

分类号_____

学校代码 10487

学号 M201276167

密级_____

华中科技大学

硕士学位论文

基于 UEFI 的可信 BIOS 系统测试方案 的设计与实现

学位申请人：涂 晶

学 科 专 业：软件工程

指 导 教 师：万 琳 副教授

答 辩 日 期：2014.11.10

**A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree for the Master of Engineering**

**Design and Implementation of Trusted UEFI
BIOS System Testing Program**

Candidate : Tu Jing

Major : Software Engineering

Supervisor : Assoc. Prof. Wan Lin

Huazhong University of Science and Technology
Wuhan 430074, P. R. China

November, 2014

独创性声明

本人声明所呈交的学位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，除文中已经标明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的研究成果。对本文的研究做出贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：

日期： 年 月 日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，即：学校有权保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权华中科技大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本论文属于 ☐ 保密， 在_____年解密后适用本授权书。
☐ 不保密。

（请在以上方框内打“√”）

学位论文作者签名：

日期： 年 月 日

指导教师签名：

日期： 年 月 日

摘 要

UEFI BIOS 是一个全新的预启动平台，具有规范化、模块化、C 语言编写等优点，支持来自不同操作系统引导前的应用，承担起网络、管理、调试、配置等不同扩展的功能。随着 BIOS 芯片模块功能和容量地不断增加，UEFI BIOS 平台的安全隐患也越来越多，相比其他组件模块，BIOS 系统在可信计算机中的地位更加突出，如何利用测试工具和测试方法来验证 BIOS 的可信性将是 UEFI BIOS 领域的重点和热点。

通过对 UEFI 框架和 BIOS 关键流程技术实现的学习，从关键硬件 TPM 芯片工作机制、可信算法的运用、软件实现几个方面梳理出了系统启动流程中各阶段安全性方案的实现原理。并针对系统中缺乏自身安全防护功能；刷新芯片机制易受到病毒的破坏或者被添加恶意危害安全系统的代码；在配置问题上存在安全问题被远程攻击者利用等隐患点，归纳出可信 BIOS 需要具有保护数据、验证身份、测量完整性以及存储报告方面的功能，利用测试技术策略结合 BIOS 在安全性方案实现的原理，设计并实现了可信 BIOS 的无网络状态下的 BitLocker 功能验证、EFI 文件完整性验证和网络状态下 WHCK 环境下可信 BIOS 功能验证、BitLocker NKP 功能验证等测试用例。

依据测试用例在不同平台、不同操作系统、不同版本 BIOS 的组合下搭建测试环境，运用测试工具、软件、脚本命令等执行测试任务，通过手动和自动的测试形式，结合冒烟测试、本地化测试、黑盒测试、回归测试等测试方法，验证 BIOS 在各环境下的可信性，并对测试结果进行了分析。

关键词：统一可扩展固件接口 可信基本输入输出系统 测试
可信计算模块

Abstract

UEFI BIOS is a new pre-boot platform with standardized, modular and other advantages of the C language. It can support for multiple operating system boot before the underlying application. And it assumes a variety of extensions configuration, commissioning, management, network and so on. With the increasing of BIOS chip module function and capacity, more and more security risks highlight in UEFI BIOS platforms. Compared to other component modules, BIOS system status is more prominent in trusted computer, how to use test tools and test methods to verify BIOS credibility will be the focus areas and hot spots in UEFI BIOS.

Through the learning of UEFI BIOS framework and process technology, tease out the principles to achieve the various stages of the system startup process safety programs from the use of critical hardware TPM chip working mechanism, credible algorithms, software implementation aspects. And for the lack of protection of its own security systems, refresh chip mechanism vulnerable to destruction of the virus, malicious code been added to endanger the safety of the system, security issues on configuration been used by remote attackers and other hidden spots, summarize the capabilities of trusted BIOS that protecting data, verifying the identity, integrity measurement and report storage. Combine the use of test technology strategy with BIOS security scheme in principle, to design and implement a credible verification BIOS BitLocker functionality without network state, EFI file integrity verification, under WHCK trusted BIOS functional verification environment and BitLocker NKPfunctional verification test under network status.

A test environment is set up based on test cases under different platforms, different operating systems, different versions of the BIOS combinations. Test tools, software, script commands to perform testing tasks are used. Manual and automated test form is combined with smoke testing, localization testing, black box testing, regression testing and other testing methods to verify the credibility of BIOS in each environment and analyze the test results.

Key words: UEFI Trusted BIOS Test Trusted Computing Module

目 录

摘 要	I
Abstract	II
1 绪论	
1.1 研究背景和意义	(1)
1.2 国内外研究情况	(2)
1.3 关键技术介绍	(4)
1.4 研究内容和论文组织结构	(10)
2 可信 BIOS 系统测试方案的需求分析	
2.1 可信 BIOS 系统测试的总体结构分析	(12)
2.2 可信 BIOS 系统的安全方案分析	(14)
2.3 可信 BIOS 系统的测试目标	(18)
2.4 可信 BIOS 系统的测试流程	(18)
2.5 本章小结	(19)
3 可信 BIOS 系统测试方案的设计	
3.1 测试方案的总体设计	(20)
3.2 测试工具的设计	(24)
3.3 测试用例的设计	(27)
3.4 本章小结	(30)
4 可信 BIOS 系统测试方案的实现	
4.1 测试平台硬件的搭建	(31)
4.2 测试平台软件的搭建	(32)
4.3 测试用例实现	(38)
4.4 本章小结	(43)

5 测试结果及分析

5.1 测试结果..... (44)

5.2 结果分析..... (47)

5.3 本章小结..... (51)

6 全文总结与展望

6.1 全文总结..... (52)

6.2 展望..... (53)

致 谢 (55)

参考文献 (56)

1 绪论

本章首先介绍了基于 UEFI 框架下可信 BIOS 的测试背景和国内研究现状，并对其进行了分析；然后对文章中系统测试使用到的关键技术进行了阐述；最后对文章要研究的内容进行描述。

1.1 研究背景和意义

随着 IT 技术不断发展，保证软件的正确性、完整性、安全性和质量的工作变得越来越重要，为计算机提供最底层、最直接的硬件设置和控制的基本输入输出系统的安全测试则更显重要，为了测试基本输入输出系统，我们首先来了解它。上世纪七十年代，Gary Kildall 提出了引导固件的概念，也即为操作系统和系统硬件提供一个抽象层，用来引导系统同时也为系统提供外设的引导。随后，IBM 在开发第一台计算机的时候，对固件引导进行了扩展，将开机的前导程序以及一些输入/输出子程序存储到 PROM(可程序只读存储器)，并取名为 BIOS(Basic Input Output System)即基本输入输出系统^[1]，它包括两个重要的部分：一部分是上电自检，另一部分是运行时库，它们给较早的计算机操作系统提供基础的服务。

传统 BIOS 是针对 8086 架构提出来的，此时的处理器是 16 位的，能访问 1MB 的内存，BIOS 一直都受限于这样的平台架构^[2]。同时，为了节约 BIOS 程序的存储空间，一般采用汇编语言编写，这造成了代码的可读性、可移植性都较差，也只有较少专业工程师才能从事 BIOS 的编程维护工作^[3]。另外，BIOS 的服务通过中断完成，自身采用 16 位汇编代码编写，在 64 位处理器的时代，这样的运行模式，无疑使得启动时间长，也成为传统 BIOS 最致命的缺点。

鉴于传统 BIOS 的弊端，1998 年英特尔、微软、HP 和其它一些公司开展了超越 BIOS 行动，开始研发新一代 BIOS，即可扩展固件接口(EFI, Extensible Firmware interface)。EFI 把现代计算机的软件架构概念引入固件程序设计^[4]，开始使用 C 语言来创建一些特定的组件，主要是 Setup 程序，典型的 BIOS 开发工具同样也用 C, C++ 以及 Perl 这样的高级语言来实现。为了统一 EFI 标准和规范，

2005 年计算机软硬件厂商发起成立了国际 UEFI (Unified EFI) 联盟。UEFI 是一个开放的业界标准接口^[5], 在平台固件和操作系统之间定义了一个抽象的编程接口, 但并没有规定这个编程接口的具体实现, 因此它可以在 IA32、IA64、ARM 等许多不同架构实现, 但是都表现为相同的接口。伴随着英特尔的积极推广, UEFI 被更多人了解, 多家 PC 生产商联合成立了 UEFI 论坛, 众多开发者也都加入到 UEFI 开源社区, 在该网站有四个与 UEFI BIOS 相关的开源项目, 即 UDK2014、EDKII、EFI Dev Kit、EDKII Build Tools。目前世界有三大 BIOS 厂商, 即 Phoenix、AMI 和 Insyde, 在国际 BIOS 生产商向 EFI 架构转型之际, 给国内研发者们提供了一个新的契机, Byosoft 是中国大陆第一家得到 Intel 授权的 BIOS 公司, 与 Intel 合作紧密, 研发国产 BIOS^[6]。

层次关系上来看, 它位于计算机硬件层和操作系统层之间, 一般以固件 (Firmware) 的形式存在, 向下负责计算机的硬件管理, 向上对操作系统提供统一的硬件使用和管理接口, 起到一个桥梁的作用。

针对不断推出的新版本 BIOS, 如何用最准确高效的方法进行测试是我们研究的目的。目前针对 BIOS 的稳定性, 可操作性, 平台兼容性等相关测试已经有相对完善的测试工具来完成, 测试效果也比较理想, 在但是随着 BIOS 不断发展, 其安全问题也慢慢突显出来, 主要包括这样两个方面: 首先, 针对 BIOS 自身而言, 破坏 BIOS 代码或者在代码中间添加一些恶意代码, 从而破坏系统的硬件和软件功能^[7], CIH 病毒就是其中一个典型的代表。这类威胁处于计算机底层, 不容易清理, 更隐蔽, 给计算机造成极大的安全隐患。另外, 通过网络的远程控制同样给 BIOS 的安全带来了隐患, 远程攻击者在本地计算机关机时都可以启动系统来存储访问^[8]。由此可见, 保证 BIOS 的安全性在一个完整的计算机系统中的重要, 本文通过验证 BIOS 部分安全保护功能的可行性, 从而进一步加强 BIOS 的可信性^[9]。

1.2 国内外研究情况

目前, 计算机技术掀起了一场新的互联网革命, 且已广泛应用于社会各个领域,

打破了地域和空间的限制，改变了人们信息获取和知识传播的方式，真正形成了信息化^[10]。于此同时，它也潜藏着一定的安全风险，甚至影响到国家安全战略。我国政府安全相关部门，一直倡导国产化 PC 系统，由中国科学院计算所自主研制的通用龙芯 CPU 为中国信息产业无“芯”的局面迈出了重要的一步，在 BIOS 领域，南京百敖软件是国内唯一的专业固件（BIOS）产品及服务供应商。对于计算机系统来说，BIOS 是开机执行的第一个步骤，同时也掌控着所有的计算机硬件资源，它的重要性甚至在操作系统之上。

UEFI 作为新一代 BIOS 技术最大的特点是运用模块化设计^[11]，从而弥补了传统 BIOS 的缺点^[12]，开机之后，UEFI 首先就会进行初始化，主要的操作是检测硬件设备和加载硬件的驱动程序，免去了由操作系统来加载的弊端，在重装系统后不需重新安装驱动。UEFI 运用 C 语言编程，可扩展性、可视化、兼容性以及网络功能都增强了，这让 UEFI 逐渐接近初级操作系统，UEFI 凭借这些技术特征，广泛运用于网络电脑、嵌入式应用等产品。这些开放性和灵活性，同时给 BIOS 黑客们提供了温床，1999 年 CIH 病毒的大爆发使得全球六千万台电脑遭到破坏，主要是通过破坏 BIOS 造成的。2007 年中国黑客将当时微软最新的操作系统 Windows Vista OEM 验证机制破解了^[13]。这次攻击事件中黑客利用反编译 BIOS 代码的技术，实现了在没有源码编译的情况下，对 BIOS 的代码植入，从而造成破坏。随着 UEFI 技术源码的公开，部分程序甚至不需要反编译就能被侵入，攻击者可以利用模块化结构和改写 BIOS 内容将恶意代码嵌入，隐藏在 BIOS 中的恶意代码比常规程序更具有破坏性和隐藏性，这对 UEFI 安全性^[14]提出了更高的要求。

为了从整体上提高计算机系统的安全性，1999 年，由惠普、IBM、英特尔、微软等公司组成的可信计算平台联盟 TCPA(Trusted Computing Platform Alliance)诞生了，后来被正式改为可信计算组织 TCG (Trusted Computing Group)^[15]。TCG 定义了可信平台模块 (TPM)^[16]，目前已经有具有 TPM 功能的 PC 机上市（惠普、IBM 等）^[17]。可信计算的核心思想是通过在硬件上引入可信芯片，从结构上解决计算机体系结构简化带来的脆弱性问题^[18]。基于密码硬件芯片，从平台加电开始，到应用

程序的执行，构成完整的信任链^[19]。2014 年 4 月 17 日由中国电子信息产业集团、中国信息安全研究院等 60 家单位发起的可信计算产业联盟建立起来了，为我国有效提高自主系统的安全性和可靠性提供支撑。

目前测试 BIOS 安全性主要运用动态测试技术，根据测试使用的方法不同，又分为黑盒测试法和白盒测试法。黑盒测试法，也称功能测试，在测试中，测试人员不必知道内部程序和结构，只要按照规定来验证功能是否正确即可。主要用于发现是否有不正确或者遗忘的功能，在接收输入数据后，是不是能够准确地输出信息。白盒测试法是按照内部结构测试程序，设计测试用例对程序的路径进行验证，确定是否与预期一致，主要用于检测数据结构是否有错。

1.3 关键技术介绍

为了实现可信 BIOS 的系统测试工作，首先需要了解 UEFI BIOS 方面的技术和可信计算平台^[20]在结构上的工作机制。

1.3.1 UEFI 整体框架

作为业界的一个开放标准接口，UEFI^[21]在操作系统和系统固件之间定义了一个类似于操作系统 API 的抽象编程接口，UEFI 的设计基于以下要素：

（1）重用现有的对应表基础接口，为了维护在操作系统和固件上代码的投资，许多在平台兼容处理器上的现有规范必须在 UEFI 框架下完成。

（2）系统分区，定义了分区和文件系统，目的是让多个供应商之间安全共享，这样增值软件就会减少对磁盘空间的占用。

（3）EFI 引导服务，为系统和设备提供了统一的标准接口，设备的访问是通过句柄和协议概念抽象出来的。

（4）EFI 运行时服务（Runtime Services），是为硬件平台提供的最小单元接口，可以调用这些接口与平台固件进行交互。

图 1.1 是组成系统的各个模块之间的相互关系^[22]。

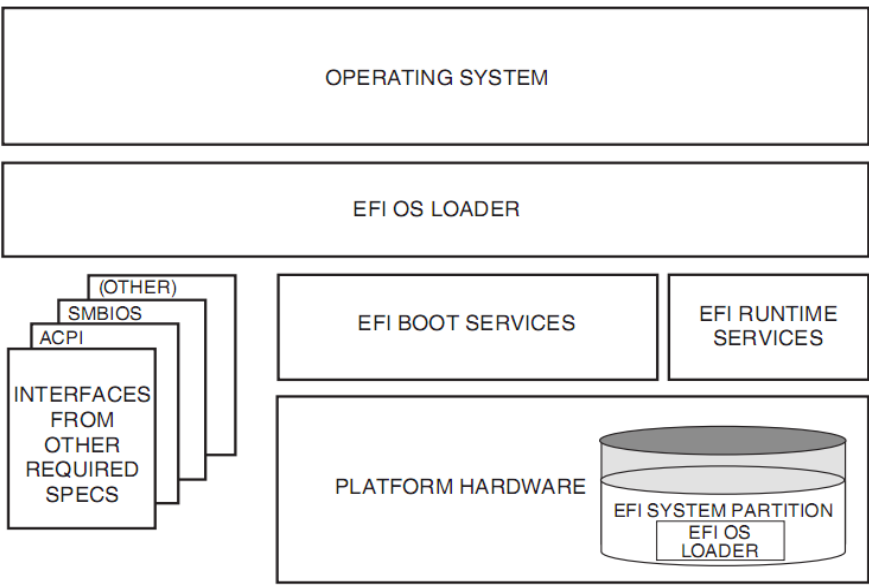


图 1.1 UEFI 框架图

平台固件可以从系统分区中检索找到系统加载程序。EFI 规范支持各种大容量设备，包括硬盘、DVD-ROM、CD-ROM 甚至是远程网络启动。通过修改系统加载程序，也能利用可扩展协议接口，添加其他类型的设备。当操作系统加载程序准备引导操作系统时，引导程序就会调用服务接口和运行时服务。

1.3.2 UEFI 启动顺序

在框架中，正常的启动程序需要经过 SEC 阶段、PEI 阶段、DXE 阶段，BDS 阶段、TSL 阶段、RT 阶段及 AL 阶段^[23]，如图 1.2 所示。

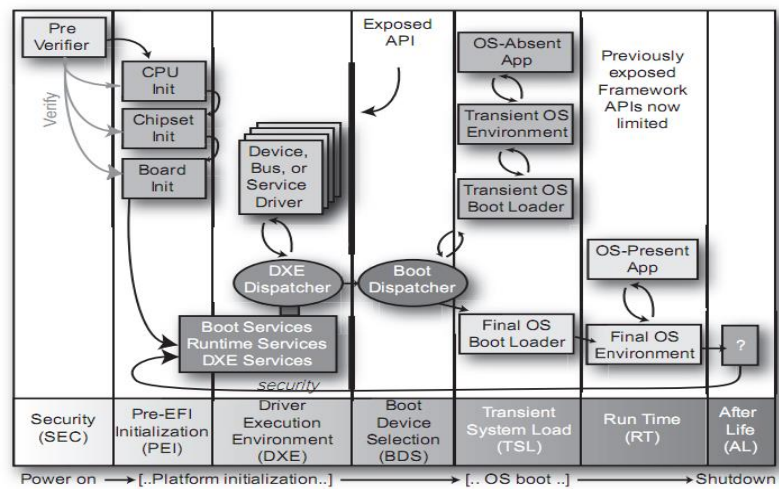


图 1.2 UEFI 启动顺序

（1）第一阶段是 Security 阶段（SEC）

SEC 阶段是上电后执行的第一个阶段，因此是整个系统可信的根源。SEC 阶段最重要的就是要确保处理器开始执行的代码是安全可信的。在这一阶段利用处理器产生一些临时内存，这些内存可以作为系统早期使用的临时内存，也可能用于静态 RAM。TPM 芯片的初始化工作也在这个阶段完成。

