学校代码：10255

学号：2070968

Pre-boot 环境下 IPv6 安全架构设计与实现

Design and implementation of Security Architecture for IPv6 under the Environment of Pre-boot

专业：系统工程

作者：孙小霞

指导老师： 王直杰 教授

日期：2009-12-20

东华大学学位论文原创性声明

本人郑重声明：我恪守学术道德，崇尚严谨学风。所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已明确注明和引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品及成果的内容。论文为本人亲自撰写，我对所写的内容负责，并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：孙小霞

日期：2010年3月3日

东华大学学位论文版权使用授权书

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，同意学校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，允许论文被查阅或借阅。本人授权东华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。

保密 □，在 ____ 年解密后适用本版权书。

本学位论文属于

不保密 ☑。

学位论文作者签名：孙小霞　　指导教师签名：王连生

日期：2010年3月3日　　日期：2010年3月3日

Pre-boot 环境下 IPv6 安全架构设计与实现

摘 要

随着计算机的发展，传统 BIOS 在计算机应用中逐渐老化，并伴随着可控性差等问题，严重影响了计算机的发展速度。为了改善计算机的性能，英特尔等公司提出了下一代 BIOS 即本文的 UEFI BIOS。UEFI BIOS 从根本上克服了传统 BIOS 的很多弊端。但是 UEFI BIOS 存在很多安全方面的威胁。因此 IPSec 在 UEFI BIOS 中的应用成为必然。

本课题针对 UEFI BIOS 的 Pre-boot 环境设计并实现了 IPSec 安全架构，从而在网络层有效地保证了通信安全；另外，本文实现了椭圆曲线加密算法应用于 IPSec 架构中的思想；本文还对实现的 UEFI BIOS 网路安全系统进行了功能和性能方面的测试。本文的具体工作如下：

首先，分析了 UEFI BIOS 环境下 TCP/IP 架构中存在的网络安全缺陷。本文基于 UEFI BIOS 中安全问题的考虑，分析了 UEFI BIOS 的主要功能模块和各个启动阶段的主要任务；总结了 BIOS 中 TCP/IP 协议的组成架构并从代码的角度对其进行安全性分析；针对 TCP/IP 在安全方面的缺陷阐明了将 IPSec 应用进来的必要性；用对比的方法给出了 ECC 算法与其它加密算法相比的优越性。

其次，设计并实现了UEFI BIOS下的IPSec架构。因为TCP/IP协议族从最初的设计开始就没有考虑通信安全方面的因素，因此该架构在安全方面存在很多安全隐患。Pre-boot环境下引入TCP/IP是一个新的尝试。它能够带来很多传统BIOS下不能实现的性能。因此，通过综合考虑这些方面的因素，本文设计并实现了IPSec来保证Pre-boot环境下的网络安全。本文重点设计了IPSec在Pre-boot环境下的通信架构，对设计过程中的细节进行了阐释，论述了ECC加密算法的具体实现方法。

第三，设计实现了IPSec密钥交换模块，并实现了ECC算法在密钥交换中的运用。本文设计了IKE的两个工作阶段：第一阶段每次传输往返数据报的封装格式，具体数据段的定义和IKE参与Pre-boot环境下网络通信的方式；第二阶段采用积极模式实现了Child_SA的建立。另外，本文实现了ECC算法加入到Pre-boot环境中来的思想。

最后，从功能和性能上对在Pre-boot环境下实现的IPSec安全方案进行测试。实验证明本文设计的安全框架符合Pre-boot环境下的各项协议标准，该架构能够很好的运行在TCP/IP协议中；在Pre-boot环境下的性能测试进一步验证了椭圆曲线加密算法在IPSec中能够正常运行，也证明了IPSec的引入在时间上对启动的影响在可接受范围之内。

关键字　椭圆曲线　Pre-boot 环境　IPSec　BIOS

DESIGN AND IMPLEMENTATION OF SECURITY ARCHITECTURE FOR IPV6 UNDER THE ENVIRONMENT OF PRE-BOOT

ABSTRACT

With the development of computer, the traditional BIOS is facing the problem of aging and poor control which prevents the speed of the computer development. In order to improve computer performance, Intel has introduced the next generation BIOS, that is, UEFI BIOS. It fundamentally solved the disadvantages of the original BIOS. However, the new BIOS is facing security challenges. Therefore the application of IPSec in the next generation of BIOS meets an urgent need.

This thesis designs and implements the IPSec security architecture under Pre-boot environment of UEFI BIOS, providing an effective guarantee for the communication security in IP-layer. This thesis implements the application of ECC encryption algorithm to IPSec. This thesis carries out the system optimization and performance testing. The detailed work of this thesis is as follows:

First, the network security flaws of UEFI BIOS are analyzed. Based on security considerations for UEFI BIOS, this thesis analyzes the main function modules and the main tasks of every boot phases of UEFI BIOS. Then this thesis gives a summary of the structure of TCP / IP protocol of BIOS, analyzes the security from the view of code. This thesis explains the need for incoming IPSec application due to the security flaws of TCP/IP. In this thesis, the comparison between ECC and other encryption algorithm is given to show the superiority of ECC algorithm.

Second, the IPSec framework under UEFI BIOS is designed and implemented. Because the layers of TCP/IP do not take security considerations into account, safety hazard exists from the beginning of the design. The introduction of TCP/IP to Pre-boot environments is a new attempt. It can bring a lot of performance beyond the reach of the traditional BIOS. Therefore, through comprehensive consideration of these factors, this thesis designs and implements IPSec to ensure the network security for Pre-boot environment. This thesis focuses on the design of the IPSec architecture in the Pre-boot environment. At the same time, this thesis demonstrates the good performance of IPSec for the security protection under Pre-boot, by giving the details of the design process of IPSec. The concrete realization of ECC encryption methods is also given.

Third, this thesis designs and implements the IKE module, and implements the application of ECC encryption algorithm. The two phases of IKE are implemented. The first phase is the design for the format of data package, the definition of data fragment and the way how the IKE work for the traffic under Pre-boot. The second phase is the building of Child-SA with active mode. In addition, this thesis shows how the ECC algorithm is added to the Pre-boot.

Finally, from the perspective of function and performance this thesis carries out the testing for IPSec security solution under Pre-boot environment. The experiment proves that the designed security framework meets the protocol standard of Pre-boot environment. The architecture can run well with TCP/IP protocol. The performance test further verifies the ECC encryption algorithm in IPSec is able to run properly in Pre-boot environment, and demonstrates that the introduction of the IPSec impacts little on the total time.

KEY WORDS: ECC Pre-Boot IPSec BIOS

目 录

第一章 绪论

1.1 传统 BIOS 的缺陷

BIOS（Basic Input Output System）全称是基本输入输出系统，是计算机中的一组可执行程序[1]。BIOS被固化到计算机内主板上的ROM芯片中，因此也常称BIOS为固件；它保存着计算机最重要的控制基本输入输出的程序，还有系统的设置信息，开机时运行的自动检测程序和系统启动后执行的自举程序；BIOS保存了计算机底层中直接对硬件进行设置的和实施控制的信息。随着计算机领域的高速发展，作为Pre-boot的环境的传统BIOS固件在设计时本身存在的很多缺陷限制了计算机技术的提高[3,4]。

首先，传统BIOS刷新机器的方法过于繁琐。Windows刷新BIOS的方法仅仅能对BIOS的简单功能进行刷新，不能改变BIOS整体架构性能。相比其它高速发展的计算机配置，BIOS固步自封，越来越跟不上现代科技发展的步伐。

其次，传统BIOS使用底层的16进制的机器语言进行代码实现，寄存器调用参数和小于1MB的固定编址方式存储。这些都使程序实现过于复杂。随着计算机功能和架构的不断完善，BIOS实现代码越来越庞大。容量的限制使ROM固件满足不了各方面发展的需求，虽然人们提出用可选的ROM增加一些额外的存储空间，但是这并不能从根本上改变BIOS容量有限所面临的问题。这些给BIOS开发者提出了很大的挑战。

再者，BIOS实现了计算机平台的硬件与操作系统软件之间的连接，可是这个连接功能没有完备的规范，不同BIOS制造商生产的BIOS各不相同。制造商可以通过任意的改变配置或者设定配置的默认值来适应自己平台的需求。用户可以利用BIOS对计算机进行设置或进行机器的错误诊断，但是不能从根本上改变BIOS的默认配置。

当前，BIOS的开发的复杂性使得开发只掌握在具有丰富经验的专家手中，这些局限性使BIOS的发展裹足不前。当今人性化是社会发展的总趋势，传统BIOS的弊端也越来越满足不了各方面发展的需求。因此，英特尔公司提出了可扩展固件接口EFI架构，为下一代BIOS开发提出了崭新的思路。

1.2 UEFI BIOS 的诞生

可扩展固件接口(Extensible Firmware Interface, EFI)BIOS可以解决传统BIOS面临的问题。EFI全称是可扩展固件接口。EFI不是某个具体的软件，而是OS与平台硬件之间的一整套接口规范。EFI由众多数据表和系统服务组成，包括与硬件的基本信

息，加载OS的工具以及OS所用的启动服务和运行时服务。这些数据表和系统服务实现的功能构成了BIOS的核心。

EFI由英特尔发起，该接口方案受到全球业界的广泛认可，并相应成立了UEFI(统一可扩展接口)论坛，这一国际组织主要负责UEFI的推广，平台初始化规范制定，UEFI规范制定和UEFI规范的SCT验证。EFI基于模块化特性主要被应用于64位的安腾处理器平台上。从2000年开始，全球多家知名公司纷纷加入到UEFI的研发团队中来，如惠普，LSI，微软等公司[3]。目前，UEFI的发行版本是2.3。

就UEFI BIOS核心架构来说，它像一个被简化的OS，工作于平台硬件以及高级OS (比如Windows或者Linux)之间。UEFI BIOS摒弃了纯文本工作方式下的界面单一的缺陷，UEFI BIOS采用支持鼠标操作的图形化界面管理方式。同时，UEFI BIOS采用很多开发者熟悉的高级的C语言编程，从而使更多的技术人员参与其开发。这样，不仅可以改善平台之间的兼容性，也可以缩小开发的成本，缩短代码开发的时间，从而加快BIOS的更新速度。由于UEFI的可扩展性增强，计算机制造商可以通过外挂等方式使用户根据自己的喜好设置机器特性，更能满足机器人性化的需求。可扩展性也能很好的实现不同驱动模块的引入，从而增强BIOS的功能[4]。UEFI推出了类似于Windows下驱动模型的驱动模式，这种模式采用C语言编程，可以在不同的平台中运行，使UEFI BIOS的兼容性更强。UEFI比传统的BIOS的启动时间更短，所占空间也更小。UEFI BIOS作为一个Pre-boot的环境，在进入操作系统之前用户便能进行很多传统BIOS不曾支持的功能操作。远程网络访问就是其中之一。UEFI BIOS增加了网络功能模块，从而使用户在进入操作系统之前便可进行远程机器的访问或者因特网的在线查询。

本文主要针对UEFI BIOS下实现的网络功能模块实现安全保障。因为UEFI只是在因特尔的EFI得到业界肯定后作为EFI组织的统称，在代码实现级别跟EFI没有任何差别，所以本文不具体区分UEFI和EFI。Pre-boot环境是遵循UEFI规范的UEFI BIOS的运行。因此本文所讲的Pre-boot环境就是UEFI BIOS的执行过程。

1.3 IPSec 的发展

UEFI 的驱动模块增加了网络特性，其网络架构满足网络基本模型。因特网是许多个互不相同网络的互联。互联网通过一系列的协议规范实现不同主机间的通信。目前，因特网协议为 TCP/IP。UEFI BIOS 中网络实现支持 TCP/IP 协议架构，其中 TCP 协议和 IP 协议是该架构的核心协议[6]。本文主要针对网络 IP 层实现安全机制。

目前 IPv4 的网络安全机制只建立在应用层上，如电子邮件加密，超文本链接协议等，无法从网络层来保证通信的安全。从 1995 年开始，因特网工程项目组（Internet

engineering task force，IETF）推出了一套 IP 安全（IP Security，IPSec）协议用来实现网络层的安全[34]。这些协议通过 RFC 的形式规定下来，并成为业界的标准[2]。IPv6 安全架构的重要组成部分就是 IPSec，它通过实现 IPSec 保证网络层级别的安全。

IPv6 通过 IPSec 实现对了网络层数据的身份认证和加密处理。IPv6 的网络层协议都实现了 IPSec 安全。为了将现有的 IPv4 协议栈过渡到 IPv6，在 IPv4 协议栈中也可以实现 IPSec 协议。IPSec 已经成为 IPv6 的标准组件。IPSec 协议也存在不足：这个协议只能在网络层上运行，不能实现应用层的安全需求；对 IPSec 中协议的处理会影响现有网络安全协议的实施。比如在 ESP(文章后面会有具体介绍)中新的 IPv6 报头添加以后，原来的入侵检测无法获得原目的地址，这样对数据的扫描过程就不能很好实现，这些缺陷的存在给 IETF 工作组提出了新的挑战。

对于应用 IPSec 的产品来说，可扩展性是体现竞争优势的重要特性，因为它能直接影响到产品的服务质量。本课题充分考虑到这一点，本文所使用的 Pre-boot 环境下的 IPv6 通信协议栈，能够满足未来 IP 发展的需求；UEFI 的可扩展性为性能的增加提供了很好的平台，这些为扩展性提供了良好的环境。安全性能的增加，会影响网络的特性。IPSec 的引入使网络的时间延迟增加，网络吞吐量减少，这些都会干扰网络的使用质量，并在总体上降低网络性能，所以功能与性能测试对 IPSec 的应用尤为重要。

1.4 本文研究的意义

当前，UEFI一个最大的悬而未决的议题就是安全问题[5]。UEFI BIOS就像一个小型的OS，C语言代码实现需要更多的安全性保障，因此对它的安全性有更高的要求。现在的网络安全产品大都针对操作系统，并没有考虑太多BIOS的安全问题。而且，在操作系统获得计算机控制权以前BIOS掌管计算机的所有资源。如果它被病毒感染，整个计算机系统都会瘫痪。因此BIOS的安全性能的增加就显得尤为重要。

IPSec能够实现网络层通信安全服务，这些服务有网络的访问限制、数据完整性检测、消息源验证、抗重播机制和机密性保护等[2]。这些目标是通过数据流安全协议AH(验证头)和ESP(封装安全载荷)以及IKE(IPSec密钥交换)来实现的。IPSec可以为电子商务、电子政务提供网络的安全性和可靠性。而IPv4没有实现安全和服务质量的保障机制，容易遭受黑客的袭击。IPSec VPN技术可以为企业提供网络通信安全服务，实现远程连接并能很好的节省通信成本。因此将IPSec应用于BIOS，实现BIOS的安全通信势在必行。

UEFI BIOS将BIOS的功能模块化[13]，有利于用户为BIOS增加更多所需功能。

UEFI BIOS支持TCP/IP网络通信协议。随着IPv6的发展，UEFI BIOS逐渐的增加了IPv6的支持性能。本文立足于BIOS环境下IPv6中对IPSec的安全依赖性，将IPSec应用到UEFI BIOS中来，一定程度上解决了BIOS网络通信的安全问题。

同时，本文考虑到BIOS是固化在计算机上的一段程序，代码的存储量大小是有限的，因此在BIOS中添加程序时把重点放在代码量的大小问题上。同时，BIOS是开机到进入到操作系统之前运行的固件，所以时间要求也非常严格。椭圆曲线加密(Elliptic Curve Cryptography，ECC)算法的最重要优点是存储量小，计算时间短。因此本文借助ECC算法在这些方面的优势，将其运用到IPSec的IKE中来。该设计在保障Pre-boot环境下的通信安全性的同时也借助ECC算法的优越性，更大程度上节省了Pre-boot的时间和硬件资源。

课题进行了性能和功能方面的测试来论证方案的可行性和正确性。IPSec产品研发与应用面临严峻的挑战，主要包括如何确保IPSec协议不同开发者平台具有兼容性，如何从网络性能和功能方面对实现的IPSec的安全性进行评定。IPSec要求在发送数据之前建立通信管道。通道建立速率（Tunnel Setup Rate）定义为一个机器每秒钟所能建立的通道个数[45]。通道建立速率是评定网络性能的重要指标。在建立通道之后，IPSec将对传输的数据进行加密，并对传入数据进行解密。加密和解密的过程需要大量的运算。此外，加密与解密过程会大量的增加时间延迟。增强安全性必定会降低网络的性能，在IPSec的实施过程中，一个重要的考虑因素就是网络安全性与通信性能的之间的平衡。本文将从功能和性能两个方面对所设计的Pre-boot环境下的IPSec进行测试，从而评估其性能。

1.5 论文的研究内容及章节安排

本课题针对 UEFI BIOS 的 Pre-boot 环境设计并实现了 IPSec 安全架构，从而在网络层有效的保证了通信安全。另外，本文实现了椭圆曲线加密算法应用于 IPSec 架构中的思想并对实现的 UEFI BIOS 网路安全系统进行了功能和性能方面的测试。相比以往的一些算法，在本课题的研究中将椭圆曲线算法的理论引入到 Pre-boot 环境下的 IPSec 的研究中，该算法的引入符合 Pre-boot 环境的性能要求。同时，本文对实现的系统进行了测试。

本文的组织结构如下：

第一章 绪论。主要是提出问题，介绍BIOS的发展现状，TCP/IP架构的发展和本课题的研究的意义。

第二章 分析Pre-boot环境中网络的通信安全缺陷。通过分析Pre-boot环境下各个模块的组成，相互衔接关系以及TCP/IP的工作机理，分析BIOS架构的缺点及面临的

网络安全威胁， 给出ECC加密方法应用于其中的思想。

第三章 基于IPSec的UEFI BIOS系统安全架构设计。TCP/IP作为UEFI BIOS的通信协议，使BIOS在开机进操作系统之前便可以与外界进行通信。但是存在通信安全方面的隐患。本章进行了在Pre-boot环境下在TCP/IP中IPSec安全架构的设计。

第四章 在Pre-boot中IKE的设计和代码实现。本章设计了IKE协议的消息交互过程中数据报通信格式，并将ECC算法运用到Pre-boot环境下。本章实现了TCP/IP中DH交换的过程并实现了ECC算法的应用。

第五章 Pre-boot环境下IPSec架构系统实验及测试。本章通过搭建测试环境，安装测试所需的配置文件并书写设计用例来实现系统功能和性能的测试。测试证明，本文设计实现的在Pre-boot环境下的IPSec系统工作正常，对系统性能方面的影响在可接受的范围内。

第六章 总结。

第二章 Pre-boot 环境中网络的通信安全缺陷

2.1 引言

伴随着计算机技术的飞速发展，UEFI BIOS引起了越来越多人的关注。本章着眼于UEFI BIOS中网络通信安全问题的考虑，介绍了UEFI BIOS的发展状况和主要功能；通过图示等方式总结并分析了UEFI的层次结构；分析了UEFI BIOS启动过程中（PEI,DXE,BDS等）各个阶段的主要任务。然后给出了UEFI BIOS的通信模块中的TCP/IP协议的组成结构并从安全性角度对其代码进行了分析。本章论述了TCP/IP在安全方面的缺陷和将IPSec应用于TCP/IP中的必要性。本章用对比的方法介绍了ECC算法与其它加密算法相比的优越性。

