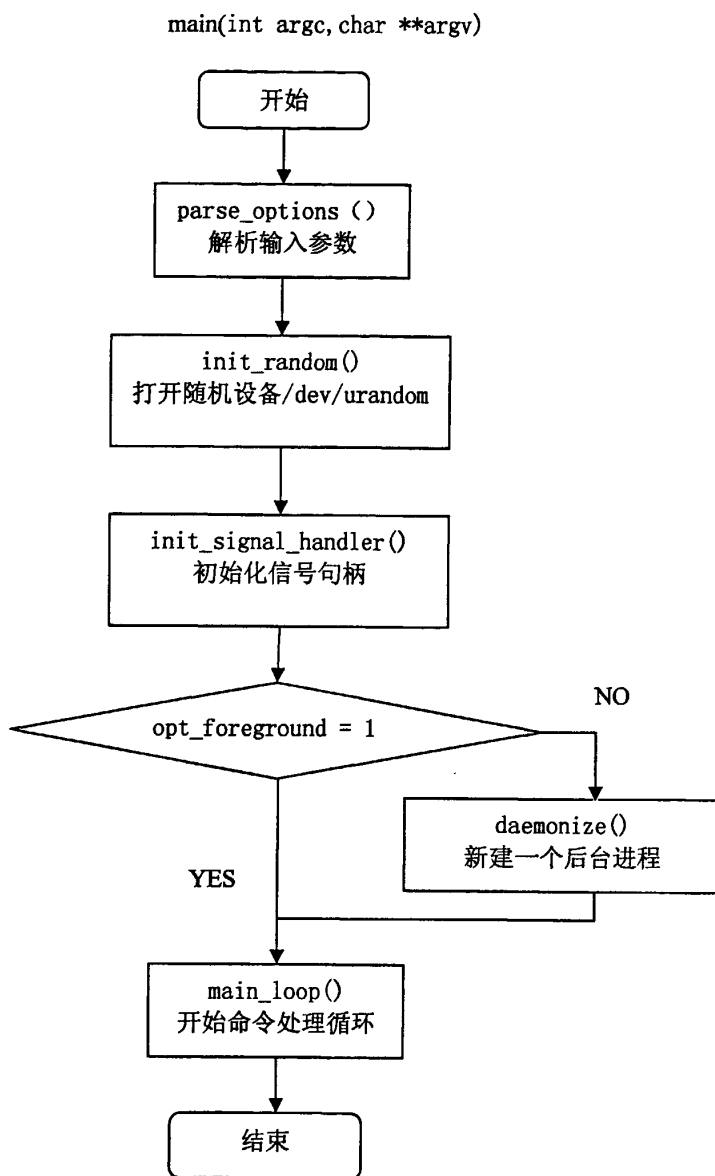


图 4-4 tpm.c 程序流程图

Figure 4-4 Flow Chart of tpm.c

`main_loop()`函数主要是实现了命令的传送功能。函数通过 `socket` 建立的连接, 使用 `read()`和 `write()`函数, 读出以及写入数据, 实现与 `tddl` 的通信。

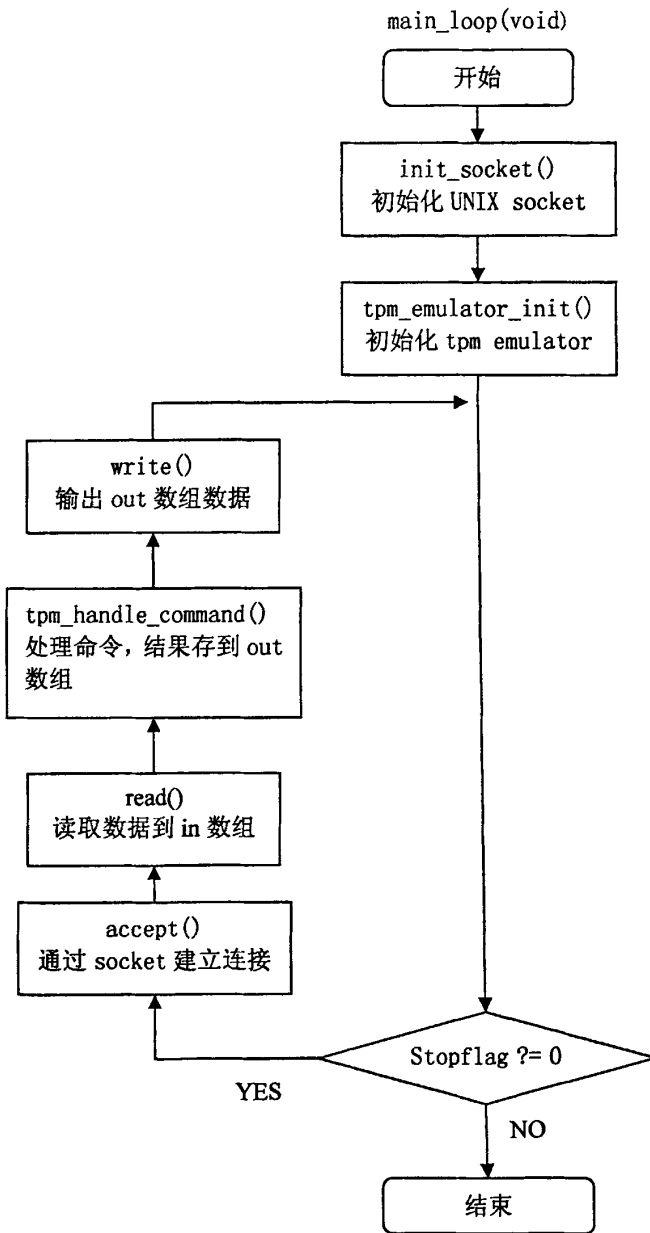


图 4-5 main_loop()函数流程图

Figure 4-5 Flow Chart of main_loop() Fuction

4.3.2 tddl 目录程序工作原理分析

tddl 目录中包含如图 4-6 所示的几个文件：

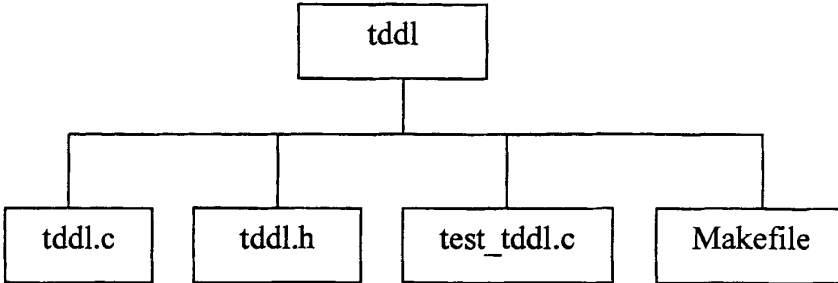


图 4-6 tddl 目录组成
Figure 4-6 Structure of Diretory in Tddl

其中 `test_tddl.c` 是用来测试设备驱动库（tddl）是否已经装载好，本人把它改造之后用于进行 TPM 命令的测试与试验。假设 `tpmd` 程序已经运行，则测试过程如下：

```
# make test_tddl.c  
# ./test_tddl
```

