ForCES 架构 VPN 的 IPsec 关键模块的设计及在 NP 上的实现

摘 要[1]

针对当前网络设备的封闭性、垄断性和灵活性差等缺点，国际互联网工程任务组中路由领域的 ForCES 工作组提出了转发件与控制件分离的开放性路由器体系结构。随着互联网的迅速发展和计算机网络的广泛使用，网络安全问题也越来越得到重视。因此，网络设备的开放性和安全性成为了 ForCES 工作组的重点研究方向。

本文主要研究基于网络处理器（Network Processors,NP）的 ForCES 架构 VPN 中实现的若干关键技术，包括：（1）根据 ForCES 体系结构对 VPN 作整体架构设计；（2）根据 ForCES FE 模型对 VPN 进行 LFB 建模；（3）根据网络处理器的特点对其资源进行高效率的分配；（4）基于 ForCES 体系结构对 VPN 的配置协议进行研究等等。

在对以上关键技术研究的基础上，本文主要做了以下几方面的工作。

1、 提出了基于 NP 的 ForCES 架构 VPN 的体系结构，使得 VPN 的扩展非常灵活。

2、 提出了基于 ForCES 架构 VPN 的 IPsec 入站和出站处理 LFB 模

[1]资助项目：国家 863 计划项目“ForCES 体系结构与关键技术研究和国际标准制定”(2007AA01Z201)；国家自然科学基金：“松散耦合型分布式路由器的若干关键技术研究”(60603072)；浙江省科技计划重大专项“ForCES 国际标准制定及产品研发与产业化应用”(2006C11215)；浙江省教育厅重点项目“基于 ForCES 结构的分布式路由器的关键技术研究”(20061070)。

型。第三方的安全管理软件可以基于 IPsec LFB 进行开发而不用关心 VPN 底层硬件的实现细节。

3、 在基于 Intel 网络处理器的可移植性框架中开发实现了 IPsec VPN 的 LFB，验证了本文提出的 IPsec LFB 模型实施的可行性。

4、 设计并实现了一种 NP 的嵌入式 Linux 系统中基于字符设备驱动的用户空间与内核空间通信的 IPsec SA 的配置协议，用户空间与内核空间的 IPsec SA 配置遵循这种协议，使得 IPsec SA 的管理具有一定的规范性和扩展性。

最后，对本文所实现的基于 NP 的 ForCES 架构 VPN 的样机进行了测试，结果表明本文提出的基于 ForCES 架构的 VPN 模型是正确的和高效的。

关键词：网络处理器; ForCES; 逻辑功能块; VPN; IPSec;字符设备驱动

DESIGN OF IPSEC KEY MODULES OF FORCES BASED VPN AND IMPLEMENTATION ON NETWORK PROCESSOR

ABSTRACT

To overcome the disadvantages of closeness, monopolistic and poor flexibility of current network equipment, ForCES working group of Internet Engineering Task Force in routing area proposes architecture of router with the separation of Forwarding Element and Control Element. With the development of Internet and the wide use of computer networks, information security is getting more and more important. Therefore, the openness and security of the next generation network has presently become the ForCES working group's important research direction.

The primary research works of this paper are several key technologies in the implementation of VPN on ForCES architecture which based on network processor, such as: (1) Designing the overall VPN construction according to the ForCES architecture. (2) Modeling of VPN related LFBs according to ForCES FE model. (3) Assigning the resource of network processor efficiently according to the characteristic of it. (4) Researching on the configuration protocol of VPN which based on the ForCES architecture and so on.

Based on the research of above key technologies, the main work of this paper is included as following aspects:

- Proposed a ForCES-based VPN architecture based on network processor, causing the VPN to expand extremely flexibly.
- Proposed the model of IPsec inbound and outbound processing LFB based on ForCES architecture. The third party of safety management software may carry on the development based on these LFBs while not to care about the detail of hardware realization.
- Developed and implemented IPsec VPN related LFBs in the Intel IXA architecture, and verified the validity of LFB models proposed by this paper.
- Designed and implemented of configuration protocol of IPsec SA between kernel space and user space of embeded linux system of NP, which is based on the Character Device Driver mechanism. The management of IPsec SA is of specification and expansion when the user space and kernel space followed this protocol.

We have implemented a ForCES-based VPN prototype based on Intel NP platform. Several tests are done on the prototype. Test result shows that the function of the VPN is correct and highly efficient. The architecture proposed by this paper testifies the feasibility of ForCES specification and provides the important technical parameter for the

ForCES application.

KEYWORDS: network processor; ForCES; LFB; VPN; IPSec; character device driver

独 创 性 声 明

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

签名：[illegible] 日期： 年 月 日

关于论文使用授权的说明

本学位论文作者完全了解浙江工商大学有关保留、使用学位论文的规定：浙江工商大学有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅和借阅，可以将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、汇编学位论文，并且本人电子文档的内容和纸质论文的内容相一致。

保密的学位论文在解密后也遵守此规定。

签名：[illegible] 导师签名：[illegible]

日期： 年 月 日

1. 绪论

Internet 从诞生到现在，以惊人的速度在发展。Internet 的互联性和开放性使得世界各地的人们可以实现信息的交换和共享，使得我们的生活和工作都变得更加便捷。但是，在 Internet 给我们提供以前无法想象的方便的同时，它也为黑客和计算机犯罪提供了可乘之机，使得我们需要考虑更多的安全因素。目前，外部网络对内部网络的非法访问和攻击以及在 Internet 中传输的信息被窃取或非法篡改是我们面临的最重要的不安全因素。具备数据认证和加密功能的安全网关是目前保证网络安全行之有效的手段之一。

1.1. 研究背景

1、ForCES 技术

ForCES 是 IETF（互联网工程任务组） 路由领域(Routing Area)的一个工作组，专门研究开放可编程的路由器体系结构和协议问题，是当前开放可编程网络研究最受关注的研究组织之一。它的基本思想是把路由器分成转发件（Forwarding Elements，FE）和控制件（Control Elements，CE），并认为其可由 CE、多个（可达几百个）FE 以及连接它们的 ForCES 协议[1]构成。

将转发件与控制件分离需要定义一个结构化的框架以及相关的协议，使一个 ForCES 网络单元(Network Element，NE)内部的控制平面（Control Plane）和转发平面（Forwarding Plane）之间的信息交换标准化，从而使 CE 与 FE 成为在物理上分离的标准组件。这种在物理上分离网络设备组件，并使用统一的标准和规范来指导产品的开发与运营的做法，充分发挥了开放网络的优势，使得各个组件按照相应的功能划分，各自独立发展，互不干涉，在一定程度上加速了这些组件的开发与发展。另一方面，各组件间相应的标准协议接口，又能迅速地把这些组件有机地组合成为一个整体，实现与普通组件组成的系统完全相同或更多的功能。

迄今为止，IETF ForCES 工作组已经完成了 ForCES 需求[2]（ForCES Requirements，RFC3654）、ForCES 框架[3] (ForCES Framework，RFC3746)，当前的工作重点是 ForCES 协议和 ForCES FE 模型[4]。自从 IETF ForCES 工作组成

立以来，本课题研究组的部分成员和其他来自 Intel，Nokia，IBM 和 Znyx 等公司的代表一起通过邮件和参加 IETF 国际会议等形式相互交流和讨论，并作为主要作者参与了 ForCES 协议规范(ForCES Protocol Specification)的设计与编写工作。

2、网络安全与 VPN

随着 Internet 在全世界范围内的迅速扩展，人们越来越依赖它进行方便、快捷的信息交流，Internet 使用 TCP/IP 协议将各种信息封装成 IP 包在网络上传递，然而，目前流行的 TCP/IP(IPV4)协议在设计之初侧重于效率而忽略信息的保密、认证等安全性问题，网上传送的信息可能被偷看(无法保密)，可能被篡改(无法保证完整性)，人们对接收到的信息无法验证它是否真的来自于可信任的发送者(无身份验证)，人们也无法限制非法或未授权用户侵入自己的主机等等。TCP/IP 的安全缺陷让网络黑客有可乘之机发动各种攻击(比如地址欺骗攻击、拒绝服务攻击等)，同时也无法满足人们对网络传送信息越来越高的安全要求(比如个人电子邮件、信用卡号的保密，电子商务活动中商家、客户及银行的各种敏感信息的安全，分散办公的企业内部信息的保密和安全访问控制等)。为了弥补 TCP/IP 的安全缺陷，人们制定了各种安全措施，有的在应用层实施(如 EMAIL，客户端使用 PGP(Pretty GoodPrivacy，一种加密软件)来保障电子邮件安全)，有的在传输层实施(如 WWW 目前使用 TLS 协议在 TCP 顶端提供身份验证、完整性校验和保密性安全服务)。IPSec（IP Security）则是在网络层针对 IP 包提供数据保密性、数据完整性、数据源认证和抗重播的安全服务[5]，由于 Internet 上所有信息都通过 IP 包传送，因此 IPSec 是目前唯一一种能为任何形式的 Internet 通信提供安全保障的协议，可以不用修改任何上层应用协议和传输层协议，“无缝”地从网络层获得安全保障，共享 IPSec 提供的安全服务，实施 IPSec。以实现端到端、安全的虚拟专用网 VPN，满足人们对 Internet 传送信息的安全要求[6]。

在国外，Internet 已成为全社会的信息基础设施，企业端应用也大都基于 IP，在 Internet 上构筑应用系统已成为必然趋势，因此基于 IP 的 VPN 业务获得了极大的增长空间。在 2003 年，全球 VPN 市场达到了 180 亿美元，预计到 2008 年将超过 300 亿美元。VPN 将会成为人们网络生活的重要组成部分。在不远的将来，VPN 技术将成为广域网建设的最佳解决方案，它不仅会大大节省广域网的

建设和运行维护费用，而且增强了网络的可靠性和安全性。同时 VPN 对推动整个电子商务、电子贸易将起到不可低估的作用[10]。

3、NP 技术

NP（网络处理器）技术的采用实现了 CPU 的业务控制和 NP 的数据转发这两个层面的分离，能够使 CPU 专注于它所擅长的方面。将 CPU 从繁琐的业务控制流程里分离出来，从而能够保证网络更好的安全策略管理。现在操作系统基本上也是多核系统，虽然在安全保证问题上表现良好，但是它在可扩展性和性能上的提高是存在瓶颈的。CPU 对于业务的控制和网络数据的转发引擎处理分离以后，可以降低对操作系统的要求。CPU 可以做非常复杂的安全控制，即使我们处理策略不是最优化的，功能还需要持续增加，但是由于网络数据处理引擎能够保证数据的快速转发，CPU 载荷会大为降低，NP 技术最大的优势就是它在系统整体性能上表现优异。NP 技术采用了以后能够实现网络安全数据转发和过滤的全线速，但在安全领域涉及到安全的过滤、加/解密处理时性能会有所下降，从系统的整体表现而言，还是要大大优于 CPU 架构的网关。

网络设备技术和路由交换技术经过了二十多年的发展，起初由于网络带宽小，设备都是以 CPU 的处理为中心的体系结构，随着网络带宽的技术的升级发展到 ASIC 和 NP 体系结构。而安全网络产品，直到前几年才有了快速发展，但对于大多数网络安全设备而言，在计算机架构方面还停留在以 CPU 集中处理为主的技术阶段。

NP 是一种可编程器件，它特定的应用于通信领域的各种任务，比如包处理、协议分析、路由查找、声音/数据的汇聚、防火墙、QoS 等。

1.2. 国内外研究现状

1.2.1. ForCES 框架研究

在 2004 年初，ForCES 工作小组将 GRMP、FACT、Netlink2 三个候选协议合并生成了 ForCES 协议。目前，ForCES 协议还没有生成 RFC，所以现阶段的主要工作是研究实现 ForCES 系统结构。现今，国际上对 ForCES 系统结构研究如下[9]：

1、Intel CP-PDK

Intel 公司开发出了一套基于 NPF 标准的控制平面软件开发套件（Control Plane-Platform Development kit，CP-PDK），CP-PDK 作为 Intel IXA-SDK4.1（Internet Exchange Architecture-Software Development Kit）（IXA-SDK 是 Intel 为其网络处理器提供的一套集成开发环境）的一个组件提供。网络处理技术论坛（Network Processing Forum，NPF）是由 70 多家网络行业公司组成的一个组织，负责制定通用规范，旨在推动和加快基于网络处理技术的网络和电信产品的开发。PDK 构建于可扩展的体系结构之上，符合 NPF 软件基础和 IETF ForCES 消息发送协议。它可将控制平面与数据平面的功能相分离，从而实现了产品的并行开发，进而大幅缩短产品上市时间。Intel 在其 CP-PDK 中说明使用 ForCES 协议作为其控制平面和转发平面的通信标准，但是并没有在其中实现 ForCES 协议。

2、我们的工作

我们在通用处理器上开发 CE；以网络处理器为硬件平台，以 MontaVista Linux 为软件平台，在 Intel IXA-SDK4.1 的基础上开发 FE，并依据最新的 ForCES 协议草案（Internet Draft），对 ForCES 协议软件进行研究与实现。目前，我们正在根据 ForCES 协议开发 ForCES 原型机 v3。

3、Flexinet

Flexinet 也是基于 ForCES 结构。和我们一样，该项目的转发件也是基于网络处理器进行开发的。和我们课题组不同的是控制件采用 Java 而不是 C/C++作为编程语言。

4、Distributed Control for Decentralized Modular Routers

瑞典的 M. Hidell，O. Hagsand，P. Sjodin 实现的路由器，也是基于 ForCES 结构的。其系统控制件基于 Unix 和 Zebra 路由软件，转发件采用通用 Linux。

1.2.2. VPN 研究

虚拟专用网是近几年迅猛发展起来的网络技术，研究的范围和深度也越来越广泛和深入。IETF 已经制定了多个关于 VPN 方面的 RFC。同时国内外很多厂商也已经根据这些 RFC 以及一些业界标准生产出了许多适合不同应用的 VPN 设备。特别是国外的网络设备、操作系统厂商在 VPN 方面都走在了最前端，Cisco、

ASCEND、NetScreen 和 Microsoft 等大公司都从不同的方面做着完善 VPN 的工作。根据所选协议的不同，不同的公司会侧重不同的 VPN 协议，例如 Cisco 和很少的几个厂家倾向于 L2F，Novell，在某种程度上包括 Cisco，也积极探索 IPSec[7][8]。

就产品而言，在国外 Cisco、Netscreen 的产品处于领先的位置，Netscreen 500、1000 系列和 5000 系列产品为支持 VPN 的高端路由器。在使用 3DES168bit 的算法时其 VPN 的处理速度为 250Mbps 到 6Gbps。之所以能够达到如此高的速率，是因为它使用最多达 6 个处理器并行的工作，网络接口使用的是千兆模块。其它的特性有：1、最大实现 25000 个隧道；2、支持手工密钥配置、IKE 密钥协商和 PKI 公共密钥体制 X.509 证书系统；3、支持 3DES168bit、AES128bit 和 DES56bit 的加密算法。低端产品有 NetScreen 5 和 50 系列防火墙产品。其 VPN 的通过率可达 10Mbps。支持 DES 和 3DES 加密算法。

Cisco 支持 VPN 的路由器很多。在使用 3DES 结合 SHA1 的情况下测试其 VPN 的处理速度如下：Cisco 2600 10Mbps；Cisco 2650 14Mbps；Cisco 3660 40Mbps；Cisco 7100 140Mbps；Cisco 7200 145Mbps。这些设备都实现了 IPSec、GRE、L2TP 和 PPTP 隧道协议，使用 DES、3DES 加密算法，支持 X.509 证书系统、RSA 签证、共享密钥、Radius 协议、CHAP/PAP 协议等，使用 IKE 密钥管理协议。Cisco 的 PIX 防火墙在安装了 VPN 模块后也支持 VPN 功能，这些产品都是用专用的平台和 Cisco IOS 操作系统。