2.2 UEFI BIOS 的通信优缺点分析

UEFI 架构设计中定义了很多架构相关的基本概念如内核，协议和驱动[4]。下面将本文中涉及到的基本概念做简要概述。

UEFI 系统内核(UEFI System Kernel)是 UEFI 通过对底层平台的抽象提供的一些系统底层服务。UEFI 内核提供的服务包括两种：启动引导服务（BOOT Service）为加载过的 UFEI image 提供通用的引导环境，也为系统提供引导过程中的调用服务；运行时服务（Runtime Service）在启动引导中和操作系统运行过程中都可以被调用。运行时服务能够实现物理内存地址与虚拟内存地址之间的转换。

UEFI 协议(UEFI Protocol)是 UEFI 对平台设备的抽象。通过对与设备对应的协议提供的接口进行操作，实现对不同设备的操作。每个协议都有一个全局唯一标识符 GUID 与之对应。协议设计中，用一个数据结构存储私有数据和用函数的指针表示各个外接接口，这种实现方式，使设备更换驱动程序时的操作简单化，进而能简化了驱动程序开发的步骤。

UEFI 用驱动来管理硬件模块。驱动通过协议的接口被调用。UEFI 的驱动模型能够使驱动程序的开发过程简单化，生成的可执行代码量更小。设备驱动程设计需要符合 UEFI 驱动模型，然后加载驱动程序的映像句柄(Handler)会产生一个与该驱动程序对应的驱动程序绑定协议即 UFEI Driver Bang ding Protocol。该协议产生后系统的固件把设备驱动程序跟一个控制器相连。最后，设备驱动程序会在控制器的句柄(Handler)上产生一个提供统一的输入输出操作的协议。

2.2.1 UEFI BIOS 的基本组成架构分析

本节主要分析 UEFI BIOS 的基本架构组成和启动过程。通过对 UEFI BIOS 整个架构和组成过程的分析，阐明该架构相对于传统 BIOS 存在的优势；同时也进一步阐明在 UEFI BIOS 架构中引入网络安全的重要性。

1. UEFI 功能分析

因特尔的UEFI平台核心架构(以下简称架构)是一个通用的可扩展的平台环境。UEFI可以通过接口的形式分离平台和固件。UEFI为标准的总线提供了统一接口，而且总线类型的列表具有可扩展性。正因为此，操作系统开发人员可以只关注操作系统的开发。即使固件有所变化，这个接口对所有符合UEFI规范的架构依然有效，相应的操作系统也不需要进行任何修改。

本文基于的Pre-boot环境，根据UEFI Spec 2.3架构设计而成。与传统BIOS的功能类似，该Pre-boot环境即本文所说的UEFI BIOS主要用于初始化和配置系统，然后加载操作系统。

如图2.1所示，UEFI架构的最底层是硬件系统，包括主板上的CPU，内存和各种外部接口[3]。UEFI通过UFEI的驱动执行环境实现与硬件的信息交互。驱动执行环境包括协议框架，平台驱动和架构驱动以及兼容性支持等模块。UEFI通过驱动执行环境的协议框架模块向下实现与硬件平台的通信。通过驱动执行环境的平台驱动和框架驱动等向上实现对UEFI的支持。兼容支持模块保证了UEFI BIOS对传统BIOS的支持，从而有利于实现由传统BIOS向UEFI BIOS的平缓过渡。UEFI架构的最高层是UEFI操作系统装载器，预启动应用程序，可选ROM和传统操作系统装载器。他们一起实现启动过程中操作系统引导，然后逐渐启动操作系统。传统操作系统装载器实现对传统MBR分区方式的支持，从而实现对传统操作系统的兼容。操作系统装载器包括引导服务和运行时服务。UEFI各层次间并不是独立的，上层需要底层提供的系统启动信息才能进行工作。

UEFI BIOS作为一种固件接口，继承了传统BIOS的很多功能，如在启动操作系统之前的磁盘管理；引导服务；脱离操作系统的平台管理方式；远程信息设置等。UEFI引导固件主要完成从开机启动后到操作系统运行之前收集所有的硬件配置信息。固件在启动时不需要首先去初始化，这种设计方式能够在芯片生产时更好的节约成本。用高级语言设计对外的接口，通过这种方式，可以提高相关BIOS产业中基于UEFI架构的编辑模块间的互操作性。其他如独立的BIOS提供商(IBVs)，最初的设备生产(OEM)，独立的软件提供商(ISV)，独立的硬件提供商(IHV) 便可以围绕一个

中心架构参与到平台革新中来，作为EFI驱动的多个组件来运行[1]。

UEFI除了可以替代BIOS以外，很多功能得到了增强，如对系统死机的处理上，用户可以通过进入UEFI来改变系统设置或者载入其它驱动程序等方式解决操作系统

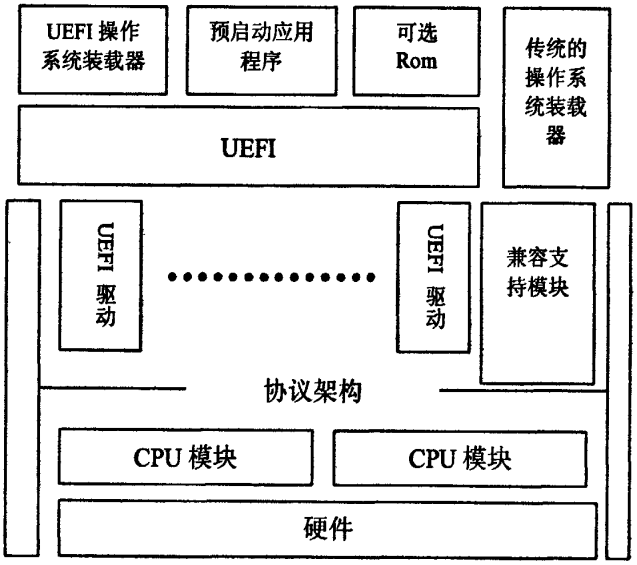

图2.1 UEFI架构

产生的问题。因为 UEFI 的可扩展性，开发人员可以根据需求为 UEFI 增加功能模块，如自动化配置程序，故障检测和故障报告等等。UEFI 最突出的功能是它支持 TCP/IP 网络架构协议，因此，用户可以在 UEFI 界面下连接因特网，使用网络资源，进行远程诊断。本文的主要工作就是针对 UEFI 支持的网络特性，更好的保证其网络通信的安全。

2. UEFI BIOS 的启动过程分析

图 2.2 给出了一个该架构基础上的固件的启动过程中各个阶段的描述图[5]。启动阶段要经历 PEI 到 DXE，从 DXE 阶段到 BDS 阶段，最后启动操作系统的过程。可以简单的将 UEFI 的工作模式归纳为：启动系统，初始化标准固件平台，然后加载 EFI 驱动程序库以及执行相关程序，在 EFI 系统启动菜单中选取所要进入的系统并向 UEFI 提交启动引导代码，正常的话将进入系统，否则将中止启动服务并返回 UEFI 系统启动菜单。启动过程具体如下。

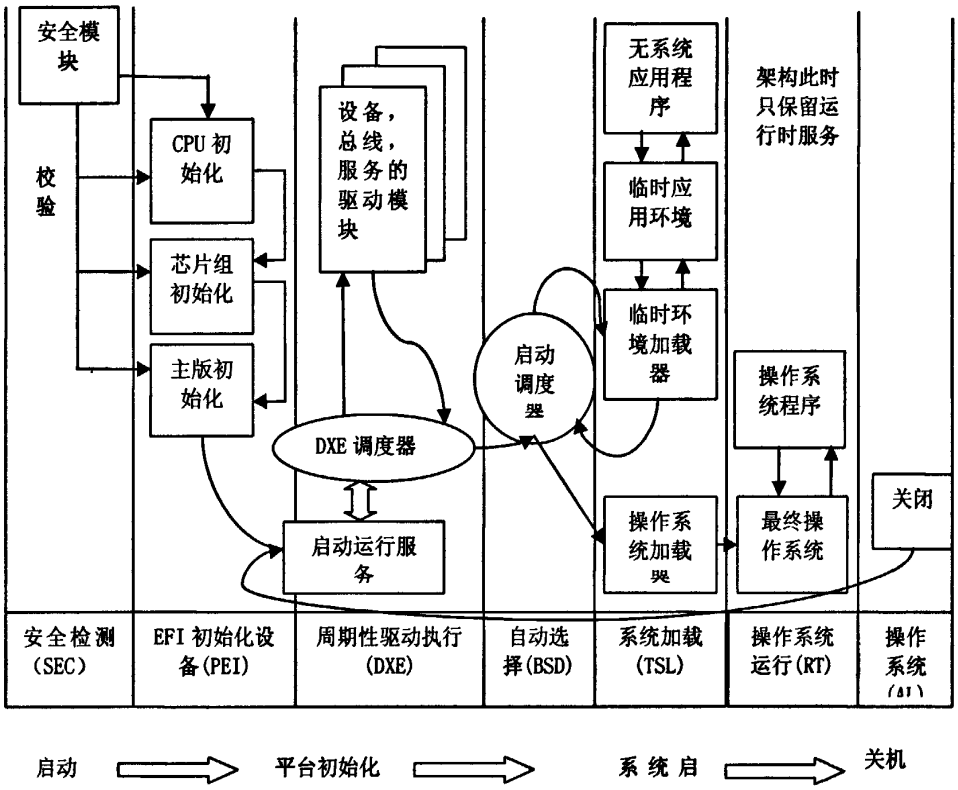

图 2.2 UEFI BIOS 的启动流程

开机安全检查后便进行系统固件的初始化。PEI(Pre-EFI Initialization)阶段激活最小量的C代码来查找和初始化内存及其他的资源，以把执行权交给C语言代码。PEI代码完成查找和初始化内存的基本工作，大多数芯片集和其它组件的初始化工作在驱动执行环境DXE下才开始进行。一旦发现内存，PEI交付状态信息给DXE使用，用这些内存来描述平台资源映射表并进行初始化然后跳转到DXE的基本代码。

在DXE阶段完成UEFI应用程序和驱动程序的加载。通过对UEFI映像的加载完成了UEFI环境的建立。在DXE中的EFI驱动可以在Pre-boot环境下实现硬件设备管理并产生相关的服务。同时EFI驱动包含平台中的大多数代码，它能枚举和初始化平台组件，并负责把管理和控制权交付给最后的操作系统。

当UEFI操作系统载入程序准备好后，它关掉UEFI启动服务，然后调用操作系统载入程序启动操作系统。其中，UEFI操作系统载入程序和真正的操作系统载入程序是不同的应用程序。前者负责调用UEFI的启动代码为后者做好准备，然后由后者来载入操作系统。如果UEFI 操作系统装载器或者UEFI定义的系统分区存在，UEFI便引导系统。图2.2 给出了UEFI BIOS从开机启动到关闭的整个执行流程。

3. UEFI BIOS 的优缺点

与传统BIOS相比，UEFI BIOS在很多方面存在优越性。

UEFI BIOS的操作界面更直观，运用更方便；驱动模块的可扩展，可以使UEFI BIOS的功能更完善；容错和纠错能力更强。与传统BIOS明显不同的是，UEFI是用堆栈方式传递参数、动态链接来构建系统。UEFI BIOS比传统BIOS更易于实现，从而缩短了系统研发的时间。更加重要的是，UEFI BIOS运行于64位模式并兼容32位模式，突破了传统16位代码的寻址能力限制，实现了现阶段处理器的最大寻址，从而加快了BIOS代码运行的运行速度。除此之外，它的优点还表现在驱动开发模型统一，速度快、兼容性好等方面。

UEFI 在实现全面革新的同时也存在很多缺陷[18]。

(1) UEFI 用硬盘上分割出的一个小分区作为自身的存储空间。存储容量的大小在很大程度上限制了 UEFI 功能的扩展。这也是本文设计的一个重要考虑点。容量的限制，要求加密算法占用的空间要尽量小，在这种情况下选择 ECC 算法是必要的。后面章节将有更加详细的介绍。同时，硬盘的不稳定性，会造成了 UEFI BIOS 的不稳定，这是未来 UEFI BIOS 发展需要考虑的方面。

(2) 基于软件系统设计的 UEFI 对病毒、黑客的抵抗力相对于传统 ROM 中的 BIOS 有很大程度的减弱。而且 UEFI BIOS 程序代码基于普遍应用的 C 语言编写而成，所以很多人都可以很容易地进行破译。这给 EFI 的安全性提出了更高的要求。

(3)UEFI BIOS 增加了对 TCP/IP 协议的支持功能，但是没有考虑相应的通信安全保护机制，使其在进行网络通信过程中更容易受到黑客的攻击。

安全架构方案在网络的基础架构方面影响很小，但是它需要安全通信双方装载相应的驱动程序，这对容量有限的 UEFI BIOS 来说是一个很大的负担。同时，安全方案的引入使通信双方通信的质量受到影响。密钥的产生和加密过程的完成都会在通信的时间上造成一定程度延迟。安全协议所需要的计算成本和空间很大，因此选取恰当的安全通信机制，保证网络的通信安全显得尤为重要。在网络安全传输机制中，对网络性能的评估和网络架构的设计直接影响整个网络架构性能。

本文针对 Pre-boot 环境下通信的安全方面提出了 IPSec 的保护机制。从而在一定程度上克服了 BIOS 中通信安全没有保障的隐患。同时，ECC 加密算法的引入也能适应 Pre-boot 环境容量有限的限制，克服通信时间延迟过大的缺陷。

2.2.2 Pre-boot 环境下 TCP/IP 架构分析

UEFI BIOS采用高级语言编写，它拥有先进的驱动模式[32~35]。在UEFI架构中，计算机在不进入操作系统时就能实现很多附加功能：支持TCP/IP协议和通用网络设备驱动，不用进入操作系统便可以浏览网页。用户可以在UEFI界面中使用各种网络

资源。后面章节中将具体分析UEFI BIOS中TCP/IP协议的工作原理。

1. 网络组成架构

OSI网络模型由国际标准化组织创建，该模型提供了通信任务应该遵守的一些指导性建议。OSI分层思想将网络分为七层包括物理层、数据链路层、网络层、传输层、会话层、表示层和应用层。OSI给每一层都分配了一组特定的功能。每一层都使用下层的服务并为其上层提供服务。

7	应用层	应用处理协议
6	表示层	
5	会话层	
4	传输层	TCP/UDP
3	网络层	IP
2	数据链路层	数据链路层
1	物理层	物理层

图 2.3 ISO七层模型和TCP/IP模型

TCP/IP协议栈表示了与OSI模型类似的网络体系结构。TCP/IP层次模型共分为四层：应用层、传输层、网络层、数据链路层。如表1所示，本文主要基于TCP/IP模型设计。

在通信过程中，数据由应用层开始逐层封装如图2.3所示[6]。用户数据在应用层加上应用层首部，然后封装的报文被传递给传输层。传输层添加TCP或者UDP首部后将数据传递给网络层。在网络层，数据被添加上IP头，随后数据报便被转交给数据链路层封装成帧，通过物理层以比特格式发送出去。

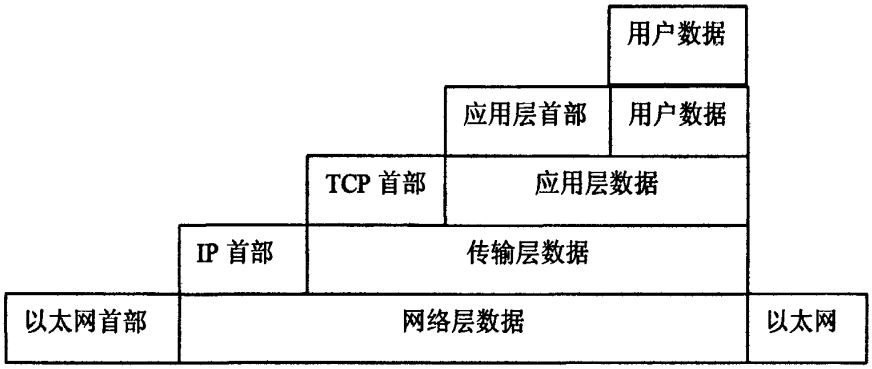

图 2.4 数据进入协议栈时的封装过程

2. 基于 TCP/IPv4 和 TCP/IPv6 网络结构分析

在 TCP/IP 协议栈中网络层 IP(Internet Protocol)协议面临从 IPv4 向 IPv6 的过渡。IPv6（Internet Protocol Version 6）是 IETF 设计的下一代 IP 协议版本，用于替代现行 IP 协议版本 4。目前使用的第二代互联网基于 IPv4 技术，它的最大问题是网络地址资源有限。IPv6 相对 IPv4 最大的优势是网络地址几乎是无限的。其次如果每个人都可以使用不止一个的 IP 地址，网络的安全性能将会有根大提高。

在早期的 UEFI 架构中，IPv4 的协议组成框图如图 2.4 所示[2]。在该 Pre-boot 环境中定义了很多网络相关的协议。传输层协议包括 TCP 和 UDP，网络层协议为 IP，同时，在网络层中，兼容 ARP 协议。通过网络管理层实现网络系统管理。UEFI BIOS 通过 Pre-boot 执行环境基本码协议 PXE BASE CODE Protocol 来定义图 4 所示 TCP/IP 架构访问网络方式并定义了如何访问通过网络启动的，与 PXE 兼容的设备。PXE 为远程设备启动提供了一个很可靠的从启动管理器获得控制的方式即提供了一个远程访问设备的网络通信方式，该协议通过调用网络的各层实现通信。UEFI_PXE_BASE_CODE_PROTOCOL 用来控制 PXE 兼容设备。关于这些设备的特性在 PXE spec 里面有定义，这里不再叙述。为了进行封装层的传输 EFI_PXE_BASE_CODE _PROTOCOL 被放在网络管理层协议 EFI_MANAGED_NETWORK _PROTOCOL 协议的顶部。EFI_PXE_BASE_CODE_PROTOCOL 句柄也支持在简单的网络中实现的 EFI_LOAD_FILE_PROTOCOL 协议。

启动完整性服务协议(Boot Integrity Services Protocol)BIS 通过检测数据块中与

数据认证相关的数字签名来保证数据完整性并进行相关的认证检测。BIS 可以被应用程序直接调用。网络管理层中的管理网络服务绑定协议（MNSBP）和管理网络协议（MNP）提供最初的 I/O 口封装的网络同步的相关服务，这些服务有利于多事件驱动，应用层访问控制和系统网络接口的应用。MNSBP 用来定位被 MNP 驱动所支持的通信设备，并为潜在的通信设备建立和销毁 MNP 的实例。EFI_VLAN_CONFIG_PROTOCOL 协议为 VLAN 配置提供可管理性接口。为消息发送者设立的 EAP Protocol 协议使 EAP 的架构可配置和扩展。EFI_EAP_PROTOCOL 被用来为 EAP 架构配置所期望的 EAP 认证方法和通过在端口中注册新的 EAP 认证方法扩展 EAP 架构。在 UEFI 中的 TCP/IP 架构设计之初功能还不完善。