`test_tddl.c` 测试程序流程如图 4-7 所示：

程序在 `Tddli_Open()` 函数中执行了 `open_socket()` 和 `open_device()` 两函数，使程序与 `tpmd.c` 程序建立了连接。而在 `Tddli_TransmitData()` 函数中执行 `send_to_tpm()` 和 `receive_from_tpm()` 函数来发送和接收数据。

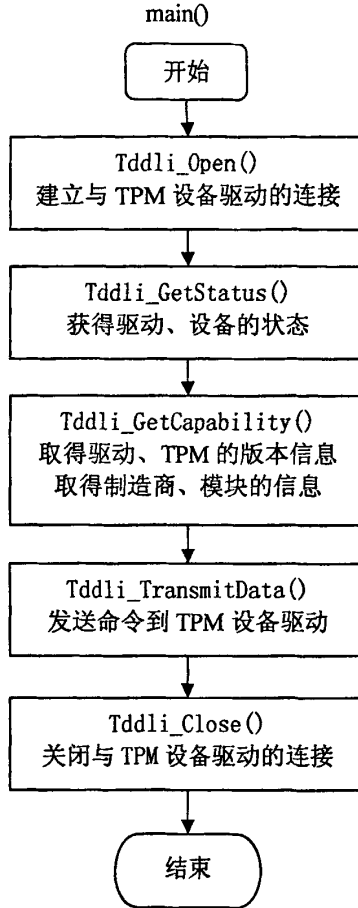


图 4-7 test_tddl 测试程序流程图

Figure 4-7 Flow Chart of test_tddl Test Program

4.3.3 tpm 目录程序工作原理分析

1. TPM 初始化

TPM 初始化包含 tpm_startup.c, tpm_testing.c, tpm_data.c。tpm_startup.c 包含 tpm_emulator.h, tpm_commands.h, tpm_data.h 和 tpm_handles.h, tpm_testing.c 包含 tpm_emulator.h, tpm_commands.h, tpm_data.h, crypto/sha1.h, crypto/hmac.h 和 crypto/rsa.h, tpm_data.c 包含 tpm_emulator.h, tpm_structures.h, tpm_marshallling.h。

TPM 初始化分三个步骤进行:

1) tpm_data.c 中的函数 tpm_init_data() 被执行, TPM 中的永久数据被初始化为缺省值;

2) 在执行 tpm_startup.c 中的函数 TPM_Init() 时, 首先完成一个内部自检, 即完成 tpm_testing.c 中的 TPM_SelfTestFull(), 它由以下四部分组成: a、检测随

机数发生器; b、用给定的测试值检测 SHA1 算法; c、用给定的测试值检测 HMAC 算法; d、通过产生新的密钥对和执行一些加解密运算来检测 RSA 引擎和凭证密钥。

如果上述的任一项自检失败了, TPM 仿真包就会进入到一个失败模式, 并且在调用命令时会返回一个错误信息。但有个例外, TPM_GetTestResult()用来获得确切的错误原因;

3) 如果自检通过, TPM 仿真包就运行在指定的模式, 并且其剩余的内部数据也会被初始化, 在特定的运行模式下, 可以通过 tpm_data.c 中的函数 tpm_restore_permanent_data()保存且恢复永久数据值。

在关闭 TPM 时, 首先 tpm_startup.c 中的函数 TPM_SaveState()会被调用来保存永久数据值, 其次调用 tpm_data.c 中的函数 tpm_release_data()来释放分配的资源, 比如内存和句柄。下图为 TPM 的启动及关闭的流程:

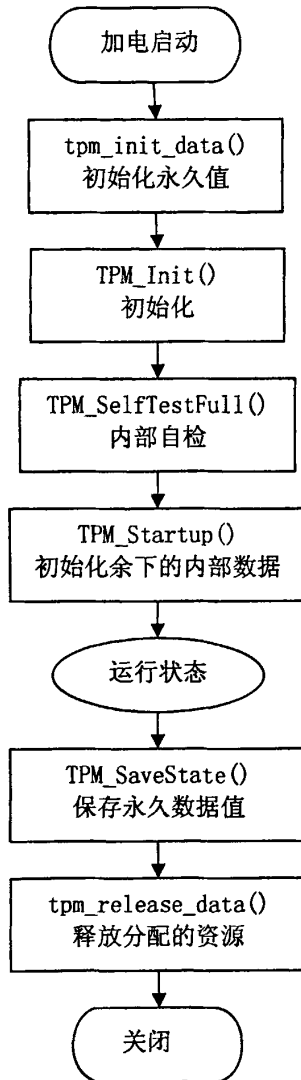


图 4-8 TPM 的启动及关闭的流程图

Figure 4-8 Flow Chart TPM Start and Close

2. TPM 命令处理

TPM 命令处理主要是 `tpm_cmd_handler.c`，`tpm_cmd_handler.c` 包含 `tpm_commands.h`，`tpm_data.h`，`crypto/sha1.h`，`crypto/hmac.h`，`tpm_marshallng.h`，`tpm_handles.h`。

TPM 命令的执行是从形成一个 TPM 命令编组（如字节数组）开始的。通过 socket 的传送，而后通过调用 `tpm_cmd_handler.c` 中的函数 `tpm_handle_command()` 把命令传送到 TPM 仿真子。

在仿真包中，命令的执行分为三个步骤：命令首先解码成三个部分，分别为请求包头，命令参数，认证尾（可选的）。在检验命令是否允许在当前 TPM 状

态执行后，计算出输入参数的摘要，并且根据特定的 TPM 命令进一步分组参数。

执行实际的命令并且设置返回的参数。在执行前，认证尾被检验来保证命令已经授权和输入参数的完整性。

计算响应授权认证并与返回参数和响应头组织在一起。最后，响应编组为它的字节表示方式并返回给调用者。

以下为命令执行过程的流程图：

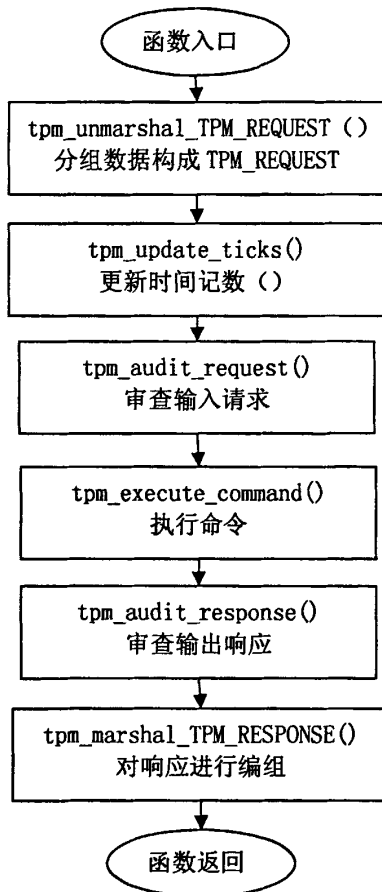


图 4-9 tpm_handle_command()函数流程图

Figure 4-9 Flow Chart of tpm_handle_command() Function

4.4 TPM 命令仿真运行测试

通过分析软件仿真包的结构，对自测程序进行了修改和完善，将自测程序修改为能够对 TPM 命令进行测试和试验，这里以 TPM 授权 OIAP 命令为例，具体测试步骤和测试结果为：