（2）第二阶段是 Pre EFI initialization 阶段(PEI)

PEI 阶段主要目的是初始化，首先，发现系统的启动资源，PEI 阶段代码只是做了部分工作来发现和初始化内存，但至少完成得到系统启动路径及后续需要的固件卷和系统 RAM，另外一些组件及芯片组的初始化工作都在下个阶段（DXE）来完成的。PEI 阶段还提供了一套标准方法来加载和调用处理器、芯片组和系统主板的初始配置程序。PEI 到 DXE 是一个单向的过程，因为只要进入到 DXE 阶段后，PEI 阶段的代码就不可用。

（3）第三阶段是 Driver Execution Environment 阶段（DXE）

DXE 阶段是框架中最为重要的阶段，系统大部分初始化的工作都要在该阶段完成。DXE 包含 DXE 基础框架、DXE 调度程序和 DXE 驱动程序组件。DXE 基础是一个引导服务映像，产生一系列的启动服务、运行时服务和 DXE 服务，是 DXE 阶段最开始执行的组件，为后面程序执行准备好引导环境，同时在设计上具有完全的可移植性，与特定的处理器、芯片组或者平台无关。DXE 调度程序主要是用来正确地发现和运行 DXE 驱动程序。DXE 驱动程序初始化处理器、芯片和平台有关的组件，同时还要为控制台设备、系统服务及启动设备提供软件抽象。DXE 的运行结果是生成一套完整的 UEFI 接口。

（4）第四阶段是 Boot Device Select 阶段（BDS）

这一阶段是紧随 DXE 阶段，主要和 DXE 阶段共同来实现控制台功能，使得启动设备输入/输出功能正常工作。提供给用户一个 UI 界面，用户可以对启动项进行选择，以此来决定由哪个启动设备来掌控系统控制权。

（5）第五阶段是 Transient System Load 阶段(TSL)

TSL 阶段是 BDS 的后续阶段，完全根据 BDS 阶段的选择进行相应的操作。

(6) 第六阶段是 Run Time 阶段 (RT)

当操作系统开始运行后,即是 RT 阶段。这个阶段除了运行时服务还在内存中工作外,其余的都被销毁了,同时之前分配的内存也被回收了。

(7) 第七阶段是 After Life 阶段(AL)

这是完结阶段。系统将控制权返还给平台固件。操作系统通过睡眠或者调用 UEFI 启动时服务进入到 AL 阶段。

1.3.3 可信计算模块

TCG 定义了具有加密和存储功能的 TPM(Trusted Platform Module)^[24],即可信平台模块。TPM 芯片是一个可以抵制非法用户篡改内部数据的独立计算引擎^[25],包括微处理器单元、存储器单元、密钥管理单元^[26]以及 I/O 输入输出单元,为整个计算机提供了加密和安全验证的服务,是整个可信计算系统的基础。主要有对称/非对称加密,安全存储,安全签名和完整性度量几项功能。

1) 对称/非对称加密

对称加密算法是应用较早的加密算法,发送数据的一方对加密密钥及明文进行加密算法的处理,得到复杂的加密密文传送,收信方需要使用加密过程中使用的密钥和相同算法的逆运算进行解密。主要有 DES、3DES、IDEA 等算法,算法是公开的,并且计算量较小,加密速度快,加密效率高,但由于交易双方都使用同样钥匙,安全性得不到保证。

非对称加密算法^[27],有 RSA、DSA 和 DH 算法,其中 RSA 是目前使用最为普遍的公钥加密算法。它的原理基于数论,将两个大素数相乘十分容易,但是要想对其乘积进行因式分解却极其困难。在 UEFI 密码系统中 RSA 算法^[28]中使用中国余数定理 CRT(Chinese Remainder Theorem),生成的 RSA 私钥由五个元素组成, $p, q, dP, dQ, qInv$ 。

具体步骤如下:

(1) e 的值为 65537

(2) 随机生成大素数 p ,直到 $GCD(e, p-1)=1$, $GCD(A, B)$ 表示 A, B 取最大公约数

(3) 随机生成不同于 p 的大素数 q ,同样要求 $GCD(e, q-1)=1$

- (4) 接着计算 $pq=n$, $\phi(n)=(p-1)(q-1)$
- (5) 计算出 d , $de \equiv 1 \pmod{p-1}$
- (6) 计算 dP , $dP = d \pmod{p-1}$
- (7) 计算 dQ , $dQ = d \pmod{q-1}$
- (8) 计算 $qInv$, $qInv = q^{-1} \pmod{p}$
- (9) 将 (n,e) 作为 RSA 公钥
- (10) 将 $(n,e,p,q,dP,dQ,qInv,d)$ 作为 RSA 私钥

2) 安全存储

可信 BIOS 系统可以提供保护存储的功能,该功能模块被称为存储可信源(RTS),它用来保护进入 TPM 的密钥和数据。RTS 管理的内存区域主要是用来存储当前 TPM 使用的密钥。由于 TPM 内部的存储空间有限,不活动的密钥就会被替换出 TPM,这些被替换出的密钥通过存储根密钥 Storage Root Key (SRK) 的密钥保护,同时 SRK 还负责保护 TPM 所保护的数据。

3) 安全签名

密钥被保存在硬件之中,签名则是在外部无法访问的硬件私密存储区域进行,因此任意签名都和存放密钥的硬件相关,另外口令会被存放在对应的口令块中,同时封装起来^[29]。如果有人来破坏时,操作系统会接收来自 TPM 的通知。根据不同的需要,可以用不同的密钥来登录不同的网站,TPM 可以产生无限多的密钥供用户使用。

4) 完整性度量

平台完整性^[30]是可信平台的一个重要特性,在 TPM 中平台配置寄存器(Platform Configuration Register, PCR)是保证平台完整性的基础。每个 PCR 都是一个 160 位的存储空间,对于长度小于 2^{64} 位的消息,安全散列算法 SHA-1 会产生一个消息摘要,刚好存储在 PCR 中。要使得 PCR 中能够存储更多的测量信息,就有了 PCR 的扩展操作,即 PCR 新的值是建立在原有值和当前要加入的新值的基础上。TPM 芯片中存在 24 个 PCR,都用于存储特定的散列值。我们通过观察 EFI 日志中的值,利用散列值的唯一性,可以检测测量实体是否改变。

1.3.4 BitLocker 驱动器

BitLocker 通过加密 Windows 操作系统卷上存储的所有数据可以更好地保护计算机中的数据^[31]，是 Windows Server 2008 及以上系统可选的数据保护功能，使用 128 位或者 256 位高级加密标准（Advanced Encryption Standard, AES）进行全卷加密^[32]。在计算机的日常使用中用户来说是完全透明的。BitLocker 主要有两种工作模式，TPM 模式^[33]和 U 盘模式，也可以同时启用这两种模式。如果要使用 TPM 模式，计算机中就需要具有不低于 TPM1.2 版的 TPM 芯片。如果要使用 U 盘模式，需要一个专用保存密钥文件的 U 盘，在重启系统的时候必须在开机前将 U 盘连接到计算机上。BitLocker 工作在磁盘底层，加密范围非常广泛。

1.3.5 工业标准测试

对 BIOS 测试的日常软件测试方法^[34]按照范围分为三种：冒烟测试、稳定测试、扩展测试，这里的测试工作大多都需要手动进行，工业标准的测试^[35]是手动测试中最重要的测试。符合 UEFI 的 BIOS 必须能够支持各种工业的驱动程序，同时由于不同厂商生产的驱动程序标准不同，就需要对这些产品进行工业测试，看其是否符合工业标准。

（1）冒烟测试

它是三者中范围最小的测试，也是确定和修复软件缺陷最经济有效的方法。在 TIANO 项目中，每天都要利用冒烟测试^[36]方法对不同版本 BIOS 进行测试，检查其最基本的功能是否正常并发布冒烟测试报告。主要运用的命令^[37]包括 `reconnect -r`（卸载设备驱动，再装载设备驱动到设备），`map -r`（检查是否装载了所有的设备），进入安卓系统、小红帽系统、WINDOWS 系统等后，进行 S3\ S4 唤醒操作。冒烟测试是检测一天的修改是否影响到系统的正常运行。

（2）稳定测试

相对于冒烟测试范围较大的测试，一般的功能测试都能覆盖到，在 TIANO 项目中，包括是否能够正常进入到系统，显卡、声卡、硬盘、USB 等设备是否能够识别，shell 是否能够正常进入退出。

（3）扩展测试

扩展测试是三者中范围最大的测试，所有 PCI-E 和 PCI 显卡、声卡、SCSI 卡能否正常工作，IDE、SATA 接口硬盘内存是否可以被正确读取。

1.4 研究内容和论文组织结构

针对目前计算机普遍面对的安全问题，文章介绍了基于 UEFI BIOS 平台上引入可信计算的思想，以英特尔 TIANO 项目为背景，使用英特尔硬件软件及测试工具，对可信 BIOS 平台的安全性能进行了测试，本文主要研究了：

（1）UEFI 平台架构

UEFI 使用模块化设计，在逻辑上分为硬件控制和操作系统软件管理两部分，启动分为七个部分，其中 PEI 和 DXE 是最主要阶段。

（2）可信计算模块相关技术

可信计算广泛运用于提高系统整体的安全性，本文主要研究了可信计算模块 TPM 的实现原理及部分加密算法。

（3）基于 UEFI 框架下测试环境搭建及其实现原理

要对 BIOS 可信性进行测试，首先要搭建测试环境，包括硬件和软件两方面。同时针对不同测试项，也有较大差异，于是针对 BIOS 需要实现保护数据、测量完整性等功能又将测试环境分为网络和无网络两种状态进行实现，并结合 UEFI 框架分析了实现的原理。

（4）根据可信计算模块相关技术原理设计测试用例

根据可信 BIOS 的功能，从主要硬件 TPM 芯片，可信算法的运用方面将验证过程实例化，设计并实现有效的测试用例。

（5）执行测试用例，分析测试结果

依据测试用例的步骤在不同平台、不同操作系统、不同版本 BIOS 的组合下执行用例。并设计测试表格记录测试结果，对测试结果进行分析。

下面是论文章节的安排：

第一章是绪论。首先介绍了可信 BIOS 系统测试的研究意义，接着描述了国内外研究情况，然后介绍了本文涉及的一些关键技术，最后给出了文章的整体章节安排。

第二章是可信 BIOS 系统测试的需求分析。首先从整体上对可信 BIOS 系统测试的结构进行了分析，接着对该系统 BIOS 的安全方案进行了分析，然后根据总体结构和方案提出了测试目标，最后给出了规划的测试流程和章末总结。

第三章是可信 BIOS 系统测试的设计。首先，给出了测试系统的设计机制，接着对测试中需要的工具进行设计，然后是对具体测试用例的设计，最后是本章小结。

第四章是可信 BIOS 系统测试的实现。首先是测试平台硬件环境的搭建，接着是软件环境的搭建，然后是具体测试用例的实现，最后是对本章的小结。

第五章是测试结果及分析。首先是对测试结果的阐述，然后根据测试结果从多方面进行了分析，最后给出了本章小结。

第六章是对本文工作的总结，同时也提出了需要改善和进一步提高的地方。

最后是对研究生阶段指导、帮助、关心我的人的感谢以及文章参考的文献。

2 可信 BIOS 系统测试方案的需求分析

测试系统的需求是整个测试过程的基础，在这一章中，首先依据 BIOS 的核心机制对可信 BIOS 系统测试的总体结构进行了介绍；然后对可信 BIOS 系统测试需要实现功能的技术方案进行阐述；接着，提出了可信 BIOS 系统测试要达到的目标，最后根据目标给出了可信 BIOS 系统的测试流程。

2.1 可信 BIOS 系统测试的总体结构分析

BIOS 是计算机开机上电启动的第一个步骤，因此只有保证 BIOS 的可信性，才能保证整个计算机系统的安全正常运转。在计算机启动中，利用可信根为起点建立可信链，保证 BIOS 的完整性、有效性和保密性，从而实现安全有效的引导启动过程。基于 UEFI 的可信 BIOS 系统平台结构如图 2.1。

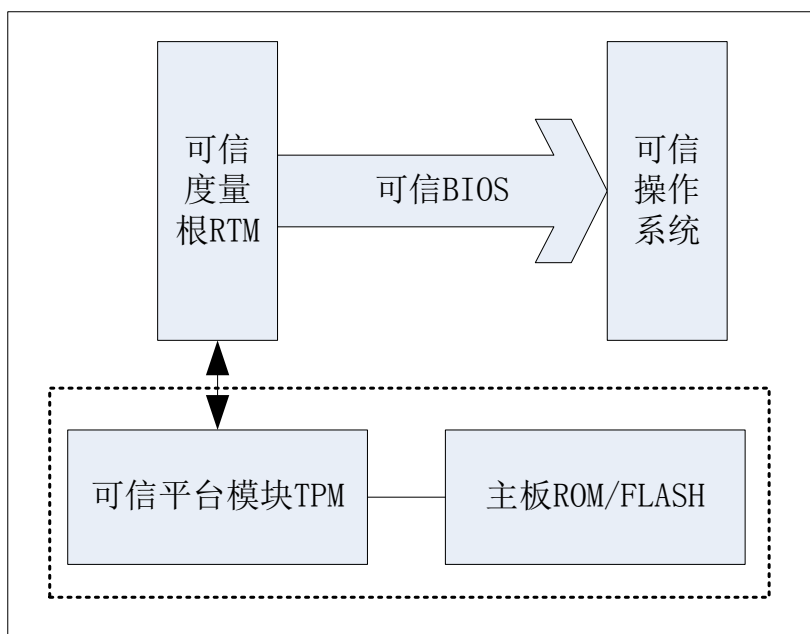
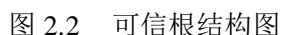


图 2.1 基于 UEFI 的可信 BIOS 平台结构图

一般来说，平台上的硬件设备实例化是一个被动的过程，TPM 芯片同样如此，它作为可信存储根（RTS）和可信报告根（RTR）作用于平台，RTS 用来存储根密钥和 PCR 寄存器，RTS、RTR 与可信度量根（Root of Trust for Measurement, RTM）一



TCG 通过 UEFI 设计说明、保护配置文件、一致性检验等不仅描述了对 TPM 的需求同时也对绑定在平台上的相关组件进行了说明，主要是为了能够建立一个可信构建块 (Trusted Building Block, TBB)。TBB 包括构成平台的组件，可以是 TPM。TPM 如何绑定到平台、系统固件闪存和部分必须被信任的固件，同时它还起到引导到物理平台的指示作用。平台上的主动代理是 RTM，它有动态和静态两种。静态核心可信度量根 (S-CRTM) 是平台固件必须信任的部分，通过 TCG 秘密模块的规定，S-CRTM 有促使可信计算开始，启动 TPM，检测平台物理状态的功能。S-CRTM 是 TBB 的一部分，它贯穿在平台固件和其他平台可信任根之间。在随后的阶段中，S-CRTM、核心可信度量根和静态可信度量根 (Static Root of Trust for Measurement, S-RTM) 是可以交换使用的。动态度量可信根 (Dynamic Root of Trust for Measurement, D-RTM) 支持测量执行环境 (measured launch environment, MLE)。它的启动需要一个可信安全事件来触发，引导更小的任务控制部件 (Task Control Block, TBC)，减少表面的攻击从而提高系统的安全性能。S-RTM 和 D-RTM 功能可以在同一个平台上共存，也可以单独存在发挥作用。相对比 D-RTM，S-RTM 被更多的组件触发，因此本文的讨论主要基于 S-RTM。图 2.3 是可信 BIOS 平台启动流程，它建立在 S-RTM 的基础上。

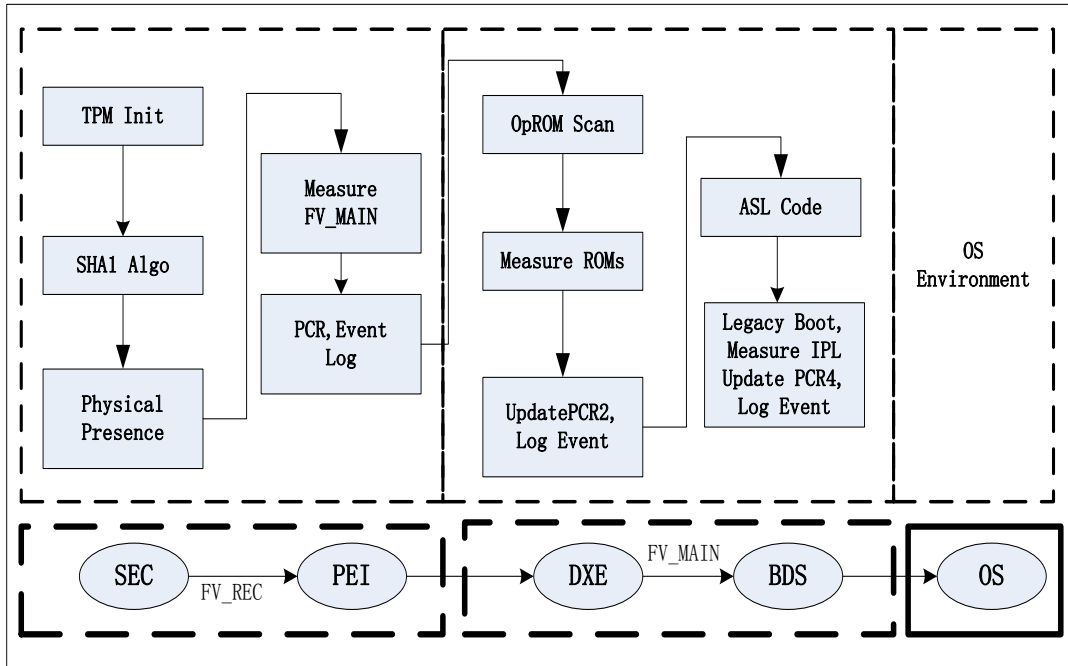


图 2.3 基于 S-RTM 的可信 BIOS 平台启动流程

在可信度量中最重要的是将可执行代码和数据散列，通过度量程序记录到 TPM 芯片的 PCR 寄存器中。这些 PCR 寄存器是只写的，并且在平台重启的时候会被擦除（至少对于 S-RTM 的静态 PCR 寄存器）。PCR 寄存器在加密和解密中都会被使用。这实际上是指如果平台在安装和连续调用软件后重启发生未被授权的变化，PCR 寄存器就会发生改变，从而达到检测系统是否被篡改的目的。