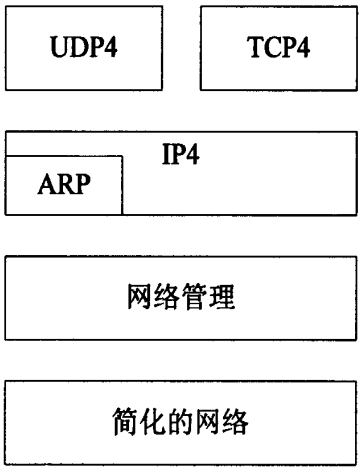

图 2.5 基于 IPv4 的 UEFI 网络层次结构

随着 Pre-boot 环境下，网络架构的不断完善。最新的 TCP/IP 协议架构如图 2.5 所示。TCP/IP 的五层实现分别为：物理层，数据链路层，网络层，传输层和应用层。在本文的 UEFI BIOS 环境中，分别实现了 IPv4 和 IPv6 的网络通信架构。本文着重针对 IPv6 进行分析。

应用层是实现了对 MTFP,DHCP 等协议的支持；传输层协议包括 TCP 和 UDP，负责为发送者和接收者之间提供应用程序的进程间通信机制，其中 TCP 实现了面向连接的服务，UDP 实现了面向无连接服务，并且都实现了对相应上层协议的支持；网络层主要使用 IP 协议。IP 协议是网络层的协议，也是计算机网络通信的最重要协议。网络层实现了将数据独立的从数据发送者传输到数据接收者功能，该层主要实现了拥塞控制，路由选择和网络互联等功能。因此，该层也是 TCP/IP 体系结构的核心；通过硬件实现的链路层负责将 IP 数据报封装成帧格式进行传输。本文研究内容主要针对传输层和网络层分析，不涉及链路层和物理层相关原理，因此，这里不

对这两层做详细分析。

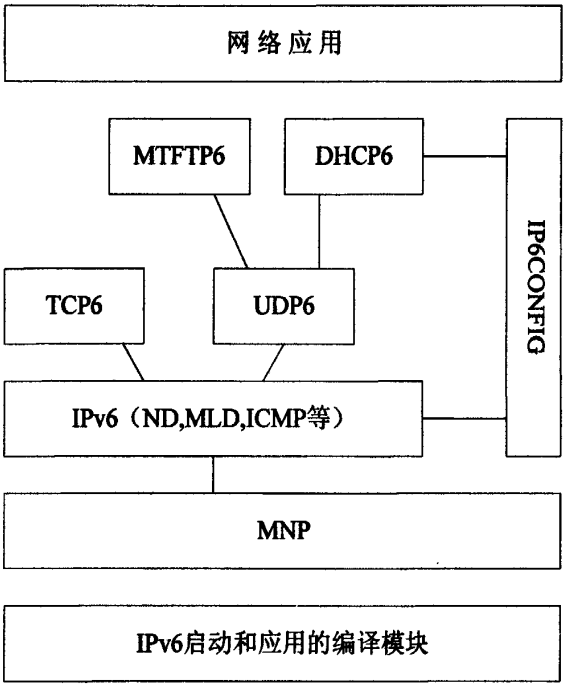

图2.6 UEFI IPv6支持架构

对于传输层，UEFI 的 TCPv6 (Transmission Control Protocol version 6) 中 UEFI_TCP6_SERVICE_BINDING_PROTOCOL 用来定位 UEFI TCPv6 协议驱动，从而建立和消除驱动的子程序，以用 TCP 协议来实现和其他主机进程间的通信。需要 TCPv6 的输入输出服务的网络可以在应用层调用其中一个协议句柄来产生服务，服务会查找设备并公开 EFI TCPv6 服务绑定协议 GUID。成功调用 EFI_TCP6_SERVICE_BINDING _PROTOCOL.CreateChild()函数之后，驱动在信息配置状态结束后进行数据包的接收和发送。而且该调用必须有对应的 EFI_TCP6_SERVICE_BINDING_PROTOCOL.DestroyChild() 函数调用。用 EFI TCP6 Variable 来维护通信中的 IPv6 地址和端口号。EFI TCPv6 Protocol 提供了发送和接受数据流的服务。EFI_TCP6_PROTOCOL 定义了 EFI TCPv6 Protocol 子协议，用来在子驱动中发送和接受数据流。它能在指定的端口监听或者连接到远程的对等客户端。每个实体都有自己的独立通信信息设置如路由器信息等。

UEFI IPv6 (Internet Protocol version 6) 协议接口可以分为以下几部分：UEFI IPv6服务绑定协议；UEFI IPv6变量和UEFI IPv6协议。UEFI IPv6 协议提供了基于IPv6封装格式的网络I/O服务，包括支持Neighbor Discovery Protocol (ND)协议，广播监听发现协议协议(MLD)和一系列的因特网控制消息协议 (ICMPv6)。UEFI IPv6

Protocol实现了一个简单的封装接口，通过驱动用来传输和接受网络包。UEFI_IP6_PROTOCOL 定义了一系列IPv6和ICMPv6类型的服务。关于IP协议的细节内容后面文章将有更加详细的介绍。

UEFI UDP (User Datagram Protocol) 协议定义了与UEFI IP协议通信的接口和建立在UEFI UDP Protocol 之上的应用层接口。UDP6协议 UEFI_UDP6_PROTOCOL 提供了简单的封装性的服务来传送和接受UDP包。UEFI UDPv6 Protocol Session 可以被任何网络驱动所应用。

综上所述，UEFI BIOS支持IPv6的相关协议。网络通信的各层通过接口，协议和服务实现各个网络层次上协议间的通信。

2.2.3 IPv4 和 IPv6 网络安全性分析

在TCP/IP架构中增加网络安全特性会降低网络通信速率，在UEFI BIOS的TCP/IP的协议族的实现中，没有考虑网络安全因素，因此TCP/IP架构本身是不安全的，很容易遭受黑客袭击，被窃听或者欺骗，基于TCP/IP通信架构的各项服务也受到不同程度的安全威胁。这些弱点给网络通信带来了很多不安全隐患。

在安全性上，IPv6与IP安全(IPSec)体制和服务密切相连。目前虽然两种IP标准（IPv4和IPv6）都支持IPSec协议，但是IPv6将IPSec作为自身标准的组成成分，在安全机制的设置更加协调，相关接口规范更加统一的情况下实现的。而IPv4的安全体系是通过累加方式实现的。

IPSec是开源的协议规范。对等实体间通信时IPSec可以为通信数据提供机密性，完整性性，认证等安全方案。不管是在IPv4通信体系还是IPv6通信体系中，网络层通信安全服务都由IPSec提供。但是IPv6对通信的安全性要求更高。IPv6必须支持IPSec所提供的安全方案。本文的设计就是基于IPv6通信体系。

IETF通过一系列的RFC定义了IPSec相关协议标准。IPSec为IPv6的通信提供加密或者身份认证等安全保护，从而实现了企业内部网的永久性的远程接入。除了这种强制性安全保护体系外，IPSec提供用于保证数据的一致性认证报头(AH)和用于保证数据的保密性和一致性的封装的安全负载报头(ESP)。在IPv6包中，AH和ESP都是作为报头的扩展功能，可以一起使用，也可以仅使用一种。虚拟专用网是IPSec的一项重要应用，IPv6内部实现了虚拟专网的功能。

在信息化的今天，因特网越来越普及。网络通信使用的 IP 协议在设计时很少考虑安全性使因特网的安全隐患越来越明显的暴露出来。主要隐患为：黑客容易伪造 IP 数据报的地址。本文主要针对 IPv6 通信中存在的不安全因素，通过设计 IPSec 安全框架，实现了 ECC 加密算法在安全机制中的运用。关于 ECC 算法的分析，后面

文章会有详细说明。

2.3 椭圆曲线加密算法分析

本文选择IPSec的重要原因是它的安全和数据机密性。机密算法是用于对数据进行加密和解密的一对数学函数：一个函数用来对数据进行加密，另一个函数用来对数据进行解密。现在使用的加密方法中，数据的安全性保障需要密钥来实现。加密的算法分为两种类型：对称加密算法和非对称加密算法。

在对称加密算法中，消息的发送方和接收方协商后使用相同的密钥进行交互。常见的对称加密算法包括数据加密标准(DES),3DES和高级加密标准(AES)。非对称加密算法也叫公钥算法，它使用两个密钥：一个是加密密钥，另一个是解密密钥。加密密钥是可以公开的也被称为公钥；用于解密的私钥需要进行保护。

其中，公钥加密方法很少用来对消息进行加密，因为它的加密速度比对称加密慢很多；然而，公钥加密算法可以用来解决对称密钥算法中对称密钥的分发问题，然后再使用对称加密算法产生的密钥对消息进行加密。本章下面介绍ECC加密算法。ECC属于公钥加密体制。

2.3.1 ECC 基本架构分析

椭圆椭圆曲线是指由韦尔斯特拉斯(Weierstrass)方程

$$y_2 + a_1 * x * y + a_3 * y = x_3 + a_2 * x_2 + a_4 * x + a_6 \tag{2.1}$$

所确定的平面曲线[7]。若 F 是一个域，$a_i \in F$，i=1, 2, …, 6… 满足式的数偶 (x, y) 称为 F 域上的椭圆曲线 E 的点。F 域可以是有理数域，还可以是有限域 $GF(P_r)$。椭圆曲线通常用 E 表示。椭圆曲线的组成中，除了曲线 E 上的所有点，还需要再加上一个叫做无穷远点的特殊点 O。

在椭圆曲线加密(ECC)中，利用了某种特殊形式的椭圆曲线，即定义在有限域上的椭圆曲线。其方程如下：

$$y_2 = x_3 + a * x + b(\mathrm{mod}\, p) \tag{2.2}$$

这里 p 是素数，a 和 b 为两个小于 p 的非负整数，它们满足：$4 * a_3 + 27 * b_2(\mathrm{mod}\, p) \neq 0$ 其中，$x, y, a, b \in F_p$，则满足式(2)式的点 (x, y) 和一个无穷点 O 就组成了椭圆曲线 E。

椭圆曲线离散对数问题(ECDLP)基本原理总结为：已知素数 p 和椭圆曲线 E，对于等式 $Q = k * P$，基本问题是已知 P, Q 求解小于 p 的正整数 k。已经证明，已知 k

和 P 计算 Q 相对简单，而由 P,Q 很难推算出 k，目前为止不存在很好的方法来进行解的计算，这就是椭圆曲线加密算法的本质所在[37~43]。

椭圆曲线计算过程就是对等式 $Q=k*P$ 进行点加和点乘运算的过程[41]。本文设计主要运用将点乘转换为点加和倍点运算的方法实现。本文采用平方-乘算法实现具体的运算功能。平方-乘运算运算的原理就是将参数 k 化为二进制序列表示：$a_0,a_1...a_{l-1},a_i \in \{0,1\}$，其中 l 为 k 的二进制表示中的最高位，然后将 k 进行二进制展开，则 k 表示为式(2.3)：

$$k=2^{l-1}+a_1 2^{l-2}+...+a_{l-2}2+a_{l-1} \tag{2.3}$$

对于标量乘，可以表示为式(2.4)：

$$\begin{aligned} kP &= (2^{l-1}+a_1 2^{l-2}+...+a_{l-2}2+a_{l-1})\ P \\ &= 2(...(2(2(2P+a_1P)+a_2P)+a_3P)+...+a_{l-2}P)+a_{l-1}P \end{aligned} \tag{2.4}$$

平方-乘算法实现过程简单概括如下，其中 P 为椭圆曲线基点，Q 是与 P 相同数据结构的椭圆曲线上一点：

(1)将基点 P 作为初始值赋给另一个椭圆曲线类型的点 Q；

(2)从 1 到 l-1 计算 Q=2Q 和 a_i=1 时计算 Q=Q+P；

(3)输出点 Q

则点 Q 就是最终点乘的结果。

2.3.2 ECC 加密算法的优势分析

椭圆曲线算法与RSA算法的相比，存在很多方面的优势。RSA 加密算法有可能被椭圆曲线公钥系统所代替[8]。椭圆曲线加密方法与 RSA 方法相比，主要优势体现在：

(1) 椭圆曲线加密算法的安全性能更高，如表 2.1 所示。位数很少的 ECC 就可以达到很多位 RSA 的密码强度，如 160 位 ECC 算法与 1024 位 RSA 算法具有相同的安全性。

(2) 椭圆曲线计算量更小，密钥产生和加密速度更快。在私钥的产生和使用中，椭圆曲线的加密，解密和签名速度更快，ECC 密钥处理比 RSA 快得多。

(3) 椭圆曲线加密算法占用的存储空间小，密钥大小和体系架构中的参数相比 RSA 小很多。所以椭圆曲线加密算法占用存储空间更小。

(4) 椭圆曲线加密算法中密钥的大小决定了传输所需带宽低，这一特性使得 ECC 算法应用前景更广泛。

表2.1 RSA和ECC的速度比较

功能	Security Builder 1.2	BSAFE 3.0
	163位ECC(ms)	1024位RSA(ms)
生成密钥对	3.8	4708.3
签名	2.1(ECNRA)	228.4
	3.0(ECDSA)	
认证	9.9(ECNRA)	12.7
	10.7(ECDSA)	
Diffie-Hellman密钥交换	7.3	1654.0

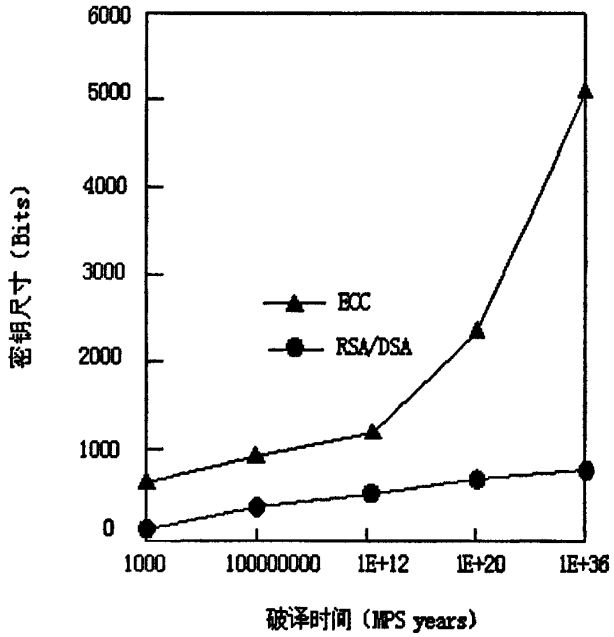

图2.7 密钥长度和破译时间

ECC 的这些特点使它必将取代 RSA，成为通用的公钥加密算法。

2.4 本章小结

本章分析了组成 UEFI BIOS 架构的模型及启动的各个阶段：PEI,DXE,BDS 等，并对其中 TCP/IP 协议的各组成模块的进行了详细的描述。通过本章的分析，可知在 UEFI BIOS 的通信过程中，存在很多安全方面的漏洞，安全架构 IPSec 的引入能在很大程度上改善其安全方面的性能。

本章还分析了ECC算法相对于其它算法在生成密钥和加密方面的优势所在。ECC能用更少的时间和空间来产生所需要的通信过程中的密钥。下面章节将对其架构进行相应的设计和实现。

第三章 基于 IPSec 的 UEFI BIOS 系统安全架构设计

3.1 引言

通过第二章对UEFI BIOS环境下，TCP/IP通信安全性能的分析，可知在BIOS的网络通信中存在着很多安全方面的隐患。本章重点进行IPSec在Pre-boot环境下的架构和代码设计。同时，通过设计展示出IPSec在保障UEFI BIOS的网络安全方面的良好性能。本章还重点介绍了ECC加密算法的具体实现方法和实现策略。

3.2 Pre-boot 环境下 IPSec 基本架构设计

本文设计的Pre-boot环境下IPSec架构如图3.1所示。架构设计主要包括Pre-boot环境下TCP/IP网络安全接口设计，IPSec协议AH和ESP协议栈设计，IPSec功能数据库模块设计和IKE密钥协商方案设计。通过网络接口设计，解决了IPSec与TCP/IP网络的连接问题。针对IPSec本身功能模块AH(Authentication Header)，ESP(Encapsulating Security Payload)和IKE(Internet Key Exchange)等协议[2]方案的设计，实现了IPSec的基本组成架构，为安全机制建立打下了基础。各种与安全传输相关的数据库的设计保证了IPSec功能正常的运作。网络协议族和IPSec安全协议协调工作，使网络通信安全得到了保障。本文假设通信双方发送方为Alice和接收方为Bob。

在实现中，IPSec在通信主机的网络接口间建立一个边界。进行通信的过程中，穿越IPSec边界时必须访问被指定的（由用户或者控制者指定）配置IPSec的控制器即本文的SPD（后面章节将有详细描述）。控制器决定这个穿越边界的数据包是否被阻止，是否通过AH或者ESP支持安全服务还是被丢弃。IP层通过选择恰当的安全协议、加密算法和加密密钥来提供安全服务。本文IPSec在由Hub连接的主机的网关中实现从而保证IP通信的安全。下面结合通信过程中双方的工作过程阐述IPSec设计与实现过程。

本文IPSec架构设计中，使用inbound代表由不受保护接口进入IPSec的通信或者由未保护边界向保护接口发出的消息。outbound是从保护接口进入IPSec实现的通信，或者是从保护边界发出直接到非保护的边界的通信。IPSec实现可以支持多个接口，在通信的双方边界都实现IPSec。Discarding将丢弃数据包，它是在边界两边实现的。BYPASS则允许不受保护的数据包直接穿过边界。IKE有保护加密密钥和安全管理功能。IPSec选择性支持IP压缩包的谈判。