1. 打开命令运行监视终端窗口:

```
tpmd -f clear
```

此命令以 clear 模式启动 tpm，用户态 tpm 执行最初的若干启动命令操作直至 TPM_Startup(), 并等待其它命令的执行。

2. 打开 TPM 命令输入终端窗口:

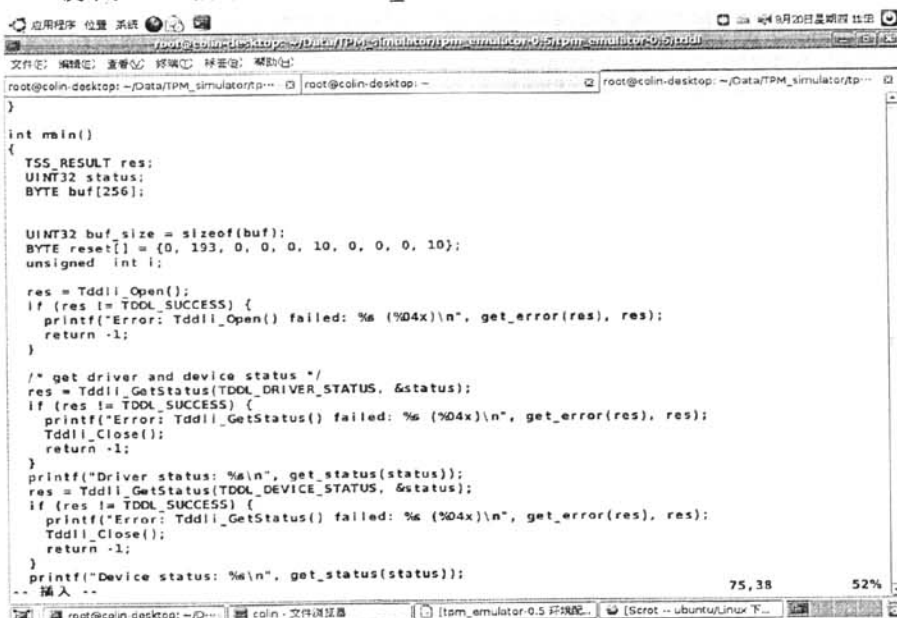
以 TPM_OIAP 命令为例，在 test_tddl.c 相应数组中输入 TPM_OIAP 命令对应的数组命令代码 {0, 193, 0, 0, 0, 10, 0, 0, 0, 10}，在终端中运行 test_tddl 后，在命令运行监视终端窗口中可显示此命令执行结果。具体步骤为:

```
cd tddl
```

在终端命令行进入此命令 TPM_emulator-0.5 安装路径下的 tddl 目录。

```
vi test_tddl.c
```

使用 vi 编辑器对 test_tddl.c 进行编辑 (如图 4-4 所示):



```

int main()
{
    TSS_RESULT res;
    UINT32 status;
    BYTE buf[256];

    UINT32 buf_size = sizeof(buf);
    BYTE reset[] = {0, 193, 0, 0, 0, 10, 0, 0, 0, 10};
    unsigned int i;

    res = Tddl_Open();
    if (res != TDDL_SUCCESS) {
        printf("Error: Tddl_Open() failed: %s (%04x)\n", get_error(res), res);
        return -1;
    }

    /* get driver and device status */
    res = Tddl_GetStatus(TDDL_DRIVER_STATUS, &status);
    if (res != TDDL_SUCCESS) {
        printf("Error: Tddl_GetStatus() failed: %s (%04x)\n", get_error(res), res);
        Tddl_Close();
        return -1;
    }
    printf("Driver status: %s\n", get_status(status));
    res = Tddl_GetStatus(TDDL_DEVICE_STATUS, &status);
    if (res != TDDL_SUCCESS) {
        printf("Error: Tddl_GetStatus() failed: %s (%04x)\n", get_error(res), res);
        Tddl_Close();
        return -1;
    }
    printf("Device status: %s\n", get_status(status));
}

```

图 4-10 仿真包测试程序

Figure 4-10 Emulator Software Test Program

```
make test_tddl
```

运行 make 文件编译 test_tddl.c 文件生成相应可执行文件 test_tddl, 通过上层 tddl 进行命令序数及参数的传递, 从而执行相应命令操作。

```
./test_tddl
```

运行可执行文件 test_tddl。

此时在命令运行监视终端窗口中可见 TPM_OIAP 命令成功执行。如图 4-11 所示:

```

root@colin-desktop: ~/Data/TPM_simulator/tp... root@colin-desktop: - root@colin-desktop: ~/Data/TPM_simulator/tp...
root@colin-desktop: ~/Data/TPM_simulator/tpm_emulator-0.5/tpm_emulator-0.5# tpm -f clear
tpmd.c:380: Info: 北京电子科技学院 (besti)
tpmd.c:381: Info: starting TPM Emulator daemon
tpmd.c:145: Info: parsing options
tpmd.c:187: Info: opening random device /dev/urandom
tpmd.c:203: Info: installing signal handlers
tpmd.c:290: Info: staring min loop
tpmd.c:263: Info: initializing socket /var/tpm/tpmd_socket:0
../tpm/tpm_startup.c:30: Info: TPM_Init()
../tpm/tpm_testing.c:242: Info: TPM_SelfTestFull()
../tpm/tpm_testing.c:260: Info: Self-Test succeeded
../tpm/tpm_startup.c:46: Info: TPM_Startup(1)
../tpm/tpm_capability.c:605: Info: TPM_GetCapability() (not fully implemented yet)
../tpm/tpm_cmd_handler.c:4125: Info: TPM command succeeded
../tpm/tpm_authorization.c:181: Info: TPM_OIAP()
../tpm/tpm_cmd_handler.c:4125: Info: TPM command succeeded

```

图 4-11 仿真包测试程序运行结果

Figure 4-11 Output of Emulator Software Test Program

从测试程序的运行结果可以看出，测试程序成功地执行了程序中编辑的测试命令 TPM_OIAP(), 成功地设置了 TPM 的授权协议，获得 TPM 授权是执行 TPM 其它功能的基础。

4.5 可信平台认证程序分析

可信平台认证的实现主要包含如下几个命令: TPM_MakeIdentity, TPM_ActivateIdentity, TPM_LoadKey2, TPM_Extend 以及 TPM_Quote。 TPM_MakeIdentity 生成身份密钥， TPM_ActivateIdentity 获得身份证书， TPM_LoadKey2 将 AIK 证书加载到 TPM 中， TPM_Extend 向 PCR 加载度量值， TPM_Quote 采用 AIK 对度量值进行签名。命令执行顺序如下图所示：

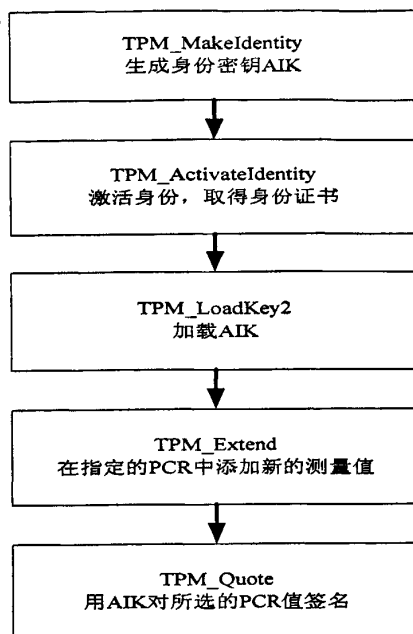


图 4-12 可信计算平台认证相关的 TPM 命令执行顺序

Figure 4-12 Exective Sequence of TPM Command
Related With Trusted Computing Platform Authentication