在国内，VPN 产品也很多，有软件实现的 VPN，如安联软件 VPN；有软硬结合的 VPN，如 SWAN VPN、西马 VPN、大卫 nGuarder VPN 系统等。

其中，安联软件 VPN 可以使用 128bit 硬件加密卡。SWAN VPN 在 10M Ethernet 的环境下，其使用 3DES 与 SHA1 相结合的处理速度只有 3Mbps。不支持远程 VPN 客户端。西马 VPN 为西马安全网关的可选模块，其处理速度不详，支持 RSA 和 3DES 算法。大卫 VPN 系统由安全客户端、安全网关和目录服务器组成。其中安全客户端为软件实现，安全网关和目录服务器是基于专有硬件平台实现的，其安全网关的 VPN 处理速度不详[9]。

就我国互联网的使用现状而言，随着企业上网、政府上网工程的推进，互联网普及到了学校、政府部门、企业、科研单位、军事单位等等。这些企事业单位

的网络基础设施已经建立，但却没有任何保护，信息安全问题亟待解决。而且目前国内的相关产品和处理速度过低、或存在安全隐患，远远不能满足需求。

1.2.3. 基于 NP 的 ForCES 架构 VPN 研究

IETF 已经就基于 ForCES 架构的 VPN 相关技术进行了研究，并根据 ForCES 协议制定了相关的一些逻辑功能块（Logical Function Block，简称 LFB）。例如华为作为 IETF 路由领域的一个工作组成员，就根据 ForCES 协议提出了一套 VPN 相关的 LFB 模型库，并列在了 IETF 的 draft 中。可以参考：

http://www.ietf.org/internet-drafts/draft-halpern-forces-lfblibrary-vpn-00.txt

其它一些大学及其研究机构也在针对网络处理器开发 VPN，并发表了多篇关于在网络处理器中实现 VPN 的论文。例如：中国科技大学研究所、中国科学院以及福州大学计算机学院等研究和发表的“基于网络处理器的高性能 IPSec VPN 的设计方案”[11]、“基于网络处理器的 IPSec 协议实现技术的研究”[12]、“基于网络处理器的 IPSec 的实现方案研究”[13]等多篇论文。

同时一些个人和组织也在开发开源的 IPSec VPN[14][15]，为 VPN 的进一步应用和改进做出了巨大贡献，例如 OpenSwan[16],它是一个免费的、公开源代码的 VPN 安全软件，它的开源以及应用为开发基于 ForCES 架构的 VPN 提供了重要参考，一些公司和组织就是用它来作为 ForCES 架构路由器中的第三方软件。

一些公司也在针对网络处理器开发 IPSec VPN，例如信雅达股份有限公司致力于研究开发基于 Intel IXP1200 架构的 VPN 网关产品，实现了国际上网络安全产品的先进设计理念，研制成功了以自我核心技术为主体的虚拟专用网系统(VPN)。 初步形成信雅达(10M/100M)安全系列产品：信雅达 VPN-Gateway、信雅达 VPN-Remote、信雅达 VPN-SMC、信雅达 VPN-OPTS、信雅达 VPN-miniCA 。还有 Intel 开发的 IXP2855 网络处理器，它也是利用转发件和控制件分离的思想进行实现的。他们在单个高性能处理芯片上集成实现数据的加解密，完成的算法有 DES、3DES、AES 和 SHA-1，数据的加解密速度达到了 10 Gbps。

此外一些参加 ForCES 标准协议制定的公司[17][18]，如 ZNYX，也在研究和设计基于 ForCES 架构的 VPN，他们采用 SG8001 作为 FE，加解密速度达到 50Mbps。

1.3. 本文的研究内容

本论文依托浙江省科技计划重大专项“ForCES 国际标准制定及产品研发与产业化应用”，主要研究基于网络处理器的ForCES架构开放可编程路由器中VPN功能的实现技术。主要研究内容如下：

内容 1：ForCES VPN 的体系结构研究

在 ForCES VPN 体系结构中，LFB 作为系统资源是提供各种网络安全服务的实体和要素。CE 可以实现对 LFB 的完全控制，包括 LFB 拓扑配置和 LFB 属性配置。

内容 2：ForCES VPN 中相关 LFB 的研究

ForCES VPN 内的资源被表示成各种不同的 LFB，LFB 及它的属性都是可以由控制件通过 ForCES 协议进行控制的，各个 LFB 之间的连接关系也是由 CE 经过 ForCES 协议定义，以形成不同的 LFB 拓扑结构、进而实现动态资源配置、以完成各种不同类型的网络安全服务。模型化的 LFB 允许我们用很少的 LFB 来准确的描述 FE 的功能（例如 IPv4 forwarder），而且能够描述更加复杂的网络功能，本文主要研究了 IPsec VPN 相关 LFB 的建模技术。

内容 3：基于 Intel 网络处理器的 ForCES VPN 中 LFB 的开发实现

IXA 可移植性框架[19][20][21]中的资源管理库为应用提供了访问微引擎的 API，比如硬件的资源管理接口[22]。开发人员可以利用资源管理库来实现对 LFB 属性库的开发。

内容 4：基于 Intel 网络处理器的 VPN 的配置协议研究

VPN 配置具有操作的多样性和参数的复杂性，通过对基于 Intel 网络处理器的 VPN 的配置协议使得软件的开发更具规范性和扩展性。

1.4. 本文的创新点与主要贡献

本文的研究内容基于 ForCES 体系结构的开放可编程路由器[18][19][20][21]，这种路由器的结构设计遵循 IETF ForCES 工作组提出的 ForCES 协议、ForCES FE 模型、ForCES 框架和 ForCES 需求等协议。本课题组部分成员作为 ForCES 协议规范的主要作者，参与了该协议的设计与编写工作，对该领域有颇深的理解与认

识。

本文关注 ForCES 架构下如何实现 VPN 的问题，主要的创新点和贡献有：

1. 提出了基于 NP 的 ForCES 架构 VPN 的体系结构，使得 VPN 的扩展非常灵活，厂商可以在保持控制件上的软件不变的情况下实现新的 VPN 转发件。
2. 提出了基于 ForCES 架构 VPN 的 IPsec 入站和出站处理 LFB 模型。第三方的安全管理软件可以基于 IPsec LFB 进行开发而不用关心 VPN 底层硬件的实现细节。
3. 在基于 Intel 网络处理器的可移植性框架中开发实现了 IPsec VPN 的 LFB，验证了本文提出的 IPsec LFB 实施的可行性。
4. 设计并实现了一种 NP 的嵌入式 Linux 系统中基于字符设备驱动的用户空间与内核空间通信的 IPsec SA 的配置协议，用户空间与内核空间的 IPsec SA 配置遵循这种协议，使得 IPsec SA 的管理具有一定的拘范性和扩展性。

由于 ForCES 协议还没有成为标准协议，并且基于网络处理器的 ForCES 架构路由器的开发在国内外尚处于研究阶段，所以本文的开发工作具有一定的创新性，为 ForCES 架构路由器支持网络安全功能提供了实例，而且为 ForCES 协议的可行性和推动 IETF ForCES 标准协议的制定提供重要依据，同时也为 ForCES 协议的继续研究提供了参考。

1.5. 本文结构

本文的正文部分共分为六章，现将其内容概述如下：

第一章是绪论，首先介绍了本课题的研究背景，包括 ForCES 技术、网络安全与 VPN，NP 技术，然后介绍了 ForCES 框架研究、VPN 研究以及 ForCES 架构 VPN 研究的现状和意义。在此基础上，提出本文的研究内容、主要贡献及创新点，最后是本文的内容安排。

第二章介绍了 ForCES NE 的体系结构和 ForCES FE 模型，然后介绍了 VPN 的一些基本概念、IPsec 协议以及 NP 架构与网络设备，最后在此基础上给出了基于 NP 的 ForCES 架构 VPN 的实现需求。

第三章详细介绍了 ForCES VPN 中实现 IPsec 的一些关键技术。包括基于网络处理器 IXP2850 的资源研究和分配，IPsec LFB 及其拓扑结构的研究和设计，安全关联数据库管理的研究和设计，以及基于 NP 的 IPsec SA 的用户空间与内核空间的通信机制研究和配置协议的设计等。

第四章重点阐述了 ForCES VPN 内 IPsec 的模块划分和关键模块的具体实现。包括基于 NP 的 IPsec 出站和入站核模块的实现，SADB 管理微模块和核模块的实现，并实现了 IPsec SA 的用户空间与内核空间通信，验证了本文设计的 IPsec SA 的通信协议的正确性和合理性等。

第五章主要根据本文给出的模型及实现方案进行测试，并对测试结果进行了分析，证明了本文提出的基于 NP 的 ForCES 架构 VPN 方案的正确性与可行性。

第六章对现有研究工作做了总结，并对进一步的研究工作提出了自己的想法和见解。

2. 基于 NP 的 ForCES 架构 VPN 实现的需求分析

传统的安全技术是将基于各种安全技术的产品彼此独立的实现、部署实施及维护，这样的防御系统不断需要巨大的管理配置开销及软硬件成本，重要的是面对隐藏性强、混合性高、从探测漏洞到发起攻击快的组合型攻击，单个技术已经不能抵御。基于 ForCES 的 VPN 研究不仅是针对各个技术本身分支的研究，它通过将网关设备分离为转发件与控制件，同时进一步将内部各资源部件以及相互连接的接口进行模块化和标准化。ForCES 技术可使网关的开发设计过程成为一个积木化搭建过程，在此基础上可实现网关设备的控制软件和基础硬件分离，不同的模块、软件和硬件可由不同厂家独立研发生产。基于 ForCES 的 VPN 允许各种网络安全功能的快速配置和重组，实现智能化的动态安全网络，可大大加快和方便网络安全升级。本章首先介绍了 ForCES 的体系结构以及 FE 模型，然后对 IPsec 协议及 NP 技术进行了研究，最后提出本文所要做的一些具体的研究内容。

2.1. ForCES 体系结构及 FE 模型

按照 ForCES 的基本思想[1][2][3][4]，将 ForCES 路由器分为 CE 端和 FE 端，CE 主要处理 ForCES 协议消息，并负责建立、配置和更新 FE 在处理数据包时需要查找的表和数据结构等。例如，CE 处理路由信息协议(Routing Information Protocol，RIP)的数据包，并发送 ForCES 协议消息通知 FE 更新本地的转发表。

FE 主要负责按线速处理和转发数据包，独立完成大部分数据包的转发，而对于一些涉及路由和其它协议消息的数据包，则需要递交给 CE 进行进一步处理。在 FE 内部，根据对数据包进行的不同处理操作，可将其分成不同的 LFB，常见的有分类 LFB，转发 LFB 和调度 LFB 等。

转发件 FE 内的体系结构主要由 ForCES FE 模型标准定义，FE 内基本结构如图 2-1 所示。

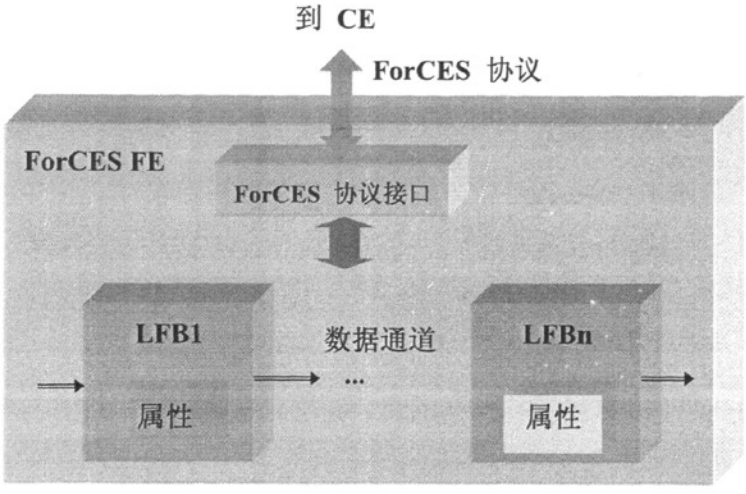


图 2-1 ForCES 转发件 FE 模型

FE 内的资源被表示成各种不同的 LFB。LFB 及其属性都是可以由 CE 通过 ForCES 协议进行控制的，各个 LFB 之间通过用于 IP 数据包传递的数据通道（Datapath）相互连接。该连接关系也是由 CE 经过 ForCES 协议定义，以形成不同的 LFB 拓扑结构，进而实现动态资源配置、以完成各种不同的 IP 类型服务。典型的 LFB 如分类器（Classifier）、 调度器（Scheduler）、IPv4 或 IPv6 转发器（Forwarder）等。

FE 模型可以表示出 FE 的能力、状态和配置。CE 根据这些信息对 FE 执行相应的控制操作。确切地说，该模型描述了 FE 的逻辑功能，以及这些功能所支持的能力和如何实现这些能力的。

FE Model 由两个主要部分组成：LFB 模型和 FE 属性。LFB Model 提供了定义 LFB 类的内容和数据结构。FE 属性描述了 FE 的能力以及 LFB 的拓扑结构等信息。

2.2. IPSec 协议研究

利用隧道方式来实现 VPN 时，除了要充分考虑隧道的建立及其工作过程之外，另外一个重要的问题是隧道的安全。第二层隧道协议只能保证在隧道发生端及终止端进行认证及加密，而隧道在公网的传输过程中并不能完全保证安全。IPSec 加密技术则是在隧道外面再封装，保证了隧道在传输过程中的安全性。

IPSec 是一个第三层 VPN 协议标准，它支持信息通过 IP 公网的安全传输。IPSec 系列标准从 1995 年问世以来得到了广泛的支持，IETF 工作组中已制定的与 IPSec 相关的 RFC 文档有：RFC2401、RFC2402、RFC 2409、RFC 2451 等。其中 RFC 2409 介绍了互联网密钥交换（IKE）协议；RFC 2401 介绍了 IPSec 协议；RFC 2402 介绍了验证包头（AH）；RFC2406 介绍了加密数据的报文安全封装（ESP）协议。

IPSec 兼容设备在 OSI 模型的第三层提供加密、验证、授权、管理，对于用户来说是透明的，用户使用时与平常无任何区别。密钥交换、核对数字签名、加密等都在后台自动进行。另外，为了组建大型 VPN，需要认证中心来进行身份认证和分发用户公共密钥。IPSec 可用两种方式对数据流进行传输：隧道方式和传输方式。隧道方式对整个 IP 包进行加密，使用一个新的 IPSec 包打包，这种隧道协议是在 IP 上进行的。传输方式时 IP 包的地址部分不处理，仅对数据净荷进行加密。

IPSec 的 ESP 协议和报文完整性认证的协议框架已趋成熟，IKE 协议也已经增加了椭圆曲线密钥交换协议。由于 IPSec 必须在端系统的操作系统内核的 IP 层或网络节点设备的 IP 层实现，因此需要进一步完善 IPSec 的密钥管理协议。

IPSec 支持的组网方式包括：主机之间、主机与网关、网关之间的组网。IPSec 还包括对远程访问用户的支持。IPSec 可以和 L2TP、GRE 等隧道协议一起使用，给用户提供更大的灵活性和可靠性。