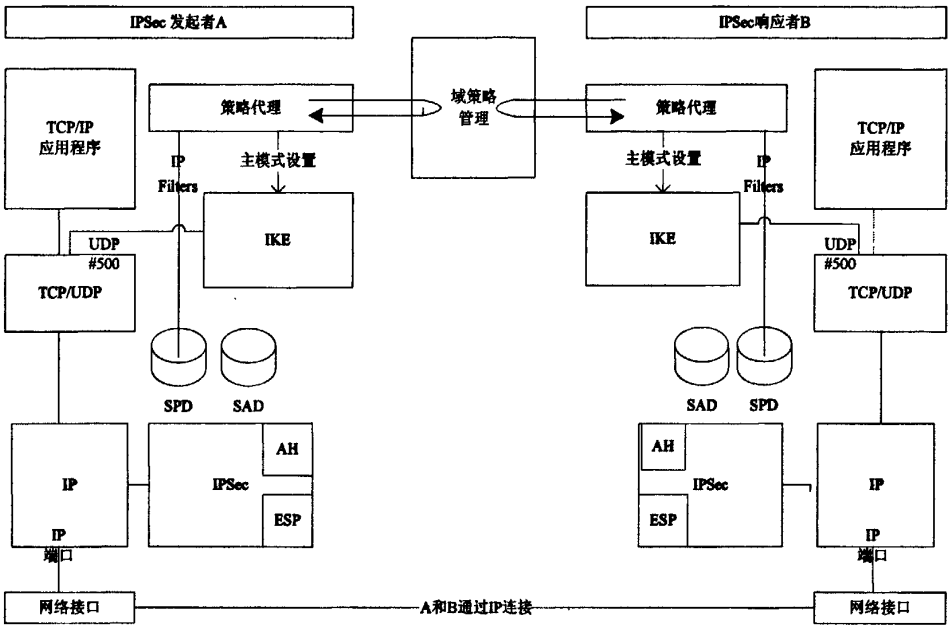

图 3.1 IPSec 通信架构

首先，发送方 Alice 给 Bob 发送数据报：

(1)源主机 Alice 的 UDP 层通过调用 IP6output()函数，将数据传输给 IP 层，令封装数据报并发送该数据报。数据报用标志位设置 IPSec 相关信息；

(2)针对发送方发送给目的主机的数据报，网关查询安全策略数据库 SPD。根据报文所设置的安全策略查询安全联盟数据库 SAD，如果 SAD 有对应的安全联盟 SA，则强制加上 AH 或 ESP 头，进行 AH 或 ESP 协议封装；

(3)如果 SAD 没有对应的 SA 则进行 IPSec 密钥交换 IKE 处理。根据安全策略建立新的 IKE SA 和 Child SA；

(4)在 SA 建立后，在 SPD 中增加与 SA 对应序列号字段，从而在策略关联数据库 PAD 中建立 SAD 和 PAD 的关联；

(5)通道模式的处理：本文的设计中，IKE 实现包含传输模式和隧道模式两种传输模式。传输模式与隧道模式在封装数据时存在差别，后面具体 IKE 设计中将会详细阐释。本文采用传输模式实现 ECC 算法密钥处理功能；

(6)发送方 Alice 向下层传递封装好的数据，通过链路层封装成帧，然后由物理层将安全的数据包发往目的主机 Bob。

接收方 Bob 的处理过程：

(1)另一端的网关收到这个包，剥去额外的 IP 头，并利用数据报的 AH 或 ESP 头的填充信息根据 SPD 调用接收方 Bob 所在网络架构中的 IPSec 层。如果是 IKE 协商所用数据报，则不会有 AH,ESP 封装，系统直接将数据上传给 IPSec 处理；

(2)IPSec 从 AH 或 ESP 头中取出携带的 SPI 信息，从 IP 头中选出源主机和目的主机的地址及使用的相关协议；

(3)IPSec 层用步骤(2)中获得的参数从 SAD 中取出对应的 SA，如果 SA 不存在，就数据报就会被丢弃；

(4)SAD 将查找到底 SA 返回，IPSec 将会按照 AH 和 ESP 协议定义的规则对这个包进行处理；

(5)对数据报的策略进行验证，进而判定 IPSec 对应用的处理是否正确。文中的策略是通过选择器查询 SPD 得到；如果验证正确，那么解密后把这个包被目的主机接收，否则丢弃该包。

3.3 IPSec 内部组成架构设计

本节主要完成 IPSec 内部协议的整体设计。在实现中，用两个协议 Authentication Header (AH) 和 Encapsulating Security Payload (ESP) [10] 为安全通信提供服务（如图 3.2 所示），AH 和 ESP 都提供访问控制功能，ESP 提供密钥的分发，加密并管理 SPD 中的通信数据。用 IKE 进行动态密钥的协商和通信安全联盟的建立。其中还涉及各种策略的建立和所需数据库之间的信息交互，本文后面将会详细阐释。

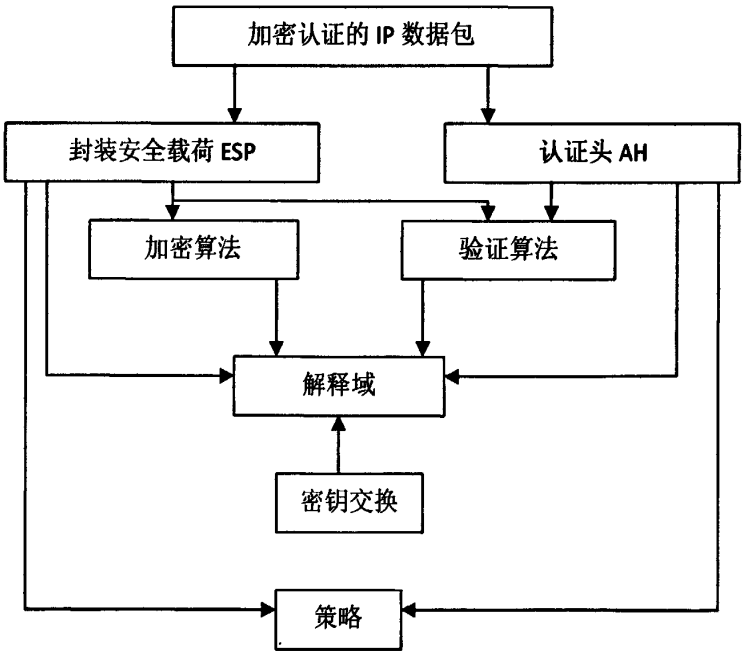

图 3.2 IPSec 架构

本文设计的 IPSec 中的 AH 协议为 IP 层的数据报提供通信双方身份验证、数据

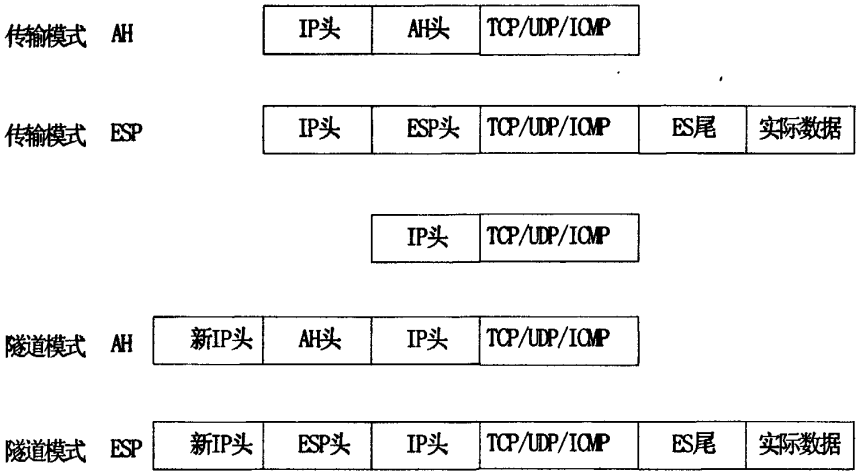

图3.3 AH和ESP插入数据包

完整性和抗重放等功能，并验证整个数据报，但不加密数据报，因此不提供保密机制。ESP 协议为数据报提供身份验证、数据完整性、抗重放及加密保护等功能。在传输模式实现中，ESP 只保护传输的数据而不保护传输的 IP 报头，如图 3.3 所示。本文设计也支持 IKE 协议。IKE 协议按照密钥交换中的相关标准实现，结合了 Oakley 和 SKEME 两种协议，IKE 设计框架符合 ISAKMP (Intemet Security Association and Key Management Protocol)协议的相关标准。IKE 协议为 IPSec 体系实现了密钥的动态协商，它能动态建立安全联盟，为通信双方协商安全通信所需要各种参数，例如加密算法的选择，会话使用的密钥产生和分发，通信双方的身份验证等。IKE 通过两个阶段协商过程建立 ISAKMP SA 和 IPSec 安全关联（IPSec SA）。下面针对 IPSec 中的各个模块，说明设计思路。

UEFI BIOS环境下的IPSec设计需要相关组件的具体实现。其中，包括不同传输模式下IPSec在数据报中的位置，组成IPSec各个安全协议和数据库的设计以及通信过程中相关控制方式的设置。如图3.3所示，本文将阐述传输模式下设计过程中的主要内容。

3.3.1 通信模式设计

本文设计的IPSec中每种协议都支持两种通信模式：传输模式和隧道模式。在传输模式中实现的AH和ESP主要为底层协议提供通信安全保障；在隧道模式中实现的AH和ESP被用在隧道模式的IP数据报中。本文采用在相互通信的网关间建立通信通道的方法保证网络安全，从而实现了网络的安全服务功能。其中，IPSec的加密功能

的实现需要相应的加密密钥来支持，通信双方的密钥交换是通过IKE来实现的。本文ECC算法主要在传输模式通信中设计，实现了通信协议和通信服务的有机结合。

3.3.2 AH 模块设计

本节设计AH协议模块，它主要用来为IP层报文提供数据完整性校验和通信双方的身份认证。此外，它还有可选择的重放攻击保护功能。但是AH不提供与数据加密相关的服务，设计符合RFC4302标准[9]。ESP则为报文提供数据完整性校验，身份认证，数据加密以及重放攻击保护等，即除提供AH所提供的服务外，还提供机密性服务。关于ESP的实现，下一节会具体介绍。在不同模式下，AH的插入方式如图3.4所示。

实际上，在数据出现在数据报中之前，AH 便已包含数据。因此，即使在传输模式下，AH 提供的保护也会包含一些 IP 数据包头。在隧道模式下，整个数据报处于 IPSec 数据包头的保护之内。数据报由外部 IPSec 数据包头以隧道模式保护。

数据包进行 AH 保护后，则除可变域之外的所有数据都受 AH 的保护。相应的字段信息如图 3.4 所示。AH 头插入在 IP 头和传输层数据之间，对整个除可变域之外的所有数据进行认证。AH 协议端口号是 51，因此 IP 协议字段对应值为 51。

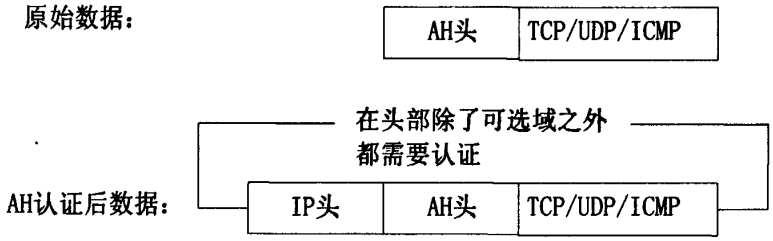

图 3.4 AH 认证模式头

AH插入数据报之后的各字段信息的填充方式如图3.5所示。首先是IP头，主要包括IP协议版本字段，流量类型，流标签。其中，IP版本初始值是IPv6。流量类型和流标签主要实现了数据包的标识作用，从而加速传输，这里不做讨论。还有有效载荷长度，下一报头和跳数限制等，这三个字段的功能相对IPv4进行了扩展，本文对此不做详细解释。

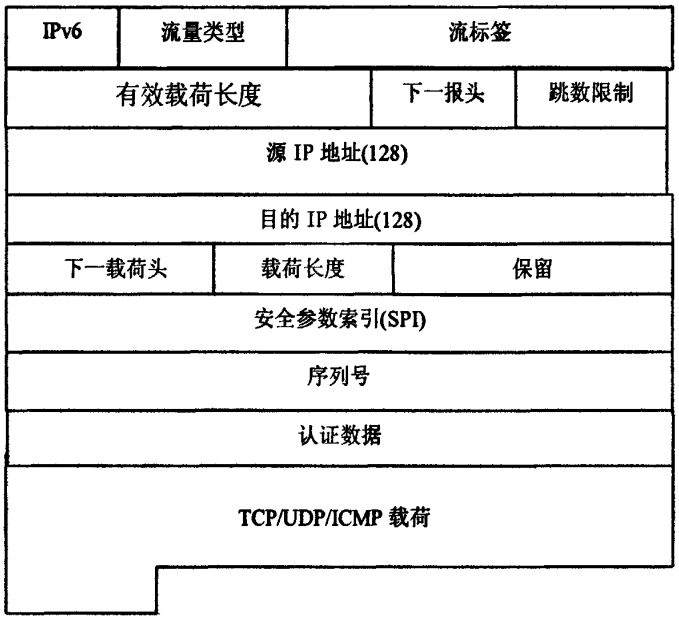

图 3.5 AH 填充后数据报字段表示

其次是IP数据报的源和目标地址用来记录通信双方的地址信息，IPv6地址兼容IPv4地址。然后是AH头，它主要包括安全参数索引(SPI)，序列号等控制AH的标志信息。用SPI进行SPD数据库中SAD安全联盟的信息检索，关于SPI文章后面会具体描述其功能。认证数据包含AH协议的认证信息，从而保证数据的安全。AH协议字段加上传输数据便组成了AH数据包。数据报封装后送至链路层，通信双方便开始进行通信数据的传输。

3.3.3 ESP 协议设计

本节设计实现 ESP 协议。在图示 3.6 所示的 ESP 实现中，ESP 头被插入在 IP 头和传输层协议之间，ESP 尾插入在传输层协议和所传输的数据之间。协议对从 ESP 头到 ESP 尾的所有数据实现认证保护，同时，对 ESP 头后面到实际传输数据之前的所有字段进行加密保护。加密功能的引入，使 ESP 协议能更好的为网络安全通信提供保障。ESP 协议号为 50[10]。认证数据包含了一个完整性校验值(ICV)，它由 ESP 包的最小认证数据计算而来，该部分是可选域，只有当认证服务需要时才发挥作用。以上选择由软件实现，通过索引到 SPI 的哈希表来进行选择。下文将具体阐释 SPI 工作原理。

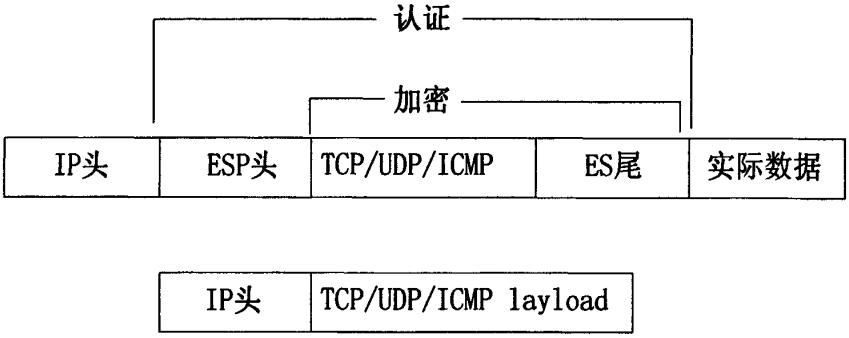

图 3.6 ESP 添加后数据报格式

ESP 协议在传输模式中的字段如图 3.7 所示。与 AH 协议的插入方式相同首先是 IP 头。IP 头定义了数据报长与协议类型等相关信息，该信息与 AH 协议基本相同，这里不再赘述。IP 协议头之后便是 ESP 头，包括 SPI，序列号等。之后是传输层协议头，如 TCP, UDP 等以及该层用到相关协议信息。然后是 ESP 尾的认证数据还有下一载荷的信息。最后则是最终需要传输的数据。

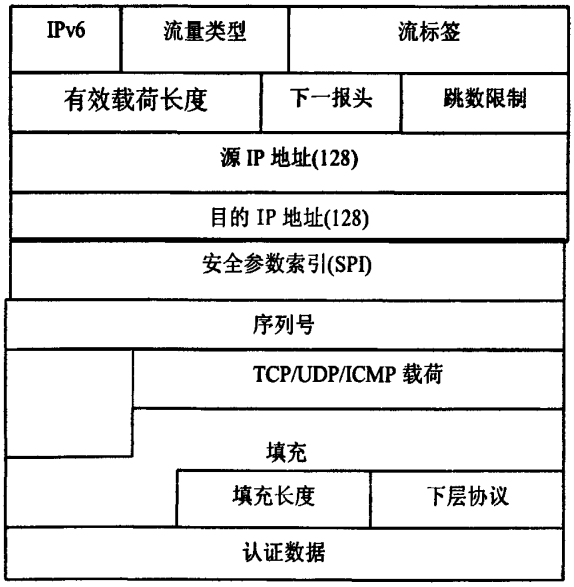

图 3.7 ESP 添加后数据报字段表示

3.3.4 SPD,SAD 和 PAD 设计

本节实现三个主要的数据库：安全策略数据库(SPD)，安全联盟数据库(SAD)和对等认证数据库 (PAD)。其中，SPD 决定了 inbound 或者 outboard 的 IP 通信信息的部署。SAD 包含了与建立的 SA 相关的参数。第三个数据库 PAD 提供了 SA 管理协议如 IKE 与 SPD 之间的连接关系。本文在设计 IPSec 过程中，各种功能实现遵循 Pre-boot 相关协议规范。其使用的数据库按照 RFC4301 的相应标准设计。

本文设计的 SPD 规定了 IPSec 进行通信传输中的安全策略信息。本文使用的安全联盟是一个管理架构，它可以为穿越 IPSec 边界的通信数据执行安全策略。安全联盟处理过程建立在 SPD 之下，SPD 指定了为 IP 数据包提供什么样的服务和以哪种方式提供这种服务。SPD 的入口函数设计为：

```
typedef struct _IPSEC_SPD_ENTRY {
  EFI_IPSEC_SPD_SELECTOR   *Selector;
  IPSEC_SPD_DATA           *Data;
  EFI_LIST_ENTRY           List;
} IPSEC_SPD_ENTRY;
```

在通信的过程中，所有通信的处理，包括 IPSec 管理通信协议(IKE)和不被 IPSec 保护的通信都要通过 SPD 入口访问 SPD。本文沿用选择器函数(EFI_IPSEC_SPD_SELECTOR)来决定选择哪个 SPD，收到数据报的接口作为函数的输入，函数的输出就是 SPD 的 ID。IPSEC_SPD_DATA 定义了与某个 SPD 对应的数据处理策略和 SA。EFI_LIST_ENTRY 则定义前向和后向列表，将 SPD 中存储的控制信息以列表的形式存储。通过 socket() 接口修改 SPD。通过函数 IPSecLookupSpdEntry()为穿越 IPSec 边界的每个数据报指定安全处理方式。

同样，本文也设计实现了 SAD。在安全联盟数据库 SAD 中，每个入口定义了跟 SA 相关的参数，每个 SA 在 SAD 中都有入口，对每一个 outbound 处理，SPD 缓存中的 SPD-S 部分的入口指向 SAD 入口。本文用函数

```
IPSecLookupSadBySpd (
    IN EFI_LIST_ENTRY              *SadList,
    IN EFI_IP_ADDRESS              *DestAddress
)
```

实现通过 SPD 对 SA 的查找。通过函数

```
IPSecLookupSadBySpi (
    IN UINT32                      Spi,
    IN EFI_IP_ADDRESS              *DestAddress
)
```