TPM_emulator-0.5 仿真包中这五个函数与可信平台认证实现相关，其中 TPM_MakeIdentity, TPM_ActivateIdentity, TPM_LoadKey2 三个命令是关键命令。课题对这三个命令的相关数据结构和实现流程进行了深入的分析，为今后在 USBkey 中实现 TPM 的可信认证打下基础。

4.5.1 TPM_MakeIdentity 的实现

1. TPM_MakeIdentity 功能

TPM_MakeIdentity 的主要功能是产生身份认证密钥 AIK，其中公钥部分存放在 TPM_PUBKEY 结构体中，而私钥部分是以 TPM_STORE_ASYMKEY 结构的形式秘密存储在 TPM 的封闭区域里。

2. TPM_MakeIdentity 涉及到的主要数据结构

TPM_MakeIdentity 命令位于仿真包的 tpm/tpm_identity.c 中，涉及到的主要数据结构包括：

```

1) typedef struct tdTPM_KEY{
TPM_STRUCTURE_VER ver; //结构的版本号
TPM_KEY_USAGE keyUsage; //密钥类型
TPM_KEY_FLAGS keyFlags; //密钥属性标志位
TPM_AUTH_DATA_USAGE authDataUsage; //显示授权状态（never 或

```

always 状态)

```
TPM_KEY_PARMS algorithmParms; //密钥算法参数
```

UINT32 PCRInfoSize; //PCR 的长度。如果密钥未绑定到一个 PCR 上, 则这个值为 0。

BYTE* PCRInfo; // PCR 结构类型, 如果密钥未绑定到 PCR 上, 则是一个空的数组

```
TPM_STORE_PUBKEY pubKey; //公钥部分
```

```
UINT32 encDataSize; //encData 参数的大小
```

```
[size_is(encDataSize)] BYTE* encData; // 私钥部分
```

```
} TPM_KEY;
```

TPM_KEY 结构定义身份密钥 AIK 的结构, AIK 私钥的数据类型是 TPM_STORE_PRIVKEY, 公钥的数据类型是 TPM_STORE_PUBKEY。

```
2) typedef struct tdTPM_IDENTITY_CONTENTS{
```

```
TPM_STRUCT_VER ver; //结构的版本
```

```
UINT32 ordinal, //TPM_Makeidentity 命令的序号
```

TPM CHOSENID_HASH labefvCADigest; // identityLabel 和 privacyCA 哈希运算值

```
TPM_PUBKEY identityPubKey; //AIK 公钥
```

```
} TPM_IDENTITY_CONTENTS;
```

TPM_IDENTITY_CONTENTS 结构用于存储 TPM 的身份信息, 包括 AIK 公钥、身份标识信息和隐私 CA 的哈希值。

```
3) typedef struct tdTPM_PUBKEY{
```

```
TPM_KEY_PARMS algorithmParms; //密钥算法参数信息
```

```
TPM_STORE_PUBKEY pubKey; //公钥内容
```

```
}TPM_PUBKEY;
```

TPM_PUBKEY 结构存储的是 TPM 公钥数据, 包括公钥的算法参数信息和公钥数据。

3. TPM_MakeIdentity 命令实现流程

TPM_MakeIdentity 命令流程图如图 4-13 所示:

idKeyParams 是 TPM_MakeIdentity 命令的输入参数, 包含身份密钥的一些参数信息, TPM_MakeIdentity 命令首先验证参数属性的正确性, 然后用 authHandle 去验证所有的输入参数是否经过所有者授权, 用 srkAuthHandle 去验证所有的输入参数是否经过 SRK 所有者授权, 并且验证 idKeyParams 的密钥应用类型和可移植性。

接着, TPM_MakeIdentity 命令对加密的 AuthData 值进行解密, 解密后的 AuthData 值在 idKey 中作为 usageAuth 存储, 加密和解密手段均由 TPM_OSAP

设定。按照 idKeyParams 的属性生成一个非对称密钥对, AIK 为 TPM_PT_ASYM 密钥类型, SRK 作为父密钥对 AIK 的私钥部分进行加密, 利用 labelPrivCADigest 和 idKey 的信息组成 TPM_IDENTITY_CONTENTS 结构, 命名为 idContentse。用 AIK 私钥对 idContents 签名, 签名结构存储在 identityBinding 返回参数中。

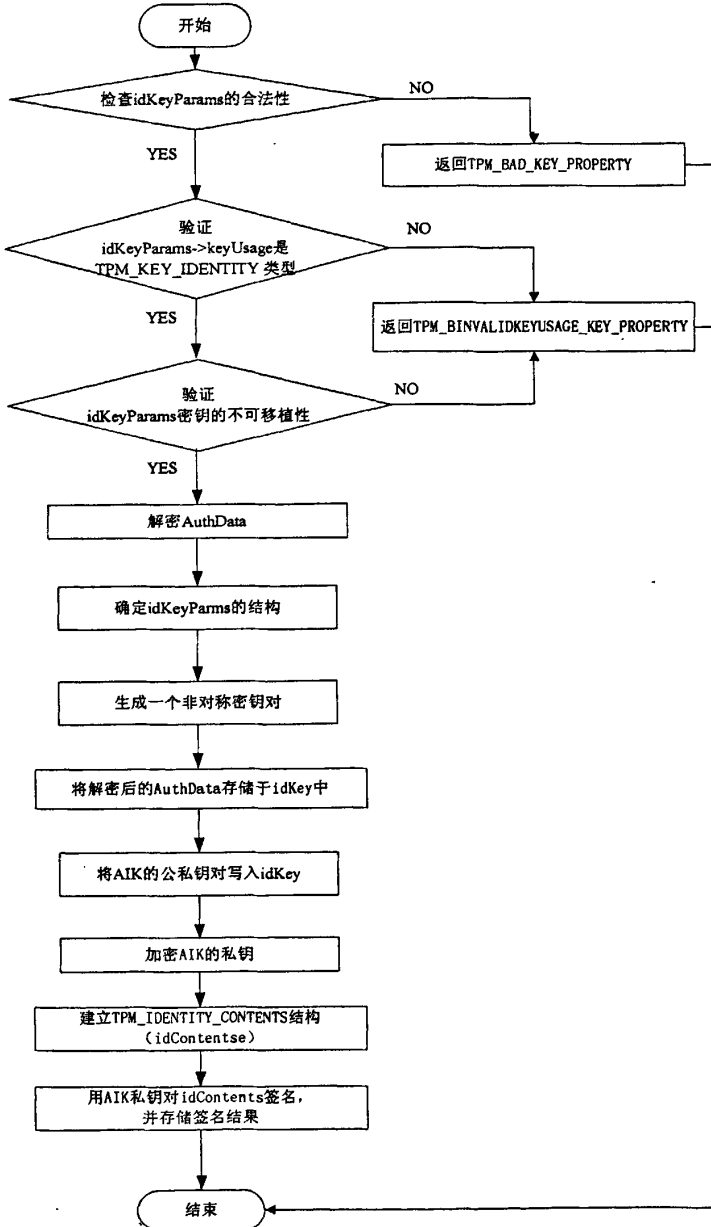


图 4-13 TPM_MakeIdentity 命令流程图

Figure 4-13 Flow Chart of TPM_MakeIdentity Command

4.5.2 TPM_ActivateIdentity 的实现

1. TPM_ActivateIdentity 功能

TPM_ActivateIdentity 命令主要有两个方面的作用，一个是确认 CA 传来的 TPM_SYM_CA_ATTESTATION 结构中的证书是当前 TPM 的，另一个是获得加密 TPM_MENTISTRY_CREDENTIAL 结构的会话密钥。只有平台所有者能够执行此命令。