2.2.1. IPsec 协议概述

IPSec 协议族是 IETF 制定的一系列协议，它为 IP 数据报提供了高质量的、可互操作的、基于密码学的安全性。特定的通信方之间在 IP 层通过加密与数据源验证等方式，来保证数据报在网络上传输时的私有性、完整性、真实性和防重放[23]。

- 私有性（Confidentiality）指对用户数据进行加密保护，用密文的形式传送。
- 完整性（Data integrity）指对接收的数据进行验证，以判定报文是否被篡改。

- 真实性（Data authentication）指验证数据源，以保证数据来自真实的发送者。
- 防重放（Anti-replay）指防止恶意用户通过重复发送捕获到的数据包所进行的攻击，即接收方会拒绝旧的或重复的数据包。

2.2.2. IPsec 体系结构

IPSec 将几种安全技术结合形成一个完整的安全体系，它包括安全协议部分和密钥协商部分。

（1）安全关联和安全策略：安全关联（Security Association，SA）是构成 IPSec 的基础，是两个通信实体经协商建立起来的一种协定，它们决定了用来保护数据包安全的安全协议（AH 协议或者 ESP 协议）、转码方式、密钥及密钥的有效存在时间等。

（2）IPSec 协议的运行模式：IPSec 协议的运行模式有两种，IPSec 隧道模式及 IPSec 传输模式。隧道模式的特点是数据包最终目的地不是安全终点。通常情况下，只要 IPSec 双方有一方是安全网关或路由器，就必须使用隧道模式。传输模式下，IPSec 主要对上层协议即 IP 包的载荷进行封装保护，通常情况下，传输模式只用于两台主机之间的安全通信。

（3）AH（Authentication Header，认证头）协议：设计 AH 认证协议的目的是用来增加 IP 数据报的安全性。AH 协议提供无连接的完整性、数据源认证和抗重放保护服务，但是 AH 不提供任何保密性服务。

（4）ESP（Encapsulate Security Payload，封装安全载荷）协议：封装安全载荷（ESP）用于提高 Internet 协议（IP）协议的安全性。它可为 IP 提供机密性、数据源验证、抗重放以及数据完整性等安全服务。ESP 属于 IPSec 的机密性服务。其中，数据机密性是 ESP 的基本功能，而数据源身份认证、数据完整性检验以及抗重传保护都是可选的。ESP 主要支持 IP 数据包的机密性，它将需要保护的用户数据进行加密后再重新封装到新的 IP 数据包中。

（5）Internet 密钥交换协议（IKE）：Internet 密钥交换协议（IKE）是 IPSec 默认的安全密钥协商方法。IKE 通过一系列报文交换为两个实体（如网络终端或网关）进行安全通信派生会话密钥。IKE 建立在 Internet 安全关联和密钥管理协

议（ISAKMP）定义的一个框架之上。IKE 是 IPSec 目前正式确定的密钥交换协议，IKE 为 IPSec 的 AH 和 ESP 协议提供密钥交换管理和 SA 管理，同时也为 ISAKMP 提供密钥管理和安全管理。IKE 具有两种密钥管理协议（Oakley 和 SKEME 安全密钥交换机制）的一部分功能，并综合了 Oakley 和 SKEME 的密钥交换方案，形成了自己独一无二的受鉴别保护的加密材料生成技术。

2.2.3. AH 协议

AH[24]提供无连接完整性、数据验证和可选的反重放保护，但不同于 ESP，它不提供机密性。因此其包头比 ESP 报头简单的多。图 2-2 说明了 AH 的数据格式。

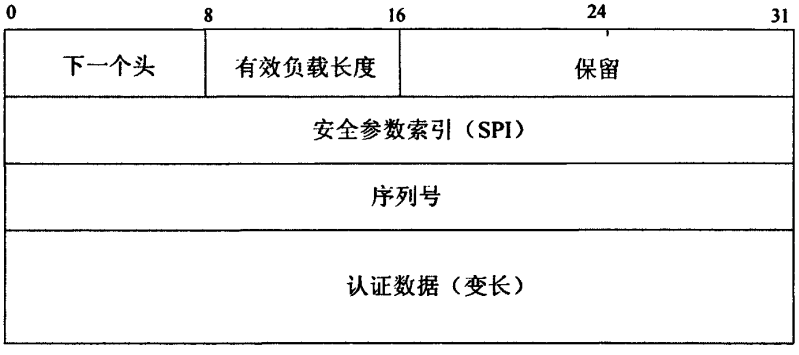


图 2-2 AH 头格式

AH 是一种 IP 协议，在 IP 包头中用 51 来表示。“下一个头”字段指出 AH 报头后面是什么内容。在传输模式下，这是被保护的上层协议（如 UDP 或 TCP）的编号；在隧道模式下，这个值为 4。

如果通信端点也是 IPSec 端点，则传输模式下的 AH 很有用。在隧道模式下，AH 封装 IP 分组，并在 AH 报头前面加入一个新的 IP 报头。虽然 AH 隧道模式可用于提供 IPSec VPN 端到端安全，但它不提供数据机密性服务，因此不是很有用。

“有效负载长度”字段指出了 AH 报头的长度。“保留”字段未被使用，因此被置为零。SPI 和序列号的作用与 ESP 中相同。验证摘要同 ESP 有一个重要的不同：在 AH 中，对 IP 报头和有效负载进行验证。由于 AH 根据包括 IP 报头在内的整个分组计算验证数据，而有些 IP 字段在传输过程中会发生变化，因此

计算验证摘要时将把 IP 报头中在传输过程中可能发生变化的字段排出在外。被排出在外的字段包括服务类型（TOS）、标记、分段偏移、存活时间（TTL）和报头校验和。之所以将这些字段排出在外，是因为验证在传输过程中被修改的值（如 TTL）将导致验证散列值与发送方的不同，进而导致分组被丢弃。

各字段含义如下：

1) 下一头（8 比特）：标识紧跟验证头的下一个头的类型。
2) 载荷长度（8 比特）：以 32 位字为单位的验证头的长度，再减去 2。例如，缺省的验证数据字段的长度是 96 比特（3 个 32 位字），加上 3 个字长的固定头，头部共 6 个字长，因此该字段的值为 4。
3) 保留（16 比特）：保留为将来使用。
4) 安全参数索引（32 比特）：用于标识一个安全关联。
5) 序号（8 比特）：单增的计数器值。
6) 验证数据（可变）：该字段的长度可变（但应为 32 位字的整数倍），包含的数据有数据包的 ICV（完整性校验值）或 MAC。

2.2.4. ESP 协议

ESP[25]提供了机密性、数据完整性以及可选的数据来源验证和反重放服务，这是通过对原始有效负载进行加密并将分组封装在一个报头和报尾之间来实现的，如图 2-3 所示：

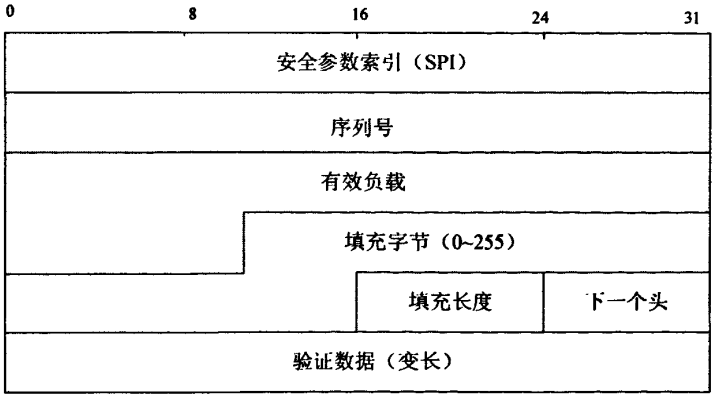


图 2-3 ESP 数据格式

在 IP 报头中，用 50 来表示 ESP。ESP 报头被插入到 IP 报头和上层协议报头

之间。在隧道模式下，IP 报头是新的；而在传输模式下，是原始 IP 分组的报头。在上节的传输模式与隧道模式中都已经作了简要的介绍。

ESP 报头中的安全参数索引（SPI）是一个 32 的值，它同前面的 IP 报头中的目标地址和协议一起指出用于处理分组的安全关联（SA）。SPI 是目标对等体在 Internet 密钥交换（IKE）协商期间随意选择的一个数字。其功能类似于索引号，可用于在安全关联数据库（SADB）中查找 SA。

序列号是由发送方插入到 ESP 报头中的，它是一个唯一的单调递增数字。序列号和接受滑动窗口一起提供了反重放服务。这种反重放保护机制是 ESP 和 AH 通用的。

被保护的数据（具体的说是被 ESP 加密的数据）位于有效负载数据字段中。用来对有效负载进行加密的算法可能需要初始化向量（IV），它也放在有效负载字段中。注意，可以对 IV 进行验证但不进行加密。如果使用的加密算法是 DES，则被保护的数据字段的前 8 个字节为 IV，3DES 和 AES 也使用 8 字节的 IV。

ESP 报头中的“填充”字段用于增加 ESP 报头的位数，填充的位数取决于使用的加密算法。“填充长度”字段指出了填充的字节数，以便解密时能够恢复原始数据。

“下一报头”指出了有效负载中的数据类型。例如：如果在隧道模式下使用 ESP，这个值为 4，说明内部是一个 IP 数据包。50 代表为 AH，51 代表为 ESP。

2.2.5. IPsec 的两种模式

AH 和 ESP 使用传输和隧道两种工作模式。其相应的 IP 包结构如

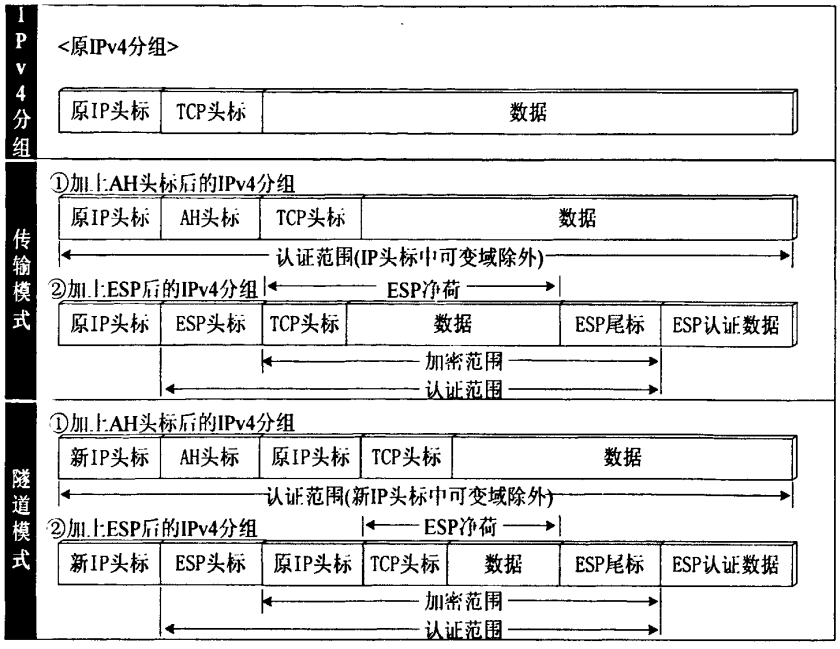


图 2-4 所示。

(1) 传输模式：主要用于保护 IP 包的负载，如 TCP/ UDP 数据段等，仅适用于主机间的点到点通信。

(2) 隧道模式：用于保护整个 IP 包。由于将原 IP 包封装在一个新的 IP 包中,因而原 IP 包在传输过程中，传输路径上的路由器都无法查看其 IP 头信息；新生成的 IP 头中的源、目的地址都与原 IP 头中的不同，从而增强了 IP 包在传输中的安全性。

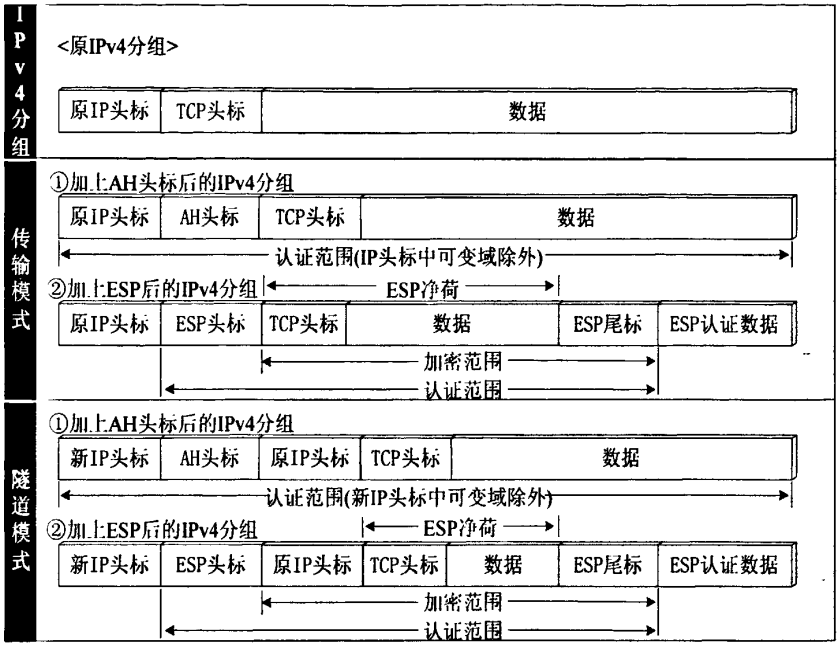


图 2-4 IPsec 的传输方式

2.3. NP 架构与网络设备

Intel 公司的 IXP 系列是网络处理器中的佼佼者，IXP 系列都是基于并行处理，为快速数据包处理而设计的。IXP 系列网络处理器的一个基本结构特征是使用一个核心处理器和多个 RISC 数据处理引擎（微引擎）并行工作，微引擎还支持多个硬件线程，以实现数据的线速转发。IXP2850[30]网络处理器是目前 IXP 系列 NP 中功能最为强大的处理器，它为高端核心设备设计，性能更为强大：支持 10Gb/s 应用，具有 16 个 1.4GHz 的微引擎，每个微引擎有 8 个硬件线程，每秒可以完成 250 亿次的操作，可实现 10Gb/s 的包转发速率。700MHz 的处理核心 Xscale，主要是面向 OC-48 到 OC-192 边缘网和核心网的应用，可以用于：骨干网的路由与交换、无线设备、IPSec 和 VPN、10Gb/s 的企业交换和路由等。同时，该处理器增加了 2 个 crypto 单元，用于加、解密，可达到 10Gb/s 的加、解密速度。

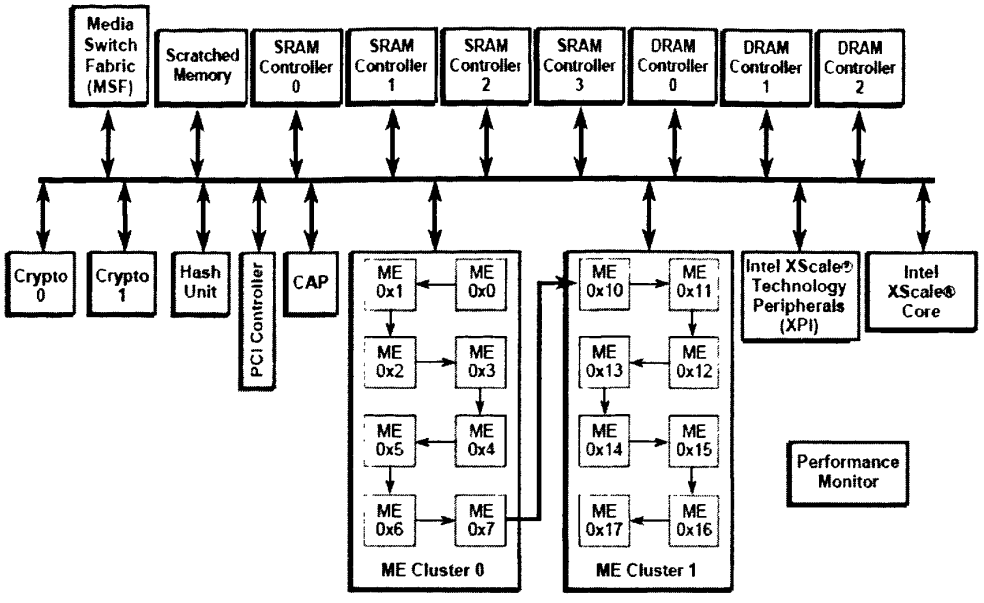


图 2-5 IXP2850 硬件结构示意图

由图 2-5 可知，IXP2850 网络处理器由 Xscale Core、Microengine、Media and Switch Fabric Interface（MSF）、SRAM Controller、DRAM Controller、SHaC、PCI、Chassis8 部分构成。以下简要介绍了各个部分的功能。