实现通过 SPI 和目标地址对 SA 的查找。IPSec 建立在安全联盟 SA 的基础之上。SA 的连接方法能为通信双方提供安全服务。一个 SA 只能针对 AH 或者 ESP，不能同时支持两种协议。在每一个通信需求中，IKE 负责精确的创建 SA 对。SAD 的入口标明了 SA 查找利用 SPD 或者 IP 地址，SPI 等。

设计中，安全联盟 SA 符合两种传输模式，传输模式和隧道模式。本文集中阐述传输模式下的通信。IKE 建立了密钥对，为了简化，我们选择两个 SA 有相同的模式。两个主机通过传输模式 SA 提供端到端的安全服务。如果需要提供安全方面的服务，则在传输模式路径上的机器就可以用传输模式 SA 实现该安全功能。

本设计方案中，实现的对等认证数据库(PAD)提供了 SPD 和 IKE 的连接机制。PAD 的入口设计为：

```
IPSecLookupPadEntry (
    IN UINT8                    IpVer,
    IN EFI_IP_ADDRESS           *IpAddr
)
```

它集成了几种重要的功能：实现 IPSec 通信双方的分类授权机制；通过访问 SAD 指定通信双方认证时使用的协议和方法；为通信双方提供认证数据；限制某些协议 ID 的类型和数值；规定通信双方网络的地址信息等。当通信双方访问 PAD 时，它可以为 IKE 协商提供所需要的功能。为了执行这些功能，PAD 为对等实体间的通信提供入口。一个入口指定一组通信双方。本文设计中，通过该入口指定了 IKE 的版本和认证方式即 IKE 的主模式，预共享密钥认证数据。

3.4 IPSec 在 UEFI BIOS 环境下的应用

本节在 Pre-boot 环境下实现 IPSec 通信安全模块。IPSec 作为 UEFI BIOS 的驱动加载进来。本节内容主要包括 UEFI BIOS 与 IPSec 之间的接口设计，IPSec 协议间信息的交互和通信过程中 IPSec 各个数据库的调用。IPSec 通过 EFI_IPSEC_CONFIG 实现与 Pre-boot 环境之间的通信。IPSec 内部协议通过上一节中定义的三个数据库实现相互之间的协调工作。下面阐述具体函数的定义。

Pre-boot 环境通过 EFI_IPSEC_CONFIG 对 IPSec 安全架构进行配置。其中，EFI_IPSEC_CONFIG_PROTOCOL 为 UEFI BIOS 的 IPSec 协议驱动定义了设置和获得安全和策略相关信息的机制。同时，UEFI BIOS 通过接口 EFI_IPSEC_CONFIG 的 HandleSa (IN IKE_UDP_SERVICE *UdpService, IN IKE_PACKET *IkePacket)实现了 Pre-boot 环境下的 UDP 和 IKE 间的通信，从而保证网络层的 IPSec 与传输层之间的通信。传输层可以通过 IKE_UDP_SERVICE 控制协商建立 SA 的性能。在 IPSec

内部，通信协商建立 SA 的过程由 IKE 完成，IPSec 通过 HandleIkeSaPacket()控制 IKE 通信所用的数据包。

首先，IPSec 配置协议通过 EFI_IPSEC_CONFIG_SET_DATA SetData 来设置 IPSec 的所访问数据库类型：安全联盟数据库 SAD 或安全策略数据库 SPD。该数据结构定义了各个数据库的接口和对请求的各种策略，同时 EFI_IPSEC_CONFIG_GET_DATA GetData 将 IPSec 所包含信息向下传输，实现了 UEFI 中 IP 层与 IPSec 的连接。UEFI 通过 EFI_IPSEC_CONFIG_GET_NEXT_SELECTOR GetNextSelector 保证了通信的连续进行。EFI_IPSEC_CONFIG_REGISTER_NOTIFY RegisterDataNotify 记录了通信过程中的各个数据的建立与撤销，从而起到标记的目的。这些功能通过 UEFI 专有的协议定义方式：EFI_IPSEC_CONFIG_PROTOCOL 来实现。

IPv6 配置协议通过 EFI_IP6_CONFIG 来设置与 UDP 和 IP 之间的通信接口。如果 IPv6 的驱动程序中 Ip6IPSecProcessPacket()接收到 IPSec 的使能信号，则调用 IPSec。通过 EFI_IPSEC_CONFIG 实现与 IPSec 的通信。具体的通信过程，上一节已经进行了详细阐释，这里不再阐述。

本文针对 IKE 第一阶段通信中 Diffie-Hellman 交换引入 ECP(ECC Groups)来改善 Pre-boot 环境下通信安全。关于 IKE 工作的具体细节，后面将会进行更加详细的阐述。

3.5 密钥库的设计

本节主要在RSA密码库的基础上实现ECC加密算法。通过定义ECC头文件和可执行程序，本节实现ECC算法密钥产生和加密的功能。然后将本文实现的ECC加密算法应用到的IPSec中，为IPSec保证安全性的同时，更能适应本文设计所使用的Pre-boot环境。该密钥库的实现，为密钥的产生和加密功能的实现提供了更多的可选方案。

3.5.1 密码库的组成结构

本节密码库设计符合基本的库设计规范。通过定义头文件和相应的执行程序代码来实现密码库功能。本节主要实现ECC算法密钥生成和加密功能。设计内容主要包括椭圆曲线参数定义，椭圆曲线点加，点乘运算函数设计，密钥生成和加密函数等。

首先，定义ECC的各个参数。椭圆曲线算法中用到的椭圆曲线参数主要包括曲线生成参数a,b，椭圆曲线的一点G，大素数和公钥，私钥等。用两个整型数a，b定

义构成椭圆曲线的参数为：

```
typedef struct _EC_CURVE {
    INTN                    *a;
    UINTN                   *b;
} EC_CURVE;
```

本节选取的参数a是负值，因此将a定义为有符号数。定义椭圆曲线上的任意一点为G(x，y)：

```
typedef struct _EC_POINT {
     UINTN                   *X;
     UINTN                   *Y;
} EC_POINT;
```

点G(x，y)的选取符合版本2.3中椭圆曲线生成规则。用数据结构定义参加计算的ECC算法的其它参数包括大素数值，基点等如下：

```
typedef struct _EC_PARAMETER {
    UINTN                   *Prime;
    UINTN                   *BasePointOrder;
    EC_CURVE                 Curve;
    EC_POINT                BasePoint;
} EC_PARAMETER;
```

通过该数据结构，就定义了参与求模运算的大素数Prime，基点的序，曲线和基点。通过_ECC_PUBLIC_KEY定义公钥：

```
typedef struct _ECC_PUBLIC_KEY
{
    UINTN                   keySizeInWords;
    EC_POINT                PubKey;
    EC_PARAMETER              Param;
} ECC_PUBLIC_KEY;
```

公钥参数主要包括按字节计算的密钥大小keySizeInWords，椭圆曲线上一点表示的公钥值PubKey和椭圆曲线参数Param。通过_ECC_PRIVATE_KEY定义私钥：

```
typedef struct _ECC_PRIVATE_KEY
  {
    ECC_PUBLIC_KEY            PublicKey;
    UINTN                   *PrivKey;
} ECC_PRIVATE_KEY;
```

私钥参数除了生成的公钥PublicKey还包括整型的私钥PrivKey本身。ECC密钥的产生需要随机数K和点乘点加等运算。参数的定义：

```
Rand_K (
    IN   UINTN                          *Order,
    IN   UINTN                          nLen,
    OUT UINTN                           *K
);
ECC_generate_keypair (
    IN OUT ECC_PRIVATE_KEY                  *PrivKey
);
```

随机数产生函数Rand_K()主要定义了随机数K的产生并返回产生的随机数K。将私钥带入ECC_generate_keypair()后产生密钥对。最后对产生密钥进行验证：

```
ECC_validate_key (
    IN        ECC_PUBLIC_KEY                *PublicKey
);
```

上述加密库参数定义符合ECP(ECC组)的RFC 4753设计规范。

ECC算法主要运算为大整数运算，通过点乘，点加实现密钥的生成和相关运算。本文实现中，通过将点乘转化为点加和倍点运算实现点乘。本文用数组来表示椭圆曲线的各个参数如CURVE_A[][]， CURVE_B[][]，BASE_X[][]等，作为椭圆曲线的各个组成参数使用。关于参数值的选取下一节会有详细介绍。运算采用汇编指令进行大整数运算，提高了代码运算的速率。ECC密钥生成与计算过程是大整数的运算过程。

通过ECC_generate_keypair()来生成所需密钥。将生成的私钥作为参数传入密钥生成函数，然后通过调用Rand_K()来生成随机数K，通过调用点乘运算实现公钥的计算。产生密钥输出接口设计为：

```
ECC_generate_keypair (
  IN OUT ECC_PRIVATE_KEY                  *PrivKey
)
{
  Rand_K (PrivKey->PublicKey.Param.BasePointOrder,
          PrivKey->PublicKey.keySizeInWords,
          PrivKey->PrivKey
          );
  return PointMul (&PrivKey->PublicKey.Param.BasePoint,
                   PrivKey->PrivKey,
                   PrivKey->PublicKey.keySizeInWords,
                   PrivKey->PublicKey.Param.Prime,
                   PrivKey->PublicKey.keySizeInWords,
```

```
                &PrivKey->PublicKey.Param.Curve,
                  &PrivKey->PublicKey.PubKey
                  );
}
```

其中，椭圆曲线点乘和点加的实现主要代码结构：

```
PointMul (
    IN      EC_POINT                    *P,
    IN      UINTN                       *K,
    IN      UINTN                       Len,
    IN      UINTN                       *GFp,
    IN      UINTN                       FieldSize,
    IN      EC_CURVE                     *Curve,
    OUT      EC_POINT                    *Result
    )
  {
    EFI_STATUS                          Status;
    EC_POINT                            *Q;
    INT8                               *Buff;
    INTN                               Count;
     Buff = (INT8*) CPL_allocb(Len << ALIGN);
     if (Buff == NULL)
     return EFI_OUT_OF_RESOURCES;
     Status = Alloc_Point (&Q, FieldSize);
      if (EFI_ERROR (Status))
     return Status;
     Count = Hex2Bin(K, Len, Buff);
     for (Count--; Count >= 0; Count--) {
     PointAdd(Q, Q, Curve, GFp, FieldSize);
     if (Buff[Count] == 1)
     PointAdd (Q, P, Curve, GFp, FieldSize);
}
```

通过将已获得的参数 K，曲线参数等带入函数 PointMul()后，通过 PointAdd()运算，实现密钥的生成过程。通过上述代码库各参数和相应运算，椭圆曲线便可进行相应的密码生成和加密。加密库的实现过程中还涉及到相应的运算方式，内部函数调用等考虑。本文在此不做赘述。到此，ECC 加密库已经实现。

3.5.2 ECP 算法参数选取

本节为 ECC 加密库实现具体参数的数值选取和密码库密钥生成功能的测试。根据 ECC 密钥将用于 Pre-boot 环境下的需求，密钥参数定义应该符合 Pre-boot 环境的各个接口规则。ECP 密钥产生符合 IPSec RFC-4753 规则[11]。RFC4753 实现了 ECC 组的扩充，增加了三个 ECP 组。其中，本文实现了 256 位随机 ECP 组。256-bit 随机 ECP 组基于以下 Mersenne 大素数：

$$p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1 \tag{3.1}$$

产生的整数。符合椭圆曲线：

$$y^2 = x^3 - 3 * x + b \tag{3.2}$$

通过式(3.2)可知，椭圆曲线的参数 a 值为-3，其它参数符合 RETF 组织规定的 ECP 组参数设置方式为：

(1)域大小：256。

定义参数 EFI_KEY=256。该参数定义了生成密钥的大小。

(2)b 为：5AC635D8 AA3A93E7 B3EBBD55 769886BC
651D06B0 CC53B0F6 3BCE3C3E 27D2604B

考虑在通信过程中，默认为大端模式，我们初始化参数 CURVE_B[][]为：
0x27D2604B, 0x3BCE3C3E, 0xCC53B0F6, 0x651D06B0,
0x769886BC, 0xB3EBBD55, 0xAA3A93E7, 0x5AC635D8

同样，按照椭圆曲线参数 b 的转化方式，也对组序，产生器等进行大端模式的转换。这里不再一一给出。

(3)组序：0XFC632551, 0XF3B9CAC2, 0XA7179E84, 0XBCE6FAAD,
0XFFFFFFFF, 0XFFFFFFFF, 0X00000000, 0XFFFFFFFF

(4)组产生器(g_x, g_y)为 BASE_X[][]和 BASE_Y[][]：

BASE_X[][]: 0xD898C296, 0xF4A13945, 0x2DEB33A0, 0x77037D81,
0x63A440F2, 0xF8BCE6E5, 0xE12C4247, 0x6B17D1F2

BASE_Y[][]: 0x37BF51F5, 0xCBB64068, 0x6B315ECE, 0x2BCE3357,
0x7C0F9E16, 0x8EE7EB4A, 0xFE1A7F9B, 0x4FE342E2

Diffie-Hellman 组的 ID 值为 19。我们用以下进行 Diffie-Hellman 交换式交换密钥模型来验证 ECC 加密库。我们假设 Diffie-Hellman 发起者 Alice 的私钥为

i: C88F01F5 10D9AC3F 70A292DA A2316DE5
44E9AAB8 AFE84049 C62A9C57 862D1433

通过计算公钥值为$g^i = (g_x^i, g_y^i)$其中

g_x^i：DAD0B653 94221CF9 B051E1FE CA5787D0
98DFE637 FC90B9EF 945D0C37 72581180

g_y^i：5271A046 1CDB8252 D61F1C45 6FA3E59A
B1F45B33 ACCF5F58 389E0577 B8990BB3

这时，密钥交换 Ke_i 载荷为：

00000048 00130000 DAD0B653 94221CF9
B051E1FE CA5787D0 98DFE637 FC90B9EF
945D0C37 72581180 5271A046 1CDB8252
D61F1C45 6FA3E59A B1F45B33 ACCF5F58
389E0577 B8990BB3

假设响应者 Bob 的私钥为：

r：C6EF9C5D 78AE012A 011164AC B397CE20
88685D8F 06BF9BE0 B283AB46 476BEE53

那么公钥通过 $g^r = (g_x^r, g_y^r)$ 给出，其中

g_x^r：D12DFB52 89C8D4F8 1208B702 70398C34
2296970A 0BCCB74C 736FC755 4494BF63

g_y^r：56FBF3CA 366CC23E 8157854C 13C58D6A
AC23F046 ADA30F83 53E74F33 039872AB

密钥交换 Ke_r 载荷为

00000048 00130000 D12DFB52 89C8D4F8 1208B702 70398C34 2296970A
0BCCB74C 736FC755 4494BF63 56FBF3CA 366CC23E 8157854C 13C58D6A
AC23F046 ADA30F83 53E74F33 039872AB

则共享密钥值为：$g^{ir} = (g_x^{ir}, g_y^{ir})$ 其中

g_x^{ir}：D6840F6B 42F6EDAF D13116E0 E1256520
2FEF8E9E CE7DCE03 812464D0 4B9442DE

g_y^{ir}：522BDE0A F0D8585B 8DEF9C18 3B5AE38F
50235206 A8674ECB 5D98EDB2 0EB153A2

连接后组成：

g^{ir}：D6840F6B 42F6EDAF D13116E0 E1256520
2FEF8E9E CE7DCE03 812464D0 4B9442DE
522BDE0A F0D8585B 8DEF9C18 3B5AE38F
50235206 A8674ECB 5D98EDB2 0EB153A2

g^{ir} 被用来生成 SKEYSEED。上一章已经进行了详细的阐释，这里不做具体解释。

表 3.1 实验数据

算法\时间(ms)	1	2	3	4	5
ECP(256-bits)	3.82	3.05	3.66	4.01	3.45
MODP(1024-bits)	25.07	24.56	27.33	24.03	23.44

通过应用上述私钥和公钥的验证，该加密算法库在时间上和正确性上符合设计要求。通过时间函数的验证，五次实验结果如表 3.1 所示。与一般的密钥生成算法 MODP 所应用的 RSA 算法想比，密钥生成的时间上远远小于 RSA。

ECP 组通过函数 ECP_PRIVATEINIT()实现曲线的初始化。通过 Rand_K()生成随机私钥。密钥库通过 ECP_GENERATE_KEYPARIS()实现密钥生成算法与 DH 的连接。本文将在后面章节具体介绍。

实验表明了 ECP 实现 KE 交互有以下优点：

时间少：ECC 使用比较短的密钥就可以达到所需要的加密强度。相对于 RSA,ECC 具有安全性能更高，运算量更小，处理速度更快，需要的存储空间更小小，对传输带宽要求低等优点。本文主要目的是在 Pre-boot 环境下，将 ECP 应用于 IPSec 中的 IKE 中。从而在各方面条件的限制下，实现快速的密钥交换。

软件实现：因为 ECP 组采用整型数对大素数进行求模运算，该算法软件实现效率更高。

3.6 本章小结

本章设计了Pre-boot 下的IPSec框架，其中包括在Pre-boot环境下，IPSec基本配置信息的概述以及对Pre-boot环境下IPSec具体的协议的实现（AH, ESP和IKE)。本章设计了IPSec中所使用的数据库SPD,SAD和PAD，并对如何通过通信建立IKE SA和Child SA进行了简要概括（后面章节将对此进行详细的设计)。另外，本章还详细介绍了ECC算法的设计和具体实现过程。下一章将从代码的角度说明本章所述框架中IKE的具体实现过程。

第四章 Pre-boot 环境中密钥交换方案的具体实现

4.1 引言

上一章从框架设计的角度总体设计了Pre-boot环境下的IPSec安全架构。本章实现IPSec中的IKE密钥交换方案。本章进行IKE的消息交换方案设计，进而设计实现了IKE中第一阶段消息交换和第二阶段消息交换的具体功能。本章还将ECC算法加入到Pre-boot环境中来。

4.2 IPSec 密钥交换(IKE)的设计

本节针对 Pre-boot 环境对 IKE 进行设计，其中包括消息负载的架构设计，传输信息负载方案设计和规范消息负载被处理的顺序以及被使用的方法。本节实现了 IKE 的不同模式的消息在两个阶段中进行的交换。

从某种角度上说，本文设计的 IKE 协议是一种混合协议，它将部分的 Oakley 以及部分的 SKEME 和 ISAKMP 协议合并，在一种安全基础上，经过验证的方式协商安全联盟，并产生衍生密钥的材料。在 IKE 中用协商使用的加密算法来保证数据的机密性，用协商的方法实现验证机制。对数据进行验证的方法有许多种，如数字签名算法，可以用来进行加密的公钥算法和共享密钥算法。本文设计中主要应用共享密钥的方法进行验证。交换的机密性和验证方法是作为 ISAKMP 安全联盟的一部分来进行协商的属性。