2.2 可信 BIOS 系统的安全方案分析

结合 UEFI 可信 BIOS 系统平台的可信根作用及启动过程中各流程可能受到的威胁，为平台提出可执行验证，驱动签名，用户身份验证，网络验证和网络安全的技术方案。

1) UEFI 首先要提及的功能是驱动签名或可执行验证，驱动签名的功能如下：

(1) 扩展可以被 UEFI 识别的多种形式的签名，有 SHA-1，SHA-256，RSA2048 /SHA-1，RSA2048/SHA-256 和认证码；

(2) 有配置或好或坏签名数据库的标准方法；

(3) 当执行动作被拒绝为列表提供基于方法的升级时，驱动签名提供了标准的行为规范。

UEFI 能够提供验证的功能，S-RTM 记录平台上代码的状态和值，以便之后的实体可以对度量进行评估。对于一些策略的验证或者实施，当引导程序变成策略的一部分时，UEFI 固件能够起到可信根实施（Root of Trust for Enforcement, RTE）或者可信根验证（Root of Trust for Verification, RTV）的作用。例如，这个策略可以包含用已经经过认证码签名的 UEFI 图像对 UEFI 图像进行验证。图 2.4 是 UEFI 图像签名的必要步骤，签名包括 RSA 非对称加密和安全哈希算法中的一个哈希函数。

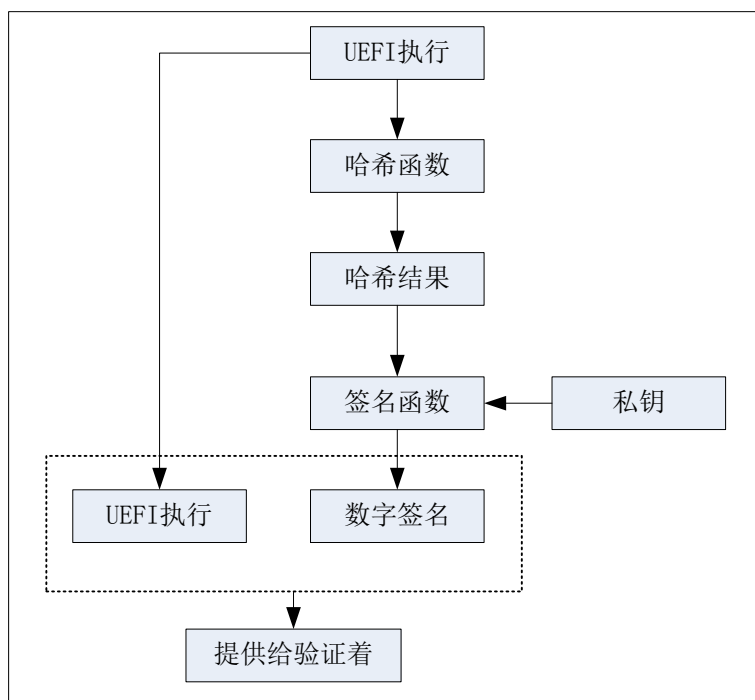


图 2.4 驱动签名的过程

这些工作大多由厂商或者第三方设备来完成，例如 VeriSign 这样的认证机构。只要签名图像被配置好了，不论是从网络端、总线适配卡还是 UEFI 系统分区加载，UEFI 固件都会验证图像的完整性，图 2.5 是验证 UEFI 图像的过程。

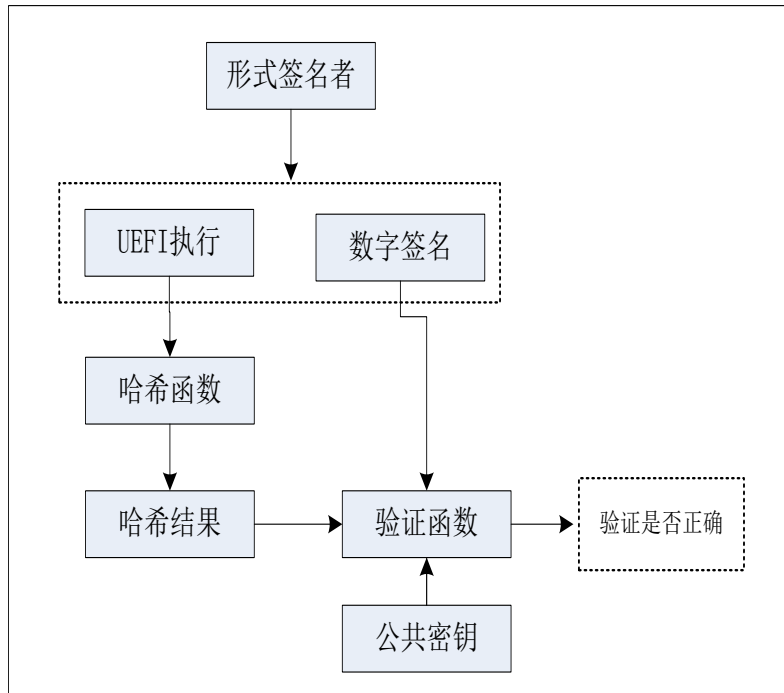


图 2.5 UEFI 图像验证过程

结合强健的 UEFI 工具和可协作的可信基础设施，可以让 UEFI 在一个安全，强健的环境中逐步增强可扩展性。

2) 在网络安全方面也提出了可用的技术方案，包括网际协议安全^[38]（Internet Protocol Security, IPsec）。

像 TPM 芯片这样的硬件可以用来存储 IPsec 证书，但是为了更加安全，确保 IPsec 密码方法和网络代码在 UEFI 固件上完整使用，需要按照前面的方法来配置平台。IPsec 可以用于网络平台加强类似基于 Internet 小型计算机系统接口(Internet Small Computer System interface, SCSI)供应的引导，平台上的 IPsec 允许基于 IPV4 和 IPV6 的 iSCSI 的引导和供应操作。iSCSI 是基于 TCP/IP 的协议，主要是用来建立和管理 IP 存储设备、主机和客户机之间的连接，并创建存储区域网络，使得数据的高速传输成为可能。对于基于 IP 实现隔离的 iSCSI 网络，在 iSCSI 与 LAN 之间存在物理连接的情况下，应该部署更加严密的 CHAP 身份验证方式，这样才能消除外部影响，总体上才能加强存储管理接口的安全，图 2.6 是 iSCSI 位于 UEFI 网络协议栈顶部图。

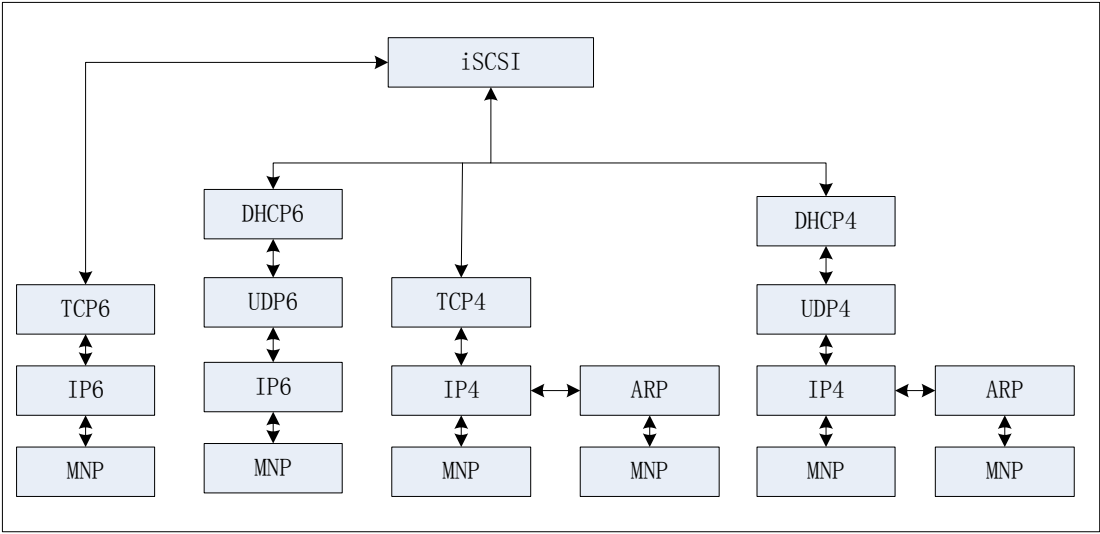


图 2.6 SCSi 在 UEFI 网络协议栈的应用

其中 MNP 是微通信网路连结协定（Microcom Networking Protocol），ARP 是地址解析协议（Address Resolution Protocol），UDP 是用户数据报协议（User Datagram Protocol），DHCP 是动态主机配置协议（Dynamic host configuration protocol）。

3) UEFI 用户身份验证（User identification, UID）

从主要供应商到抽象的用户验证过程中，允许驱动的加载需要一个基本的框架设施，包括很多因素和策略方案来确定用户是否有这样的权限来获得操作系统的控制权，同时也可能对于一些用户来说是有限制性的。标准固件中提供给用户验证的设备有智能卡、智能令牌或者指纹传感器。图 2.7 是用户身份验证的流程图。

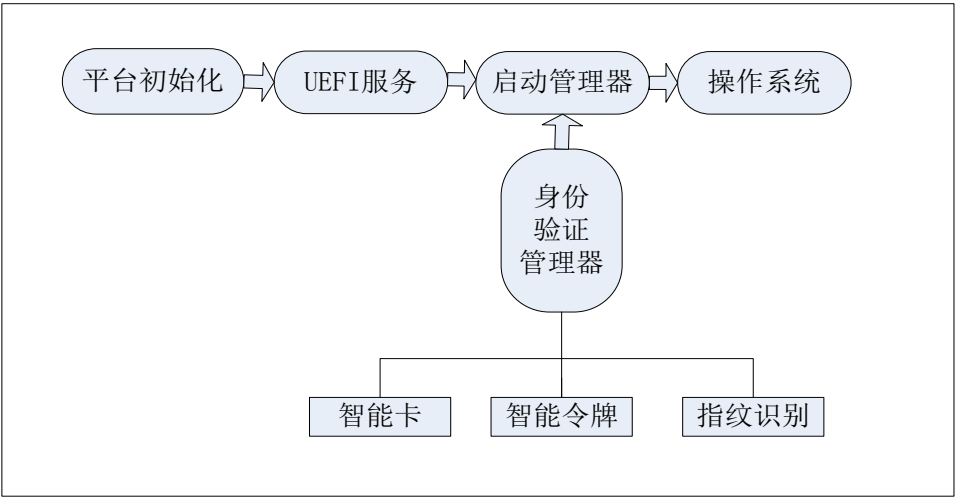


图 2.7 用户身份验证的流程图

2.3 可信 BIOS 系统的测试目标

可信 BIOS 从平台开机上电到将控制权交由操作系统都起到保护的作用，基于 UEFI 的可信 BIOS 需要实现保护数据，验证身份，完整性测量以及存储报告功能，因此我们总体目标是根据在平台启动的各个阶段，实现对这些功能的测试。

最开始的 SEC 是 UEFI 系统开机上电的初始化阶段，是整个系统可信的基础。TPM 芯片在这个阶段完成初始化的工作，因此可以为系统提供保护数据的功能。

在测试平台中，我们利用外插 TPM1.2 及 TPM2.0 芯片验证启动安全性。

(1) 在 SEC 阶段代码执行完后，系统将会进行完整性校验的工作，如果系统固件卷被检查没有被篡改过，即是完整的，那么系统进入到 PEI 阶段。在 PEI 阶段，会对驱动进行完整性检查，如果通过再对执行文件进行完整性校验，并且将得到的摘要值存储在 TPM 芯片的 PCR 寄存器中。

(2) 接下来是 DXE 阶段，这是系统的核心步骤，提供了引导服务和执行时服务。这个阶段要对大多数的驱动进行校验，并且还要对板载的 PCI 插槽的内存进行加载。DXE 驱动程序被执行的顺序由可选的依赖顺序文件和与这些驱动程序相关的依赖表达式的组合来决定，该阶段同样要实现完整性校验工作。

(3) DXE 提供 BDS 架构协议，为 BDS 阶段提供了一个入口函数，当 DXE 驱动程序调度完成后，进入 BDS，使得启动操作系统的步骤能够正常进行。在测试中，利用 TPM 芯片对系统分别进行网络和无网络情况下 BitLocker 加密，重启系统时，验证用户身份。

2.4 可信 BIOS 系统的测试流程

本文中涉及到的测试主要包括无网络状态下的 BitLocker 功能验证、EFI 文件完整性验证和网络状态下 WHCK 环境下可信 BIOS 功能验证、BitLocker NKP 功能验证。这与我们上一节提出的总体目标相一致，可信 BIOS 需要实现保护数据，验证身份，完整性测量以及存储报告功能。不论是网络还是无网络状态下的 BitLocker 功能验证都是对可信 BIOS 实现数据保护，身份验证的检验，EFI 文件完整性验证和 WHCK 环境下可信 BIOS 功能验证实现了完整性测量和存储报告功能。这四项验证不仅在系

统不同阶段分别验证可信 BIOS 的某些功能，同时还相互作用，对可信 BIOS 的性能做了更完善的检验。

我们要实现具体的测试，就要先梳理清楚测试的流程设计出具体测试用例，以用于提高测试效率。项目的测试流程大体分为以下五个步骤：

(1) 阅读需求说明（关于 UEFI 有官方发布的 Spec），根据 TIANO 项目需求定制测试计划

(2) 根据测试计划、功能点划分，设计合理的测试用例

(3) 按照设计的测试用例步骤严格执行测试用例，在执行前要确保搭建的测试环境是符合需求的

(4) 记录测试结果，并针对结果不符合预期或者测试中出现异常的情况，通过二分法、排除法、组合法等多种途径检验 BUG 出现的充要条件，并提交 BUG

(5) 测试完成后，编写测试报告，如果是回归测试，还要将多次测试结果进行对比，找出其中异同点，并对提交的 BUG 进行追踪

在测试过程中，使用到各种软件测试方法^[39]，在可信 BIOS 验证中使用到的有黑盒测试、自动化测试、冒烟测试、本地化测试、回归测试^[40]等。有了整体的测试流程和测试方法，测试就变得有章可循。在实际项目进行中，由于研发不断对 BIOS 的功能进行维护和更新，因此要对不同 BIOS 版本进行很多类似项目的测试，用来验证新功能正确性的同时，要验证修改部分是否会影响到其它部分的功能。为了辅助测试，设计测试工具能够有效提高测试效率。

2.5 本章小结

本章主要介绍了可信 BIOS 的整体框架及其关键流程的技术实现，基于可信 BIOS 相对比一般 BIOS 的扩展性，从关键硬件 TPM 芯片、软件的实现和可信算法的运用几个方面梳理出了系统启动流程中各阶段安全性的方案。从而归纳出基于 UEFI 的可信 BIOS 整体系统测试的目标和流程，相关测试项的设计将在下一章节具体介绍。

3 可信 BIOS 系统测试方案的设计

本章首先介绍了测试系统的实现机制，然后介绍了针对本文测试用例中使用的测试工具进行了介绍，最后针对可信 BIOS 实现的功能，设计并实现了无网络状态下的 BitLocker 功能验证、EFI 文件完整性验证和网络状态下 WHCK 环境下可信 BIOS 功能验证、BitLocker NKP 功能验证测试用例。

3.1 测试方案的总体设计

3.1.1 BitLocker NKP 测试框架的设计

BitLocker NKP 是指将网络解锁作为 BitLocker 保护器的一种操作。当连接到网络的操作系统重启时为系统提供自动解锁的操作，这种网络解锁更便于驱动器管理域环境下的客户端和服务端。该功能的实现需要 DHCP 为其自动分配 IP 地址，WDS 服务器为其增加私有网络解锁证书等。一般 WDS 服务器须是 AD DS 中一员或者是 AD DS 域的域控制器，但如果 WDS 服务器在独立的模式下配置 AD DS 就不是必须的。由于 WDS 使用 PXE 部署服务，依赖于 DHCP 进行 IP 寻址，因此在网络上起作用的 DHCP 是必不可少的。域名称系统 (Domain Name System, DNS) 在 WDS 运行前一定要在网络下开启工作。同时，运行的 WDS 需要 NT 文件系统 (NTFS) 来存储 Image。

当 Windows 启动管理器检测到有网络解锁保护器存在的时候，解锁步骤就从客户端测试机开始了。它利用 UEFI 中 DHCP 驱动从 IPv4 中获取 IP 地址，然后广播一个包含网络密钥和会话密钥的由特定供应商提供的 DHCP 请求用来寻求答复，所有由服务器解锁证书的加密过程都如此。在 WDS 服务器上的网络解密提供者识别出这是来自于特定供应商的请求，使用 RSA 私钥进行解密，然后通过特定供应商提供的 DHCP 的答复返回会话时网络加密密钥。在服务器端，WDS 服务器有一个可选的插件，例如 PXE 提供者，可以处理传入的网络解锁请求。提供者也可以配置子网限制，这将要求在网络解锁提出请求的客户端提供的 IP 地址是要属于一个被允许的子网，

这样的限制主要是确保网络密钥能够释放给客户端。在网络解锁提供者不适用的例子中，BitLocker 将不会为下一个可用保护器解锁驱动。在一个典型的配置中，这就意味着标准的 TPM+PIN 解锁屏幕会处于一个解锁驱动的状态。服务器端开启网络解锁的配置需要由一对 2048 字节的 RSA 私钥和公钥形成的 X.509 证书和发送到客户端的公钥证书。该证书必须由 Windows Server 2012 及以上水平的域控制器上的组策略编辑器来管理和部署。该证书是加密网络密钥的中间层公钥，这里网络密钥是解锁驱动两个密钥中的一个，另一个密钥被保存在 TPM 中。图 3.1 是 BitLocker 网络解锁的过程。