（1）Intel XScale® core:32 位嵌入式 RISC（精简指令集）处理器，时钟频率为 600MHz。负责处理网络处理器中的 Control Plane 处理任务，执行系统芯片初始化配置、系统控制/管理、运行路由协议栈、更新路由表等操作。另外，Intel Xscale Core 还负责对异常数据包进行处理。

（2）ME：微引擎，ME 是 IXP2850 的核心组件，也是 IXP2850 取得线速处理性能的关键所在，负责绝大部分的数据包处理任务。ME 能访问 IXP2850 中的所有共享资源：SRAM Controller、DRAM Controller、MSF、SHaC 以及 NNR。IXP 有 8 个 ME，分为两组：ME Cluster0 和 ME Cluster1，其中 0～7 号 ME 属于 ME Cluster0，7～15 号 ME 属于 ME Cluster1。

（3）Media and Switch Fabric Interface：介质和交换结构接口，简称 MSF。它是 IXP2850 与外部物理层设备（PHY）、交换结构（SF）的接口单元。是 IXP2850 接收、发送数据包的窗口。MSF 通过 UTOPIA、SPI 协议与 PHY 接口，通过 CSIX 协议与 SF 接口。UTOPIA、SPI、CSIX 都是经过标准化的协议，通过这些标准协议 IXP2850 能够方便地与其他厂家的产品进行接口。

（4）SRAM Controller：SRAM 控制器，用于接口 SRAM 存储设备，控制、管理 IXP2850 中其他功能单元对 SRAM 存储设备的访问、操作。IXP2850 中有 4 个 SRAM Controller：SRAM Controller0、SRAM Controller1，SRAM Controller2 和 SRAM Controller3 相互独立，可并行工作。另外 SRAM Controller 还可用于接口符合相应接口规范的协处理器，执行特定的复杂的数据包处理操作。

（5）DRAM Controller：DRAM 控制器，用于接口 DRAM 存储设备，控制、管理 IXP2850 中其他功能单元对 DRAM 存储设备的访问、操作。IXP2850 中有 3 个 DRAM Controller ，DRAM 存储设备的最大存储空间为 2GB，用于存储数据包、路由表等大型数据结构。

（6）SHaC 单元：包括 Scratchpad Memory（中间结果存储器）、Hash Unit（哈希单元）、CAP（Control Status Register Access Proxy，控制状态寄存器访问代理）3 部分。其中 Scratchpad Memory 的容量为 16K，用于微引擎之间的通信以及重要数据的内部缓存；Hash Unit 支持 48bit、64bit、128bit 的哈希运算；CAP 用于对 IXP2400 中的控制、状态寄存器进行访问、操作，用于设定 IXP2850 的工作模式和采集运行状态。

（7）PCI Controller：PCI 控制器，用于接口 Control P1ane Processor(控制面处理器)、Management Processor(系统管理处理器)、其他 IXP 网络处理器、以及 PCI 以太网卡等符合 PCI 规范的设备。PCI Controller 符合 PCIv2．2 规范，接口总线宽度为 64bit，时钟频率为 66MHz。

（8）Chassis：系统底盘，是 IXP2850 中各功能单元的内部高速通道，由多组单向高速数据总线、命令总线，以及相应的总线仲裁单元组成。

每个微引擎内部结构框图如图 2-6 所示。IXP2850 上每个微引擎都有 8 个硬件线程，为了节省开销，提高交换（context swapping）效率，每一个 context 都有自己的寄存器组（包括 Xfer reg，GPRs）、PC（程序计数器）、Local Register（本地寄存器），有单独的程序运行空间。每个微引擎都包含 4 种类型的 32 位数据通道寄存器[26]。

- 256 个 GPRs：通用寄存器，一般用作 Execution Data Path（执行数据通道）的 Source Reg（源地址寄存器）和 Dest（目的地址寄存器）。一个微引擎中有 256 个 GPRs，分为两个 Bank，A Bank 和 B Bank，各有 128

个 GPRs；

- 512 个 Xfer Register：传输寄存器，用作微引擎与其他功能单元进行数据交换的窗口，可分为 Xfer in reg 和 Xfer out reg 两大类。当进行读操作时，数据从其他功能单元写入 Xfer in reg，当进行写操作时，数据从 Xfer out reg 写入其他功能单元；
- 128 个 NNR：邻居寄存器，是 IXP2400 中相邻两个微引擎之间数据传输的快速通道；
- 640×32 位 LM：本地存储器，一个微引擎中 LM 的容量为 640×32bit。微引擎中每个 Context 都有两个 LM 的地址指针，用于 LM 的寻址。

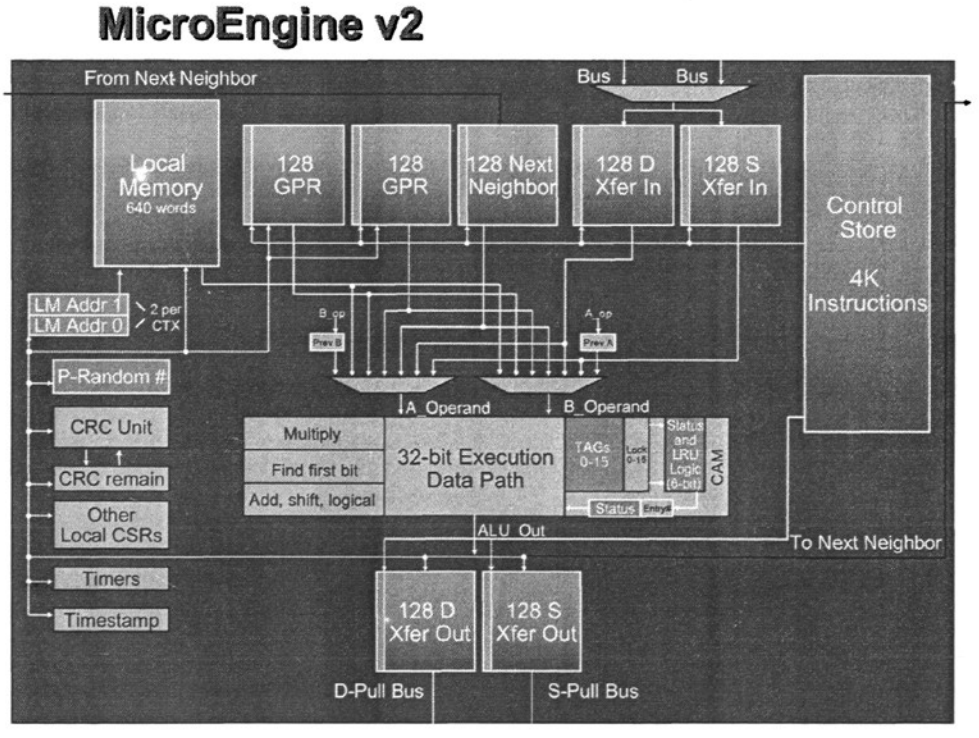


图 2-6 微引擎内部结构框图[33]

2.4. 基于 NP 的 ForCES 架构 VPN 的需求分析

根据 ForCES 技术和 VPN 技术的需求，当前研发 ForCES VPN 需要解决以下几方面的内容。

1、基于 ForCES 架构的 IPsec VPN 体系结构需求

LFB 作为系统资源是提供各种网络安全服务的实体和要素。首先，LFB 要像 Linux 内核模块那样可以实现动态加载和卸载，就必须设计好一个合理的 ForCES VPN 的体系结构模型，通过 ForCES 协议，实现 CE 对 LFB 的动态加载和卸载以及属性的配置等等。

2、基于 ForCES 架构的 IPsec VPN 体系硬件和软件架构需求

在 ForCES VPN 体系结构中，必须要采用适合开放可编程路由器的软硬件架构，必须要解决的问题有：a) 作为 FE 必须可以是即插即用的；b) 软件架构必须可扩展的，要像 Linux 内核架构一样，可以随时动态插入一些所需要的模块，提供新的功能。

3、基于 ForCES 架构的 IPsec 入站及出站 LFB 实现需求

ForCES VPN 内的资源被抽象成 IPsec 入站、出站及其他相关 LFB，LFB 及其属性都是可以由控制件通过 ForCES 协议进行控制的，LFB 之间的连接关系也是由 CE 经过 ForCES 协议定义，以形成不同的 LFB 拓扑结构、进而实现动态资源配置、以完成各种不同类型的网络安全服务。

考虑到 ForCES 工作组前期的工作重点在路由研究领域，对 VPN 并没有明确的定义。因此，当前的 ForCES VPN 研究急需解决以下几个关键问题：a）VPN 中 IPsec LFB 的模型建模方案；b）向 ForCES 工作组提交 VPN 中 LFB 的模型草案。目前初步根据 VPN 中的功能来界定 LFB：如防火墙内典型的 LFB，如过滤器、 内容处理、地址转换器、队列调度器、转发器等；VPN 网关内典型的 LFB，如数据加密、数据解密、IPsec 封装、IPsec 解封装等。

4、基于 Intel 网络处理器[27][28][29]的 ForCES VPN 中 LFB 的开发实现需求

IXA 可移植性框架中的资源管理库为应用提供访问微引擎的 API，比如硬件的资源管理接口。开发人员可以从资源管理库来实现对 LFB 属性库的开发。在核模块中的接口基础上开发，函数的功能强大，并且有一定的任务间管理机制，但是由于它在上层，缺少了对底层硬件资源管理的灵活度。

5、用户空间与内核空间 IPsec 参数配置的通信机制实现需求

基于操作系统的灵活性，用户空间与内核空间的可以采取多种通信方式，数据的单向或双向传输也有多种选择。针对 IPsec 配置参数的复杂性和多样性，可以归纳成一个通用的配置协议，完成 IPsec 配置多种操作的统一。

6、基于多核处理器的高性能系统开发技术的需求

随着互联网应用业务的迅速发展，网络威胁的不断扩张与多样性变化，用户对网络的速度与安全性要求急剧提升。面对数据包高速处理及安全方面计算密集的双重要求，围绕传统单核通用 CPU 建立的网络处理系统结构已经成为通信的瓶颈。多核体系结构则采用多核多线程技术，提高了数据处理能力，使其在网络安全领域有着十分广阔的应用前景，是当今计算技术的发展趋势。

研究基于多核体系结构的高性能系统开发涉及多核处理器体系结构、硬件单元与特性、并行处理模式、基于控制/快速数据平面开发平台的系统设计开发技术、微模块和核心组件设计、模拟与优化技术、资源分配与调度技术、ForCES 架构的 VPN 系统模块划分、高并行化模块设计等技术。以实现一体化管理服务的高性能线速处理。

综上所述，基于 Intel 网络处理器的可移植性框架中实现 ForCES 架构 VPN 以及开发实现 LFB 实体，是 ForCES 架构 VPN 的研究与实现中的重要内容。

2.5. 小结

本章首先介绍了 ForCES 体系结构及 FE 模型，VPN 技术及相关协议，最后根据所述的 ForCES 体系结构及相关技术，我们针对开发基于 NP 的 ForCES 架构 VPN 做了一些具体的需求分析，包括 ForCES VPN 的体系结构研究、VPN 相关 LFB 的分析、LFB 的开发实现以及用户空间与内核空间的 IPsec 参数配置实现等。

3. 基于 NP 的 ForCES 架构 VPN 实现的关键技术研究

本章主要是对基于 NP 的 ForCES 架构 VPN 实现所要解决的关键技术进行研究，包括体系结构的研究、NP 资源分配机制、IPsec VPN 的 LFB 建模、基于散列技术的安全关联数据库的管理机制研究以及用户空间与内核空间 IPsec SA 通信机制的研究。

3.1. ForCES 架构 VPN 的体系结构设计

根据软件任务及运行处理器位置的不同，将 ForCES VPN 的软件分布[36][37][38]在控制件和转发件上。图 3-1 描述了转发件上控制平面软件与微处理器上软件之间以及控制件软件各层之间存在复杂的信息交互与依赖关系[39]。

各个模块说明如下：

- 安全策略配置模块

包含命令行配置以及通过 UOM 界面（用户操作管理界面）进行配置，主要完成 SPD 以及 SAD 的配置。

- IKE 协议处理模块

完成 IPsec 安全关联的动态建立。使用开源的 Openswan 实现。

- IPsec 服务映射层模块

为上层应用提供 SPD 以及 SAD 的操作 API，这些 API 负责和具体的 LFB 打交道，完成安全策略以及安全关联的管理。

- IPsec 相关 LFBs 抽象层模块

实现 IPsec 相关 LFB 的控制接口。

- IPsec 核模块

完成对于 IPsec 入站处理微模块和 IPsec 出站处理微模块的管理，包括这些微模块需要的变量的分配以及 patch symbols,微模块的加载以及启动，异常数据包的处理。