2. TPM_ActivateIdentity 涉及到的主要数据结构

TPM_ActivateIdentity 命令位于仿真包的 tpm/tpm_identity.c 中。主要涉及到的数据结构包括：

```
1) typedef struct tdTPM_ASYM_CA_CONTENTS {
    TPM_SYMMETRIC_KEY sessionKey; //加密 AIK 证书的会话密钥
    TPM_DIGESTidDigest;          //密钥公钥部分的摘要，此密钥被 CA 验证
}TPM_ASYM_CA_CONTENTS;
```

TPM_ASYM_CA_CONTENTS 结构存储 CA 发给 TPM 的证书，包括加密证书的会话密钥和 TPM 公钥的摘要值。TPM 得到此证书后，取出会话密钥解密证书，并计算自己的公钥的摘要值与此结构中的 TPM_DIGESTidDigest 对比，如果相等则为自己的证书。

```
2) typedef struct TPM_SYMMETRIC_KEY{
    TPM_ALGORITHM_ID algid;          //加密算法
    TPM_ENC_SCHEME encScheme,       //加密机制
    TPM_UINT16 size;                //密钥长度
    TPM_BYTE *data;                 //密钥
}TPM_SYMMETRIC KEY;
```

TPM_SYMMETRIC KEY 结构定义会话密钥的属性，包括算法、方案、长度和密钥内容。

3. TPM_ActivateIdentity 命令实现流程

TPM_ActivateIdentity 命令实现过程的流程图如下所示：

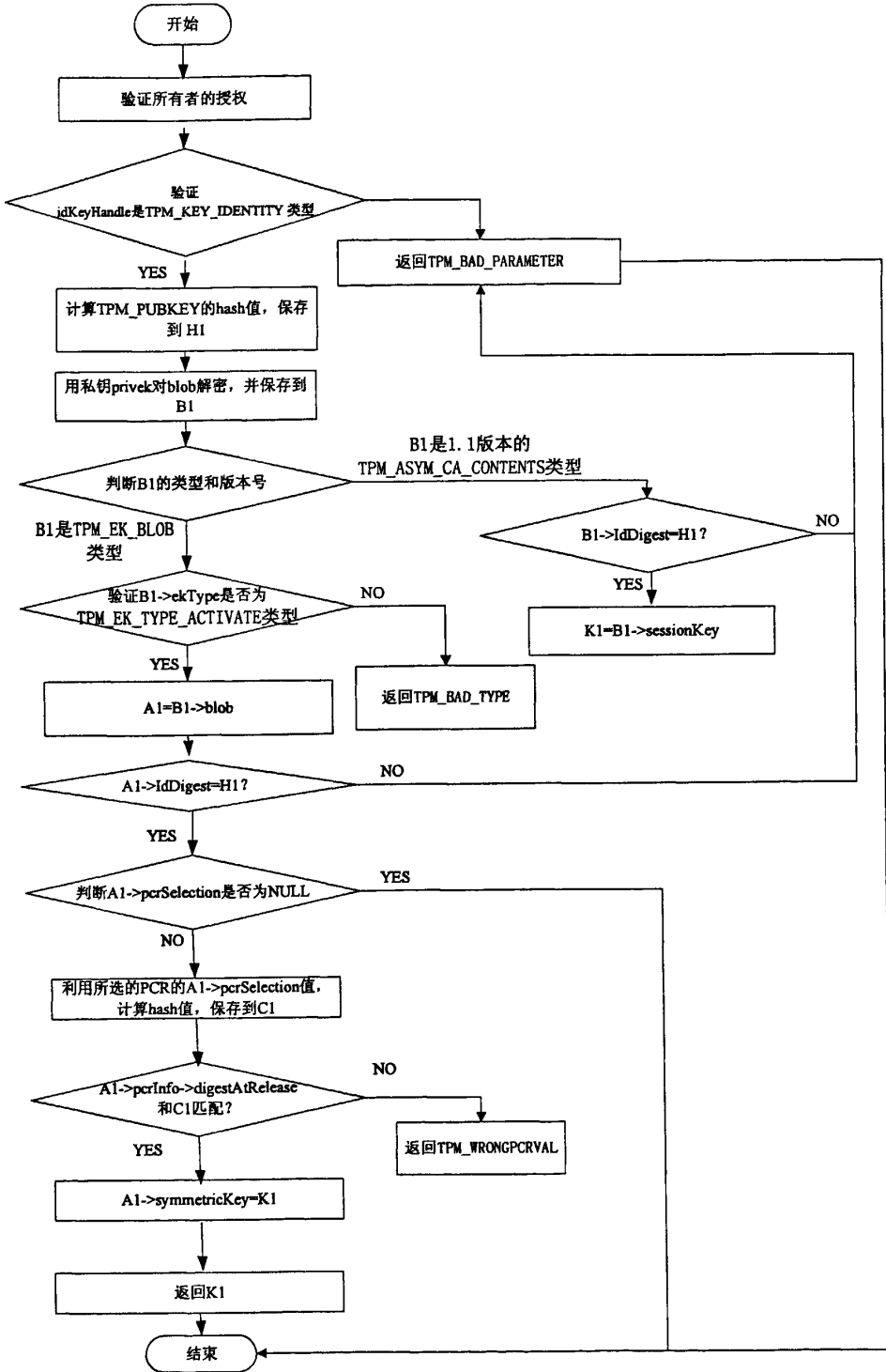


图 4-14 TPM_ActivateIdentity 命令流程图

Figure 4-14 Flow Chart of TPM_ActivateIdentity Command

从流程图中可以看出，该命令首先查看所有者的授权是否有效，检查密钥类型是否为身份密钥（TPM_KEY_IDENTITY）类型。然后取出公钥对其计算摘要值，并用私钥对证书数据进行解密，保存在 B1 中。根据 B1 的类型和版本号判断是 CA 传过来的证书，将解密的证书和计算的摘要值进行比较，相等则确认证书。如果根据 B1 的类型和版本号判断是 TPM_EK_BLOB，则解密出会话密钥。

4.5.3 TPM_LoadKey2 的实现

1. TPM_LoadKey2 功能

在 TPM 使用一个密钥进行加解密、捆绑、封装、签名或者其他功能前，需要把 key 送到 TPM 里。TPM_LoadKey2（）命令就是把 key 加载到 TPM 里面以待使用。

TPM 指派 key 的句柄，TPM 就是通过句柄来定位需要加载的 key。假定句柄可以因为 key 的管理操作而改变。上层的软件要负责保持句柄与其他外部软件使用的标签之间的映射。该命令对 key 使用要进行限制。例如，试图加载一个存储密钥时，需要检验该存储密钥的限制（2048 大小等）。