本文设计中 IKE 定义了两个阶段：第一阶段：两个 ISAKMP 通信双方的实体建立一个安全、验证过的信道来进行通信，被称为 ISAKMP 安全联盟（SA），本文中不区分 IKE SA 与 ISAKMP SA。本文用两种模式：“主模式”和“积极模式”来实现第一阶段的交换。第二阶段指协商关于服务的安全联盟，这些服务可以是 IPSec 或任何其它需要密钥材料以及协商参数的服务。“新组模式”并不真正在第一阶段或第二阶段中。它紧接着第一阶段，用于建立协商中使用的新组。“新组模式”只能在第一阶段之后使用。

在 IPSec 中，IKE 可以通过消息的协商建立 ISAKMP SA 和 Child SA，从而建立 SAD.来保障 IPSec 安全的顺利实施。IKE 的交换消息有固定数目，在第一阶段的主模式中一共有六条消息：第一条和第二条消息协商策略；第三条和第四条消息交换 Diffie-Hellman 的公共密钥值和 Nonce 值；最后的两个消息验证 Diffie-Hellman 交换。

一个第一阶段协商建立的ISAKMP SA可以用于多个第二阶段的Child SA协商。一个第二阶段协商也可以同时对多个安全联盟提出请求。经过优化后，实现中可以减少每个安全联盟的建立过程中传输往返和DH求幂运算的次数。实现中，第一阶段中的“主模式”提供了身份保护。

Alice		Bob
Hdr, SA	--> <--	Hdr, SA
HDR, KE, Ni	--> <--	HDR, KE, Nr
HDR*, IDii, HASH_I	--> <--	HDR*, IDir, HASH_R

图 4.1 主模式预共享密钥

本节主要进行IKE主模式，预共享密钥的设计和代码实现过程。主要消息交换结构如图4.1所示。实现中，通过InitIkeSaSaData (IN IKE_SA_SESSION *IkeSaSession)进行初始化，函数中用IN IKE_SA_SESSION来记录SA协商的具体阶段。然后通过NegotiateSa (

```
IN IKE_UDP_SERVICE        *UdpService,
IN IPSEC_SPD_ENTRY        *SpdEntry,
IN EFI_IP_ADDRESS         *RemoteIp
```

)

进行SA的谈判。谈判过程所需的具体信息主要包括UDP相关数据IKE_UDP_SERVICE，SPD数据库的策略查找IPSEC_SPD_ENTRY和远程目的主机的具体地址EFI_IP_ADDRESS。最后通过函数

EstablishIkeSession (

```
IN IKE_SA_SESSION       *IkeSaSession,
IN IPSEC_PRIVATE_DATA *Private
```

)

建立通信所需的ISAKMP SA。实现中，通过IKE_SA_SESSION来标记谈判的具体阶段，用IPSEC_PRIVATE_DATA传输IKE协商的具体参数。具体设计过程下面小节将做具体设计。IKE谈判的初始化的步骤包括：通过策略代理库，由本地数据库检索本地策略数据，策略代理分发安全设置信息给IKE（包括主模式设置等），通过IPSec驱动用选择器进行SPD相应列表的填写。

UDP 应用程序通过端口 500 输出“ applicative” IP 数据报，因为这个数据报的输

出接口是 IPSec 使能的，所以数据报被发送到 IPSec 驱动，SPD 检查并返回“secured”；此时，SAD 数据库检查是否存在相应的 SA，如果存在则进入 IPSec 通信步骤，否则 IPSec 请求 IKE 建立 SA，主模式进入第一阶段协商。

4.3 IKE 第一阶段设计

本节设计IKE第一阶段交换所需要的消息封装格式。本节将第一阶段的交互过程分成三个小节进行具体设计，主要包括策略谈判负载设计，密钥交换设计和认证设计。第一阶段协商中用于验证交换的预共享密钥通过其它机制衍生而来，密钥实际的建立过程不在本文设计的范围，这里不进行详细论述。通过交换获得的密钥也作认证时的交换。

4.3.1 策略谈判设计

本节主要设计策略谈判机制。设计内容包括消息封装时各字段具体的参数配置，协议封装方式和谈判消息的通信模式。消息字段的参数配置如图 4.2 所示，主要包括 ISAKMP 消息头和 SA 协商载荷。通信双方的保护策略即 ISAKMP SA 的属性或者 SA 载荷谈判采用负载交换的方式，在第一个传输往返的交换中进行。发起者 Alice 可以提出多个提议载荷 T Payload；响应者 Bob 只能用一个来回答。响应者选择一个 T Payload 的提议作为应答。在这两个消息的谈判中，主要完成双方所需协议的协商。当进行预共享密钥认证时，主模式代码设计主要包括如下几部分。

1. 头载荷的初始化

策略协商封装载荷头 IKE_HEADER 格式定义为：

```
typedef struct {
  UINT64   InitiatorCookie;
  UINT64   RESPonderCookie;
  UINT8    NextPayload;
  UINT8    Version;
  UINT8    ExchangeType;
  UINT8    Flags;
  UINT32   MessageId;
  UINT32   Length;
} IKE_HEADER;
```

当进行不同消息的协商时，可以选择性的初始化 ISAKMP HEADER 中的参数，满

足本身通信的要求。其中，第一次传输往返的交互中，消息的初始化值如图 4.2 所示，通过定义初始化 InitiatorCookie ，RESPonderCookie Cookie 来决定交互双方的身份信息。然后设置下层载荷的类型 NextPayload 和交换的类型 ExchangeType。本次消息往返中，下层载荷为需要协商的 SA，所以相应的载荷类型为 SA。本文采用主模式，所以交换类型为 Main Mode。响应方 Bob 也进行同样的配置。保护机制谈判通过函数

```
FillIkeHdrByIkeSaSession (
  IN IKE_SA_SESSION       *IkeSaSession,
  IN IKE_HEADER           *Hdr
)
```

根据 IKE 阶段信息 IKE_SA_SESSION 来填充 IKE Header。

2. SA 载荷的初始化

本文设计中，SA 载荷主要包括本身属性外，还包括协议(Proposal 简称 P)载荷，定义协商用到的协议类型，第一次传输往返用到 ISAKMP 协议，因此初始化 Proposal ID 为 ISAKMP。在 P 载荷中又包含了转换传输提议 (简称 T)载荷来提议协商所用的加密算法，交换方法等信息。T 载荷除了载荷头定义的基本信息外，还包括加密算法，Hash 算法，DH 交换类型以及认证方法，生存周期等基本信息。本文 DH 交换加入了 ECP 组进行密钥的生成和交换，后面将会对此有更加详细论述。提议载荷设计中，Hash 算法采用 SHA，因此 Hash Algorithm 定义为 SHA。认证方法为预共享密钥，因此定义 Authentication Method 为 Pre-shared Key。然后设定其他参数，如加密算法为 3DES 等。当希望双方通过 ECC 算法来进行密钥交互时，应答方选中两个 T 载荷中的第一个作为应答返回。然后通过

```
_MAIN_1_PSK_Handler (
    IN IKE_SA_SESSION *IkeSaSession,
    IN IKE_PACKET       *IkePacket
)
```

检查是否进行消息的交换。如果是，则通过

_MAIN_1_PSK_Generator (IN IKE_SA_SESSION *IkeSaSession) 产生交换所需的材料：GenerateSaPayload (IkeSaSession->SaData, IKE_PAYLOAD_TYPE_NONE); 从而进行交换所需材料（加密算法，Hash 算法，验证方法）的协商。

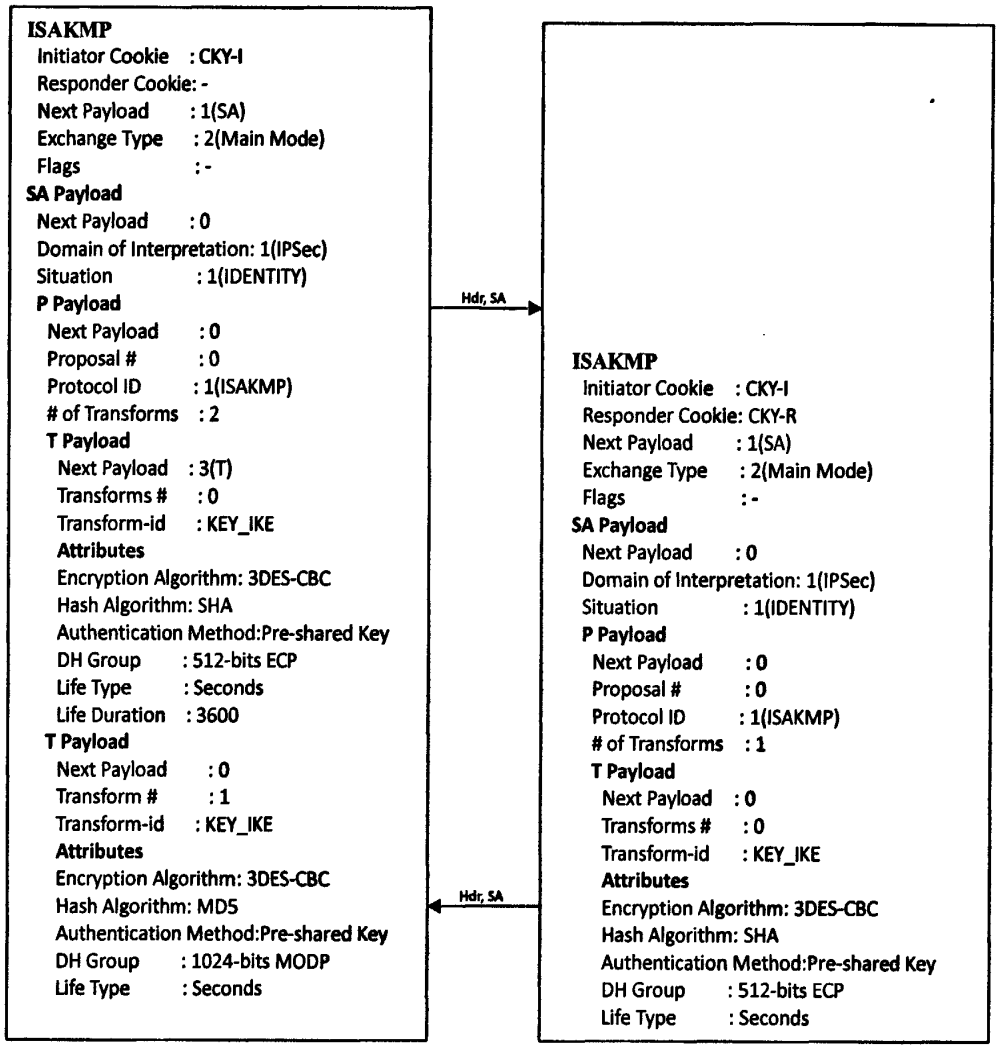

图4.2 策略谈判

（附注：图中左侧是发送者，右边是接受者，下同）

3. 通信过程设计

沿用发送方为 Alice 和接收方为 Bob 的命名方式。 Alice 发送 ISAKMP 消息“Hdr，SA”给接收方 Bob，IKE 的 IP 数据包被传送到 IPSec 驱动；SPD 检测并返回“permitted”，此时，IKE 的通信不会通过 AH/ESP 安全保护；IPSec 不对数据包进行修改，并返回；IKE 数据包被发送到 Bob；IKE 数据包被 Bob 接收；Bob 的输入接口进来的数据包是 IPSec 使能的，因此 IPSec 数据包被传送到 IPSec 驱动；SPD 检测并返回“permitted”；因为 IKE 通信不会被 AH 或者 ESP 保护，所以不用进行相应的处理；数据包被 IPSec 驱动返回；IKE 消息被 Bob 的 IKE 接收；这样便完成

了 IKE 中第一个由 Alice 到 Bob 的通信。

4.3.2 密钥交换设计

本节进行密钥交换的设计。在策略协商后，通信双方用协商好的策略进行密钥交换，通过交换生成通信双方所需要的公钥。本节设计内容包括 IKE 消息头的初始化，KE 载荷参数配置以及公钥协商后加密密钥的生成。如图 4.3 所示。在进行密钥交换消息交互时，首先也是调用 IKE_HEADER，用

FillIkeHdrByIkeSaSession (

IN IKE_SA_SESSION *IkeSaSession,

IN IKE_HEADER *Hdr

)

来进行第二个传输往返的 IKE_HEADER 初始化，确定 IKE 头的相关参数。除了发起方和接收方的 Cookie 设置外，下层载荷为 KE，所以在此初始化下层载荷 NextPayload 为 KE。模式类型是 Main Mode。

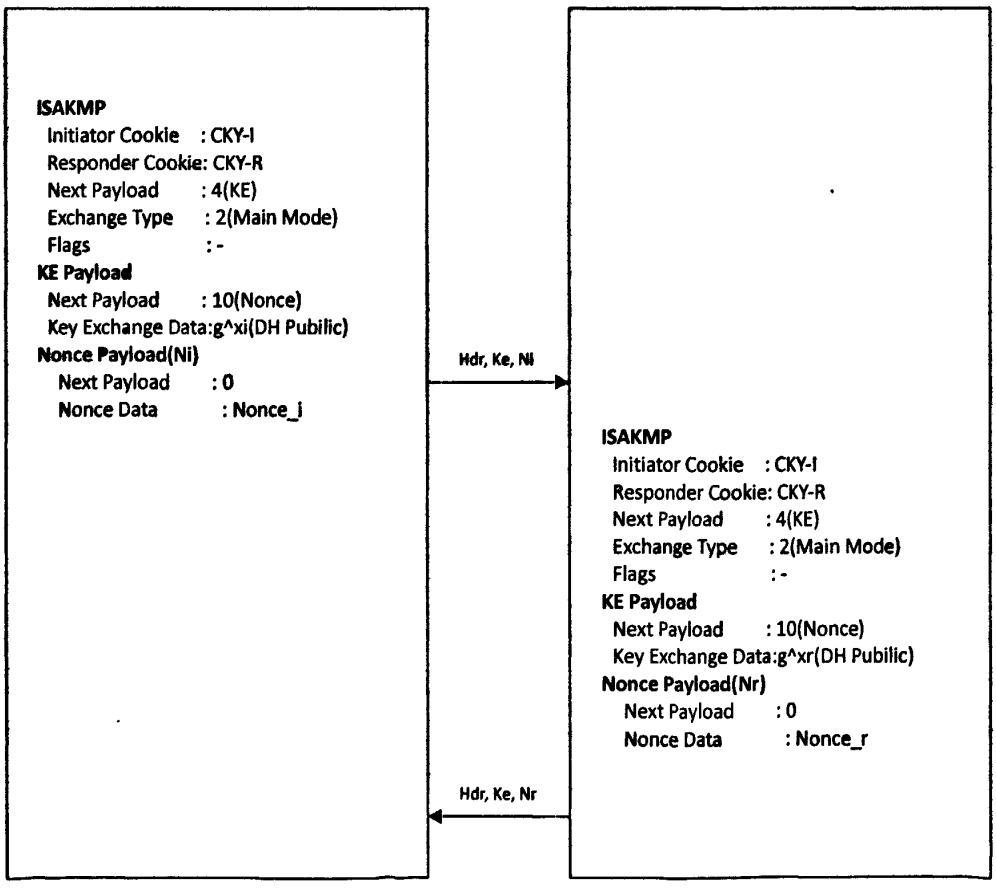

图 4.3 DH 交换过程

在 KE 交互的过程中，密钥的产生和交互过程是设计的关键。本文应用 ECC 即椭圆曲线加密算法产生密钥，应用 DH 交换方式实现密钥在对等通信体间的交换过程，使密钥产生速度和交换速度更加符合 Pre-boot 环境的要求。DH 交换过程下面会进行详细阐述。

通过图4.3可以看出，ISAKMP交互的过程中，头部是必须填充项，同样,密钥交换KE也是通过密钥交换KE PAYLOAD来实现。KE载荷主要包括下层载荷Nonce和交换的密钥数据DH公钥构成。DH公钥的具体值的生成过程后面章节会具体论述。Nonce载荷包括了具体的Nonce交换的数据Nonce_i。发送方将其发送给接收方Bob后，Bob进行处理，并返回自己的公钥值和Nonce载荷信息。

在交互过程中，双方通过

```
GenerateNoncePayload (
  IN UINT8          *NoceBuf,
  IN UINTN          NoceSize,
  IN UINT8          NextPayload
)
```

来产生交换所需的 Nonce 载荷。主要包括载荷数据，载荷大小和下层载荷的设置。然后，通过

```
GenerateKeyPayload (
  IN IKE_SA_SESSION *IkeSaSession,
  IN UINT8          NextPayload
)
```

产生交换所需密钥载荷。最后调用_MAIN_2_PSK_Generator (IN IKE_SA_SESSION *IkeSaSession)来产生第二次往返交换所需的数据包。通信过程与第一次往返通信过程类似，这里不再赘述。

交换所得公钥作为加密所需的材料，被 SKEYID 用来生成最终的加密密钥。计算方法为：

SKEYID = Hash (pre-shared-key, Ni_b | Nr_b)
SKEYID_d = Hash (SKEYID, g^xy | CKY-I | CKY-R | 0)
SKEYID_a = Hash (SKEYID, SKEYID_d | g^xy | CKY-I | CKY-R | 1)
SKEYID_e = Hash (SKEYID, SKEYID_a | g^xy | CKY-I | CKY-R | 2)

从而：

HASH_I = Hash (SKEYID, g^xi | g^xr | CKY-I | CKY-R | SAi_b | IDii_b)
HASH_R = Hash (SKEYID, g^xr | g^xi | CKY-R | CKY-I | SAi_b | IDir_b)

如文中上式所示，本文的 prf 采用 Hash 算法实现。因为 Hash 算法和加密过程不是本文的重点，所以本文不作详细赘述。

4.3.3 认证过程设计

本节进行认证过程的具体设计。设计内容包括IKE数据报头的初始化，交换数据ID计算和Hash数据生成，从而完成身份认证的过程。

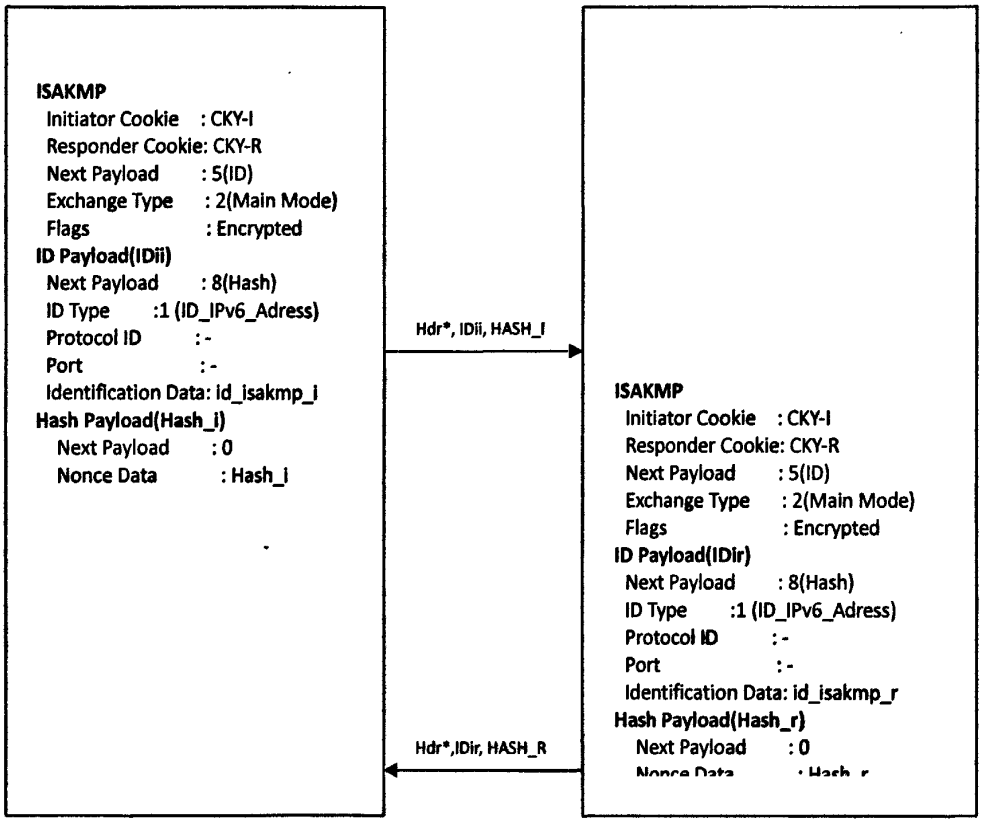

图 4.4 认证过程

与前两种交换过程类似，首先初始化IKE头。在初始化中，因为下层载荷为ID，所以Next Payload设置为ID。然后标志位设置为Encrypted，如图4.4所示，调用