- 安全策略处理核模块

完成对于数据包分类/安全策略处理微模块的管理，包括该微模块需要的分类表的分配以及 patch symbols。

➢ IPsec 入站处理微模块

完成 IPsec 数据包的入站处理功能。

➢ IPsec 出站处理微模块

完成 IPsec 数据包的出站处理功能。

➢ 数据包分类/安全策略处理微模块

完成数据包的分类和安全策略查找，以决定采用哪种方式对数据包进行处理。

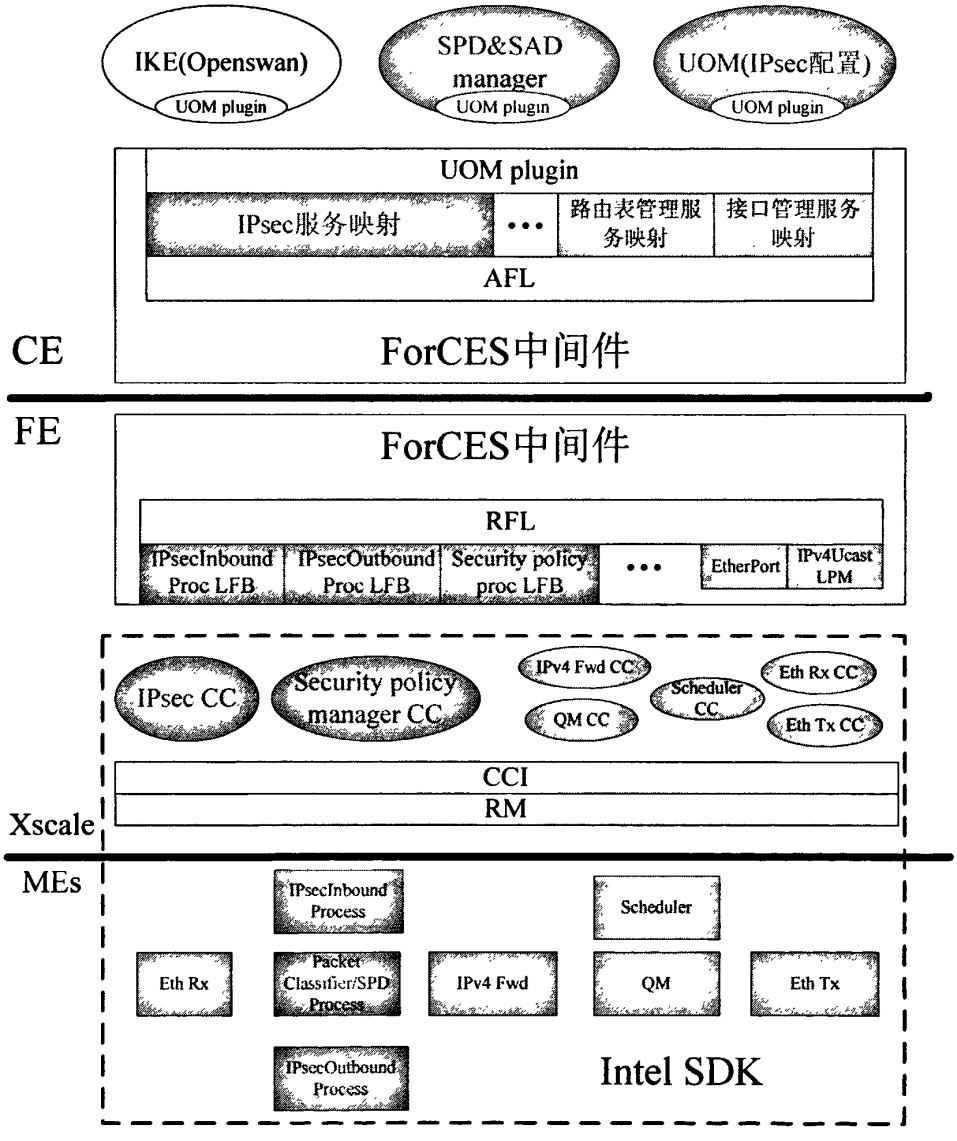


图 3-1 基于 ForCES 的 VPN 系统功能模块图

从功能上分，可以将基于 NP 的 ForCES VPN 的软件体系架构分成三个层面：

1、控制件

这里需要说明的是，控制件与上述的控制平面是两个不同的概念。控制件是 ForCES 的组成部分，而控制平面是多核体系结构的网络处理器根据代码任务及运行于转发件控制面处理器和微处理器位置的不同进行的层次划分。控制件既可以集中式存在于控制平面，又可以分布式存在于其它网络设备上，具有高度的灵活性。

控制件中的 ForCES 功能层是转发件中 ForCES 功能层的映射，将上层的操作信息定位到相应的目标 LFB。

控制件中 ForCES 功能层上方的服务 API 通过功能重组的形式向应用层软件提供各种服务，典型的服务有拓扑管理、网络管理、策略应用[31][32]等。此外，ForCES VPN 作为一个网络件，向第三方软件提供必须的支持，例如 RFC3654 已经提出 ForCES 协议对 SNMP 的支持需求。

2、转发件

转发件上根据软件运行的处理器的位置不同而分为控制平面软件和数据平面软件。数据平面主要运行在微处理器之上，其中包含了大量的微代码宏，这些宏与低层硬件结构密切相关，直接对硬件进行操作，利用这些宏[34][35]开发出的软件实现了与低层硬件的松耦合，具体完成输入端口和输出端口间高速数据包的处理（转发、分类、排队、调度、加密、检测等各种安全处理）功能，具有线速执行的特点，并充分利用数据包的无关性，采取并行处理方式，是各种 LFB 的微码层实现。控制平面运行在主控内核（Xscale core）上，负责数据平面上微代码宏的初始化配置、管理、异常数据包处理等任务，其中的 ForCES 功能层（以 ForCES 功能中间件的形式存在）对底层的 LFB 微码实体的行为动作进行抽象，抽象的结果表现为可直接控制底层的 LFB 微码实体行为的 API，这些 API 与底层的 LFB 微码实体共同组成了 LFB 的实体，即 LFB 的实现。

3、ForCES 中间件

CE 端嵌入式 ForCES 协议中间件和 FE 端嵌入式 ForCES 协议中间件作为 ForCES 协议的两个终端点，不仅规定了控制件与转发件之间的信息传递格式，同时屏蔽了 CE-FE 链路的多样性和复杂性，为上层开发者带来极大的方便。

上述给出了 ForCES VPN 的软件体系架构，该架构针对 ForCES 网络件，具有通用性，它适用于 ForCES 结构防火墙和 ForCES VPN 的软件架构实现。

3.2. NP 中高效率资源分配机制的研究

在一般程序设计中，可以不考虑操作系统和编译程序、线程调度的细节、寄存器的数量和容量，而在网络处理器的程序设计中，忽略这些因素就不能编写出优化的程序。在对网络处理器，尤其是微引擎编程之前，需要仔细了解网络处理器的系统结构和硬件资源。

3.2.1. 存储资源分配

基于 SRAM、DRAM 以及寄存器的各种不同的特点（2.3 节介绍），以及各种存储设备的读写速度差异[40]（如表 3-1 所示），我们需要根据不同的应用，定义不同的资源分配机制。就一般流程来说，接收模块的微引擎从 RBUF 取得报文，复制到 DRAM，同时读取报文头获得报文处理所需要的关键数据，作为包头描述符（packet descriptor metadata）写入 SRAM，通过一个简单的地址映射，使包头信息与 DRAM 中的包数据关联。在后续的报文处理中，可以尽量减少对 DRAM 中的访问。如，一个路由器的方案中，需要修改的包数据包括二层帧头、IP 头的 TTL/校验和等，如果每一次数据的修改都在 DRAM 中进行，无疑是非常低效的。通常的做法是维持一个全局的数据结构，存放在 Local Memory、Scratch 或寄存器中，在发送之前一起写入 DRAM 包数据中。

表 3-1 存储器访问差异

存储器	传输字节数	读操作时延(cycles)	写操作时延(cycles)
DRAM	8	295	53
SRAM	4	150	53
Scratch	4	100	40
Local Memory	4	3	3

因为 I/O 访问延迟，线程必须在适当的时候阻塞自己等待结果。连续的 I/O 操作会引起线程停止，导致系统性能急剧下降。解决方法主要是从程序流程上避免，同一线程 I/O 处理之间预留尽量多的指令周期；如果存在多个连续的 I/O 操作，可以一起等待 I/O 结束信号。

应用上述存储器特点，本文对 IXP2850 上的硬件存储器作如下分配，以完成 IPsec 数据包的处理：

考虑网络层协议头在路由器中处理的频率比数据包本身要大，而且协议头部字节比较小，所以将微引擎中每个线程可支配的两个 LM 地址指针所指向的空间中存放外部 IP 头，若为 IPsec 包，则存放的是 IPsec 隧道头，包含 IP 头以及 ESP/AH，而第二个 LM 指针存放内部 IP 头。

DRAM、SRAM、Scratch 是 IXP2850 的 3 大存储资源。其中 DRAM 的主要特点是存储空间大，但访问时延也比较大，主要用于大型数据的存储，考虑 IP 数据包的操作频率最小，且数据量庞大，所以 DRAM 中主要存储 IP 数据包；SRAM 的主要特点是访问时延较小，除了支持一般的数据读/写操作外，还集成了复杂的硬件结构支持原子操作，以及对 Linked List（链表）、Ring Buffer（环缓冲区）等复杂数据结构的自动操作和管理，SRAM 主要用于存储数据包描述符、环缓冲区、数据包发送队列（Linked List 的一种形式）等数据结构[30]，另外，本文设计将 SADB 存放在 SRAM 中，因为 SA 数据结构比较大，且使用频率相对也比较大，另外还可以使用 SRAM 支持的原子操作用于 SA 的内容进行自增操作，方便了 SA 的使用；Scratch 是 IXP2850 的片上存储单元，存储空间较小但访问速度快，主要用于进程之间的通信。

以上资源的分配，充分利用了 IXP2850 上存储资源的特点，保证了设备的充分利用和多任务并发处理的实现，大大提高系统的总体性能和处理速度。

3.2.2. 微引擎资源分配

IXP2850 上有 16 个 ME，分为两簇，ME0-ME7 为一簇，ME8-ME15 为一簇，把 IPsec VPN 的模块映射到 ME，如表 3-2 所示，需要充分考虑各模块功能以及存储器访问和数据传输时对总线使用的协调。接收模块、队列管理模块、调度模块各自需要占用一个 ME，发送模块需要占用两个 ME，处理线速达 1Gbit/s 的数

据流。包处理的工作量相对较大，它包括 IP 包分类、去二层头、IPsec 策略处理、IPsec 入站处理、IPsec 出站处理以及转发等，而且考虑以后功能扩展，需要分配多个 ME，并且分配在两个簇中，减少同簇中资源利用的冲突。

表 3-2 ME 资源分配

任务	ME
包接收	ME0
包处理	ME3~ME6 ME12~ME15
队列管理	ME8
调度	ME9
发送	ME10~ME11

3.3. IPSec VPN 相关 LFB 建模

每个 LFB 都是数据包具体处理模块的一个抽象[2]。各种不同的抽象 LFB 是 ForCES FE 模型的最重要的组成部分，这些 LFB 在一数据通道上互相连接，进行接收、以及处理、修改、发送数据包等操作，同时伴随着各种 Metadata 信息的交互。如 Classifier，Shaper 和 Meter 等都是 LFB 的示例。模型化的 LFB 允许我们用很少的 LFB 来准确的描述 FE 的功能（例如 IPv4 forwarder），而且能够描述更加复杂的网络功能。

基于 ForCES 架构 IPsec VPN 的 LFB 为 IPsec 协议服务，主要完成 IPsec 协议相关的操作，根据功能的划分，可以将 IPsec VPN 的 LFB 分为 IPsec 策略处理 LFB（IPsec_SP_Process LFB），IPSec 入站处理 LFB（IPsec_Inbound_Process LFB），IPSec 出站 LFB（IPsec_Outbound_Process LFB）等等，这里以 IPsec_Inbound_Process LFB、IPsec_Outbound_Process LFB 为例，对 LFB 的研究和设计进行介绍。

3.3.1. IPsec_Inbound_Process LFB 模型

IPsec 入站处理 LFB 完成对 VPN 网关端点到达的数据包的处理，将数据包根据协商好的安全关联进行解封装。

1）输入描述

- ✧ 正常输入：输入为需要进行 IPsec 入站处理的数据包。
- ✧ 异常输入：IPv4 包

2）输出描述

- ✧ 成功输出：IPsec 入站处理模块成功完成了数据包的入站处理，产生一个 IPv4 数据包
- ✧ 丢弃输出：数据包不能通过 IPsec 入站处理模块，例如 SA 过期，SA 状态不成熟等。

3）metadata

IPsec入站处理模块需要接收IPv4分类处理模块解析到的三元组<目的IP地址，SPI，IPsec 协议>。

4）属性描述

在 ForCES FE 模型中，LFB 属性主要指每个 LFB 中对 CE 可见的参数，可以包括：标志位、单参数、复合参数以及 CE 可以通过 ForCES 协议进行读/写的表等。LFB 属性表示 FE 的能力。可配置的属性使 CE 可以灵活地指定一个 LFB 的行为。CE 通过询问一个 LFB 的属性设置，也可以确定该 LFB 的状态。

IPSec_Inbound_Process LFB 的属性包括 IPsec 安全关联数据库 IPsec SAD（在下一节中进行具体的介绍），一些统计属性以及 IPSec SA Hash（DestIP、SPI、IPSec_Protocol 三元组的 Hash 值）。IPSec SA Hash 属性都是只读的，不能够进行配置。

综上所述：可以用

图 3-2 来表示该 LFB。

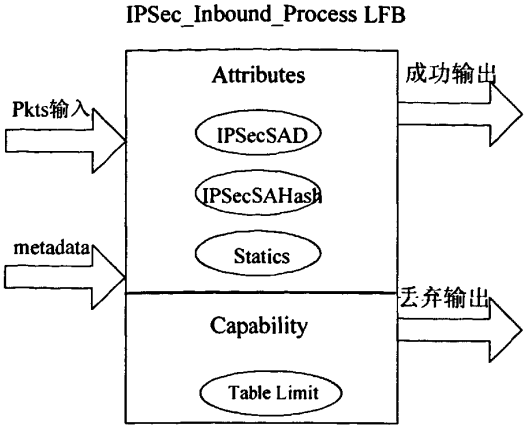


图 3-2 IPSec_Inbound_Process LFB 的定义

3.3.2. IPsec_Outbound_Process LFB 模型

IPsec 出站处理 LFB 完成受 IPsec 保护的网络数据包的处理，并将它们根据协商的安全关联进行封装

1）输入描述

- ✧ 正常输入：输入为需要进行 IPsec 出站处理的数据包。
- ✧ 异常输入：输入为不需要进行 IPsec 保护的数据包。

2）输出描述

- ✧ 成功输出：IPsec 出站处理模块成功完成了数据包的出站处理，产生一个新的 IPv4 数据包。
- ✧ 丢弃输出：数据包不能通过 IPsec 出站处理模块，例如 SA 未协商，SA 过期，SA 状态不成熟等。

3）Metadata

IPsec 出站处理模块需要接收 IPsec 策略处理模块查找到的 SA_Index。

4）属性描述

IPSec_Outbound_Process LFB 的属性包括如上节所述的 IPsecSAD 以及一些统计属性。

综上所述：可以用图 3-3 来表示该 LFB。

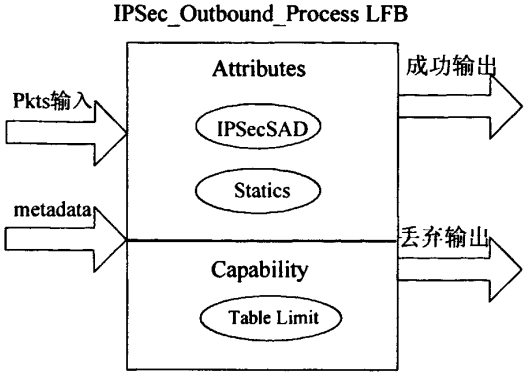


图 3-3 IPSec_Outbound_Process LFB 的定义

3.4. 基于散列技术的安全关联数据库管理机制的研究

对于 IPsec 数据流处理而言，有两个必要的数据库：安全策略数据库（SPDB）和安全关联数据库（SADB）。SADB 中的每一个元组是一个安全关联 SA，它是构成 IPsec 的基础，是两个通信实体经协商建立起来的一种协定。本文主要对安全关联数据库作了重点研究，设计了与 IPSec VPN 转发微模块相关的安全关联数据结构，并研究了基于散列技术的安全关联数据库的管理机制。

3.4.1. 安全关联数据结构设计

SA 是两个应用 IPsec 实体（主机、路由器）间的一个单向逻辑连接，决定保护什么、如何保护以及谁来保护通信数据。它规定了用来保护数据包安全的 IPsec 协议、传输方式、密钥以及密钥的有效存在时间等等。

基于 3.2 节的设计思想，SRAM 的读/写是以 4 字节对齐，即以一个长字（Long Word，LW）为基本单位，所以，为保证 SRAM 的数据存取效率，在安全关联数据结构的设计过程中必须要考虑字节对齐，尽量使需要用到的安全关联数据存放到同一个长字内。本文考虑将占用空间不够一个长字的数据合并到 SA 的第一个长字中，这样的设计不仅节省了存储空间，而且在读/写的过程中不需要对字节进行偏移，简化了程序运行的代码，提高了系统效率。