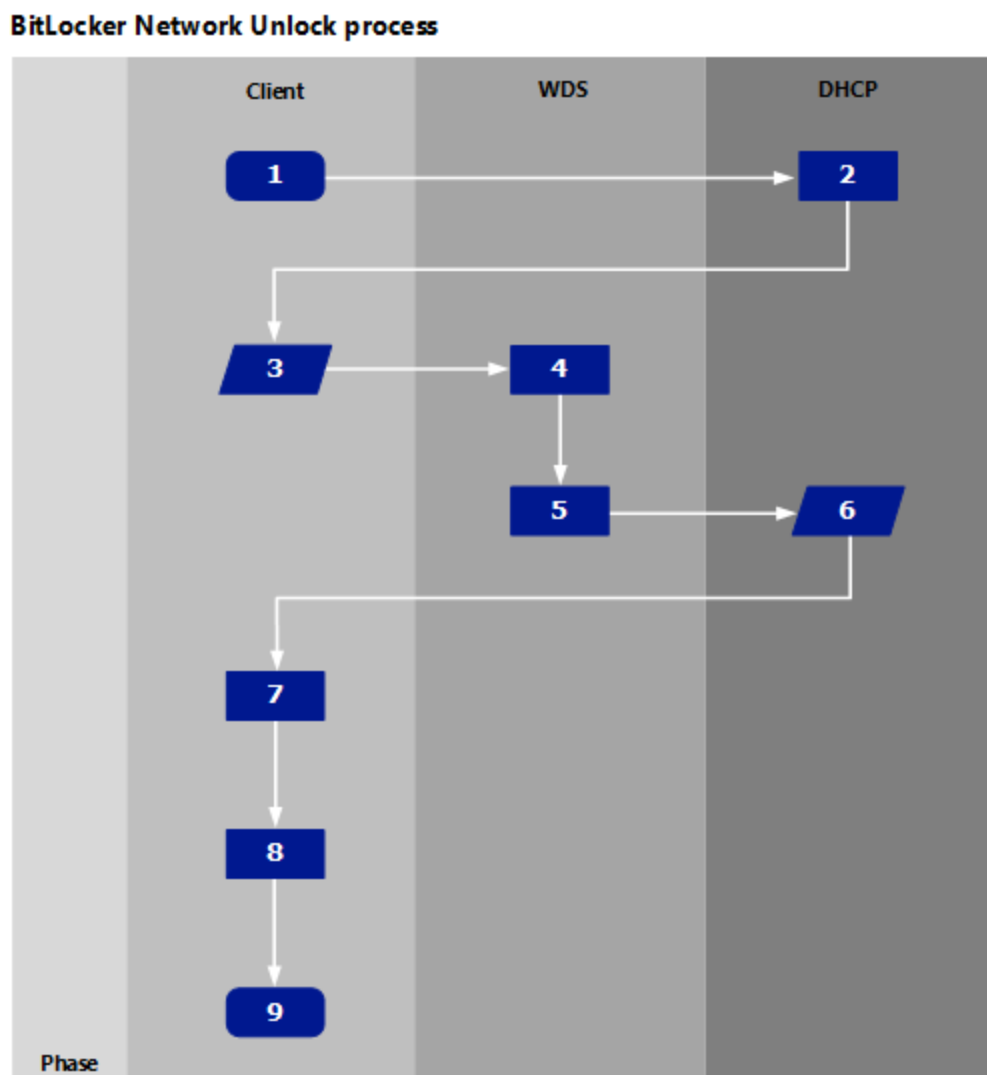


图 3.1 BitLocker 网络解锁的过程

- (1) Windows 启动管理器发现在 BitLocker 配置中存在网络解锁保护器
- (2) 客户端运用 UEFI 中 DHCP 驱动获得一个可用的 Ipv4 P 地址
- (3) 客户端广播一个包含有网络密钥（256 字节的中间密钥）和会话密钥的 DHCP 请求用于寻求答复
- (4) 在 WDS 的网络解锁提供者识别到 DHCP 的请求
- (5) 提供者用 WDS 的 BitLocker 网络解锁证书 RSA 私钥进行解密
- (6) WDS 提供者给客户端返回利用会话密钥解密的网络密钥，这形成了一个中间层密钥。
- (7) 返回的中间层密钥包含了另外一个只能由 TPM 解密的 256 字节中间层密钥
- (8) 被包含的中间层密钥被用于创建一个 256 字节的 AES 密钥来解锁卷
- (9) Windows 按顺序继续引导

3.1.2 WHCK 测试框架的设计

WHCK 是微软商标验证工具(Windows Logo Kit, WLK)的更新下一个版本，WHCK 测试是基于功能（Feature）检测，不同于以往的工具套件(Kits)，这些工具可以决定你设备哪一部分能够被验证通过。

这里功能是指设备公开的微软性能，当将设备连接到 HCK 环境时，Kit 使用聚合（Gathers）机制在设备上搜索功能，从 Windows8 操作系统开始，Features 被命名空间的形式组织，例如 Device.Graphics.WDDM12，System.Client.BluetoothController.Base 以及 Filter.Driver.Network.LWF。功能需要做什么才能获得微软硬件验证都被定义在正式规范的需求文档中，同样从 Windows8 操作系统开始，需求文档也开始使用命名空间规范，例如 Device.Imaging.Scanner.Base.RawFileFormat 就是 Device.Imaging.Scanner.Base 功能的需求文档。而在设备上完成验证功能的测试工作都是基于需求文档的，每一个测试都有一个指示对应在需求文档中。每个产品包含了预定义的可测试功能，这种产品形式代替了以前在 WLK 中自我选择系统范畴的形式，为了操作系统硬件验证的实现，产品至少要以一个产品形式来完成所有的功能。在测试之前，要确保测试环境满足测试需求，上一小节我们已经给出了搭建 WHCK 测试环境的步

骤，主要包括两个组件，一个是服务器另一个是客户端测试机。

（1）对于服务器而言

我们通常是指其控制器（Controller）。一个测试服务器包含两个部分：HCK Controller 和 HCK Studio。HCK Controller 是管理运行在测试机器上的测试项的引擎。HCK Studio 是管理工具，它能够让你选择安排任何连接在测试服务器上的客户端测试机中的测试项。Controller 和 Studio 都是通过 WHCK 安装源进行安装的。一旦安装完成，测试服务器就包含单独的安装程序，可以用来远程安装 HCK Studio 以及 HCK 客户端。

（2）对于客户端测试机而言

每个不同客户端测试机可以有不同的配置适合多样化的测试场景，不同配置包括硬件、操作系统、补丁包或者驱动等。每个客户端测试机都只能关联一台测试服务器。用户可以直接通过测试服务器的网络共享运行 HCK 客户端安装包来配置客户端测试机。

WHCK 的部署有两种场景：

（1）域（Domain-joined）环境

域环境是指当前环境中存在域控制器，所有要安装 WHCK 组件的计算机都会被加入到域控制器。如果你准备在域环境下部署 WHCK，你至少需要三台计算机：分别用作微软域控制器，WHCK 测试服务器及 WHCK 客户端测试机。确保活跃的目录是被正确配置的并且是运行在域控制器上。

（2）工作组（Workgroup）环境

与域环境相对，工作组环境是没有域控制器的一种环境状态，如果准备在这种环境下部署 WHCK，至少需要两台计算机，分别用作测试服务器和客户端测试机。本文中搭建的 WHCK 测试环境即基于这种工作组环境，此时不能够使用默认的管理员账户。

除此之外，考虑到要怎样组织资源来能充分利用 WHCK，就要合理制定控制器的数目以及连接到控制器上客户端测试机的数量，制定的机器数量与要进行验证的设备或者系统也有一定关联。如果想要减少管理控制器和客户机的开销，可以选择

分配更少的控制器，让尽可能多的客户机连接在这些控制器上，一台控制器最多能够连接 150 台客户机。当然如果可以分配更多的控制器，并且让更少的客户机连接在这些控制器上，这样由于较少的客户机与其交互，就能让控制器响应更快。

3.2 测试工具的设计

由于可信 BIOS 的实现大多都是基于 TPM 芯片，UEFI 系统启动的时候会加载文件，系统安全保护会对这些文件进行散列运算，同时将这些散列值通过 TPM_Extend 命令写入到 TPM 芯片中的 PCR 寄存中里，并记录相关的日志。PCR 寄存器可以存储 160 字节散列值的，用于验证完整性测量。在一个平台上有大量完整性指标会被测量，一些特别的完整性指标会随着时间发生变化，它们产生的新值需要被保存下来。对于完整性测量方法的来源很难验证，所以有一些完整性指标数值因此会被拒绝重写已经存在的值。因此，如果完整性指标数值被单独存储起来，它的更新同样也必须被独立存储起来，但是这项工作很难通过给内存一个上限值让其完成完整性数值的存储。PCR 寄存器被设计成用于存储任何数量的测量值，主要利用哈希散列和散列更新，伪码为：

$$\text{PCRi New} = \text{HASH} (\text{PCRi Old value} \parallel \text{value to add})$$

关于 PCR 寄存器构造有两个特别的哈希散列特性需要提及，一个是顺序，PCR 寄存器中更新的值位置是不可交换的，例如，先测量 A 后测量 B 与先测量 B 后测量 A 是不相同的；另一个特性是不可逆性，根据攻击者给定的 PCR 寄存器的输入消息确定计算是不可行的，另外，更新 PCR 寄存器的值必须是基于之前 PCR 寄存器的值或者最近一次重启时所有由 PCR 寄存器提供的输入信息。

在 TPM 加电自检完成前，PCR 寄存器以及它的保护功能都不会起作用，一旦 TPM 加电自检失败，TPM_Extend 操作也会失败，TPM_Quote 和 TPM_Seal 操作会分别报错，PCR 寄存器内容的验证也必然是失败的

由于 PCR 寄存器的值可以用来验证 UEFI 文件的完整性，那么我们需要一个工具来查看 PCR 寄存器的值。TpmTest.efi 是根据 PCR 寄存器工作原理设计出来的一款运行在 UEFI Shell 环境下的测试工具，它可以用来输出 TPM 芯片中 PCR 寄存器

的值以及系统中记录事件的日志。首先我们要进入 UEFI Shell 环境，找到存放 TpmTest.efi 的文件目录，执行 TpmTest.efi 后就进入到 TPM 的交互界面，然后在该界面执行 Pcrread n (0-7) 命令显示 PCR n 寄存器中存储的散列值，如图 3.2 所示。

```
fs0:\> TpmTest.efi
TPM Test Shell Utility (Version 0.10) ...

TPM> pcrread 0

Ordinal[0021] : PcrRead
ReturnCode[0000] : (TPM_SUCCESS) Successful completion of the operation
==> PCR[00] = 4F6F9455CF544F1A35FDBA04792953430227FDA0

TPM> pcrread 1

Ordinal[0021] : PcrRead
ReturnCode[0000] : (TPM_SUCCESS) Successful completion of the operation
==> PCR[01] = 1DD4E39621A60E5BEDDD0EBBAC47DC69F35F50A

TPM> pcrread 2

Ordinal[0021] : PcrRead
ReturnCode[0000] : (TPM_SUCCESS) Successful completion of the operation
==> PCR[02] = B2A03B0EBF2F8374299A5B2BDFC31EA955AD7236

TPM> pcrread 3
```

图 3.2 运用 Pcrread n 命令查看寄存器散列值

最后使用 GetEvent 命令可以查看系统中完整性测量的日志，如图 3.3。

```
TPM> getevent
==> Type = 0x8; PCR = 0; EventSize = 44;
Digest = 7F3023412B71C8F20263A2DF0B20A6602A29CFED
==> Type = 0x80000008; PCR = 0; EventSize = 16;
Digest = 4350EE59545F0647A091A3738DF2D11B8BF28A46
==> Type = 0x80000008; PCR = 0; EventSize = 16;
Digest = 51D44E08CC9CC9FDB9779C66A196366716857130
==> Type = 0x80000008; PCR = 0; EventSize = 16;
Digest = 15DBA1A641DC2720BF83B83C786AD6D06C48170A
==> Type = 0x80000008; PCR = 0; EventSize = 16;
Digest = DFA57C5104395FCE81DBB987E04CD5ED4B0FEB62
==> Type = 0x80000008; PCR = 0; EventSize = 16;
Digest = 5B23A120FBA2EF73BDE3BDD0D9BB3DD303A7FBA9
==> Type = 0x1; PCR = 0; EventSize = 9;
Digest = 5D620E1437E75649C7717F4781E592119FA7D2D9
==> Type = 0x80000003; PCR = 4; EventSize = 76;
Digest = 569AAF774C51A9B83C1A148B1E047877B1743101
==> Type = 0x80000009; PCR = 1; EventSize = 32;
Digest = 3F628A72B6C6AB2D58F3099E11D0C678F4129891
==> Type = 0xB; PCR = 1; EventSize = 32;
Digest = 2D2846B47F1A94785AE44601B30CDCBA9C2994F3
```

图 3.3 使用 GetEvent 命令查看系统完整性测量日志

Pcrread n 和 GetEvent 命令都是在 TPM 处于开启并且激活状态才有效，图 3.4 和

3.5 是 TPM 分别处于未开启激活状态和未开启未激活状态，执行命令后无法正常获取寄存器值和日志的界面。

```
ifs0:\> TpmTest.efi
TPM Test Shell Utility (Version 0.10) ...

TPM>>pcrread 0

Ordinal[0021] : PcrRead
ReturnCode[0007] : (TPM_DISABLED) The TPM is disabled

TPM>>pcrread 1

Ordinal[0021] : PcrRead
ReturnCode[0007] : (TPM_DISABLED) The TPM is disabled

TPM>>pcrread 2

Ordinal[0021] : PcrRead
ReturnCode[0007] : (TPM_DISABLED) The TPM is disabled

TPM>>pcrread 3

Ordinal[0021] : PcrRead
ReturnCode[0007] : (TPM_DISABLED) The TPM is disabled
```

图 3.4 TPM 处于未开启激活状态

```
TPM>>pcrread 2

Ordinal[0021] : PcrRead
ReturnCode[0006] : (TPM_DEACTIVATED) The TPM is deactivated

TPM>>pcrread 3

Ordinal[0021] : PcrRead
ReturnCode[0006] : (TPM_DEACTIVATED) The TPM is deactivated

TPM>>pcrread 4

Ordinal[0021] : PcrRead
ReturnCode[0006] : (TPM_DEACTIVATED) The TPM is deactivated
```

图 3.5 TPM 处于未开启未激活状态

由于系统中环境发生变化，PCR 寄存器散列值和事件日志都会发生变化，因此

使用可以根据 UNSEAL 操作来检验平台环境的变化，由此判断这次启动是否有被篡改过，是否可信。

3.3 测试用例的设计

测试用例是指对特定的软件产品进行测试任务的描述，其目标可以是测试某个程序路径，也可以是某个特定的需求，体现测试方案、方法、技术和策略。测试用例的作用包括：1) 在项目工作量很大的情况下，可以把测试的工作外包给第三方人员；2) 作为一个项目的管理人员，同样可以根据测试用例的种类和数量来预估测试时间；3) 当一个新的测试人员加入到团队时，不知道测试要到什么程度，测试用例就是很好的参考；4) 在 BIOS 的测试中大多采用模块化的测试，测试人员就知道测试的程序是属于哪一部分，有助于对一些测试现象前后联系进行分析，增强测试逻辑性。测试用例的设计方法是相互配合的，落实到单独的测试项目会有不同的方法，而且不同类型的软件，因特点差异也会有不同的设计方法，在具体操作中我们要灵活使用各种不同的测试方法，一般先使用黑盒测试来设计基本的测试用例，再用白盒测试法完善测试用例。对于测试项熟练掌握的测试人员来说，可以利用场景法来引导测试流程，在测试中结合各种测试方法。对于基本事件的用例设计，首要做的就是按照需求说明，把有联系的功能、操作，用路径分析法来设计用例，对于相对独立的测试点就按照功能来做就好，由于是基本事件，其用例的覆盖率要求达到 100%。

一个完整的测试用例包含的内容应该根据具体的需求来设计，本文中测试用例包含如下内容：

- (1) 测试编号，这是用来唯一区别测试用例的，便于查找
- (2) 标题，用少量的词告诉测试者测试用例的主要内容
- (3) 归纳，测试的大体目标
- (4) 前置条件，要执行该项测试所需要满足的条件，包括硬件和软件设备的型号，用例适用的平台和系统等
- (5) 执行步骤，这是测试者的指导说明
- (6) 预期结果，包括预期的输出或者起到作用的现象
- (7) 用例执行类型，可以是手工操作，也可以自动运行，特别对于一些需要现象需要较长时间出现的测试来说，告诉你是否需要人为参与

(8) 测试重要性, 包括低、中、高等级别, 这是为了测试人员能够了解当前测试的重要性级别

(9) 测试预估时间, 完成测试大概需要的时间, 便于测试人员合理安排测试顺序

(10) 测试关键字, 该用例的主旨

(11) 用例创建时间和创建人员

(12) 用例修改时间, 人员, 改动原因, 修改点

(13) 用例备注, 包含测试工具获取的路径等

结合以上测试用例包含的要素, 我们设计了本文无网络状态下的 BitLocker 功能验证、EFI 文件完整性验证和网络状态下 WHCK 环境下可信 BIOS 功能验证、BitLocker NKP 功能验证测试用例。其中 EFI 文件完整性验证的执行在 3.2 节中已经阐述, 下面以无网络状态下的 BitLocker 功能验证为例来说明测试用例的设计过程。

本文中对 BIOS 可信性的测试主要是功能测试, 即针对产品的各项功能进行验证, 根据功能测试用例, 逐项检验, 看 BIOS 是否达到了设计说明中的要求。功能测试也被称作黑盒测试, 最大的特点是测试人员不用考虑程序内部结构和逻辑结构, 而是只知道该程序输入和输出之间的关系, 在测试中一般是给出一个输入值, 得到一个输出值, 与一个期望值作对比, 如果与期望值结果相同则说明该项测试是通过的, 如果不相同则说明该项测试有问题, 接下来要进一步去确认问题源, 定位这个测试项出现与期望值不相符合结果的真正原因。黑盒测试主要是检测在接口上输入是否能正确接收并且能否输出正确的结果; 是否有数据结构错误或外部信息访问错误; 性能上是否能够满足最终的需求; 是否有初始化和终止性错误。

黑盒测试方法主要有等价类划分、边界值分析、因果图、错误推测、状态测试等。测试用例由有效等价类和无效等价类的代表组成, 从而保证测试用例具有完整性和代表性, 它是一种系统性的确定要输入的测试条件的方法。

针对无网络状态下的 BitLocker 功能验证为例进行说明, 用例的设计要检验 BitLocker 功能是否能够按照需求说明正常使用, 同时程序在适当时候接收数据是否能够输出正确的信息。