2. TPM_LoadKey2 涉及的主要数据结构

TPM_LoadKey2 位于仿真包的 tpm/tpm_storage.c 中，涉及到的主要的数据结构主要有：

```
1) typedef struct tdTPM_STORE_ASYMKEY{
TPM_PAYLOAD_TYPE payload; //实体类型
TPM_SECRET usageAuth; //密钥使用授权值
TPM_SECRET migrationAuth; //移植密钥授权值
TPM_DIGEST pubDataDigest; //TPM_KEY 结构的摘要
TPM_STORE_PRIVKEY privKey; //私钥部分
}TPM_STORE_ASYMKEY;
```

TPM_STORE_ASYMKEY 结构主要存储 AIK 的私钥，私钥存放在 TPM_STORE_PRIVKEY 结构中。

```
2) typedef struct tdTPM_KEY(
TPM_STRUCT_VER ver, //版本
TPM_KEY_USAGE keyUsage; //密钥使用类型
TPM_KEY_FLAGS keyFlags; //密钥标志位
TPM_AUTH_DATA_USAGE authDataUsage; //授权使用条件
TPM_KEY_PARMS algorithmParms; //密钥算法参数
UINT32 PCRInfoSize; //PCRInfo 的长度
BYTE* PCRInfo; //TPM_PCR_INFO, 结构内容
```

```
TPM_STORE_PUBKEY pubKey; //公钥部分
UINT32 encDataSize; // encData 的长度
[size is(encDataSize)] BYTE* encData; //加密的 TPM_STORE_ASYMKEY 结构
}TPM_KEY;
```

TPM_KEY 结构在 TPM_Makeidentity 介绍过，存储 AIK 的公私钥对。

3. TPM_LoadKey2 命令实现流程

TPM_LoadKey2 命令实现过程的流程图如下所示：

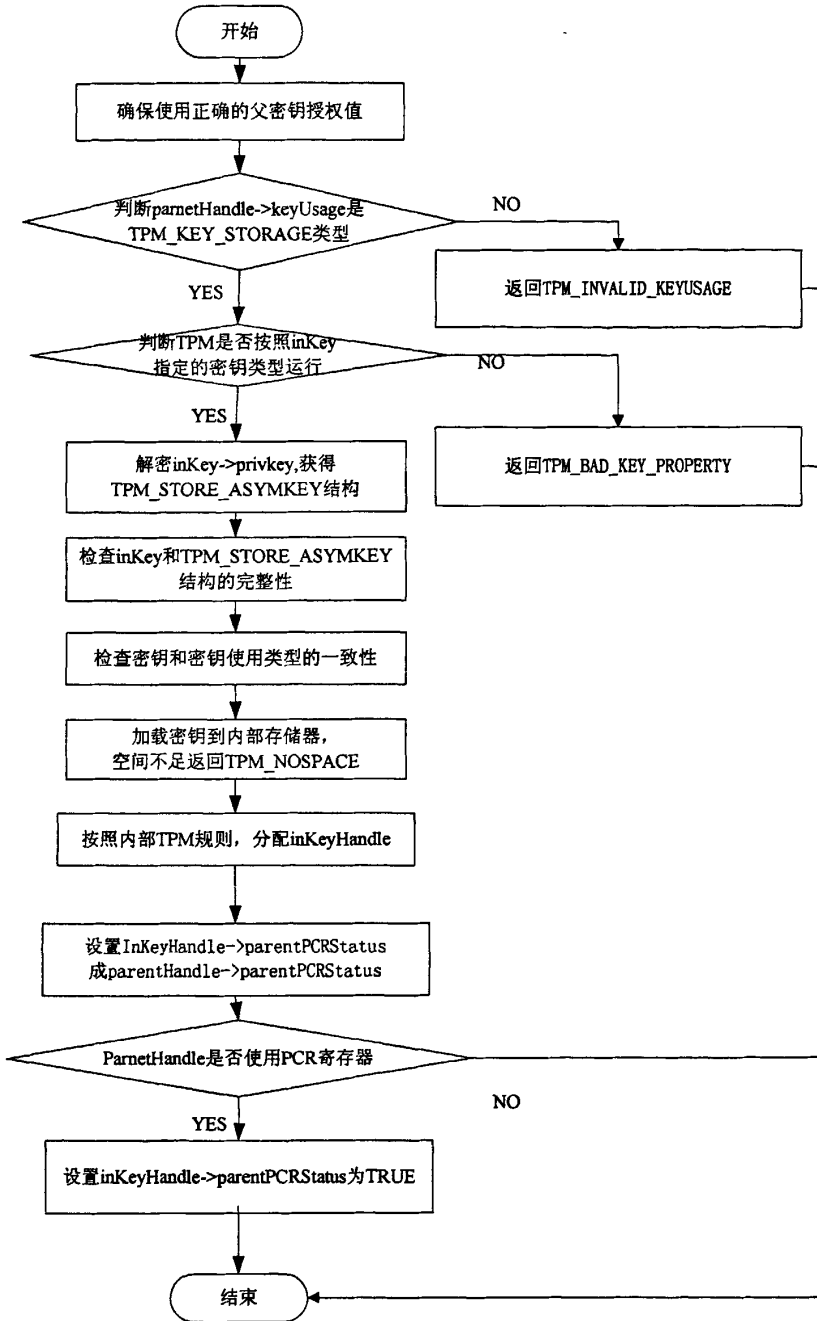


图 4-15 TPM_LoadKey2 命令流程图

Figure 4-15 Flow Chart of TPM_LoadKey2 Command

从流程图可以看出，TPM_LoadKey2 首先检查使用正确的父密钥授权值和密钥类型，然后根据密钥类型进行加载密钥。如果密钥是不可移植类型，检验非对称密钥对的公私钥的一致性。如果是身份密钥类型（TPM_KEY_IDENTITY），确定算法必须是 RSA，密钥大小必须是 2048bit。上述条件符合后将密钥装载到

内部存储器，并根据 TPM 规则给装载的密钥分配句柄。

4.6 本章小结

本章研究了 IBM 的开源软件 TPM_emulator 0.5 仿真包的结构和功能，以及本人在仿真包的编译、调试中的经验，重点对 TPM 仿真包的可信平台认证部分实现进行了分析，详细分析可信平台认证相关的主要函数的数据结构和实现流程。在此研究与分析基础上，本人在《北京电子科技学院学报（理工版）》2007 年第 4 期上发表了一篇名为《基于 USBkey 的可信计算平台研究与设计》的论文，提出了仿真实现 TPM 可信认证平台的设计方案，为今后在 USBKey 中实现 TPM 的远景目标打下了基础。

结 论

本文主要围绕可信计算技术展开研究，为了在 USBKey 上实现 TPM 功能的这一远景目标，本文主要进行了三个方面的研究与实践：一、可信计算主要安全机制的研究，弄清 TPM 的安全机制；二、可信度量根 BIOS 安全问题的研究，弄清 BIOS 改造和安全增强方法；三、可信平台模块仿真系统分析，弄清 TPM 功能实现的细节。

在可信计算主要安全机制的研究中，论文主要研究了可信平台关键组成部分可信平台模块 TPM 和可信软件栈 TSS 的结构与功能，分析了可信计算技术的主要安全机制——可信链度量机制、可信存储机制、可信认证机制。