除了系统效率因素外，本文设计的安全关联数据结构还考虑到了 SA 的安全性。即 SA existence flag，SA inused flag 两个标志位的设置。根据这两个的值即

可判断该 SA ID 在 SADB 中是否存在，或是否正在使用中。这两个标志位的设置将在 SA 的管理中起到关键作用。如果 SA 不存在或正在使用中将不允许对 SA 的删除操作，返回用户错误代码分别为 SA 不存在或正在使用。因为 SA 的某些数据在微码的处理中将被修改，例如序列号，考虑到序列号的唯一性，所以 SA 不允许其他线程同步修改。微码在使用 SA 时首先判断 SA inused flag 是否为 1，若不是将该标志位设置为 1，若为 1 则表示其他线程正在使用该 SA，本线程将转入睡眠等待其他线程使用完 SA 后唤醒本线程继续使用 SA。这两个标志位的设计一定程度上提高了数据的正确性和系统的安全性。第一个长字中 29：20 位的 10 个 bit 以及第 11bit 预留作将来使用，具有扩展性。

基于上述分析和 IPSec 进入和外出处理的要求以及 NP 上存储器的特点，本文设计了如表 3-3 所示的高效的、安全的、可扩展的安全关联数据结构。

表 3-3 SA 数据结构描述

LW	Bits	Size	Field	Description
0	31	1	SA existence flag	If bit[31] is set, the SA is existed in SA table
	30	1	SA inused flag	If bit[30] is set,the SA is inused
	29:20	10	Reservd	
	19:16	4	Encryption Algorithm	0x0: NULL 0x1: TripleDES-CBC 0x2: AES-CBC-128 0x3: AES-CTR 0x4: DES
	15:12	4	Authentication Algorithm	0x0: NULL 0x1: HMAC-SHA1-96 0x2: AES-XCBC-MAC-96 0x3: HMAC-MD5-96
	11	1	Reserved	
	10:9	2	SA State	0x0:IPSEC_SA_STATE_LARVAL, 0x1:IPSEC_SA_STATE_MATURE,

				0x2:IPSEC_SA_STATE_DYING, 0x3:IPSEC_SA_STATE_DEAD
	8:7	2	SA Lifetime Type	0x0:BYTES 0x1:TIME 0x2:BYTES_TIME
	6	1	Replay Window Over Flow Flag	0x0:Unoverflow 0x1:Overflow
	5:2	4	Transform Protocol	0x0: AH 0x1: ESP 0x2: ESP_AH
	1:0	2	Transform Mode	0x0:TUNNEL 0x1: TRANSPORT
1	31:0	32	SPI	
2	31:0	32	Sequence Number	
3	31:0	32	Local Tunnel Address	
4	31:0	32	Remote Tunnel Address	
5	31:0	32	Replay Window Size	
6	31:0	32	Replay Last sequence Number	
7	31:0	32	Replay Bit Map	
8	31:0	32	Replay maximum difference	
9	31:0	32	PMTU	
10	31:0	32	SA Lifetime Seconds Counter	
11	31:0	32	SA Lifetime Kilobytes Counter	
12	31:0	32	SA Lifetime Seconds Soft Timeout	Rfc3585
13	31:0	32	SA Lifetime Kilobytes Soft Timeout	
14	31:0	32	SA Lifetime Seconds Hard Timeout	

15	31:0	32	SA Lifetime Kilobytes Hard Timeout	
16~23	31:0	32*8	Encryption Key	256bits = 32bytes = 8 Long Word
24~28	31:0	32*5	Authentication Key	160bits = 20bytes = 5 Long Word

3.4.2. SADB 的哈希散列技术管理与查找

SADB 的哈希散列技术管理与查找包括 CC 层的管理和微码层的快速查找，SADB 在数据包的处理过程中，需要微引擎进行快速查找。如何管理 SADB 从而加快 SA 的哈希查找速度是本节需要解决的关键技术。本节重点在于介绍微码层的 SADB 的数据结构管理。

IXP2850 网络处理器提供硬件协处理器实现哈希查询，满足高速处理（线速）的要求，称为哈希单元（HASH unit）。哈希单元将输入数据变换为等长数值的索引，可以用于安全协议和高速表查找。哈希变换的输出结果满足正态统计分布，适合作查找表的索引。用哈希单元产生到表的映射地址，可以节省空间，即使用更小的查找表空间，而且可以减少冲突（collision reduction）。

IXP2850 微引擎指令集提供三种 HASH 指令：hash_48，hash_64，hash_128，分别处理 48 位，64 位，128 位数据。不同于常规的协处理器，哈希单元不能同步操作。ME 通过 HASH 指令发出的 1 到 3 个哈希请求，顺序地传递到哈希单元。处理一个请求时，操作数（被乘数，multiplicand）每次取 16 比特和系统定义的一个可编程的乘数（multiplier）运算。这种处理对于 hash_48 重复 3 次，对于 hash_64 重复 4 次，对于 hash_128 重复 8 次。整个操作数都处理完后，如果还有处理请求，立即进行下一个 HASH 操作。

对表的查询，通常把 HASH 索引的小部分（8，16，24，32 比特），用于构建哈希表。也可以采用全部索引的若干位逐个异或（XOR）的方式，这样就用到了全部的索引结果，进一步减少冲突。索引值的大小是和对应的查找表大小相关的。由于存储器容量有限，以 128 位、64 位值作偏移地址不大现实，现实中选择偏移地址为 16 位值。比如，对于硬件产生的 64 位值，可以按 16 位进行异或操作，最后得出的 16 位值为偏移地址。对性能影响较大的部分有两部分，一个是 HASH 值的计算，一个是对 HASH 表的查询，在 IXP2850 中实现这两个功能，只需要使用它的 HASH 硬件，就可以简单快捷地得到 HASH 运算结果值，

以结果值为偏移地址又可以快速地进行查询，同时 HASH 结果值的大位数也降低了 HASH 结果值的可能冲突率。

IPSec SA HASH 表是根据 SADB 数据库中的 SPI、DestIP、Protocol 三元组生成的 HASH 值，并把相应的 SA Index 值填入 HASH 表中，以方便 HASH 查找算法的实现。为了解决 HASH 算法中的冲突域，我们在 SADB 数据结构中增加了一条表项 SA Index 作为相同 HASH 值时的链接所用，构成的关系如图 3-4 所示。

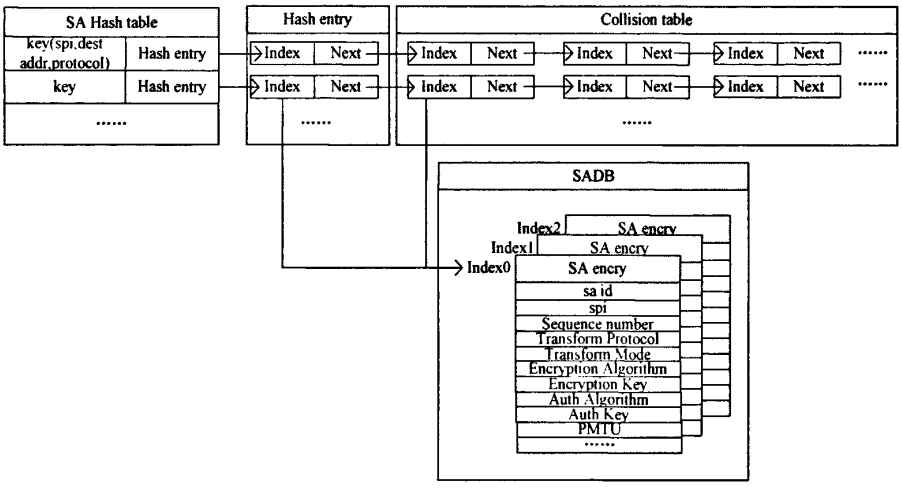


图 3-4 SADB 和哈希表之间的关系

哈希表的查表操作较为频繁，需要快速完成，本文在设计中，将 HASH 表和哈希冲突表均存放在 SRAM 中。先定义一个 HASH 表基地址，然后以计算得出的 HASH 值作为偏移地址，这样可以快速地取得表项中的值。初始化 Hash 表时将 Hash 表内容置为 0xFFFFFFFF，SA 表在添加的过程中将 Hash entry 中存放的值改为 SA 的索引值。若存在冲突域，则将引起哈希查找冲突的 SA 存放在冲突表中，用 Next 指向冲突表的偏移地址。

3.5. 用户空间与内核空间 IPsec SA 通信机制的研究与配置协议的设计

IPsec 安全关联数据库的管理需要交换用户空间和内核空间的数据，顾名思义，交换数据就是说用户空间需要向内核空间读/写数据，而且内核空间也要向

用户空间读/写数据，在数据交换的过程中可能涉及到大量数据的传输和拷贝。如何做到大容量的数据在用户空间和内核空间之间交互将是 IPsec 安全关联数据库管理需要解决的首要问题。

在基于 NP 的 Montavista Linux 操作系统中，我们需要解决 2 个关键技术问题，首先是采用何种方式进行用户空间与内核空间的数据双向传输，其次是如何采用简单的数据结构进行大容量的数据通信。本文对这两个关键技术作了详细的研究，并设计了基于 NP 的用户空间与内核空间的 IPsec SA 配置协议。

3.5.1. 用户空间与内核空间通信方式的研究

一般地，在使用虚拟内存技术的多任务系统上，内核和应用有不同的地址空间，因此，在内核和应用之间以及在应用与应用之间进行数据交换需要专门的机制来实现，众所周知，进程间通信（IPC）机制就是为实现应用与应用之间的数据交换而专门实现的。Linux 传统的进程间通信有很多，如各类管道、消息队列、内存共享、信号量等等。但它们都无法介于内核空间与用户空间使用，原因如表 3-4：

表 3-4 进程间通信方法与比较

通信方法	无法介于内核空间与用户空间的原因
管道（不包括命名管道）	局限于父子进程间的通信。
消息队列	在硬、软中断中无法无阻塞地接收数据。
信号量	无法介于内核空间和用户空间使用。
内存共享	需要信号量辅助，而信号量又无法使用。
套接字	在硬、软中断中无法无阻塞地接收数据。

Linux 用户空间与内核空间的通信方式通常由三种：netlink、/proc 文件系统，字符设备驱动。本文将对这三种通信机制进行比较：

netlink[41]是一种在用户空间与内核空间之间进行双向数据传输的非常好的方式，用户空间应用使用标准的 socket API 就可以使用 netlink 提供的强大功

能，内核空间需要使用专门的内核 API 来使用 netlink。使用 netlink 的内核部分可以采用模块的方式实现，但是它需要在 include/linux/netlink.h 中增加一个新类型的 netlink 协议定义，修改了系统内核，所以在编译的过程中只能静态编译，不能实现模块的动态加载。

在 Linux 环境下开发程序，当需要交换用户空间与内核空间的数据，以及对用户空间和内核空间的数据进行通信等处理时，一般情况下，比较常用的两种方法为创建/ proc 文件与注册字符设备驱动文件。

将两种方法进行对比之后发现，它们具有许多的相同点，也有很大的差异[43]。比如：

1) 它们的实现框架相同，都是通过加载动态模块来实现；

2) 它们的核心数据结构都一样，都是 file_operations。其原因是它们都属于文件系统，都由 Linux 文件系统统一管理；

3) 它们的核心实现函数都一样，都是通过 copy_from_user( )和 copy_to_user( ) 来实现用户空间与内核空间的通信。

但是它们还有如下不同点：

1) /proc 文件的创建不需要像在字符设备文件中的 open/release 来初始化和释放操作；

2) 字符设备文件的建立过程要比/proc 文件复杂一些，因为在建立设备文件后，还需要通过 mknod 来创建文件，但/proc 文件的实现难度、理解难度要复杂一些；

3) 字符设备文件可以建立在任何目录下，不一定在/dev 目录下，而/proc 文件则必须在/proc 目录下；

4) /proc 文件一般用来读取内核的基本信息，字符设备文件则常常用来实现读写的双重任务。

通过以上分析，得出如下结论：字符设备文件是一种经典的、人们常用的实现用户空间与内核空间通信的方法；如果仅仅从内核空间读取数据到用户空间，/proc 文件的优势则明显大于字符设备文件。

由图 3-1 可知，基于 ForCES 架构 VPN 的体系结构是一种用于开发模块化的、可重复利用的网络数据包处理软件的架构，各个模块完成自己独立的功能，利用

各个模块的功能组合可完成网络数据包的处理。LFB 要像 Linux 内核模块那样可以实现动态加载和卸载，如果我们将对 IPsec LFB 的的配置操作当作一种设备模块来使用，用户便可以通过 insmod 和 rmmod 来动态装载和卸载 IPsec 设备模块。基于这种模块化的开发思想，动态地加载模块可以保持操作系统内核的紧凑性。所以本文采用设备文件进行用户空间与内核空间的通行，完成 IPsec 配置功能。IPsec 配置系统软件结构如

图 3-5 所示：

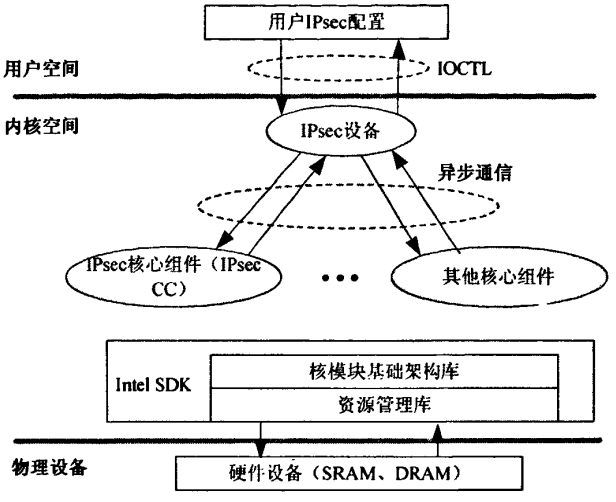


图 3-5 IPsec 配置系统软件结构图

3.5.2. Linux 内核地址空间和用户地址空间之间数据传输

Linux 内核地址空间和用户地址空间之间传输数据，不能用通常的办法利用指针或 memcpy 来完成这样的操作。出于许多原因，不能在内核空间中直接使用用户空间地址。Linux 内核提供 copy_from_user()/copy_to_user() 函数来实现内核空间与用户空间数据的拷贝，copy_from_user()函数的作用是将用户空间数据拷贝到内核，copy_to_user()函数的作用是将内核空间的数据拷贝到用户空间[44]。一般将这两个特殊拷贝函数用在类似于系统调用一类的函数中，此类函数在使用中往往"穿梭"于内核空间与用户空间。此类方法的工作原理路如图 3-6：

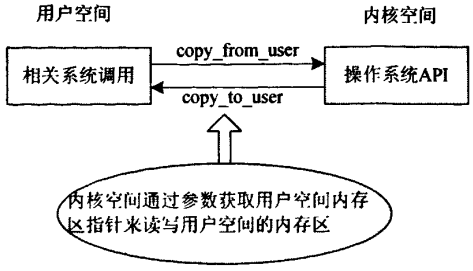


图 3-6 用户空间与内核空间数据交换工作原理

Linux 系统支持两类硬件设备：字符设备、块设备。块设备仅支持面向块的 I/O 操作，所有 I/O 操作都通过在内核地址空间中的 I/O 缓冲区进行，它可以支持几乎任意长度和任意位置上的 I/O 请求，即提供随机存取的功能。字符设备支持面向字符的 I/O 操作，不经过系统的快速缓存，并且只支持顺序存取的功能，一般不能进行任意长度的 I/O 请求。