在此, 无网络状态下启用 TPM 并且打开 BitLocker 功能后, 系统重启需要正确的密钥才能正常启动, 这里我们将密钥的输入分为三类: 错误密钥、空密钥、正确密钥。观察测试结果是否能够按照只有正确密钥才能正常启动操作系统。表 3.1 是无

华中科技大学硕士学位论文

网络状态下的 BitLocker 功能验证用例。

表 3.1 无网络状态下的 BitLocker 功能验证用例

测试用例：[TIANO-12]TPM 下 BitLocker 功能验证（由于保密性，随机数表示测试用例 ID） 第一版本 创建于 2014-04-03 最新一次修改在 2014-06-01 综述：测试在 Windows 操作系统下磁盘的加密和解密			
序号	操作	期望结果	执行类型
1	在指定的测试平台刷好 BIOS 后,启动到 SETUP 设置界面,开启并激活 TPM,保存并重启系统	TPM 的状态能够被修改为开启并激活状态	手动
2	在光驱中放入 Windows8 操作系统安装光盘,从光盘启动安装操作系统。注意这里不能使用之前就加载好的 Windows 系统,必须重新在需要测试的平台安装	Windows8 操作系统被正确安装	手动
3	启动到安装好的操作系统,确保没有 U 盘链接在系统平台上,通过右键在系统分区对其加密,选择 Turn on BitLocker 选项执行,使用 U 盘来存储密钥。注意这里不要对非操作系统驱动进行加密,只有操作系统分区需要 TPM 开启加密	BitLocker 能够在操作系统分区中起到作用	手动
4	在加密完成后,重启操作系统	操作系统成功重启	手动
5	重启平台,启动到 SETUP 界面,关闭 TPM (不用清除也不用让其处于不活动状态),验证重启到操作系统时需要恢复密钥	在关闭 TPM 时系统需要恢复密钥才能进入	手动
6	输入与步骤 3 中不同的密钥	系统不能进入	手动
7	不输入密钥直接回车等待进入	系统不能进入	手动
8	输入步骤 3 中保存在 U 盘中的密钥,系统就可以正常进入	系统正常进入	手动
9	启动进入到 SETUP 界面,重新开启 TPM,保存后重启到操作系统,关闭操作系统分区的 BitLocker	BitLocker 被正常关闭	手动
测试类型：手动测试 测试重要性级别：中级 测试时间估计：约一小时，其中解密时间近半小时			

然后在下一章搭建的测试环境下进行测试。只有在正确密钥的输入下才能正常启动操作系统,说明 BIOS 系统是可信的。

3.4 本章小结

本章介绍了测试系统的实现机制，以 BitLocker 和 WHCK 测试框架进行说明，阐述了基于 UEFI 的可信 BIOS 系统测试的设计及测试用例的设计，包含测试工具和测试用例。介绍了测试工具的设计方法，以 TpmTest.efi 核心测试工具的具体设计过程及实现方法为例进行说明，同时利用该测试工具实现了 EFI 文件完整性验证测试用例的实现。

4 可信 BIOS 系统测试方案的实现

搭建测试环境是任何测试工作在执行过程中的首要步骤，本章针对验证 BIOS 可行性的测试首先介绍了硬件平台在无网络和网络状态下的搭建，然后介绍了在两种网络状态下的软件系统实现过程，最后完成了上一章设计的测试用例的实现。

4.1 测试平台硬件的搭建

针对不同测试项的要求，硬件和软件平台搭建要求也不尽相同，这里首先针对硬件环境分为有网络和无网络两种状态进行搭建。

（1）无网络状态

首先需要进行测试的平台开发板，一台显示器，若干 SATA 线，供电电源，Windows8.1 操作系统光盘，SATA 接口硬盘，外接光驱（若开发板没有内置光驱），TPM1.2 芯片（1.2 或者 2.0 均可，由于测试时操作相同，这里我们用 TPM1.2 来进行测试说明），然后将这里准备的所有硬件设备与平台开发板进行连接，最重要的是将 TPM1.2 芯片正确外插在平台开发板，以确保其能够正常工作。

（2）网络状态

我们这里要进行的网络状态下的测试有两类，一类是 BitLocker 网络密钥保护器（BitLocker Network Key Protector, BitLocker NKP），另一类是基于微软硬件验证（Windows Hardware Certification Kit, WHCK）中可信 BIOS 的验证。

针对 BitLocker NKP 的测试环境搭建需要的硬件有三台开发板，其中两台可以用作于服务器，另外一台用作客户端测试机；一台交换机，用于三台开发板的网络连接；Windows Server 2012 R2 和 Windows 8.1 操作系统光盘；SATA 硬盘三块，SATA 线若干；至少三根网线；TPM1.2 芯片等。用网线分别将三台开发板与交换机连接起来，每个开发板外接 TPM1.2 芯片外插在客户端测试开发板上。

WHCK 测试环境相对于 BitLocker NKP 的搭建较为简单，所需的硬件设备主要

是两块开发板，其中一台用作于服务器，另一台用于客户端测试机，Windows Server 2012 R2 光盘、两块 SATA 接口的硬盘，分别连接在两块开发板上，将服务器与客户测试机用网络连接，TPM1.2 芯片外插在客户端测试机上。

4.2 测试平台软件的搭建

对于任何测试项，首先要做的是将需要测试的 UEFI BIOS 固件（也常称为 BIOS mage）烧制到开发板平台上的 BIOS 芯片，这是在 BIOS 测试中最基本也是最重要的一个步骤。UEFI BIOS mage 是基于 UEFI Release 和 Debug 不同版本的代码文件，根据不同平台大小不同，一般以 8M 和 16M 居多，将 BIOS mage 烧制到 BIOS 芯片中，该 BIOS 就对整个系统从开机上电到将控制权交给操作系统发挥作用。BIOS mage 的烧制有两种方式，一种是利用*.efi 工具进行软刷，这种方法方便快捷，但是针对不同平台的 Image 需要不同的工具，而且一定要进入到系统 Shell 环境中才能发挥作用，对于系统无法正常启动或无法开机等情况没有很好的解决办法；另一种是使用 Dediprog 工具进行硬刷，这种方法最大的优点是在系统无法正常启动的情况下可以起到起死回生的作用，但是这种方法对于一般用户来说就不太实用，因为需要将 BIOS 芯片从机器上取出，放入 Dediprog 工具来操作，对 BIOS 芯片中的程序进行完全擦除和重写。对于测试人员来说这就是万能的刷 BIOS 工具，非常实用。刷好了需要测试的 UEFI BIOS mage，接下来就其他软件环境进行配置。

对于无网络环境的可信测试软件配置较为简单，首先需要通过 Windows8.1 的光盘来安装一个纯净的硬盘操作系统，其次要通过平台进入到 SETUP 界面对 TPM 环境进行配置，进入 SETUP 中找到 TPM 的配置项，我们要打开 TPM 激活它，这样 TPM 芯片就发挥作用了。

根据上面描述的硬件环境配置，这里也分两类对软件环境进行配置。

（1）针对 BitLocker NKP 的软件配置

首先对两台服务器进行软件配置，在两个平台上分别安装好 Windows Server 2012 R2 操作系统后，接下来对 DHCP 和 WDS 进行配置。其中一台 A 开启动态主

机配置协议 (Dynamic Host Configuration Protocol, DHCP), 另一台 B 开启微软部署服务 (Windows Deployment Services, WDS)。

DHCP 是一个局域网的网络协议, 这里主要用于给内部网络自动分配 IP 地址和提供主机配置参数。为了部署测试环境, 首先打开 A 服务器管理器 (Server Manager), 点击添加角色 (Add Features), 选中 Active Directory Domain Service(ADDS), 待 ADDS 安装完成后, 需要将其提升为域控制器。我们无须单独安装域名服务器 (Domain Name Server, DNS), 因为它会在 ADDS 安装的同时自动被安装。接下来我们开始安装 DHCP, 选中 DHCP 服务器 (DHCP Server), 点击下一步进行安装, 如图 4.1 为选择安装 DHCP 的界面。

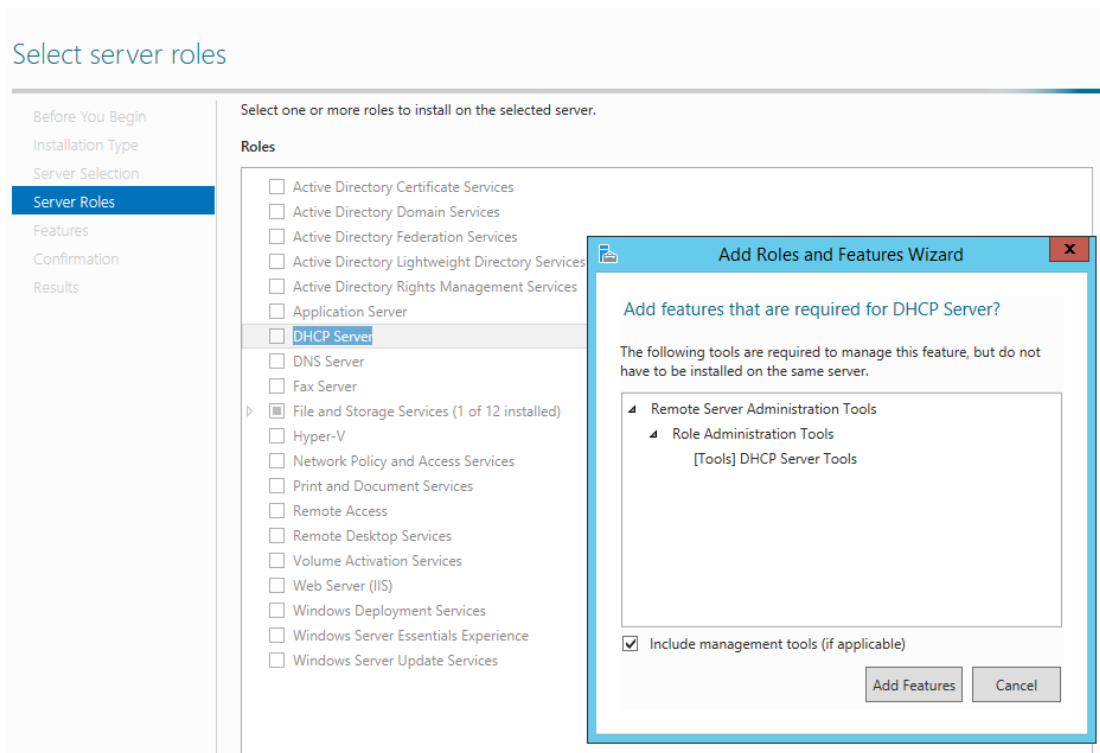


图 4.1 选择安装 DHCP 的界面

安装完成后, 要对 DHCP 进行配置, 首先要新建作用域, 即分配此作用于的地址分配, 这里我们选择起始 IP 地址为 192.168.1.10, 结束 IP 地址为 192.168.1.100, 长度默认为 24, 子网掩码为 255.255.255.0, 如图 4.2 所示为 DHCP 配置作用域。

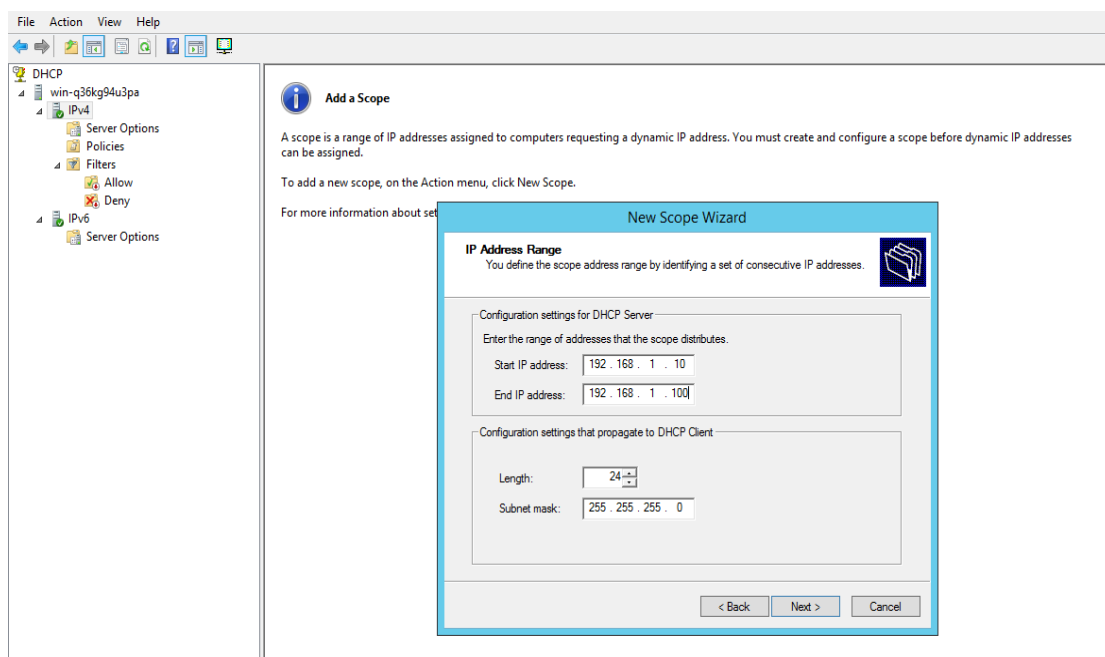


图 4.2 DHCP 配置作用域的界面

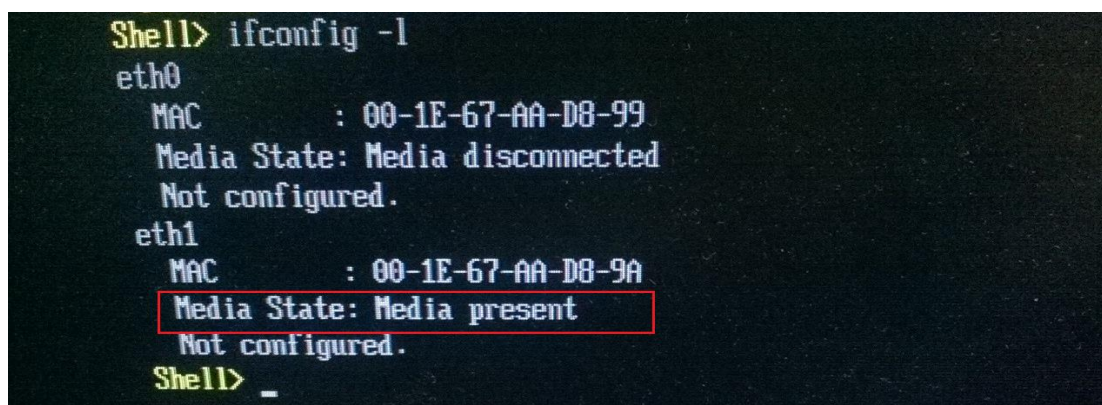
很重要的一步是要激活 DHCP 作用域，如果不激活，那么设置将是无效的。这样 A 服务器的 DHCP 服务就开启了，接下来开始配置 B 服务器的 WDS。同样在服务器管理器中选中 WDS 进行安装，等到 WDS 完成后，我们开始配置 WDS，首先在安装映像（Install images）和启动映像（Boot images）添加中添加映像，这一步可以通过将光盘插入 B 服务器，将光盘中 Boot.wim 和 Install.wim 分别倒入到启动映像和安装映像目录下。

映像的添加主要是为后续客户端测试机能够通过网络访问 B 服务器进行操作系统的安装做准备。在配置完 WDS 后，一定要记得将 WDS 开启，右键服务器出现所有任务，有启动、停止和重新启三个选项，当服务器上图标右下角黑色的原点变成绿色说明 WDS 被正常开启。

服务器被配置好后，接下来是客户端测试机。首先要通过 Windows 8.1 操作系统光盘为客户端安装一个纯净的操作系统；然后将需要测试的 BIOS image 通过硬刷或者软刷的方式刷进测试机；最后进入到 BIOS SETUP 界面，把网卡引导（EFI Network）选项打开，这是为了客户端测试机可以通过网卡引导访问服务器进行操作系统安装。

这样三台开发板各自平台的软件环境算是设置完成，但要使得三者可以进行通

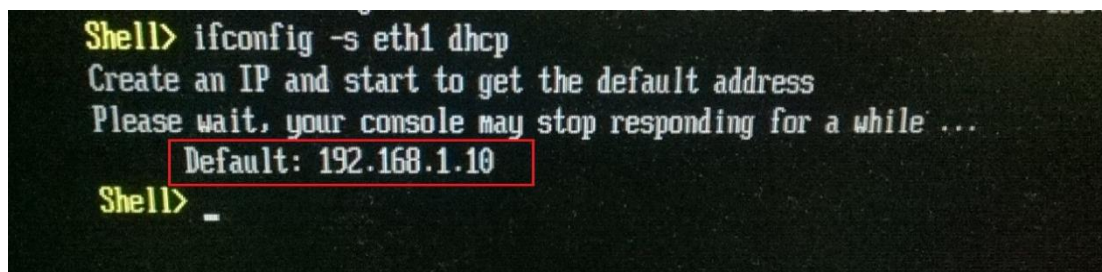
信，我们还要对 A 服务器进行静态 IP 的设置，这里我们将其设为 192.168.1.2，同时关掉其防火墙，否则就会就无法返回数据。对于 B 服务器的 IP 我们可以进行静态设置也可以动态获取，这里我们让 DHCP 为其动态分配；对于客户端测试机的 IP 需要动态分配，这样才模拟了 DHCP 为用户动态分配 IP 这样一个需求。DHCP 有三种机制分配 IP 地址：（1）自动分配，DHCP 给客户端分配永久性的 IP 地址；（2）动态分配，DHCP 给客户端分配过一段时间会过期的 IP 地址；（3）手工配置，由网络管理员给客户端指定 IP 地址。管理员可以通过 DHCP 将指定的 IP 地址发给客户端。这里我们在客户端测试机的 Shell 环境下，通过 `ifconfig -l` 命令来查看本机的 IP、MAC 地址及其使用状态，如图 4.3。



```
Shell> ifconfig -l
eth0
  MAC      : 00-1E-67-AA-D8-99
  Media State: Media disconnected
  Not configured.
eth1
  MAC      : 00-1E-67-AA-D8-9A
  Media State: Media present
  Not configured.
Shell> _
```