BIOS 作为可信链的初始可信度量根，是可信的源头，因此可信度量根 BIOS 安全问题尤为重要。论文对可信度量根 BIOS 安全问题进行了深入研究，通过实验室环境下的 BIOS 漏洞情景再现和理论分析总结了 BIOS 安全威胁种类，研究了 BIOS 模块结构和 BIOS 采样技术，并针对典型的 BIOS 安全隐患，设计实现了 BIOS 安全检查系统。该检查系统通过建立标准 BIOS 样本库和 BIOS 安全隐患库对计算机的 BIOS 进行采样分析，并根据分析结果进行安全修补，加强了 BIOS 的安全性。在 64 位的 EFI BIOS 即将代替传统的 16 位 BIOS 系统的趋势下，EFI BIOS 在提升了 BIOS 的容量和功能同时也带来更多的安全隐患，本文提出了一种通过 USBKey 增强 EFI BIOS 安全性的设计方案。

为了弄清 TPM 功能实现的细节，论文研究了 IBM 的开源软件 TPM_emulator 0.5 仿真包的目录结构和功能，对仿真包进行了修改完善，并编译调试通过，对 TPM 命令进行了测试和试验。本文还针对 TPM_emulator 0.5 仿真包的可信平台认证部分实现进行了详细分析，描述了可信平台认证相关的主要函数的数据结构和实现流程。

论文在上述三个方面的研究与探索为在 USBKey 中实现 TPM 功能打下了坚实的理论基础和实践基础。

在论文的研究和实现上本人遇到了很多困难。最先遇到的就是大量 TCG 文档的研究问题。由于文档较多，且都是英文文献，需要细心地钻研，做大量的分析与文档整理工作。TCG 近几年的标准更新较快，在研究中还需及时跟踪最新标准发布情况，这样才能跟上可信计算技术的发展步伐。在论文的研究中，通过多次参加国内举办的可信计算技术年会和国家信息保密年会，从中学习到很多有益的思想与知识，对论文的完成很有启发。在 BIOS 安全研究与检查系统的实现过程中遇到很多困难，例如 BIOS 信息采集方法的选取、BIOS 隐患库和标准代码样本库的建立问题，由于国内外在 BIOS 安全方面的研究较少，可借鉴的资料

很少，许多方案的设计与选择都是由实验室的实验结果而得出的。在 BIOS 检查系统的实现中培养了我的方案设计与编程实现能力。在 TPM_emulator 0.5 仿真软件包的研究分析过程中，该软件包是开源代码，没有帮助文档，需要结合 TPM1.2 规范进行分析。在将软件包进行内核编译中多次编译不成功，通过在网上查找资料，深入研究 linux 操作系统终于成功搭建编译环境，最终将软件包编译成功。在对软件包主要模块进行分析的过程中，我选择对照 TPM1.2 规范和对程序源代码进行跟踪的方式进行分析，工作量大并且涉及的数据结构较多，需要比较全面的密码学知识，锻炼了我的理论联系实际的能力和进行分析设计的能力。

为了在 USBKey 中实现 TPM 功能的远景目标，下一步需要进一步分析 TPM_emulator 0.5 可信度量模块的数据结构和关键命令实现流程，尤其是 TPM_Extend 命令和 TPM_Quote 命令，这两个命令是可信度量的核心实现。另外，为了实现这些核心命令的调用，需要按照 TSS 规范编写相应的应用接口 API。同时，为了外挂 TPM 模块在操作系统加载前执行可信计算功能，需要对 BIOS 进行改造，而目前 EFI BIOS 提供了可改造的空间和兼容高级语言的规范接口，可以将 USBKey 的驱动程序按照 EFI 接口规范写入 EFI BIOS，在 EFI BIOS 中执行 USBKey 的驱动。按照 TPM 规范在 USBKey 中实现的是可信存储根与可信报告根，EFI BIOS 经过改造之后实现了可信度量根的功能，远景目标的实现将探索出一条新的可信计算模块应用之路。

参考文献

- 1 沈昌祥,张焕国,冯登国,曹珍富,黄继武.信息安全综述.中国科学 E 辑,2007 年第 37 卷第 2 期:134~140
- 2 肖政,韩英,叶蓬,侯紫峰.基于可信计算平台的体系结构研究与应用.计算机应用.2006 年 8 月.1807~1809
- 3 秦戈,韩文报.关于可信计算平台模块的研究.信息工程大学学报.2006 年 12 月:341~343
- 4 郭煜.TPM 中身份证明密钥的管理.信息安全与通信保密.2006 年 4 月: 76~78
- 5 孔维广.可信计算平台的工作原理与应用研究.武汉科技学院学报.2003 年 6 月:81~82
- 6 谭亶烈.可信计算平台中的关键部件 TPM.信息安全与通信保密. 2005 年 2 月:29~31
- 7 Trusted Computing Group.TCG Specification Architecture Overview Revision1.2,2004.<https://www.trustedcomputinggroup.org/downloads/specifications>
- 8 Trusted Computing Group. TPM Main:Part 1 Design Principles, 1.2 edition,2005. <https://www.trusted.computinggroup.org/downloads/specifications>.
- 9 Trusted Computing Group. TPM Main:Part 2 TPM Structures, 1.2 edition,2005. <https://www.trustedcomputinggroup.org/downloads/specifications>.
- 10 Trusted Computing Group. TPM Main:Part 3 Commands, 1.2 edition,2005. <https://www.trutedcomputinggroup.org/downloads/specificafons>.
- 11 The Trusted Computing Group, TCG PC Client Specific TPM Interface Specification,USA:TCG, July 11, 2005.
- 12 余发江,张焕国.可信安全计算平台的一种实现.武汉大学学报, 2004,50(1):69~72
- 13 胡中庭,韩臻.操作系统安全可信链的研究与实现.信息安全与通信保密.2007 年第2期:47~49
- 14 The Trusted Computing Group, TCG PC Client Specific Implementation Specification for Conventional BIOS,USA:TCG, July 13, 2005.
- 15 Edward W.Felten.Understanding Trusted Computing:Will Its Benefits Outweigh Its Drawbacks? IEEE Security and Privacy,2003,1 (3):60-62
- 16 TCG Software Stack (TSS) Specification Version 1.10 Golden.Trusted Computing Group,Incorporated.August 2003.