IPsec 安全关联数据库的管理需要交换用户空间和内核空间的数据，本文采用建立字符设备文件，通过对设备文件的读写，从而实现用户空间与内核空间的通信。Linux 将设备看成特殊文件，设备文件不使用文件系统中的任何数据空间，它仅仅作为驱动的访问入口点，用户利用该入口点就可以像操作普通文件一样来操作设备。

设备驱动程序需要注册一组文件操作来访问设备，这些文件操作定义了设备提供的特定功能，Linux 系统使用 file_operations 结构来向系统说明这些操作入口点，用来完成设备的打开、关闭和命令控制等功能。

```
static struct file_operations ipsec_config_fops = {
    ioctl: ipsec_config_ioctl,
    open: ipsec_config_open,
    release: ipsec_config_close,
};
```

ioctl 是设备驱动程序中对设备的 I/O 通道进行管理的函数。它的调用函数如下：

```
int ioctl(int fd, ind cmd,char *cmdArgStr);
```

其中 fd 是用户程序打开设备时使用 open 函数返回的文件标示符，cmd 是用

户程序对设备的控制命令，cmdArgStr 是该设备控制命令操作的参数。

1）基于 NP 的 IPsec 安全关联数据库的操作包括 SA 的添加、删除、查询、打印以及清空数据库。ioctl 的 cmd 包括：

```
/* defines for ioctl commands */
#define ADD_IPSEC_SA        0x0
#define DEL_IPSEC_SA        0x1
#define GET_IPSEC_SA        0x2
#define DUMP_IPSEC_SAD      0x3
#define PURGE_IPSEC_SAD     0x4
```

2）综合 IPsec SA 的各种操作，本文设计了用户空间与内核空间通信时传递的参数格式，并将其定义为 IPsec 安全关联数据库管理的用户空间与内核空间的通信协议，如图 3-7 所示：

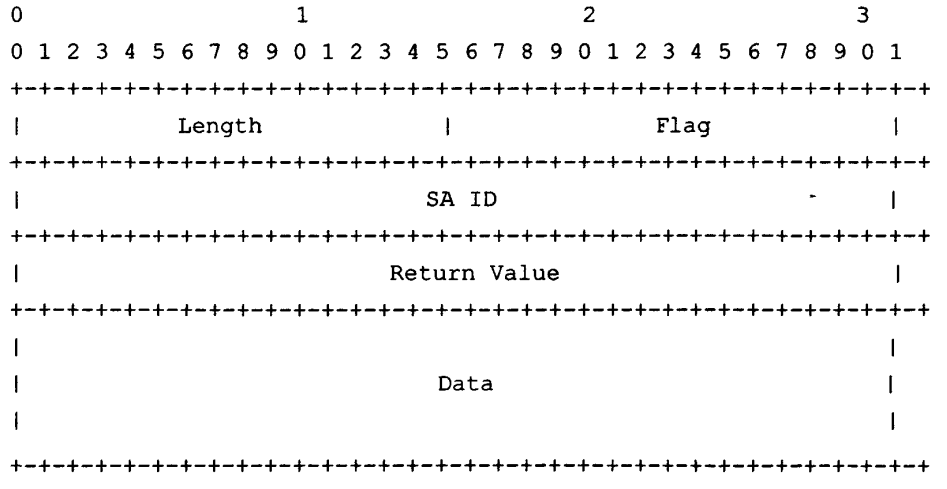


图 3-7 IPsec SA 配置协议格式

每个字段的具体含义如下：

1) Length（2 bytes）：Data 的长度

IPsec SA 的配置协议由协议头和协议体组成，如图 3-7 中，协议头包括除 Data 部分外的其他所有字段。Length 记录的长度为整个协议头部和协议体的长度。

2) Flag（2 bytes）：标识符

标识符用于标志操作结果成功与否，Flag 值的定义为：

'IPSEC_FAILURE'(0b00)——表示需要的操作未成功。默认值为 0，当内核进行相关操作后，返回时将该域根据操作结果作相关修改；

'IPSEC_SUCCESS'(0b01)——表示操作成功。若内核操作成功，将该域填写为 1。

Flag 标识符还可以有其他用途。对于 IPsec SA 的添加、更新、删除和查询操作，Flag 表示的是操作的结果是否成功，而对于打印操作来说，它表示的是 IPsec SADB 中的数据是否完全打印结束。若未全部打印结束，用户空间将继续调用函数向内核申请打印数据库信息。所以，在 dump 操作时，Flag 值的定义为：

'IPSEC_DUMP_UNOVER'(0b00)——IPsec SADB 未打印结束。默认值为 0，当内核将数据库的内容全部取出之后，将其修改为 1；

'IPSEC_DUMP_OVER'(0b01)——IPsec SADB 打印结束。

Flag 字段目前只使用了 1 位，基于协议格式的需要（本文设计的配置协议为 4 字节对齐），本文将前 31 位用于预留使用，便于扩展。Flag 字段如图 3-8：

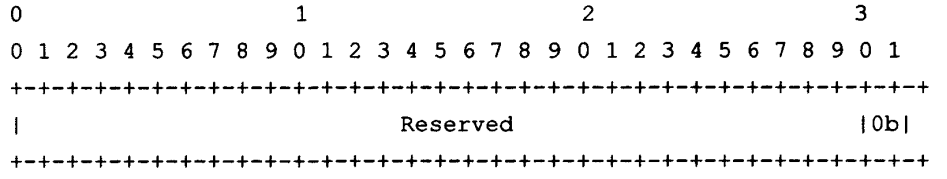


图3-8 Flag

3) SA ID（4 bytes）：被操作的 SA 索引号，该字段为 32 位；

4) Return Value（4 bytes）：由数据库操作返回的值

当对 SADB 进行添加、更新、删除操作时，Return Value 保存内核操作返回的代码，包括操作成功和错误代码，具体的返回值将会在本文的对此关键技术的具体实现中一一说明。而对于打印操作，该字段将被拆成两个部分，如图 3-9 所示，SADB Dumped Num（16bit）表示 SADB 中当前已经取出的 SA 条目，而 SADB Total Num（16bit）则表示 SADB 中 SA 的总条目数。若返回的 SADB Total Num 为零，则表示用户 SADB 中没有 SA 数据。

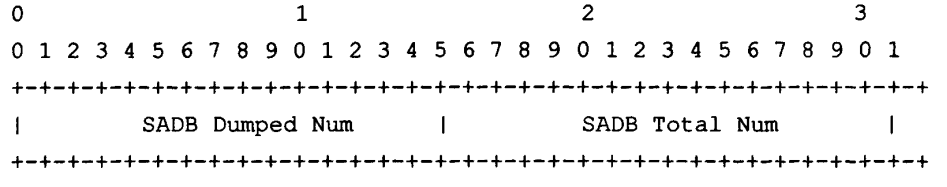


图 3-9 Return Value

5) Data（不定长）：用户空间的输入数据或者内核空间的返回数据，此字段可以为空。

IPsec SA 的配置协议格式由通用的 header 和 body 构成，body 部分的数据是可变长部分，某种情况下也可以为零字节。综上所述，本文设计的 IPsec SA 配置协议具有以下特性：

- 规范性：IPsec SA 在用户空间与内核空间通信的协议格式通用于 IPsec SA 管理的所有操作；
- 灵活性：Return Value 字段可以根据不同的操作用于返回不同的代码；
- 扩展性：Flag 字段和 Length 字段以及可变长的 Data 字段的结合使用可以用于扩展操作的内容；

3.6. 小结

本章是对第三层隧道 VPN 各种关键实现技术进行研究，其中对 IPSec 出站处理和入站处理模块、安全关联数据库的管理模块进行了较为详细的研究和设计，对它们的整体架构、处理流程、相关 LFB 的设计以及拓扑结构的研究、数据结构的设计等等都做出了相应的分析，有利于第四章对各个模块的具体实现。此外，还对 NP 的资源作了高效的分配，提高了系统的处理性能。

本章还分析了 Linux 内核模块的运行环境与传统进程间通信的方式，并分析比较了 Linux 内核地址空间和用户地址空间通信的三种常见方法，总结出采用字符设备驱动的方式完成基于 NP 的 IPsec SADB 用户空间与内核空间的通信。并设计了 IPsec SADB 在用户空间与内核空间通信时采用的配置协议，解决了不能将数据库里的大量数据打印显示到用户界面的难题。

4. 基于 NP 的 ForCES 架构 VPN 的具体实现

本章将在前面几章讨论的基础上，具体介绍基于 NP 的 ForCES 架构 VPN 的具体设计实现。实现内容包括 SADB 哈希管理的实现、IPsec VPN 入站与出站处理的具体实现和用户空间与内核空间 IPsec SA 通信的具体实现。

4.1. SADB 哈希管理的实现

本节具体介绍了采用哈希算法查找和管理基于 NP 的 SADB 包括 IPsec 微模块（CC 层）和 IPsec 微模块（微码层）的实现。

4.1.1. IPsec 核模块的哈希算法管理实现

SA 被添加至 SRAM 的 SADB 后，IPsec 核模块需要对 SADB 采用哈希算法管理，以加快微码层上的查找速度。

首先，IPsec 核模块将 SA 的三元组<SPI，目的 IP 地址，IPsec 协议>作为 hash 算法的关键字添加到 hash 函数_ix_cc_ipsec_hash_sa_entry(*arg_saEntry)的入口参数。arg_saEntry 的数据结构如下所示：

```
typedef struct ix_s_cc_ipsec_sa_entry
{
    ix_uint32 spi;
    ix_uint32 DestIPAddr;
    ix_uint32 IPsecProto;
}ix_cc_ipsec_sa_entry;
```

_ix_cc_ipsec_hash_sa_entry（）函数采用 128 位的哈希算法返回 hash 值，并将此值作为 hash 偏移量累加到 hash 表的基地址上。同时在此偏移量处存放 SADB 的索引值（数据结构如下），若 index 为初始化值 0xFFFFFFFF，则直接将 SADB 的索引值赋给 index，否则则存在冲突，应该把索引值赋给 next 存放到哈希冲突链上，待微码层查找时匹配哈希冲突表以解决哈希算法产生的冲突问题。

```
typedef struct ix_s_cc_ipsec_sa_hash_entry
```

```
{
    ix_uint32    index; /* index into SA Table */
    ix_uint32    next;  /* index into Collision Hash Table */
} __attribute__ ((packed)) ix_cc_ipsec_sa_hash_entry;
```

4.1.2. IPsec 微模块的哈希算法查找实现

基于 NP 的 IPsec 微模块的哈希算法查找主要用于 IPsec 入站处理，具体流程如下：

1) 当数据包入站后，IPsec 入站处理微模块从数据包头中提取<SPI，目的 IP 地址，IPsec 协议>三元组，并将此三元组作为关键字传递给哈希函数 _get_sa_hash_value(hash_value, in_spi, in_ipaddr， in_next_proto)，其中 hash_value 保存 hash 算法产生的哈希值；
2) 将此哈希值作为偏移量查找哈希表，提取 SADB 的索引值；
3) 根据 SADB 的索引值和基地址查找 SA，取出三元组信息与 SA 的内容做比较，若相等则取出 SA，查找完毕；
4) 若存在哈希冲突则将三元组信息与哈希冲突链上的 SA 的内容逐一比较；
5) 若查找结果为空，则将数据包丢弃处理，并记录日志，否则将进行数据包的解封装处理；

4.2. IPsec VPN 入站与出站的具体实现

4.2.1. IPsec 数据包处理功能模块划分

根据基于 ForCES 架构 IPSec VPN 的具体需求以及网络处理器本身的软件架构，本文将 IPsec 数据包的处理模块作了如图 4-1 所示的划分，以完成 IPsec VPN 的入站与出站处理。

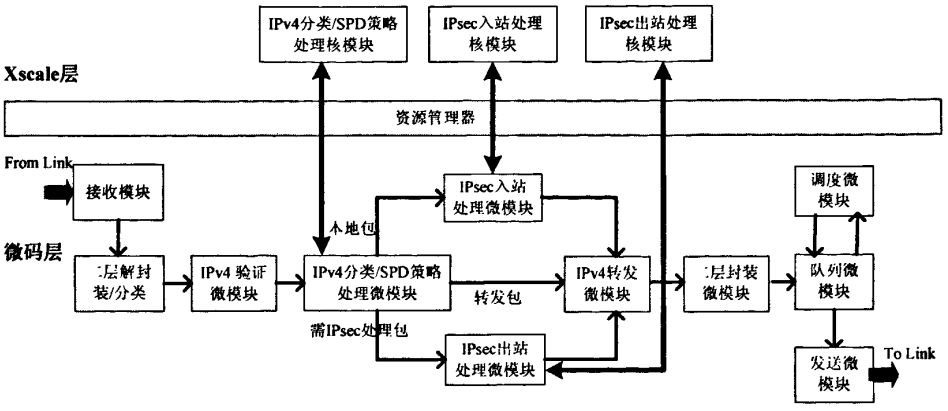


图 4-1 IPsec 数据包处理模块结构图

4.2.1.1. IPv4 分类/SPD 策略处理子模块划分

本模块实现对图 4-1 中 IPv4 分类/SPD 策略处理模块的详细划分，完成 IPv4 数据包的分类和 IPsec 策略处理。如图 4-2 所示，该模块包括 IPv4 分类、SPD 策略查找以及 SA 处理子模块。

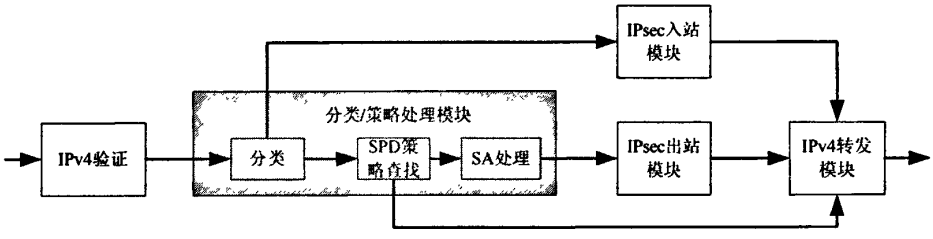


图 4-2 IPv4 分类/SPD 策略处理模块图

1. IPv4 分类子模块

IPv4 分类子模块执行数据包的分类操作，收到来自上一级数据包处理模块的 signal（信号）之后，将 IP Header 读入微引擎的本地寄存器（LM）中，根据存储在 SRAM 中的本地地址表识别出数据包的类型，分为本地包和非本地包；

2. SPD 策略查找子模块

SPD 策略查找子模块执行数据包的安全策略查找操作，其输入是包含协议类型、源和目的端口号、源和目的地址的选择符，根据存储在 SRAM 中的安全策略数据库决定出数据包采取 3 中可能的处理方式：

A. 丢弃：丢弃数据包，并记录出错信息；

B. 绕过 IPsec：给数据包添加 IP 头，然后交给 IPv4 转发模块处理；

C. 应用 IPsec：查找出应用 IPsec SA 的索引号，交给 SA 处理子模块验证 SA 的有效性后进行 IPsec 处理。

3. SA 处理子模块

SA 处理子模块验证 SA 的有效性，SA 的维护视 SA 的建立方式而定。前面提到过，SA 的建立可以通过手工键控的方式，也可以通过调用 IKE 动态协商一个新的 SA。如果采用后者以自动方式协商 SA，则需要根据 SA 生存期的状态和序号计数器的溢出标志来决定 SA 的有效性。生存期分为软生存期和硬生存期，软生存期状态决定发送方是否可用 SA 发送数据包，硬生存期状态决定接收方是否可用 SA 来处理收到的数据包。当一个 SA 的软生存期期满时，发送方不能继续用其来发送数据包，此时，可以启动或触发 IKE 再协商一个。使用软、硬生存期机制可以保证通信的持续性[14]。