图 4.3 通过命令在 Shell 下查看 IP 连接状态

发现 DHCP 暂时没有给测试机分配到任何 IP，且 eth1 是正在使用的，那么可以通过 `ifconfig -s eth1 dhcp` 的命令人工让 DHCP 给测试机分配一个 IP，这里可以看到测试机分配到一个 192.168.1.10 的 IP 地址，如图 4.4。



```
Shell> ifconfig -s eth1 dhcp
Create an IP and start to get the default address
Please wait, your console may stop responding for a while ...
Default: 192.168.1.10
Shell> _
```

图 4.4 手动让 DHCP 给测试机分配 IP

对于 BitLocker NKP 的软件和硬件配置框架如图 4.5

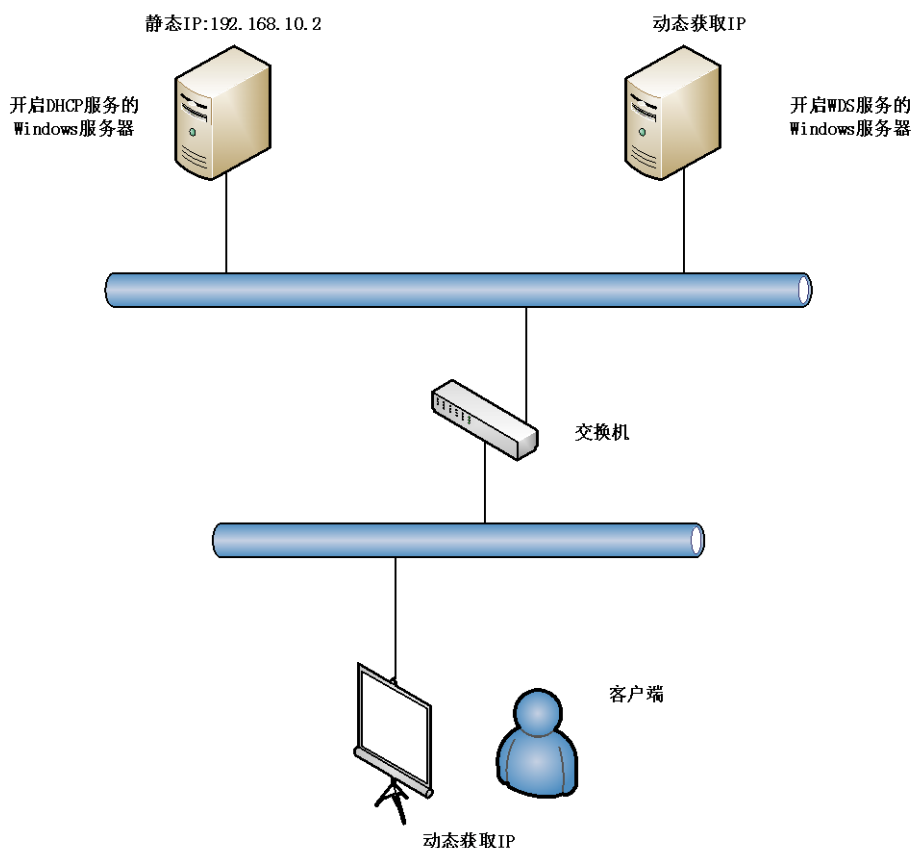


图 4.5 BitLocker NKP 的软件和硬件配置框架

(2) 针对 WHCK 的软件配置

相对比 BitLocker NKP 的软件配置，WHCK 就相对简单。将服务器端和客户端测试机分别通过光盘安装 Windows Server 2012 R2 操作系统，最主要的配置环节在服务器端。首先我们下载 Windows HCK2.1 的工具包，这个工具包适用于 Windows 8.1、Windows7 以及 Windows Server 2008 R2 到 Windows Server 2012 R2 的各个版本操作系统。在 Windows Server 2012 R2 上以管理员身份安装该 Controller，Studio 将自动安装在 Controller 上。然后在测试机上通过网络访问服务器，安装 Client，安装完成后在 Studio 右上角的管理机器默认池中会出现客户端机器的名称，由于在配置机器时无法选择默认池的机器，因此我们要在管理机器池中新建一个测试用的机器池，这里我们将其命名为 WHCK，右键将其状态改为 Ready，即为可用状态。接着在 Studio 上创建一个 Project，在设置系统选项中选择名称为 WHCK 的机器池，将客户端被测机器加载到 Project 中，这样在 Studio 中 Test 选项就会出现测试项，整个 WHCK 的

环境也就配置好了，可以在该测试环境中完成可信 BIOS 相关功能的测试。如图 4.6 是配置 WHCK 的界面。

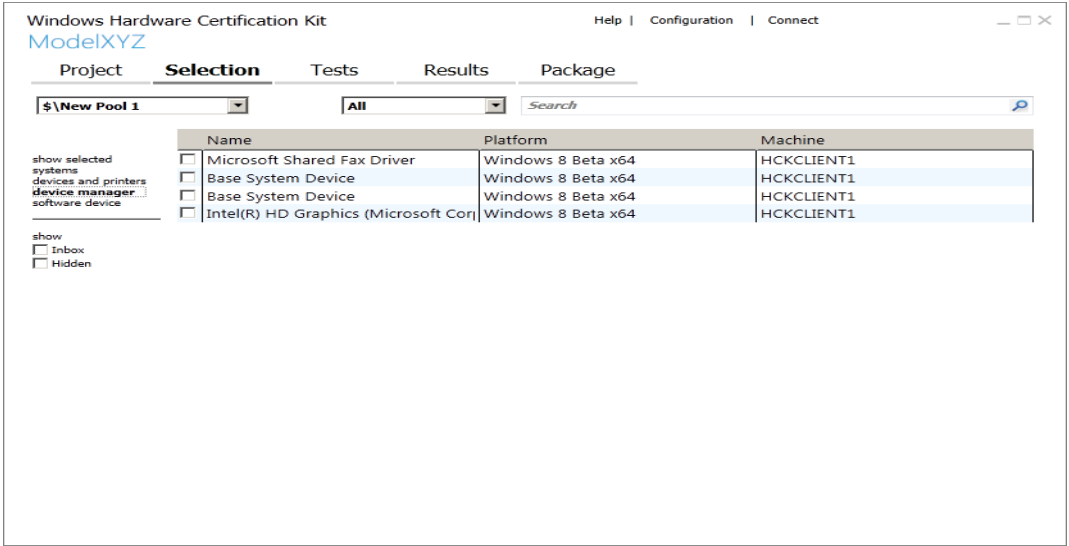


图 4.6 配置中的 WHCK 界面图

对于 WHCK 的软件和硬件配置框架如图 4.7。

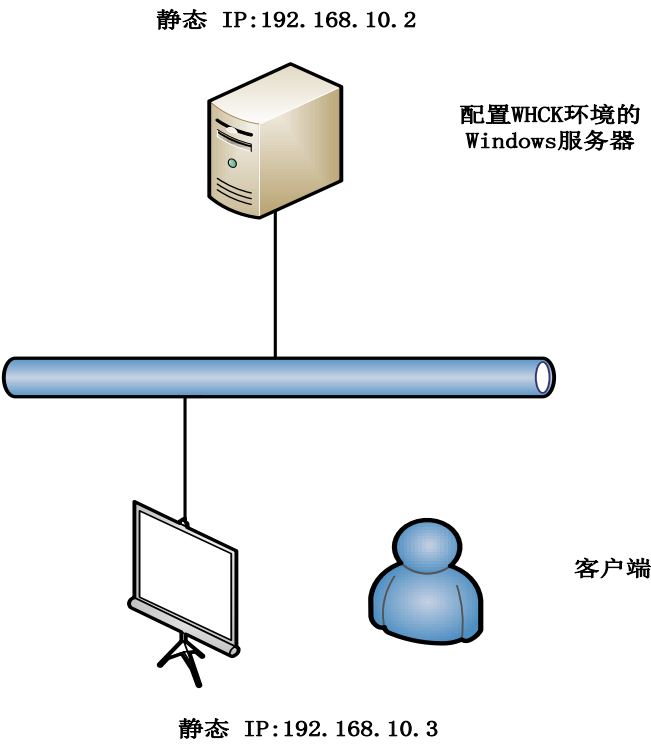


图 4.7 WHCK 的软件和硬件配置框架

4.3 测试用例实现

在搭建好测试环境后，依据第三章中测试用例的设计，就可以来测试我们的 BIOS 系统是否符合需求说明。在具体的操作过程中，我们用到各类测试方法，包括回归测试、安装测试、冒烟测试、稳定测试、扩展测试、压力测试、卸载测试等，首先来看这些测试项在实现具体测试用例中所起的作用。

回归测试主要是指研发人员在修改了代码后，测试人员要重新测试以保证修改的代码没有带来新的问题，不仅仅是代码修改的地方，同时对于其他有关联的功能都要进行测试。回归测试在整个测试过程中占了很大比重，有时候是小范围的回归测试，一定周期还要进行新版本的回归测试。对于这样重复性暂时未能实现自动测试的项目，建立完善的测试用例库是非常有必要的，对于人员的变动和新员工接手工作有很大的帮助，对于这点，在实际的测试工作中，我深有体会。这个过程中，测试用例也是需要不断维护，包括删除过时的测试用例；改进不受控制的测试用例；删除冗余的测试用例和增添新的测试用例。维护测试用例库不仅给下一次测试带来了便捷，同时更大程度上提升了测试的可信性和可用性。同时，在回归测试中有两点要注意，一方面，在项目中做的修改要在当前这一轮测试中完成回归测试的工作，不要放在下一轮；另一方面，我们进行回归测试时，对于同一轮测试中 BIOS 的版本尽量不要变动，对于出现的问题要集中进行回归测试，不能更换测试版本，否则测试结果将会非常混乱，导致测试结果无法分析，不能准确定位 BUG 出现的位置和原因。在具体 BIOS 测试过程中回归测试基本流程为：（1）确定回归测试的 BIOS 版本号；（2）等到需进行回归测试的 BIOS，根据相应的测试用例进行测试；（3）测试结果通过的话，按流程就可以关闭跟踪单了，但是一般在实际操作中，还会跟研发确认，测试通过是否是从根本上解决了上次的问题。如果测试结果不通过，就要返回 BUG 跟踪单，继续修改。这个过程要持续到问题完全解决为止。

黑盒测试可让测试人员探查所有的攻击面，特别针对两种白盒测试不易发现的错误有较好的解决能力，一种常见的错误是在产品发布前，没有去掉那些仅用于调

式的代码；另一种是在产品即将结束时给出一些不在设计文档中正确记录的功能。

安装测试贯穿在我们设计的每个测试用例中，安装测试是不论在 TPM 开启还是关闭状态，各类不同版本的操作系统、软件、服务器、网络版和单机版都能够按照需求说明正常安装或者出现预期的错误提示。

性能测试在这里被运用到了三个方面：客户端和服务端性能的测试、网络上性能的测试。在开启和未开启 TPM 芯片状态下，给系统加密的时间测试，由于 TPM 使用的是 RSA 算法，因此这直接影响 TPM 的性能。网络和非网络状态下对不同大小硬盘系统加密和解密时间等，在网络环境下对系统进行解密操作很大程度上为用户带来了便捷，同时也让网络攻击者有了途径，在客户端与服务器之间的通信中，程序的网络接口是最容易受到威胁的路径，系统能否在网络环境获取正确的秘钥不受到干扰都在我们的测试用例中体现了。

在上述及前文提及上的测试方法指导下，同时针对我们的测试目标，可信 BIOS 需要实现保护数据，验证身份，完整性测量以及存储报告功能，来看具体测试用例的实现。

4.3.1 无网络状态下的 BitLocker 功能验证

该测试用例是测试在 Windows 操作系统下磁盘的加密和解密，主要用于测试 BIOS 可信性中是否能够实现保护数据，验证身份的功能。由于保密性，该测试用例的版本号用随机数[TIANO-12]表示，版本号方便测试人员查找相应用例。下面针对该测试用例的实现步骤给出表 4.1 无网络状态下 BitLocker 功能验证。

表 4.1 无网络状态下 BitLocker 功能验证

序号	操作步骤	期望结果	与期望是否一致
1	在指定的测试平台刷好 BIOS 后，启动到 SETUP 设置界面，开启并激活 TPM，保存并重启系统	TPM 的状态能够被修改为开启并激活状态	一致
2	在光驱中放入 Windows8 操作系统安装光盘，从光盘启动安装操作系统。注意这里不能使用之前就加载好的 Windows 系统，必须重新在需要测试的平台安装	Windows8 操作系统被正确安装	一致

续表 4.1 无网络状态下 BitLocker 功能验证

3	启动到安装好的操作系统，确保没有 U 盘链接在系统平台上，通过右键在系统分区对其加密，选择 Turn on BitLocker 选项执行，使用 U 盘来存储密钥。注意这里不要对非操作系统驱动进行加密，只有操作系统分区需要 TPM 开启加密	BitLocker 能够在操作系统分区中起到作用	一致
4	在加密完成后，重启操作系统	操作系统成功重启	一致
5	重启平台，启动到 SETUP 界面，关闭 TPM（不用清除也不用让其处于不活动状态），验证重启到操作系统时需要恢复密钥	在关闭 TPM 时系统需要恢复密钥才能进入	一致
6	输入与步骤 3 中不同的密钥	系统不能进入	一致
7	不输入密钥直接回车等待进入	系统不能进入	一致
8	输入步骤 3 中保存在 U 盘中的密钥，系统就可以正常进入	系统正常进入	一致
9	启动进入到 SETUP 界面，重新开启 TPM，保存后重启到操作系统，关闭操作系统分区的 BitLocker	BitLocker 被正常关闭	一致

4.3.2 网络状态下 WHCK 环境下可信 BIOS 功能验证

该测试用例是测试网络状态下 WHCK 环境下可信 BIOS 功能，主要用于验证 BIOS 可信性中完整性测量和存储报告功能。各测试项所需时间和测试类型均不同，需要测试人员灵活处理节点，在合适的时候进行手动测试的干预部分，整个测试需要消耗的时间较长，但是由于是自动和手动测试交叉进行，不能直接放到晚上让其自动进行测试。由于保密性，该测试用例的版本号用随机数[TIANO-13]表示，所需测试软件可以通过 WHCK 官网链接下载 [Windows Hardware Development Downloads](#) 下面针对该测试用例的实现步骤给出表 4.2 网络状态下 WHCK 环境下可信 BIOS 功能验证。

表 4.2 网络状态下 WHCK 环境下可信 BIOS 功能验证

序号	操作	期望结果	与期望是否一致
1	在服务器端安装 HCK, 安装过程大约需要 45 分钟, 从开始>所有程序>Windows Kits>HCK Studio 确认 WHCK 被成功安装	Studio 被成功安装	一致
2	在客户端打开浏览器, 输入 \\ControllerName\HCKInstall\Client\Setup.exe, WHCK 安装向导出现了, 直接点击安装, 安装完成后通过控制面板进入到卸载程序中, 查看 WHCK 客户端在程序列表中	客户端 WHCK 成功安装	一致
3	在服务器, 打开被安装好的 HCK Studio, 在 Project 条目上, 点击创建 Project, 默认为 Project 1, 这个名称可以更改	成功创建 Project	一致
4	在服务器端 HCK Studio 右上角, 点击配置, 出现机器池, 创建一个新的机器池, 默认名称是 New Pool 1, 可以更改。然后点击默认池, 确认客户端的测试机器 ID 在其中, 将其拖拽到新建在机器池中。在 ID 上右键将机器的状态为 Ready。	成功创建机器池, 并使得机器状态可用	一致
5	在服务器端 HCK Studio 中选择“Selection”条目, 在下拉菜单中找到包含测试设备的机器池。在左边面板中选择要测试的项目, 这里是“Systems”	正确选取要测试的设备	一致
6	在“Test”条目中包含要验证的所有任务, 在 Type 中 A 是自动, M 是手动。单击任务前的备选框, 然后点击 Run Selected 命令, 任务就会开始运行。	正确选取要完成的任务	一致
7	在“Results”条目中可以看到测试结果, 失败是红色的叉, 成功是绿色的对勾, 在失败的任务下可以展开加号查看 Log	查看测试结果	一致

4.3.3 网络状态下 BitLockerNKP 功能验证

该测试用例是测试网络状态下 BitLocker NKP 功能验证, 主该测试用例主要用于验证 BIOS 可信性中网络状态下的验证身份和客户端、服务器端的通信远程保护数据功能。在测试操作中用到各类 PowerShell 命令, 关于该类命令的使用可参考 [WDS Cmdlets in Windows PowerShell](#)。该用例中客户端适用的操作系统版本有 Windows 8 Client x86 及以上版本; Windows 8 Client x64 及以上版本; Windows 8 Client ARM 及以上版本。测试中的关键点包括: 1) 支持有线局域网中 BitLocker 网络密钥保护的系统在重启时必须支持 UEFI2.3.1 版本中定义的 EFI_DHCP4_protocol 和 EFI_DHCP6_protocol; 2) 在重启时, BitLocker 网络密钥保护器需要通过 DHCP 协议找到 WDS, 由 WDS 发送密钥给客户端测试机来解锁操作系统分区。由于保密性, 该测试用例的版本号用随机数[TIANO-14]表示。下面针对该测试用例分别给出服务器端和客户端的设置操作步骤, 分别见表 4.3 和 4.4。

华中科技大学硕士学位论文

服务器端设置

表 4.3 网络状态下 BitLocker NKP 功能验证服务器端

序号	操作	期望结果	与期望是否一致
1	安装版本高于 8000 的 Windows Server 操作系统	操作系统正常安装	一致
2	对于 Ipv4 进行测试, 就需要一个 DHCP 服务器, 系统版本在 Windows Server 2008 R2 到 Windows Server 2012 R2 之间	系统和 DHCP 正常安装	一致
3	安装 WDS 任务以及 BitLocker 网络解锁控制台, 这一步骤在 PowerShell 下通过指令完成, 1) PS> mport-module servermanager 2) PS> add-windowsfeature -name WDS 3) PS> add-windowsfeature -name BitLocker-NetworkUnlock	命令在 PowerShell 中执行显示成功, 无报错	一致
4	为了确认功能是都被安装成功, 可以执行 get-windowsfeature 来查看	正确显示所添加的功能	一致
5	初始化 WDS 服务器, 执行 wdsutil /Verbose /initialize-server /reminst:"c:\RemoteInstall"	显示成功	一致
6	添加私有网络解锁证书, 指令为 1) certutil -f -p 1234 -importpfx RSA2048NKP.pfx 2) certutil -delstore My KeyNumber(这里 KeyNumber 代替密钥) 3) certutil -f -addstore FVENKP RSA2048NKP.cer 4) certutil -repairstore FVENKP KeyNumber	每步操作提示成功	一致
7	重启 WDS 服务器, 使其处于工作状态, 指令为 1) Net stop wdsserver 2) Net start wdsserver	显示为停止成功和启动成功	一致