- 17 B. Blakley, D.M. Kienzle. Some Weaknesses of the TCB Model. 1997 IEEE:1~5
- 18 Andreas Pashalidis, Chris J. Mitchell, Single Sign-On Using Trusted Platforms. Proc. of Information Security: 6th International Conference, ISC2003, Bristol, UK, October, 2003.
- 19 魏志东, 冯登国. 一种微型 PKI 客户端密钥管理设备的实现方案. 计算机工程. 2002, 28(10):58~60
- 20 周振柳, 刘宝旭, 池亚平, 许榕生. 计算机 BIOS 安全风险分析与检测系统研究. 计算机工程. 2007 年 8 月:114~116
- 21 张焕国等. 可信计算平台技术研究. 第十届全国容错计算学术会议论文集:169~171
- 22 (美) Sean W. Smith 著. 冯登国, 徐震, 张立武译. 可信计算平台: 设计与应用. 清华大学出版社. 2006 年 10 月:147~155.
- 23 池亚平, 方勇, 吴艺圆. 构建可信网络的研究. 北京电子科技学院. 2005 年第 13 卷第 4 期:19~20
- 24 方艳湘, 黄涛. Linux 可信启动的设计与实现. 计算机工程. 2006 年 5 月:51~52
- 25 徐娜, 韦卫. 基于安全芯片的可信平台设计与实现. 计算机应用研究. 2006 年第 8 期:117~119
- 26 王斌, 谢小权. 可信计算机 BIOS 系统安全模块的设计. 计算机安全. 2006 年 9 月:35~36
- 27 张焕国, 罗捷, 金刚, 朱智强. 可信计算机技术与应用综述. 计算机安全. 2006 年 9 月:8~10
- 28 杨鑫坤, 王薇, 杨进波. 可信计算技术的研究与应用. 洛阳大学学报. 2006 年 12 月:69~70
- 29 刘毅, 余发江. 可信计算平台应用研究. 计算机安全. 2006 年 6 月:13~15
- 30 孙琪, 杨昌等. 可信计算软件栈的体系研究. 楚雄师范学院学报. 2006 年 12 月:33~34
- 31 秦中元, 胡爱群. 可信计算系统及其研究现状. 计算机工程. 2006 年 7 月:111~113
- 32 张焕国等. 可信计算研究进展. 武汉大学学报(理学版). 2006 年 10 月:513~518
- 33 李春燕. 可信赖计算平台关键技术分析及应用. 计算机工程. 2006 年 12 月:124~125
- 34 肖敬, 喻超. 一种基于嵌入式安全系统的可信计算机系统. 计算机工程. 2006 年 7 月:246~247
- 35 李红霞, 傅鹏. 主流平台中的可信路径安全机制分析. 微计算机信息. 2006 年第 22 卷 6-3 期:97~98
- 36 UEFI, Unified Extensible Firmware Interface Specification version 2.10, USA: Intel, January 23, 2007.

- 37 Shuanghe Peng, Zhen Han. Trust of User Using U-Key on Trusted Platform. ICSP2006 Proceedings:56~58
- 38 Shuanghe Peng, Zhen Han. Enhancing the Security of PC with U-Key. IEEE Security & Privacy, September, 2006.
- 39 凌有慧. BIOS设置逐个看. 微型计算机2006年8月:135~136
- 40 臧锐, 贺也平, 朱继峰. Linux可信路径安全机制研究及改进. 计算机应用研究. 2007年3月第24卷第3期:109~111
- 41 高万鹏. 可信计算组织软件栈的研究与实现. 北京科技大学硕士学位论文. 2006:25~28
- 42 陈山. 基于可信计算和PKI的身份认证技术研究. 北京科技大学硕士学位论文. 2006:6~18
- 43 李佳蕾. LINUX下基于可信平台模块的开机认证和可信引导的设计与实现. 北京交通大学硕士论文. 2006:23~26
- 44 Jan Axelson. USB 大全. 陈逸译. 中国电力出版社. 2001:23~34
- 45 Shuanghe Peng, Weimin Tang, Yali Mou. The Implementation of T=0 Protocol of Smartcard based on USB Control Chip. Journal of Computer Applications, Vol. 24, October 2004.
- 46 Ateniese G, Camenisch J, Joye M, Tsudik G. A practical and provably secure coalition resistant group signature scheme. // Advances in Cryptology. 2000:255~270
- 47 D. Chadwick, S. Otenko, W. Xu, Adding Distributed Trust Management to Shibboleth. 4th Annual PKI R&D Workshop Proceeding. IST, August 2005.
- 48 Edward W. Felten. Understanding Trusted Computing: Will Its Benefits Outweigh Its Drawbacks? IEEE Security and Privacy, 2003, 1 (3):60~62
- 49 Einarsson S, Rausand M. An Approach to Vulnerability Analysis of Complex Industrial Systems. Risk Analysis, 1998, 18(5): 535~546.
- 50 Compaq, Phoenix, Intel. BIOS Boot Specification v1.01. (1996-10). <http://www.phoenix.com/NR/rdonlyres/56E38DE2-3E6F-4743-835F-B4A53726ABED/0/specsbbs101.pdf>.
- 51 Brickell E, Camenisch J, Chen L. Direct anonymous attestation. // ACM Conference on Computer and Communications Security. 2004:132~145
- 52 Daniel Gustafsson, Tomas Stewén. Trusted Computing & Digital Rights Management – Theory & Effects. Reports from MSI. 2004-9:10~29
- 53 毛德操, 胡希明著. Linux 内核源代码情景分析. 浙江大学出版社. 2001年8月: 615~740

攻读硕士学位期间发表的学术论文

1 池亚平, 王全民. 基于 USBkey 的可信计算平台研究与设计. 北京电子科技学院学报. 2007 年第 4 期 (理工版).

2 池亚平, 王全民. 一种基于 USBKey 的可信度量根安全增强设计方案. 信息安全与保密通信. 2007 年第 11 期.

致 谢

经过多年的不懈努力，在老师的指导和帮助下终于完成了论文，这一天的到来对于将近不惑之年的我来之不易。这里首先要感谢我的指导老师王全民副教授，感谢王老师在论文的最后阶段对我的严格要求和认真把关。王老师在学术研究上的严谨求实的精神，对我的学习和工作产生了深远的影响。

感谢北京电子科技学院的方勇副院长，在他带领的可信计算研究梯队中的研究工作对论文的研究帮助很大，研究梯队中开放的学术研讨氛围和严谨的科学研究精神使我受益匪浅。感谢北京电子科技学院与中国科学院高能物理研究所计算中心的网络安全联合实验室为我提供的研究与实验环境。

感谢北京电子科技学院通信工程系主任蒋华教授和全体教师，他们在工作中给予的热忱支持和帮助，使我在多次不堪负荷中坚持下来完成了论文。

感谢北京工业大学计算机学院的各位老师对我的培养，感谢林屹和罗琼老师耐心细致的指导和帮助！

感谢李鹏、葛明明、张利霞等同学，感谢他们在学习中对给予我的帮助！在将近不惑之年，这段美好、愉快的同学时光将永远留在我的记忆深处。

最后，感谢我的家人给予我的理解与支持，在我学习和工作最困难的时候，是他们的关爱和理解给了我前进的动力和战胜困难的决心！

个人简历

池亚平,女,1969年12月生。1992年7月毕业于重庆大学计算机系计算机软件专业,获得工学学士学位。1992年8月大学毕业分配至北京电子科技学院通信工程系工作至今,主讲《计算机局域网》、《计算机广域网》等课程,主要研究方向为网络安全。

工作期间发表的与论文有关的论文包括:

池亚平,方勇,吴艺园.构建可信计算网络的研究.北京电子科技学院北京电子科技学院学报.2005第13卷4期.

池亚平,刘平,方勇.可信计算中的密钥托管问题研究.中国计算机学会信息保密专业委员会论文集.2006年9月.

工作期间参加的与论文有关的科研项目包括:

北京电子科技学院重点实验室研究课题:构建基于可信计算的分布式网络的研究,担任课题组长。

北京电子科技学院工程科研项目:计算机BIOS安全检查研究,担任项目组长。

北京电子科技学院工程科研项目:基于BIOS的高安全计算机网络接入认证系统研究,担任项目组长。