FillIkeHdrByIkeSaSession (

IN IKE_SA_SESSION *IkeSaSession,

IN IKE_HEADER *Hdr

)

初始化 IKE 头。

本文通过GenerateMessageId (IN IKE_SA_SESSION *IkeSaSession)产生交换所需的ID身份信息。通过ID Payload记录ID数据。由

```
SaGenerateHashPayload (
    IN IKE_SA_SESSION *IkeSaSession,
    IN UINT8          NextPayload,
    IN IKE_PAYLOAD    *IdPayload
)
```

产生认证所需的数据Hash_i，然后填充Hash Payload，从而封装整个数据报。其中，Hash_i的计算上一节中已经阐释过，这里不再赘述。

然后交换该ID和Hash所得数据信息。通信过程与前两次过程类似，这里不再赘述。从而完成最后两条消息的交互。第一阶段完成，ISKMP SA建立。为进一步的数据交换和相关信息协商打下基础。

4.4 IKE 第二阶段设计

本节实现了 IKE 的第二阶段消息交互。与 IKE 第一阶段类似，第二阶设计也包括了数据报的参数设置和消息负载的模式设计。本节主要设计了第二阶段的交换模式和具体的通信过程。

1. 交换模式设计

本文第二阶段用快速模式进行交换。快速模式本身并不是一次完整的交换（因为它和第一阶段交换相关联），但又作为 SA 协商过程的一部分用来衍生密钥材料和协商非 ISAKMP SA 的共享策略。快速模式交换的信息必须由 ISAKMP SA 来保护即除了 ISAKMP 报头外，所有的负载都要加密。在快速模式中，Hash 负载必须跟随在 ISAKMP 报头后，SA 负载必须紧跟在 Hash 负载之后。Hash 用于验证消息，同时也提供了参与的证据。

在一个特定的 ISAKMP SA 中，在 ISAKMP 报头中的消息 ID 标识了快速模式正在进行中，而 ISAKMP SA 本身由 ISAKMP 报头中的 Cookie 来标识。因为每个快速模式的实例使用唯一的初始向量，这就有可能在一个 ISAKMP SA 中的任一时间内同时有多个快速模式在进行中。

2. 通信过程设计

IKE 阶段二的通信过程实现主要完成了三次消息的交互。首先，快速模式谈判将产生一个 outboard SA，一个 inbound SA，分别用 SPI-a，SPI-b 表示。本文假设 Alice 为 SPI-a，Bob 为 SPI-b；Alice,Bob 的 SAD 被更新；在发起端，IKE 通知 IPSec 驱动来回应以前的请求；推荐为“application”数据包的延迟检测 SAD 中的 SA 参数；数

据包通过 ESP 修改后送回到 IP;安全数据包发给 Bob；安全数据包被 Bob 接收；数据包被送到 IPSec 驱动；inbound SA 的 ESP 头的 SPI 值被检测，同时进行校验；ESP 头和尾被移除，IP 数据包被发送回 IP 模块；载荷被送往更高层次处理。完成第二阶段的谈判，Child SA 建立。

4.5 Diffie-Hellman 密钥交互过程设计

本节设计了 Diffie-Hellman 进行密钥交换的架构，并将 ECP 算法引入交换过程中，从而节省了密钥交换所需的时间。为了进一步增强 Diffie-Hellman 密钥交换的可靠性，本文增加了 DH 的交换环节，通过两次密钥交换过程来实现。

Diffie-Hellman 密钥交换框图如图 4.5 所示。DH 的 KE 交换由第一阶段的密钥交换中实现。本文主要在主模式，预共享密钥中，实现了 ECP 组在 KE 中的应用。算法流程图 4.6 所示。通过采用 OakleyGroupId 来选择所需 OKELEY 组。如果采用 MODP 组产生密钥，则调用 MODP 产生密钥接口。如果采用 ECP 组，则调用 ECP 产生密钥接口。程序实现了密钥诊断和程序判断报错功能。

DH 交换的具体设计细节如下。

1. DH 参数定义

在 Diffie-Hellman 协议中，通信双方建立对称会话密钥。在创建对称密钥之前，这通信双方选择两个数 p 和 g。通过

```
typedef struct _DH_PARAMETER {
    UINTN           *p;
    VOID            *g;
} DH_PARAMETER
```

第一个数 p 是一个大素数。第二个数 g 是一个 p−1 阶的生成元,可以根据采用密钥生成算法的不同来定义参数类型。当采用 ECP 组时，该参数取为整数型；当采用 ECP 时该参数取为椭圆曲线类型。两个数可以在通信过程中传输。

2. DH 交换过程

DH 交换步骤可以分为内部密钥协商和外部密钥协商，数据传输进而产生交互所需的公钥。具体步骤为：

首先，Alice 选择一个大的随机数 a，使得 $0 \le a \le p-1$，并计算出 $A = ag \bmod p$。

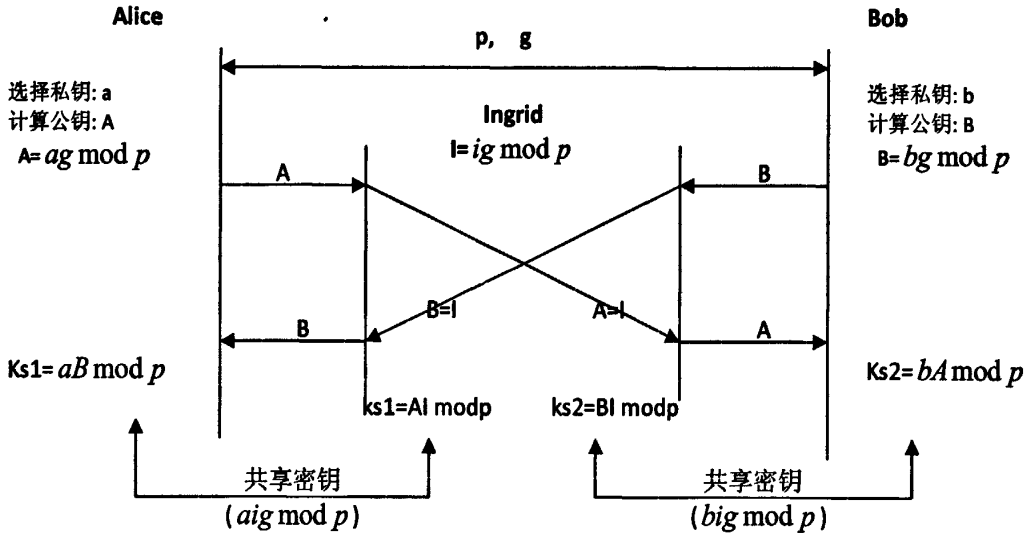

图 4.5 DH 交换

其中参数 a 和 g 进行点乘运算，以下产生过程中不再指出。

其次，Bob 选择另一个大的随机数 b，使得 $0 \le b \le p-1$ 并算出 $B = bg \bmod p$ 。

为了提高安全性，增加了交换密钥 I 的环节。Alice 和 Bob 首先协商 I，I=$ig \bmod p$.

经过安全确认后，再相互交换 A,B.

第三，Alice 发送 A 给 Bob。Alice 不发送 a 的值，只发送 A。

第四，Bob 发送 B 给 Alice。Bob 不发送 b 的值，只发送 B。

第五，Alice 算出 $k_{s1} = aB \bmod p$ 。

第六，Bob 也算出 $k_{s2} = bA \bmod p$ 。

Alice 算出

$k_{s1} = aB \bmod p = a(bg \bmod p) \bmod p = abg \bmod p$ 。

Bob 算出

$k_{s2} = bA \bmod p = b(ag \bmod p) \bmod p = abg \bmod p$ 。

Bob 不知道 a 的值，Alice 也不知道 b 的值，但是两个人求出来的值是相同的。在 Diffie-Hellman 方法中，共享密钥是 $K = abg \bmod p$ 。

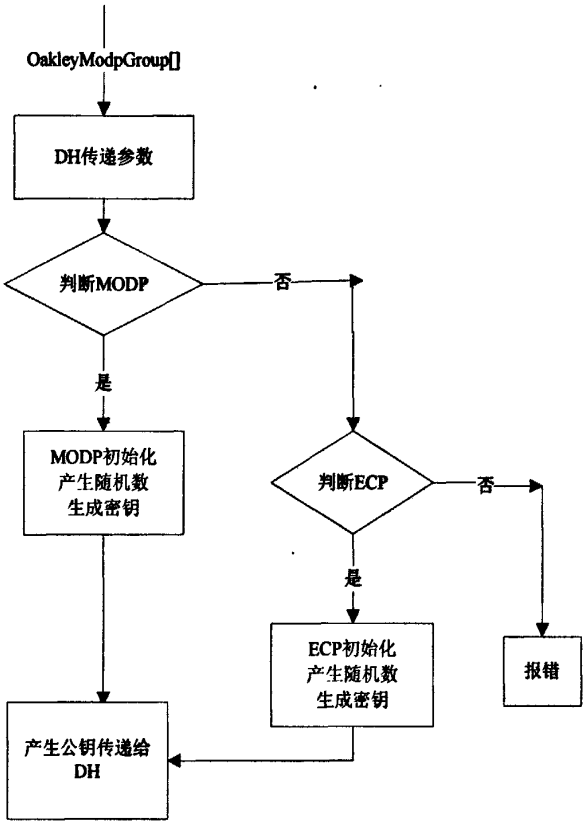

图 4.6 ECP 组在 KE 中的应用流程图

3. DH 交换与 ECP 组的接口实现

DH 交换的函数实现中包括加密算法接口定义，DH 初始化和密钥产生过程。通过

```
enum OAKLEY_GROUP_ID {
    OAKLEY_GROUP_MODP768        = 1,
    OAKLEY_GROUP_MODP1024       = 2,
    OAKLEY_GROUP_GP155          = 3,
    OAKLEY_GROUP_GP185          = 4,
    OAKLEY_GROUP_MODP1536       = 5,
    OAKLEY_GROUP_MODP2048       = 14,
    OAKLEY_GROUP_MODP3072       = 15,
    OAKLEY_GROUP_MODP4096       = 16,
    OAKLEY_GROUP_MODP6144       = 17,
    OAKLEY_GROUP_MODP8192       = 18,
    OAKLEY_GROUP_ECP_256        = 19,
```

```
    OAKLEY_GROUP_MAX,
};
```

增加ECP加密算法为OAKLEY_GROUP_ECP_256组，其值为19。ECC密钥计算过程见上一章中介绍，这里不再阐述。通过函数

```
DH_Init (
  OUT      __CPL_KEY              *Key,
  IN       UINT8                  OakleyGroupId
)
```

将DH初始化，初始化时，通过OakleyGroupId确定所使用的Oakley类型，并返回__CPL_KEY类型密钥参数。通过调用DH_Generate_Key(__CPL_KEY *Key)产生密钥，产生用来交换真正密钥的密钥。然后通过

```
DH_Compute_Key (
    __CPL_KEY                  *Key,
    UINTN                      SharedDataLen,
    EC_POINT                   *SharedData,
    UINTN                      SecretLen,
    EC_POINT                   *Secret
);
```

计算IKE所需要的密钥。此时，密钥的计算包括两种类型，通过MODP计算所得密钥和通过ECP计算所得密钥。ECP密钥的产生过程在上一章已经论述，这里不再详细阐释。假设Alice和Bob正在通信。为了增加DH交换的可靠性本文增加的中间密钥计算过程，即通过计算第三方公钥来实现密钥对的传输，从而提高了密钥交换的安全性。ECP和MODP组所使用的RSA算法，是现在最流行的加密算法，有很好的安全性。将该加密算法应用到Pre-boot环境中，能很好的保证网络通信的安全。

IKE 的可扩展的框架支持定义更多的组，使用椭圆曲线组会大大的增加使用同等长度密钥时的安全性。交换中的 Diffie-Hellman 指数在使用过后就从内存中删除，从而使保密性进一步得到改善。

4.1 本章小结

本章根据 Pre-boot 环境下 IKE 实现通信的过程：消息的发送，数据包的代码实现和数据通信的方法，设计并实现了 IKE 第一阶段和第二阶段数据报的封装格式。同时，本章也实现了将 ECP 引入到 IKE 中来的思想，从而实现 IKE 的 DH 交换过程。下一章将对本章设计的安全架构进行功能和性能方面的测试。

第五章 Pre-boot 环境下安全系统的实验及测试

5.1 引言

本章将从功能和性能上对在Pre-boot环境下实现的IPSec安全方案进行测试。功能测试是对IPSec加入Pre-boot环境后运行情况进行的测试。性能测试主要测试IPSec方案加入Pre-boot后对该环境的时间参数的影响进行的测试。通过对所设方案的不同情况下的测试，可以进一步证明本文设计的Pre-boot安全隐患的避免方案与其它方案相比存在优越性。

5.2 IPSec 性能测试

本节主要对 IPSec 架构引入 Pre-boot 环境之后，Pre-boot 启动环境的时间参数进行性能测试。通过将 IPSec 打包进 Pre-boot 环境下的 image，UEFI BIOS 便可正常启动。通过记录启动过程中各个时间参数来验证安全解决方案对总体环境的时间影响是否在可接受范围内。性能测试包括平台的搭建，测试软件的装载和测试结果的收集等。

5.2.1 性能测试所需硬件设备

性能测试需要测试平台的搭建。首先，测试功能的实施需要相关硬件的支持。如CPU选择Intel Core™ 2 Duo 2.40GHz从而保证指令处理速度符合软件运行要求；选取内存大小为512M/DDR-II *2，保证程序运行时有足够的内存空间；芯片集为Bearlake/ICH9。本次试验硬盘配置为：164G/HITACHI/SATA；CDROM 配置为:PLEXTOR；软盘配置为:TEAC；键盘和鼠标分别包括PS2和USB。网卡采用intel-proX100。

5.2.2 软件工具版本及其它配置

该测试使用性能测试工具PerformanceTools进行测试。该工具通过端口读取BIOS启动各阶段的时间值，来产生四个文件文本文件。通过对四个文件的时间分析，最后自动发送测试结果报告。测试工具运行于Windows XP环境下，通过软件包直接装载。

5.2.3 性能测试步骤设计

本节主要设计性能测试的具体步骤。首先是搭建测试环境。在 Bearlake 平台的 MBR 分区盘中装入 Windows XP 操作系统。启动机器到 UEFI BIOS 环境，在 UEFI SHELL 里面用“ map –r” 来查找整个硬盘块。有下列类似信息：

Blk3: BlockDevice – Alias (Null)

Acpi(PNP0A03,0)/Pci(1F|2)/Ata(Primary, Slave)

说明硬盘可以被机器识别。在Windows XP中装载所需的驱动配置文件，因为文件不是本文重点，因此不再赘述。然后重启机器进入UEFI BIOS，将编译好的image烧入系统中。具体测试步骤如下：

(1)收集正常启动和 S3 重启的时间

在平台上重烧编译好的image。同时，配置硬盘为第一启动项。并设置启动时间为零。关闭机器并启动以保证环境是不被其他影响，测试工具自动读取数值，并保存至文本文件中。做S3并重启，运行脚本命令dpx.efi。软件读取S3时间到文件中。

(2)收集启动到 shell 环境的时间

增加启动选项 shell64.efi(shell64.efi 是支持 shell 的 BIOS 平台工具)；配置新添加的选项为第一启动项；启动到 shell 并关机重启；关掉电源，重启，以避免静电对结果产生负面影响；执行脚本，脚本自动记录启动到 shell 的时间值。

(3)收集 recovery 所需的时间

关掉机器；拔掉电源；用 USB 做 recovery 测试。测试步骤不在本文的讨论范围，所以本文不做详细阐述。

5.2.4 测试结果分析

通过 PerformanceTools 对文件进行处理，并计算各个启动时间的值。表 5.1 显示了加入 IPSec 模块之前和之后的时间变化。+表示时间的增加，-表示时间的减少。实验将 IPSec 加入之前和加入之后的值进行了比较，并将时间变化的差值进行计算并列出。

如表 5.1，PEI 在 IPSec 加入之前与之后运行的时间减少了 1ms。IPSec 加入对这个启动模块没有直接的影响，所以-1ms 的误差是正常的，可以忽略。而在 BDS 阶段，系统读取 IPSec 的相关信息并进行初始化，时间有了很大增长值为 20。其它时间参数类似。在 Recovery 的整个过程中增长数值为 94，对整个系统来说这是一个可接受的数字。可见，在 IPSec 加入之后，时间变化只有几十毫秒，因此，安全方案的加入对 UEFI BIOS 正常的启动时间的影响在可容许范围之内。

表 5.1 性能测试实验结果

测试项		执行文件		
		IPSec 加入前	IPSec 加入后	时间变化
阶段 (ms)	PEI 阶段 (ms)	789	790	-1
	DXE 阶段 (ms)	965	962	+3
	BDS 阶段 (ms)	7938	7918	+20
	CSM 阶段 (ms)	5334	5318	+16
	总时间 (ms)	15026	14988	+38
启动到 shell 时间 (ms)		7771	7787	+16
Recovery USB	Recovery 阶段(ms)	22547	22453	+94

5.3 IPSec 功能测试

本节对 IPSec 功能测试进行设计。主要包括测试环境搭建，测试脚本文件配置和测试步骤设计。通过功能测试，验证 IPSec 在 Pre-boot 环境下的功能是否能达到预期的安全性。本节实验证明 IPSec 在 Pre-boot 环境下能够正常运行。