客户端设置

表 4.4 网络状态下 BitLocker NKP 功能验证客户端

序号	操作	期望结果	与期望是否一致
1	安装一个版本高于 8000 的客户端操作系统, 系统必须保证有一个独立的系统分区	保证独立分区	一致
2	在 PowerShell 下执行 client-gp-usepin.cmd 命令, 来设置工作组策略	REG 添加八条策略	一致
3	添加网络解锁公钥, 执行 RSA2048NKP_FVE_NKP.reg	无错误提示	一致
4	开启 BitLocker, 状态为 TPM+PIN(1234)+全零恢复密钥, 执行 client-turnontppin.cmd	系统会打印出密钥等信息	一致
5	重启客户端操作系统, 若系统无需手动输入密钥便正常启动	系统无需密钥正常启动	一致
6	测试完后, 在系统分区中将 BitLocker 关闭	无报错	一致

4.4 本章小结

本章介绍了基于 UEFI 的可信 BIOS 系统测试的软硬件环境搭建,分为网络和无网络两种状态分别进行说明,同时依据上一章测试用例的设计结果,实现了基于 UEFI 的可信 BIOS 需要实现保护数据,验证身份,完整性测量以及存储报告功能四个目标的测试用例。

5 测试结果及分析

上一章介绍了 BIOS 可信功能的测试用例的实现,本章具体分析按照这些测试用例在不同平台、不同系统、不同版本 BIOS 具体执行后的结果,通过设计合理测试报告来存储测试结果,让测试及项目相关人员清晰明了地观察测试结果,对其中存在缺陷的测试项能够清楚查找并重现 BUG。用例的执行结果一方面验证了测试用例的正确性,另一方面验证了可信 BIOS 在保护系统数据、维护 PCR 寄存器中散列值不被篡改、实施远程密钥获取等功能方面的作用。

5.1 测试结果

在测试结束时,我们需要记录测试结果,一个基本的测试结果要表明测试内容,测试中依据的测试用例编号,针对测试 BIOS 的版本号和测试结果。在实际工作中,我认为测试结果中的 BIOS 版本号具有非常重要的意义,特别是在回归测试中,经常需要对各个不同版本 BIOS 的测试结果进行比对,分析 BIOS 在一些性能或功能上是否得以改进,甚至可以用来排除系统硬件带来的测试干扰因素。

以下是本文选取的具有代表性的三个测试项的测试结果展示,表 5.1 是一个基本的测试报告表格部分,由测试人员填写。其中测试用例编号和 BIOS 版本号都是随机填写。

表 5.1 测试报告表格

测试结果					
分类项	测试用例编号	测试项	结果	BIOS 版本	备注
网络状态下 BitLocker	TIANO-12	Windows8 操作系统安装	Pass	1234	编号随机
	TIANO-12	操作系统加密	Pass	1234	
	TIANO-12	操作系统解密	Pass	1234	
网络状态下 WHCK 环境下 TPM 功能验证	TIANO-13	TPM 1.2 TCG OS nterface Client Test	Pass	1324	
	TIANO-13	TPM 1.2 TCG Physical Presence nterface 1.2 Test	Pass	1324	
	TIANO-13	TPM Attestation Test	Pass	1324	

续表 5.1 测试报告表格

	TIANO-13	TPM 1.2 UEFI Preboot nterfaace Test	Fail	1324	
	TIANO-13	TPM Based Virtual Smart Card nterface Test	Pass	1324	
网络状态下 BitLocker NKP 功能验证	TIANO-14	DHCP 自动给客户端测试 机分配 IP 地址	Pass	1423	
	TIANO-14	系统加密	Pass	1423	
	TIANO-14	系统加密情况下，重启系 统无需手动输入密钥	Pass	1423	

通过文中测试用例的实现，我们很好地验证了可信 BIOS 系统安全功能，实现了测试的目的。同时，为了保证误操作或者由于测试用例设计不当带来的隐患，我们对测试用例的可用性和正确性也进行了一定的测试。在第一轮测试结束后，测试通过率偏低，着重对其中测试失败项进行分析，采用分步排查、二分法等测试方法，尽可能排除测试中的干扰因素，最后通过与 BIOS 研发人员沟通讨论，解决了测试用例中存在的少量漏洞。由于 BIOS 的代码不断更新，我们的测试用例也要跟随实际情况灵活变动。下面是具体测试用例实现的结果。

（1）无网络状态下的 BitLocker 功能验证

无网络状态下的 BitLocker 验证是在 Windows 8 操作系统下实现的。BitLocker 的验证是利用 PCR 寄存器和平台状态相绑定的功能，当 PCR 寄存器值由于驱动等软硬件的改变而发生变化时，重启 Windows 操作系统时，平台状态也处于保护状态，在没有输入加密时保存下来的 BitLocker 恢复密钥，系统将不能继续运行。图 5.1 是系统磁盘被加密的界面。

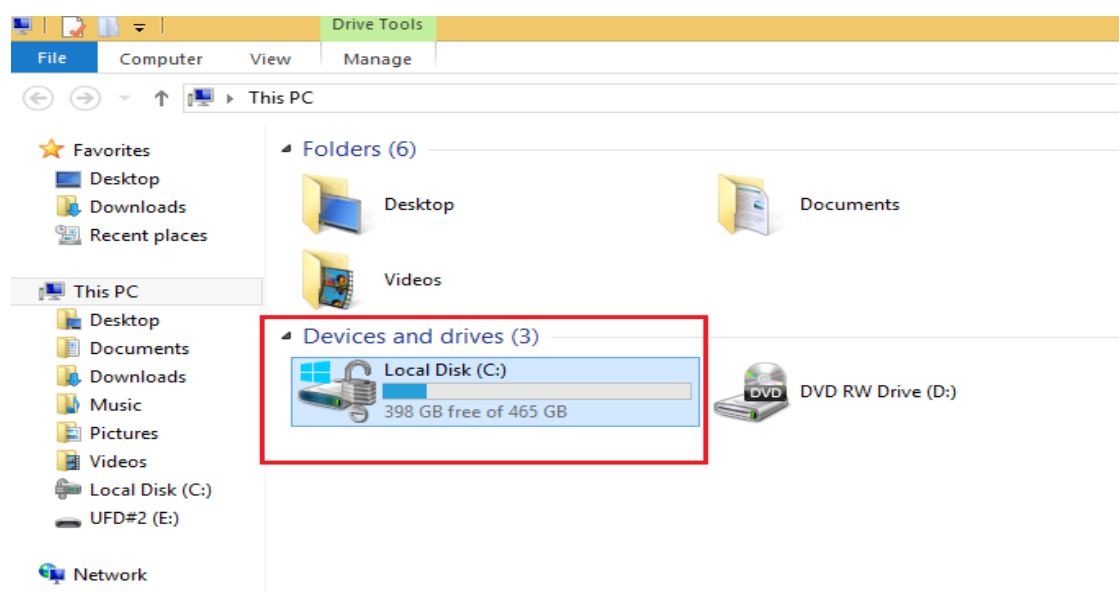


图 5.1 Windows 操作系统下磁盘被加密界面

(2) 网络状态下 WHCK 环境下可信 BIOS 功能验证

在进行 WHCK 环境下可信 BIOS 功能测试前，要经过服务器端安装 HCK Controller 和 Studio、测试机安装 HCK Client、创建 Project、创建机器池、选择验证的功能、选择测试任务执行等一系列步骤。针对测试任务中的每一项测试都有不同的配置操作，在满足测试要求的前提下，我们执行了可信 BIOS 功能验证。在测试过程中存储了测试报告，如图 5.2。

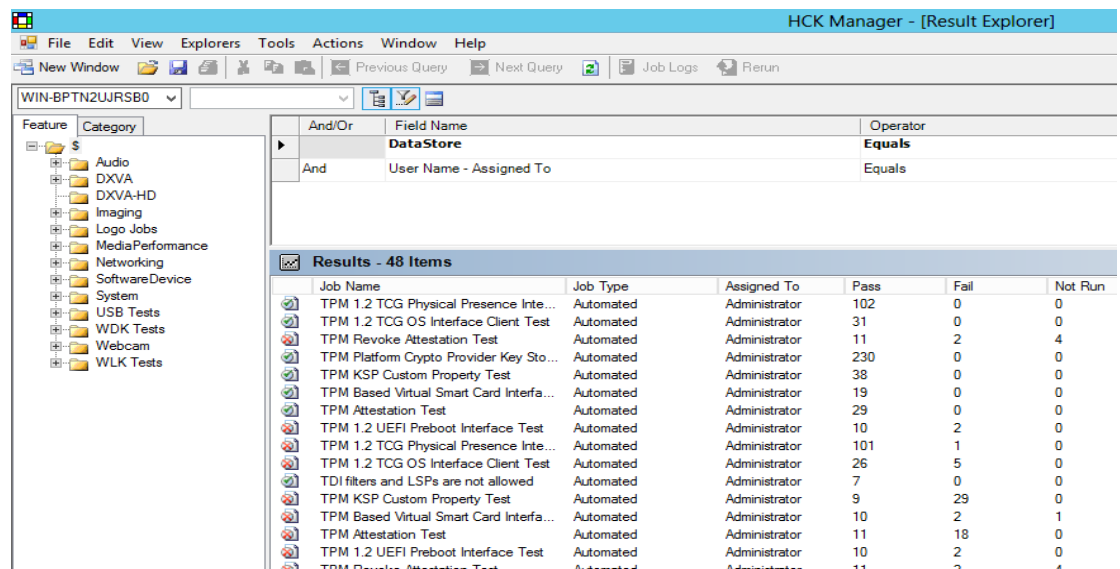


图 5.2 WHCK 环境下查看系统测试日志界面

(3) 网络状态下 BitLocker NKP 功能验证

BitLocker NKP 通过网络解锁的方法实现对 BitLocker 保护器的操作，在加密解密过程中运用脚本工具及 RSA2048 算法生成的密钥证书等执行客户端、WDS 服务器、DHCP 服务器三者之间交互。图 5.3 是在 PowerShell 下安装 BitLocker 网络解锁的过程。在各个步骤按照测试用例操作的情况下，Windows 操作系统可以在 PCR 寄存器值改变的情况下，通过网络获取密钥，验证后正常进入到操作系统。但是如果网络传输失败了，操作系统不能获得正确的密钥，只有在输入正确的密钥后才能正常运行。

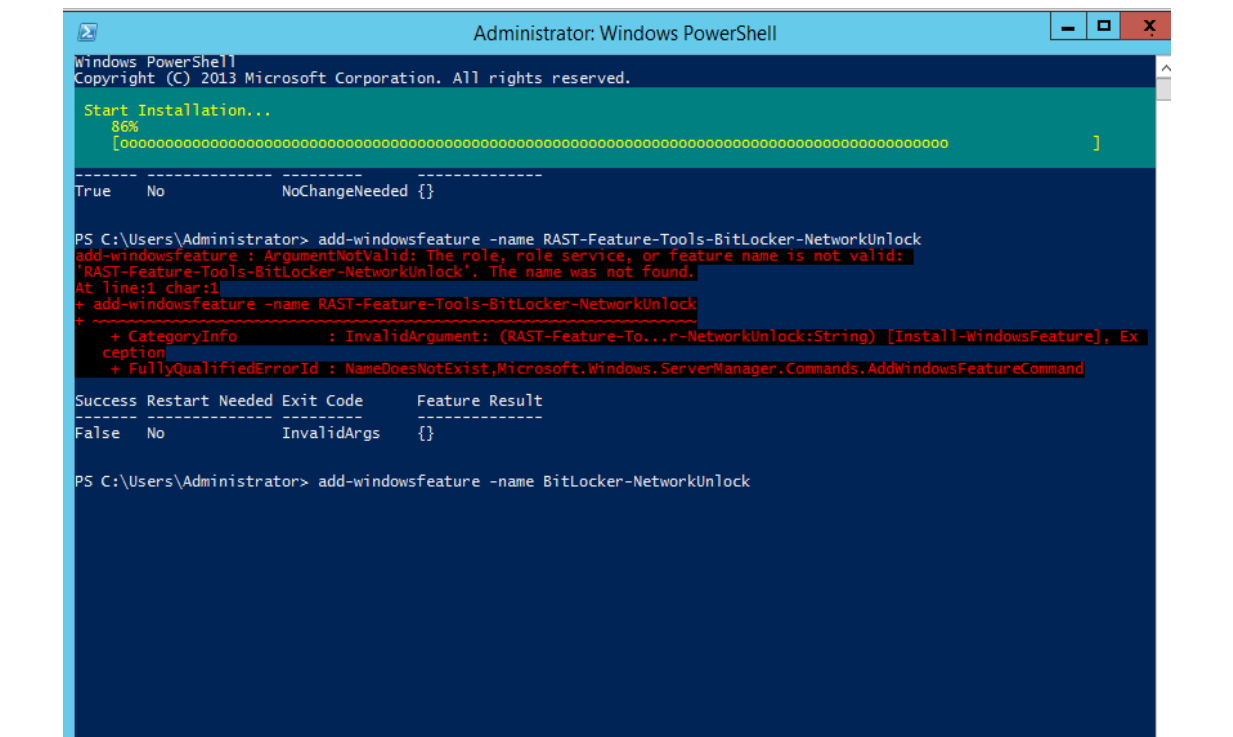


图 5.3 PowerShell 环境下执行命令的界面

5.2 结果分析

首先,从整体上对测试结果进行分析。本文中针对可信 BIOS 系统测试在指定的硬件和软件配置下,实现了 BIOS 保护数据,验证身份,完整性测量以及存储报告功能的测试。文中 BIOS 可信性模块测试有 139 个测试项,首次测试中有 124 项成功,13 项失败,2 项由于硬件无法测试,通过率为 89%,如图 5.4 首次测试项结果占比。

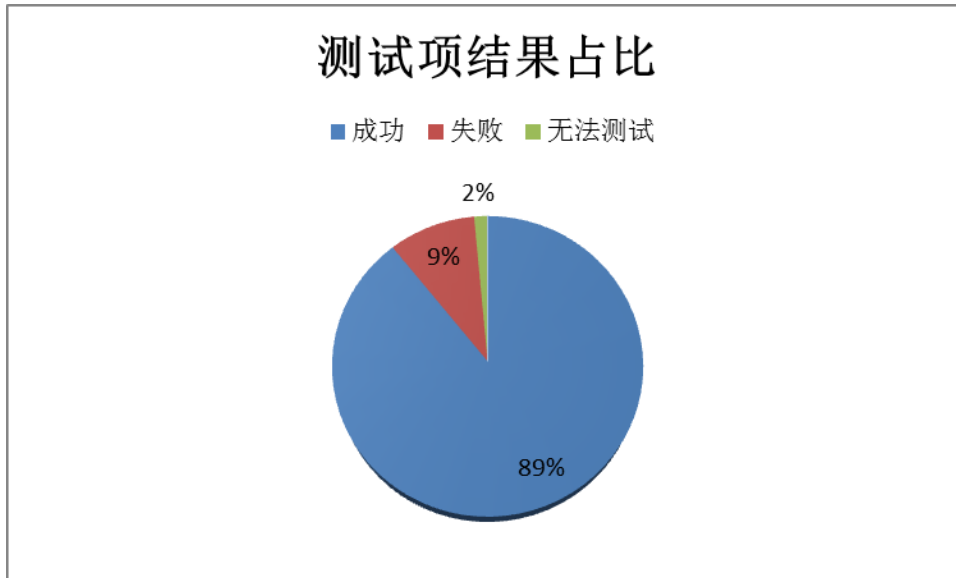


图 5.4 首次测试项结果占比

主要问题集中在网络状态下 WHCK 环境下可信 BIOS 功能验证测试项中，由于 WHCK 的各项测试占用时长不确定，手动和自动测试交替，对于测试项的执行顺序和手动参与测试时间点要求严格等一系列问题，使得首次测试进度较慢。但是为了确保结果的正确性，对于不确定影响因素时，对测试项反复测试，经验证和排除，发现仍然存在接口和定义上的一些问题^[41]。在第一次回归测试后，一些 BUG 被修正了，同时也有影响到其他测试项的问题出现，但总体上测试结果中失败项变少了，通过率达到了 94.2%，第二次回归测试后，通过率在 98.6%。从测试结果来看，测试是有效的，并且经过回归测试，成功地找到软件的错误，有效的提高了 BIOS 系统的可信性。

接着，我们总结在可信 BIOS 系统测试中测试项的优先级和容易出错的测试项，并且针对测试项易错原因进行简要分析。本文中是针对 BIOS 的安全性能进行的测试，因此测试项的优先级排定是按照需求说明中对用户的影响程度而定。可信 BIOS 从平台开机上电到将控制权交由操作系统都起到保护的作用，其中 SEC 是 UEFI 系统开机的初始化步骤，是整个系统的基础，那么 SEC 阶段的完整性验证优先级最高，且最不易出错，测试起来相对较快。DXE 阶段是系统的核心步骤，提供了引导服务和执行时服务，测试中，利用 TPM 芯片对系统分别进行网络和无网络情况下

BitLocker加密,重启系统时,验证用户身份。对于用户来说,无网络状态下的 BitLocker 加密优先级要高于网络状态下的优先级,因为网络状态下的操作更大程度是为用户提供方便,而无网络状态下的加密则是用户保护数据的刚需,对于测试人员来说,无网络状态下的测试环境搭建更为简单,且测试中出错概率更小。WHCK 下可信 BIOS 的功能验证用例虽然是测试中非常重要的一部分,有存储报告、验证完整性等功能,帮助测试及研发人员更好地定位错误,但相对于用户层面来说,优先级就没有前面三者高,并且测试项所需时间长,测试过程中易错,经常要反复测试。如图 5.5 是本文中测试用例的优先级降序排列示意图。