5.3.1 功能所需硬件设备

本节沿用用户Alice与用户Bob之间通信，并假设Alice是发起方，Bob是响应方。本节设计IPSec加入Pre-boot环境后的功能测试。选取平台设备为Bearlake；根据安全方案运行环境选择CPU为Intel Core™ 2 Duo 2.40GHz；选取内存为512M/DDR-II *2；实验所用硬盘型号为164G/HITACHI/SATA；CDROM 配置为PLEXTOR；键盘和鼠标分别包括PS2和USB；两块intel-proX100网卡；连接机器的Hub等。

5.3.2 软件工具版本及其它配置

本节测试采用软件抓包工具 wordhark 进行测试。抓包工具工作在网络层，通过在线侦听通信双方的消息传输，提取测试有用的数据报，并将数据包以文档的形式保存下来。其它软件工具主要包括 IPSec 打包后的 image 和相关的配置文件。其它如 OpenSwan 不是本文设计内容，这里不做详细介绍。

5.3.3 功能测试场景搭建

将两台 Bearlake 平台用 Hub 进行连接。一个平台安装 Linux 操作系统，另一个平台运行 UEFI BIOS。在 Linux 操作系统中安装 StrongSwan，UEFI 平台上烧有 IPSec 封装的 image。如图 5.1 所示，配置相应的 IP：

Alice：fec0::200:1

Bob: fec0::200:101

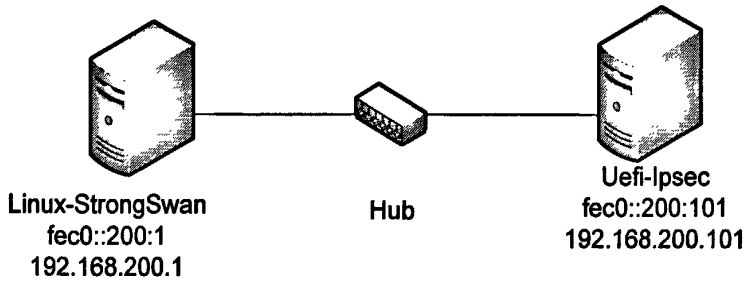

图 5.1 UEFI 和 Linux 连接图

IPSec 测试包括很多测试脚本的基本命令，如 IPSecconfig –enable 可以在 UEFI 中激活 IPSec；IPSecconfig –disable 在 UEFI 中关闭 IPSec；IPSecconfig –status 可以查看 IPSec 状态；IPSecconfig –p SPD –l 则列出 SPD 内容；IPSecconfig –p SAD –l 列出 SAD 内容；IPSecconfig –p PAD –l 列出 PAD 内容。同时，测试需要相关配置文件的应用，下面对不同平台用到的脚本文件进行设计。

5.3.4 功能测试的配置文件

Linux 运行 OpenSwan 需要安装相应的软件。除此之外，还包括许多性能参数的配置，配置文件包括 IPSec.conf 文件，IPSec.secrets 等。UEFI BIOS 运行 IPSec 封装的 image 时，也需要配置文件的设置，包括 Startup.nsh 等。

在 Linux 中安装配置文件 IPSec.conf 文件。其中，连接到 IPv6 参数设置为：

```
Alice=fec0::200:1
Bob=fec0::200:101
 authby=secret
 type=transport
```

另外，还有 IPSec.secrets 文件，主要完成预共享密钥 PSK 的设置：

```
fec0::200:1 fec0::200:101 : PSK "Hello World"
192.168.200.1 192.168.200.101 : PSK "Hello World"
```

在 UEFI 中的配置文件为 Startup.nsh。在该文件中实现了通信所需参数的设置。首先通过装载 Arp.efi，Ip6.efi 和 IPSec.efi 等使 ICMP6 通过，命令为：

IPSecconfig -p SPD -a --local ::/0 --remote ::/0 --proto 58 --action bypass --local-port 0 --remote-port 0

在该文件中，使 IKE 协商数据包通过的命令为：

IPSecconfig -p SPD -a --local ::/0 --remote ::/0 --proto udp --action bypass --local-port 500 --remote-port 500

设置 UEFI BIOS 的预共享密钥：

IPSecconfig -p PAD -a --peer-address fec0::200:1 --auth-method PreSharedSecret --auth-data "Hello World"

IPSecconfig -p PAD -l

来保护 fec0::200:1 和 fec0::200:101 之间的数据包：

IPSecconfig -p SPD -a --local fec0::200:101 --remote fec0::200:1 --proto 254 --action protect

IPSecconfig -p SPD -l

其次，装载 Udp6.efi，Dhcp6.efi 从而设置 UDP 和 DHCP 的参数。设置静态 IPv6 地址：

ifconfig6 -s eth0 man host fec0::200:101

然后便给 Linux 发送消息：

IPSectest6 -I fec0::200:101 -o fec0:200:1 hello

5.3.5 功能测试步骤设计

本节设计实现功能测试的具体测试步骤。初始化 Linux 和 UEFI BIOS 的 IKE。然后通过相互之间发送数据报实现通信。

首先，通过 Linux 对 UEFI BIOS 的通信方法如下：

(1)在 Linux 中用"IPSec up IPv6" 来初始化 Linux 和 UEFI 之间的 IKE。

(2)IKE SA 成功激活后，用"./SendIpv6 fec0::200:101 Hello" 给 UEFI 服务器发送 IPSec 保护的 IP 数据包"Hello"。

(3)在 UEFI 端，用"IPSecTest6 –i" 接收 IPSec 保护的数据包。如果数据包被 IPSec 成功解密，则打印数据包的内容，如果内容被打印，则可以认为数据包被成功解密。否则失败。

其次，UEFI BIOS 对 Linux 的通信方式如下：

(1)在 Uefi 用 "IPSectest6 –i fec0::200:101 –o fec0::200:1 Hello"初始化 UEFI 和 Linux 通信的 IKE 阶段。

(2)IKE SA 成功建立以后，用"IPSectest6 –i fec0::200:101 –o fec0::200:1 Hello"发送 IPSec 保护的"Hello"数据包给 Linux.

(3)在 Linux 部分，用"./RecvIpv6" 来接收来自 UEFI 的 IPSec 数据包.

通过以上测试步骤，便实现了 Linux 和 UEFI BIOS 之间的通信过程。

5.3.6 测试结果分析

首先，通过 IPv6 进行测试，测试结果如图 5.2，界面给出了所获得数据包的时间，源和目的地址和协议相关的信息。调试对话框显示了通信的具体起止时间和通信所用端口等信息。通过该图片信息可知网路正常通信。

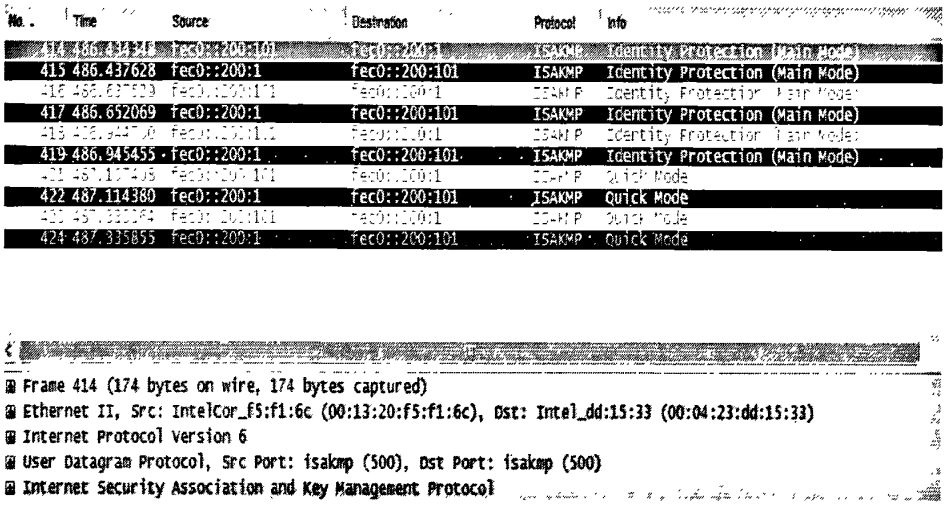

图 5.2 IPv6 通信消息

图 5.2 中分别给出了具体消息交换的时间 Time，源主机地址 Source，目的主机地址 Destination 和通信协议类型，得到的数据报的信息。其中数据 ISAKMP 通信实现了六条消息的主模式，预共享密钥的交互，四条消息实现了积极模式交互，如图 5.3 所示。

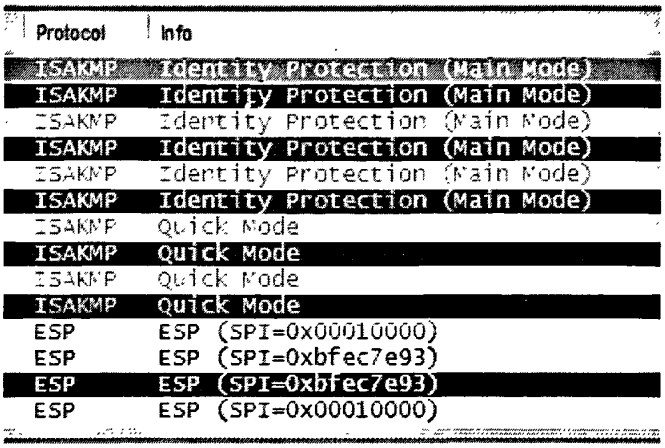

图 5.3 抓包结果

调试窗口给出了具体的调试信息。其中第二行指明了消息的主机信息和具体收发数据时间，如图 5.4 所示。调试窗口同时给出了协议通信过程中用到的端口号，如 5.5 所示，端口号为 500.

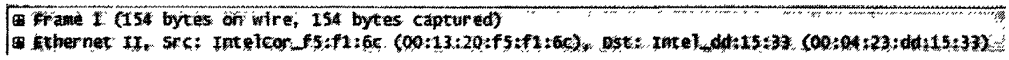

图 5.4 时间信息

通过 wordhark 抓包工具抓取通信数据包。该包主要包括消息发送和接受方的 IP 地址(Source 和 Destination)，通信所使用的传输协议如图 5.2 所示的 ISAKMP 协议和消息的传输模式类型。

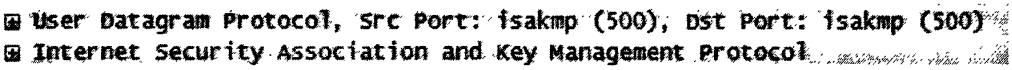

图 5.5 端口信息

本文采用主模式，所以传输类型为 Main Mode。通过抓包工具抓取的数据包显示，UEFI 和 Linux 完成正常通信。并且相应的加解密功能能够正常执行。

5.4 本章小结

本章对本文所设计的Pre-boot环境下设计的安全方案进行了功能和性能测试。测试包括搭建测试环境，测试脚本的配置和具体测试步骤的进行。最后本章对测试结果进行了分析。通过测试结果表明，该方案在时间和性能上都满足要求。

第六章 总 结

针对课题所需基本知识，查阅了大量相关文献，总结UEFI BIOS的主要功能和组成框架，分析了UEFI BIOS下通信过程中存在的安全缺陷。考虑UEFI BIOS中安全问题，本文设计并实现了IPSec安全架构。

本文通过应用ECC算法，使Pre-boot环境下的通信更加安全的同时，节省了时间和空间，充分满足了Pre-boot环境空间小，时间要求高等特点。最后本文对设计的安全模块进行了功能和性能的测试。测试结果显示该设计方案的载入对Pre-boot环境的启动时间的影响在可接受的范围内；在功能上，能够很好的保证Pre-boot环境下通信的安全。

本文的实现网络安全通信上还有很多不足。

首先是运用网络协议上。随着网络的迅速发展，网路相关的规范文档也在不断更新。IPSec 运用在 Pre-boot 环境下，是实现网络安全的内在要求。在 UEFI BIOS 中实现 IKEv2 是大势所趋。IKEv2 在实现方法和具体字段的运用上比 IKEv1 有更大的提高。因此，将 IKEv2 运行于 Pre-boot 环境下，实现 Pre-boot 环境的安全通信是大势所趋。

其次，本课题主要针对网络功能和性能方面做了基本的测试，保证设计架构能够有效的正常工作。但是，测试的范围有限，软件还需要更多测试工作的执行来保证其性能的进一步完善。

最后，本文在设计上主要考虑 Pre-boot 环境下的安全方案设计。随着通信网络的高速发展，安全方案的实施需要进一步完善。

总而言之，本文主要实现了Pre-boot环境下的IPSec架构设计和实现，从而有效的保证了该环境下网络通信的安全。但是本课题的研究还需要更多人的努力来进一步完善，进而使UEFI BIOS能够更快更好的发展。

参 考 文 献

[1] EFISpecification2.3,ver.1.1[DB].IntelCorporation,http://developer.intel.com/technoledgy/efi,2009.

[2] S. Kent，K. Seo. Security Architecture for the Internet Protocol. RFC4301, December,2005.

[3] 胡可杨. 基于 EFI 的软件测试在 TIANO 中的应用与实现. [学位论文]. 山东大学. 2007.

[4] 石浚菁.EFI 接口 BIOS 驱动体系的设计实现与应用. [学位论文]. 南京航空航天大学. 2006.

[5] 潘林，王晓箴，刘宝旭. 基于 EFI BIOS 的 Ukey 设备驱动的设计与实现. 计算机工程与应用. 2008，44(34)：69-71.

[6] 王健. 基于 TCP/IP 协议栈的网络技术教学软件研究与开发. 设计:[学位论文]. 天津大学. 2008.

[7] 张清华，尹龙军，刘勇. 椭圆曲线机密体制在数字签名中的应用. 重庆邮电学院学报，2006, 18(2).

[8] 曹建，陆建德. 一种基于 ECDH 方案实现 IKE 密钥交互的分析与设计. 计算机应用与软件, 2008, 25(5).

[9] S. Kent. IP Authentication Header. RFC4302,December 2005.

[10] S. Kent.IP Encapsulating Security Payload (ESP). RFC4303,December 2005.

[11] D.Fu, J.Solinas. ECP Groups for IKE and IKEv2. RFC4753,January 2007.

[12] D.Fu, J.Solinas. IKE and IKEv2 Authentication Using the Elliptic Curve Digital Signature Algorithm (ECDSA). RFC4753,January 2007.

[13] 朱贺新. 基于 UEFI 的可信 BIOS 平台研究与应用. [学位论文]. 山东大学.2007.

[14] D. Fu，K. Seo. ECP Groups For IKE and IKEv2. RFC 4753, January 2007.

[15] C. Kaufman, Ed. Internet Key Exchange (IKEv2) Protocol. RFC4306, December,2005.

[16] 张禾瑞. 近世代数基础，高等教育出版社，1978.

[17] 谢勇, 来学嘉, 张熙哲. 基于双核 EFI 的安全框架. 计算机工程. 2008,34(22).

[18] 徐娜. 基于可信赖计算平台的可信执行环境研究与实现. [学位论文]. 北京：中国科学研究院. 2006.

[19] 王海波, 张申生, 孙元浩. EFI/TIANO 下的图形界面新方案 NUWA. 计算机应用与软件. 2008，25(6).

[20] 赵宇. 基于 TPM 规范的 HMAC/SHA-1 IP 设计. [学位论文]. 上海交通大学. 2007.

[21] 王新成. 可信计算与系统安全芯片. 计算机安全, 2005,(5)

[22] 许顺利. 基于 BIOS 的计算机安全子系统的研究与实现. [学位论文]. 清华大学.2005.

[23] 潘登. EFI 结构分析及 Driver 开发. 计算机工程与科学,2006,28(2):115-117

[24] 张琳, 张永平. IKE 协议及其安全性分析. 计算机工程与设计.2005，9（26）：2473-2475

[25] 韩秀玲. IPSec 中密钥交换协议认证过程的研究及协议的改进. 计算机工程与应用. 2002,38(18):29-32.

[26] 张焕国, 刘玉珍. 密码学导论. 武汉大学出版社. 2003，10-19.

[27] 吴恩平, 奚自立, 马定贤. TCP/IP 安全问题. 计算机应用与软件.1997,23.

[28] 雷震甲. 网络工程师教程. 清华大学出版社. 2004.

[29] 王英, 刘迎国, 朱培栋. TCP/IP 协议栈基于语法的健壮性测试. 计算机技术与自动化.2004,23(4).

[30] 王春森. 系统设计师. 北京:清华大学出版社,2001.

[31] 彭汉礼, 王华伟. TCP/IP 网络环境下文件传输功能的设计与实现. 武汉工业大学学报.2000，22(4).

[32] 郭军. 网络管理. 北京：北京邮电大学出版社, 2001.

[33] 劳帼龄. 网络安全与管理. 北京：高等教育出版社,2003

[34] 白以恩. 计算机网络基础及应用. 黑龙江：哈尔滨工业大学出版社,2003.

[35] 张世永． 网络安全原理及应用. 科学出版社.2003,248-249.

[36] 李学俊, 敬忠良. 基于椭圆曲线离散对数问题的公钥密码. 计算机工程与应用. 2002,38(6)：20-22.

[37] 张明志. 用圆锥曲线分解整数. 四川大学学报(自然科学版).1996,33(4):356-359.

[38] 周玉洁, 冯登国. 公开密钥密码算法及其快速实现. 第一版. 北京：国防工业出版社.2002,1-5.

[39] 张泽增, 赵霖. 计算机加密算法. 西安：西安电子科技大学出版,1997.

[40] 刘淳, 张凤元, 张其善. 基于智能卡的 RSA 与 ECC 算法的比较与实现. 计算机工程与应用.2007,43(4).

[41] 徐海霞，李宝．分布式密钥分发方案的安全性证明．软件学报.2005,16(4):570-576.

[42] 白国强, 马润年, 肖国镇. 离散对数问题为特殊的椭圆曲线离散对数问题. 西安

电子科技大学学报.2001,28(2):254-257.

[43] 侯红祥. 椭圆曲线上快速标量乘算法的研究. [硕士论文]. 扬州大学.2008.

[44] 葛学锋. 基于 ECC 和 IPSec 的无线局域网. [硕士论文]. 浙江大学.2008.

[45] 姚砺. 面向对象软件测试的研究. [硕士论文]. 浙江大学.2002.

攻读硕士期间发表论文

孙小霞，王直杰. Pre-boot 环境下 ECC 的应用及性能优化. 计算机应用与软件(中文核心)，已录用.

致　谢

本硕士学位论文从选题、调研、研究到最后定稿都是在王直杰导师的悉心指导下完成的。王老师渊博的学识、严谨的治学态度、谦虚求实作风、诲人不倦的治学精神、虚怀若谷的为人品格都深深地感动着我，对我今后的学习和工作产生深远的影响。感谢王老师在学习和生活中的悉心指点和无微不至的关怀和帮助，在此论文完成之际，谨向我敬爱师致以衷心的感谢和崇高的敬意。

感谢我的同学，他们在论文写作期间提供了无私的帮助与支持，没有他们的帮助论文将不能顺利的完成，同窗之谊将终身难忘！感谢在这段时间里给予我支持与鼓励的同学。

最后感谢各位专家评委对本论文的悉心批评和指正！