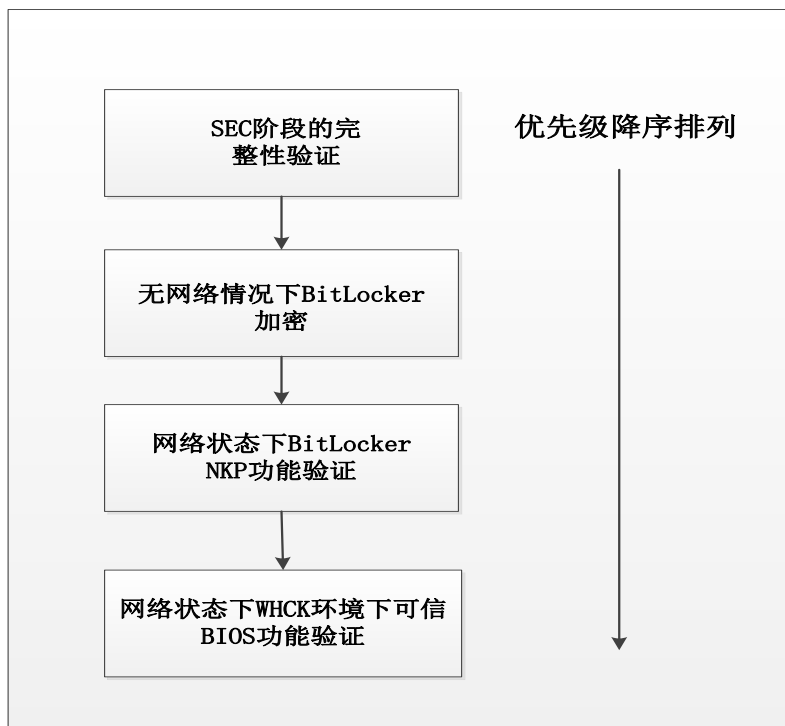


图 5.5 测试用例的优先级降序排列示意图

在测试过程中,网络状态 WHCK 中的 TPM 测试项最易出错,其中网络传输确认项和加载.cer 包是测试过程中经常要反复进行的。网络传输确认项要在网络环境下进行,但是 TPM 测试项是自动和手动交替进行的,每个阶段的时间都不相同,有的测试项一旦开始就无法立即结束,总体时间较长,测试人员如果没有在需要恰当的时间点参与手动确认测试,就要等待它完全结束再开始重新验证,在报告中也会出现多个错误。这对于初次接触该测试项的人员来说,很容易混淆。加载.cer 包易错是

由于不同操作系统需加载的.cer 包相似,每次要手动加载的包个数较多且它们之间是相互关联的。

然后,来分析可信 BIOS 测试用例的性能。针对无网络状态下的 BitLocker 功能验证测试用例,该用例测试环境和步骤相对简单,但是在熟练操作测试项的情况下最快时间也需要 1 个小时,通过光盘给一个 250G 的 SATA 接口硬盘安装 Windows8 操作系统约需 10 分钟,加密时间约 20 分钟,解密时间约 30 分钟。TPM 在加密和解密时采用的是前文介绍的 RSA 算法,因此加解密的时间直接受到 RSA 算法的影响,同时加密的硬盘区域大小也是影响加解密时间的重要原因,硬盘小加解密时间也就相应越短,但是考虑到用户计算机的容量,通过减小硬盘容量的方法减少测试时间的方法不可行,这里应该改进的是加解密算法。

最后,针对测试目标是否实现,分析测试用例的有效性。

本文选取具有代表性的四类(包括第四章中设计的完整性验证工具 TpmTest.efi)测试项进行验证,依据测试用例执行的测试结果具有可信性,同时测试结果也辅证了测试用例的正确性。同时测试过程中科学遵循测试流程和测试计划,针对不同版本 BIOS 也采取了回归测试,针对测试中出现的问题经过有效的验证策略,确定问题出现的充要条件,从而提交 BUG。文中针对可信 BIOS 相关功能进行测试,从而验证当前版本 BIOS 是否具有保护数据、验证身份、测量完整性和存储报告等功能,当然这几项功能在实际运用中并非是完全独立的,它们共同作用来完成可信 BIOS 对系统安全性能的提升。

(1) UEFI 文件完整性测试

这里使用测试工具 TpmTest.efi 来查看 PCR 寄存器的散列值,通过对比其值是否发生变化来确定系统信息是否被非法篡改。测试工具不仅能查看每个 PCR 寄存器的散列值,同时还能够查看系统完整性测量日志。测试中通过改变 TPM 状态,验证 PCR 散列值被正确地存储在芯片中。在 BitLocker 驱动器的测试项中,同样实现了 UEFI 文件完整性验证的功能。在开启 BitLocker 驱动器的过程中,程序会提示用户将恢复密钥保存起来,一般使用 U 盘较多。在对整个硬盘加密后,我们可以通过更改一些驱动文件这种方法改变 UEFI 启动环境,这样在操作系统重启时,系统检测到 PCR 寄存器中的值发生了变化,就无法正常启动。

（2）针对系统进行数据保护

在 BitLocker 的测试中，由于 PCR 寄存器中散列值发生变化，系统无法正常启动，其余所有操作均锁死，如果没有正确的口令，硬盘整个区域的数据就一直处于保护的状态。这种方法对于一些保密级别较高的系统销毁或者保护起到了很好的作用，对想要销毁或保护的硬盘进行 BitLocker 加密，按需求保管恢复密钥，这样机器就不会因为丢失在外面被一些人经过恢复文件等手段窃取重要信息。为了远程操作被进行加密的操作系统，就出现了 BitLocker NKP 这种基于网路的测试项，由服务器保存密钥，客户端机器可以通过给服务器发送获取密钥请求，在验证通过的情况下，由服务器端直接解密，客户端就不用手动输入密钥。

（3）针对加密系统需要身份验证

最常见的身份验证方法是密码口令，除此之外还有语音识别和指纹验证，出于测试可操作性考虑，BitLocker 测试中使用密码口令进行身份验证。WHCK 环境下 TPM 相关功能的测试，实际上也是一种身份验证的过程。由于 UEFI 联盟内各个成员是相互协作，从而才能创造出一种统一的可扩展固件，为了实现统一的标准，微软给出了 WHCK 验证环境，各个厂商生产的不同版本 BIOS 需要在这个环境下进行验证，最终获得统一 UEFI 身份。

（4）存储报告功能

TpmTest.efi 工具就可以查看系统完整性测量日志，同时在 WHCK 环境下测试任务执行后，都会被记录 WHCK 的环境下，通过这些报告日志，可以分析出错测试项的原因。如果测试项出现错误，首先要确认是否是测试人员操作不当引起的错误，再者要排除测试硬件本身罢工的问题，最后才是测试项不通过。有了详细的报告就能更快定位出错位置。

5.3 本章小结

本章依据测试用例步骤来验证不同 BIOS 的功能是否实现，并将测试结果记录下来，对其进行了分析，使得基于 UEFI 下可信 BIOS 系统的测试工作变得完整。

6 全文总结与展望

6.1 全文总结

UEFI BIOS 是一个全新的预启动平台，具有规范化、模块化的优点，不仅解决了传统 BIOS 使用汇编开发的弊端，扩展延伸为一个强大的微型操作平台，可以支持多种操作系统引导之前的底层应用，承担起配置、调试、管理、网络等多种扩展功能。据保守估计，目前全世界使用 UEFI BIOS 的主板出货量已经超过了一半。随着 BIOS 芯片模块功能和容量的不断增加，UEFI BIOS 平台的安全隐患也将越来越多，相比其他组件模块，BIOS 系统在可信计算机中的地位更加突出，因此确保 BIOS 的可信性是整个系统正常运行的前提。

本文根据 BIOS 存在的一些安全隐患，如在系统中缺乏自身安全防护功能；刷新芯片机制易受到病毒的破坏或者被添加恶意危害安全系统的代码；在配置问题上存在的安全问题被远程攻击者利用，通过网络获取权限进行本地计算机的远程控制和数据存取等，针对这些隐患点，运用包括无网络状态下的 BitLocker 功能验证、EFI 文件完整性验证和网络状态下 WHCK 环境下可信 BIOS 功能验证、BitLocker NKP 功能验证等测试用例，验证可信 BIOS 在数据保护、身份验证、完整性测量以及报告存储方面功能的正确性。

在基于 UEFI 的可信 BIOS 系统测试的设计与实现中，主要做了如下几项工作：

(1) 分析相关技术

本文分析了 UEFI 可信 BIOS 平台初始化框架及启动流程，介绍了可信 BIOS 的工作原理及可信链创建的过程，对可信 BIOS 平台实现保护作用的关键技术进行分析，包括加解密算法、安全存储、安全签名及完整性验证等。从系统启动流程中各阶段关键硬件 TPM 芯片、软件的实现和可信算法的运用几个方面将可信计算思想引入到系统上电自检中。

(2) 基于 UEFI 框架下搭建测试可信 BIOS 的环境

BIOS 是嵌入在机器主板上的芯片，要对其可信性进行测试，首先要搭建测试环

境。针对不同测试项，环境搭建也不尽相同，本文分为无网络和网络两种状态对需要进行测试的项进行了软件和硬件环境的搭建。

（3）设计针对安全隐患的测试用例

针对可信 BIOS 存在的安全隐患，归纳出了可信 BIOS 需要实现保护数据、验证身份、测量完整性及存储报告的功能。为了实现对可信 BIOS 是否能够正确完成这几项功能，运用无网络状态下的 BitLocker 功能验证、EFI 文件完整性验证和网络状态下 WHCK 环境下可信 BIOS 功能验证、BitLocker NKP 功能验证测试用例。

（4）实现测试用例

根据测试用例验证不同平台，不同版本 BIOS，不同操作系统中 BIOS 的可信性刷新需要测试的 BIOS 后，依据测试用例的步骤，运用测试工具，测试软件，测试命令等执行测试任务，测试类型包括手动测试和自动测试。

（5）编写测试报告，对测试结果进行分析

测试结果的记录是测试中重要的环节，本文针对测试项设计了测试表格，并对测试用例的实现最终达到的目的进行了说明。

6.2 展望

本文在一定程度上验证了可信 BIOS 功能，但对于 BIOS 的测试，还有很多方面待完善：

（1）目前对于很多测试用例的实现都是基于手动测试，需要较多的人力，测试效率有待提高，针对这一问题，可以设计更多自动测试工具来实现。

（2）对于硬件的需求较大，在测试中涉及到硬件比较多，对于测试中出现问题分析带来了更大的工作量，因为在确认测试结果前，我们首先要排除硬件原因造成的影响。例如，测试中我们会遇到网络连接不上的问题，在没有明确报错的情况下，我们需要排除交换机、网线、接口线等硬件设备。针对该问题，可以使用更多软件来代替。

（3）在对系统分区时加密和解密时间长，一个 250G 容量的硬盘，加密解密共

需要约一小时，测试人员对于这类测试需要长时间等待。针对这一问题，可以从优化加密解密算法等方面着手。

致 谢

本课题在选题及研究过程中得到了我的导师万琳老师的悉心指导。从课题选择、实现到撰写，万老师为我做出了方向性的指引，帮助我开拓思维，最终帮助我找到了论文的定位，并完成该篇文章。

在这里要感谢万琳老师，她不仅引领我走向了一条新的学习道路，让我开拓了眼界，同时，从她身上我学到了很多做人做事的道理，她尊重每个学生的想法，提倡自由的学术精神，让学生选择自己喜欢的事情，开心地学习做研究，她希望我们在走出校园前，能够成长为有思想，会实践，善表达的人。甚至每次小组会，她都认真聆听每个学生的叙述，找到我们的闪光点，告诉我们哪个细节需要改进。

还有要感谢实验室的同学们，在校的一年多时间里，从陌生到熟悉，大家彼此互相帮助，共同成长，学习和生活中我们都是很好的拍档，我在外实习期间，他们帮助我处理了很多学校事情。在外实习期间，我会经常想起 405，想起大家一起学习，一起交流，一起分享的点滴。正如我经常对朋友说的那样，我非常幸运能够成为 405 这个大家庭的一员，遇到亦师亦友的万琳老师，遇到这样一群可爱的小伙伴们。

同时，要感谢英特尔亚太研发中心的一起工作的同事们，他们作为测试和 UEFI 系统安全领域的专家，给予我很多相关方面的指导和帮助，与他们的交流讨论让我更好地将零散的知识点串联起来，在他们的帮助下，才得以顺利完成本文。

感谢我的家人和朋友们一直以来对我的支持和鼓励。

最后，感谢百忙中抽出时间评阅该论文的各位老师和专家。

参考文献

- [1] Chris Mitchell. Trusted Computing. The institution of Electrical Engineers, London, United Kingdom, 2005: 31-45
- [2] Darmawan Mappatutu Salihun. BIOS Disassembly Ninjutsu Uncovered. First Edition, 2007: 341-364
- [3] Vincent Zimmer, Michael Rothman, Suresh Marisetty. Beyond BIOS Developing with the Unified Extensible Firmware interface. Second Edition. intel Corporation, 11/2010: 206-232
- [4] Intel Corporation. Overview of Unified Extensible Firmware interface (UEFI) / Platform initialization (PI). Intel Corporation SSG, August, 2009
- [5] UEFI, Unified Extensible Firmware interface Specification. Version 2. 3. 1, 2011: 17-33
- [6] UEFI/EDKII 中国开发团队. UEFI 与 EDKII 源代码分析. 第 1 版. 北京: 电子工业出版社, 2013: 853-867
- [7] 唐文彬, 祝跃飞, 陈嘉勇. 统一可扩展固件接口攻击方法研究. 计算机工程, 2013(13): 99-101
- [8] ZhenLiu Zhou , RongSheng Xu. BIOS Security Analysis and a Kind of Trusted BIOS. Computer Engineering, 2011, Vlo. 37(No. 4): 137-139
- [9] 王斌, 谢小权. 可信计算机 BIOS 系统安全模块 BIOSM 的设计. 见: 整本文集的王斌 ed. 第二十一一次计算机安全学术交流会论文集. 南宁, 2006. 安徽: 中国科学技术出版社, 2006: 5-8
- [10] 严霄凤. 计算机基本输入输出系统安全研究. 网络安全技术与应用, 2013, 100048: 67-71
- [11] Kinney, Steven L. Trusted Platform Module Basics : Using TPM n Embedded Systems, 06/2006: 208-222
- [12] Intel Corporation. Legacy BIOS and UEFI Boot Process. intel Corporation SSG, March, 2008
- [13] TCG TCG EFI Platform Specification. Version 1, 20. Trusted Computing Group

- [14] Intel Corporation and BM, Trusted Platforms UEFI, PI and TCG-based firmware. September, 2009
- [15] TCG Trusted Computing Group Protection Profile PC Client Specific Trusted Platform Module TPM Family 1. 2. Version 1. 2. Trusted Computing Group February 24, 2014
- [16] TCG TPM Main Part1 Design Principles. Version 1, 20, 2007. Trusted Computing Group
- [17] Trusted Computing Group. TPM Main: Part 3 Commands. Version 1. 2, 2005
- [18] 杨少谦. EFI BIOS 安全增强方案设计与实现: [硕士学位论文]. 西安: 西安电子科技大学图书馆, 2009
- [19] 章瑞, 刘吉强, 彭双和. 基于 EFI 的信任链传递研究与实现. 计算机应用, 2007, 27(9): 2174-2176
- [20] Intel Corporation. Intel Platform innovation Framework for EFI Architecture Specification, 2003: 142-201
- [21] Intel Corporation. Benefits of UEFI in Manufacturing and Test. Intel Corporation SSG, June, 2007
- [22] 司丽敏, 蔡勉, 陈银镜. 一种信任链传递模型研究. 计算机科学, 2011, 38(9): 79-81
- [23] 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, 2006(8): 134-137
- [24] 陈志浩, 谢小权. 一个基于 TPM 芯片的可信网络接入模型. 见: 整本文集的陈志浩 ed. 全国计算机安全学术交流会论文集(第 23 卷). 上海, 2008. 安徽: 中国科学技术大学出版社, 2008: 24-29
- [25] 廖红旭. 个人电脑 BIOS 密码安全缺陷分析. 计算机安全, 2004(1): 60-61
- [26] 王印明, 李阳. 一种基于 DES, RSA 的随机加密算法. 计算机技术与发展, 2012 (4): 235-241
- [27] Ken Goldman, Stan Potter. SHA-1 Uses n TPM. Version 1, 20, 2010
- [28] Michael Rothman, Robert Hale, Tim Lewis, Vincent Zimmer. Harnessing the UEFI Shell Moving the Platform beyond DOS. First Edition. Intel Corporation, 2010:

59-68

- [29] John Butterworth, Corey Kallenberg, Xeno Kovah, Amy Herzog BIOS Chronomancy: Fixing the Core Root of Trusted for Measureant ACM, 2013. ACM 978-1-4503-2477-9/13/11: 25-36
- [30] 徐顺利. 基于 BIOS 的计算机安全子系统的研究与实现: [硕士专业学位论文]. 北京: 清华大学图书馆, 2005
- [31] 王晓华. 软件安全测试方法研究: 农业网络信息, 2010(3): 122-128
- [32] 陈伟琳. 协议安全测试理论和方法的研究: [博士学位论文]. 安徽: 中国科学技术大学图书馆, 2008
- [33] 孟璟, 徐宁. 可信平台模块 TPM 安全芯片的性能测试. 测试技术, 2007, 450004: 31-35
- [34] Rex Black. 测试流程管理. 第 1 版. 天宏工作室译. 北京: 北京大学出版社, 2001: 101-110
- [35] 陈明. 软件测试技术. 第 1 版. 北京: 清华大学出版社, 2011: 16-23
- [36] 朱少民. 软件测试方法和技术. 北京: 清华大学出版社, 2005: 13-17
- [37] Intel Corporation. Using the UEFI Shell for Bare Metal Provisioning. Intel Corporation SSG, 2009
- [38] 落红卫. IP 网络安全测试研究. 电信网技术, 2009, 3(3): 18-21
- [39] Chris Wysopal, Lucas Nelson, Dino Dai Zovi Elfriede Dustin. 软件安全测试艺术. 第 1 版. 程永敬等. 北京: 机械工业出版社, 2007: 15-45
- [40] 张岩, 惠小霞, 阚丹丹. 回归测试方法的研究与分析. 现代计算机, 2011, 31: 18-19
- [41] 林九川, 孙永清, 蒋培等. Intel 可信执行技术及其潜在弱点分析. 见: 全国计算机安全学术交流会论文集. 上海, 2013. 安徽: 中国科学技术大学出版社, 2013: 446-453