首先根据 SPD 策略查找模块输出的 SA 索引号查询 SADB，确定是否存在有效的 SA。在这里分 3 中不同的情况来进行说明：

A. 存在有效的 SA，将数据包交给 IPsec 出站模块处理；

B. 尚未建立 SA，SA 处理子模块将启动或触发 IKE 协商，协商成功后交给 IPsec 出站模块处理；不成功则将数据包丢弃，并记录出错信息；

C. 存在 SA 但无效，SA 处理子模块将此信息向 IKE 通告，请求协商新的 SA，协商成功后交给 IPsec 出站模块处理；不成功则将数据包丢弃，并记录出错信息。

如采用手工方式建立 SA，则不存在生存期，仅根据序号计数器的溢出标志来决定 SA 的有效性。

4. 模块输入输出

输入为数据包的源 IP 地址、目的 IP 地址、源端口、目的端口、上层协议。

输出有三个分支：IPsec 入站处理，IPsec 出站处理，IPv4 转发处理。

1) IPsec 入站处理：IPv4 分类模块将数据包的目的 IP 地址与本地地址表进行规则匹配，若为本地包，则将数据包交给 IPsec 入站模块处理；若为非本地包则交给 SPD 策略查找模块；

2) IPsec 出站处理：SPD 策略查找模块根据选择符源 IP 地址、目的 IP 地址、

源端口、目的端口、上层协议（名字和一个数据敏感级本应用不考虑）查找安全策略数据库，若查找结果为 IPSEC_PROTECT，则得到 SA 的索引号 SA_Index，进行 SA 处理后进行 IPsec 出站处理；

3) IPv4 转发处理：若查找安全策略数据库结果为 IPSEC_BYPASS，则跳转至 IPv4 转发处理。

4.2.1.2. IPsec 出站处理子模块划分

本模块实现对图 4-1 中 IPsec 出站处理模块的详细划分，完成数据包的出站处理，如图 4-3 所示，该模块包括加密、认证和 IPsec 协议封装。

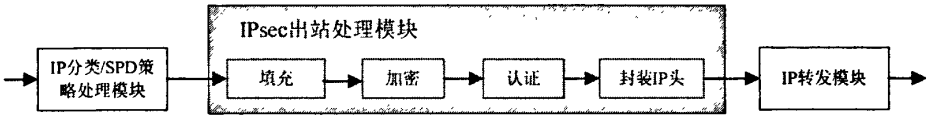


图 4-3 IPsec 出站处理模块图

1. 填充（供加密使用）

几种因素要求或者激活填充字段的使用。

1) 如果采用的加密算法要求明文是某个数量字节的倍数，例如块密码（block cipher）的块大小，使用填充字段填充明文（包含有效载荷数据、填充长度和下一个头字段，以及填充）以达到算法要求的长度。

2) 不管加密算法要求如何，也可以要求填充字段来确保结果密文以 4 字节边界终止。特别是，填充长度字段和下一个头字段必须在 4 字节字内右对齐，如上图所示的 ESP 分组格式，从而确保验证数据字段（如果存在）以 4 字节边界对齐。

3) 除了算法要求或者上面提及的对齐原因之外，填充字段可以用于隐藏有效载荷实际长度，支持（部分）信息流机密性。但是，包含这种额外的填充字段占据一定的带宽，因而小心使用。

发送方可以增加 0 至 255 个字节的填充。ESP 分组的填充字段是可选的，但是所有实现必须支持填充字段的产生和消耗。

A. 为了确保加密位是算法块大小（上面第一个加重号）的倍数，填充计算应用于除 IV 之外的有效载荷数据、填充长度和下一个头字段。

B. 为了确保验证数据以 4 字节边界对齐（上面第二个加重号），填充计算应

用于包含 IV 的有效载荷数据、填充长度和下一个头字段。

如果需要填充字节，但是加密算法没有指定填充内容，则必须采用下列默认处理。填充字节使用一系列（无符号、1 字节）整数值初始化。附加在明文之后的第一个填充字节为 1，后面的填充字节按单调递增：1,2,3,…。当采用这种填充方案时，接收方应该检查填充字段。（选择这种方案是由于它相对简单，硬件实现容易。在没有其他完整性措施实施情况下，如果接收方检查解密的填充值，这种方案粉碎了某种形式的"剪切和粘贴"攻击，提供有限的保护。）

2. 加密子模块

加密是基于 ESP 协议的处理，完成载荷的加密。查找到 SA 后，需要对报文进行加密（加密可以采用很多算法，如 DES、3DES 等）。发送方把上层协议信息（传输模式）或整个 IP 报文（隧道模式）封装到 ESP 负载中，然后添加必要的填充信息，最后用 SA 中指定的密钥、加密算法、算法模式及保密同步数据对结果数据包（包括负载、填充信息、填充长度、下一协议头类型）进行加密。如果还选择了 ESP 协议的认证服务，必须要在认证之前进行加密，并且加密数据不能包括认证数据域。在 ESP 中，完整性验证（ICV）计算的对象是除了认证数据以外的其他部分。

3. 认证子模块

认证是基于 AH 协议的处理，用来增强 IP 数据包的安全性。提供无连接的完整性、数据源认证和抗重放保护等服务。查找到 SA 后，需要对报文进行认证（认证可以采取的算法如 HMAC-MD5，HMAC-SHA1 等）。

SA 配置 AH 协议的工作方式为传输模式时只对上层协议数据（传输层数据）和 IP 头中的固定字段提供认证保护，把 AH 插在 IP 报头的后面，主要适合于主机实现。AH 协议工作在隧道模式时把需要保护的 IP 包封装在新的 IP 包中，作为新报文的载荷，然后把 AH 插在新的 IP 报头的后面。隧道模式对整个 IP 数据报提供认证保护。

4. 封装 IP 头子模块

对于隧道模式，根据 IPsec 协议格式，需要在 ESP/AH 外部封装新的 IP 头，新的 IP 头部封装 SA 中协商的隧道源地址和目的地址。对于传输模式，由于在原 IP 头和协议层之间加入了 ESP 或 AH 协议，使得缓冲区中的字节向前挪动了

ESP 或 AH 头部的长度，所以必须在数据包前重新填写 IP 头部数据。

由此看出，IPsec 出站子模块的划分具有灵活性和可裁剪性。查找到 SA 后，根据 SA 的规定可以对数据包进行加密处理，为满足用户的特殊要求，可以配置 SA 绕过 IPsec ESP 协议的加密子模块，即采用加密算法和密钥为 NULL。同理，可以配置 SA 的认证算法和密钥为 NULL，绕过 AH 认证服务。

5. 模块输入输出

输入为经过验证的有效的安全关联数据库索引 SA_Index。该模块无特别输出。

4.2.1.3. IPsec 入站处理子模块划分

本模块实现对图 4-1 中 IPsec 入站处理模块的详细划分，完成 IPsec 入站的处理。如图 4-4 所示，该模块包括对数据包的 IPsec 解封装、SA 处理、抗重放服务、解密和认证子模块。

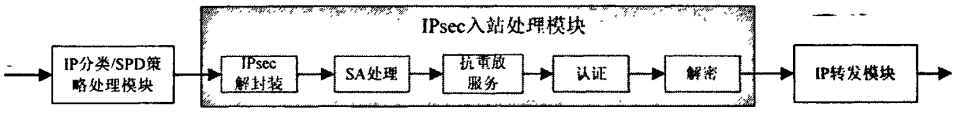


图 4-4 IPsec 入站处理模块图

1. IPsec 解封装子模块

IPsec 解封装子模块完成 IP 头的解封装操作，解析 IP 头中的下一协议类型，分为 ESP 协议或 AH 协议，否则做出错处理，将丢弃数据包，并记录日志。IPsec 解封装子模块提取 IP 头中的目的地址、IPsec 协议类型（ESP/AH）以及 IPsec 协议中的 SPI，将此三元组作为 Hash 关键字查找 SADB，得到 SA 的索引值传递给抗重放服务子模块进行重放包检查。

2. SA 处理子模块

SA 处理子模块主要完成的是 SA 的有效性验证，本子模块的功能和 IPv4 分类/SPD 策略处理模块的 SA 处理子模块的功能一致。

3. 抗重放服务子模块

所有 ESP 实现必须支持抗重播服务，虽然可以由接收方根据每个 SA 激活或者禁止它的使用。如果 SA 的认证服务没有激活，这项服务不允许激活，因为序列号字段没有进行完整性保护。

如果接收方不激活 SA 的抗重播服务，将不对序列号进行入站检查。但是从发送方观点来看，默认的是假定接收方激活抗重播服务。为了避免接收方做不必要的序列号监视和 SA 建立，如果使用 SA 建立协议，例如 IKE，在 SA 建立期间，如果接收方不提供抗重播保护，则接收方应该通告发送方。

如果接收方已经为这个 SA 激活了抗重播服务，SA 接收分组计数器在 SA 建立时，必须初始化为 0。对于每个接收的分组，接收方必须确认分组包含序列号，并且序列号在这个 SA 生命期中不重复任何已接收的其它分组的序列号。这应该是分组与某个 SA 匹配之后，对该分组进行的第一个 ESP 检验，加快重复分组拒绝。

通过采用滑动接收窗口拒绝分组重复，必须支持 32 位的最小窗口大小（接收方不会通告发送方窗口大小）。窗口“右”边界代表该 SA 接收的最高的有效序列号值。对于序列号小于窗口“左”边界的分组被拒绝。落入窗口内的分组依靠窗口内已接收分组列表进行检验。以使用位掩码为基础，实现这种检验的有效手段在安全架构文档中描述。

如果接收的分组落入窗口内且是新的，或者如果分组在窗口的右边，那么接收方进行 ICV 确认。如果 ICV 有效性失败，接收方必须把已接收的 IP 数据报作为非法而丢弃；这是可审核事件。该事件审核日志表项应该包括 SPI 值、接收的日期/时间、源地址、目的地址和序列号。

要注意的一个重点是，除非造成窗口向前滑动的那个包通过了真实性检查，否则窗口是不会真的前进的。否则的话，攻击者便可生成伪造的包，为其植入很大的序列号，令窗口错误移至有效序列号的范围之外，造成我们将有效的数据包错误地丢失。

4. 认证子模块

认证子模块完成数据包的完整性验证，防止数据包被篡改。IPsec 入站模块根据 SA 中指定的认证算法、认证密钥，计算出数据包的消息摘要，并将其与数据包中提取的 ICV 验证字段比较，若一致则认证成功，否则丢弃数据包并记录日志。

5. 解密子模块

IPsec 入站模块的解密子模块基于 ESP 协议的处理。接收方需要使用 SA 中

指定的密钥解密算法、解密密钥以及解密同步数据对负载数据、填充、填充长度和下一扩展头协议进行解密，然后用解密算法中指定的方法处理填充数据，最后重建初始 IP 报文。对于传输模式，从 ESP 负载域中提取 IP 抱头和初始上层协议信息，对于隧道模式从 ESP 负载域中提取隧道 IP 头和整个 IP 报文。如果选择了认证服务（ESP），ICV 验证和解密可以串行或并行地进行，如果串行执行，将先进行 ICV 验证，如果并行执行，那么在对解密报文进行进一步处理之前必须完成 ICV 验证[24][25]。

6. 模块输入输出

输入为三元组<目的 IP 地址，SPI，IPsec 协议>，该三元组经过 Hash 运算单元得到安全关联数据库的索引 SA_Index，根据此索引检索出该 IPsec 处理的相关参数。此模块无特别输出。

4.2.2. IPsec VPN 的数据包处理的具体实现

4.2.2.1. 数据包处理流程

根据第三章的模块设计，本文实现了基于 NP 的 IPsec VPN 的数据包处理，具体流程如图 4-5：

其中模块①为 IPv4 分类/SPD 策略处理模块，模块②为绕过 IPsec 处理，直接进行数据包转发模块，模块③为 IPsec 入站处理模块，模块④为 IPsec 出站处理模块。

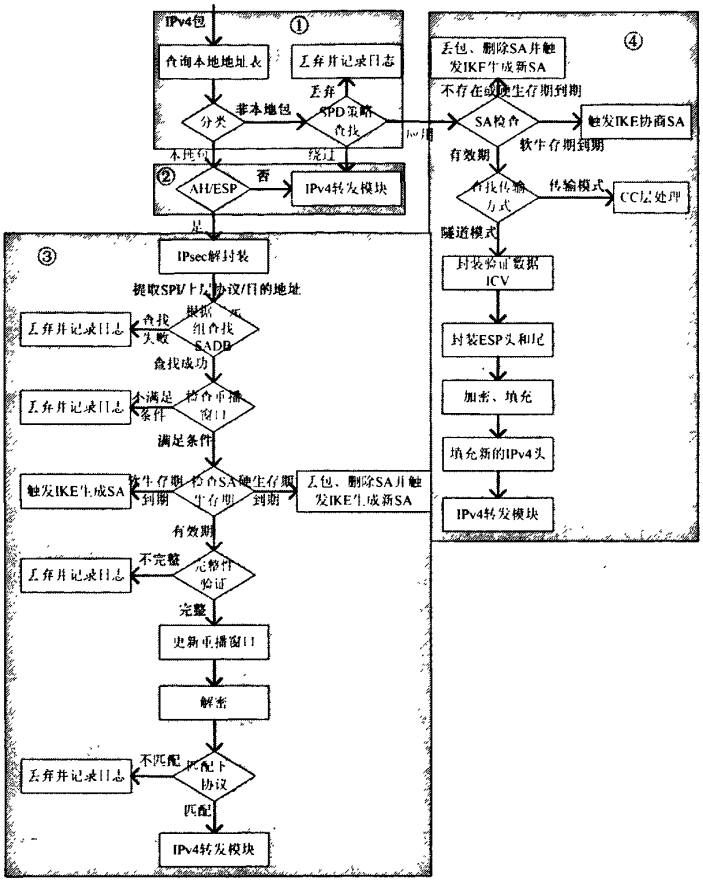


图 4-5 数据包处理流程图

数据包的模块化处理流程为：

1）IPv4 分类/SPD 策略处理模块从 IPv4 验证模块接收到 IPv4 数据包（已重组），首先对数据包进行分类；

2）根据本地地址表对数据包进行分类，若目的地为本地包跳转至模块②，若非本地包则进行 IPsec 策略处理；

3）若本地包为 AH/ESP 类型，跳转至模块③进行 IPsec 入站处理，若不是 AH/ESP 类型，则交给转发模块；

4）若目的地非本地包，则将数据包内容（源地址、目的地址、源端口、目的端口、上层协议）进行 SPD 策略查找，若查找结果为空，丢弃该包；若查找结果为“绕过”，则跳转至转发模块；若结果为“应用”，则继续实施 IPsec 出站处理；

关于 IPsec 入站和 IPsec 出站数据包的实现过程将在以下两节中详细介绍。

本文实现的 IPsec VPN 的入站和出站处理都是基于 ESP 协议的处理。

4.2.2.2. IPsec 出站微模块的实现

基于 NP 的 ForCES 架构 IPsec VPN 出站核模块出 ipsec_outbound_process() 函数来完成，流程图如图 4-6：

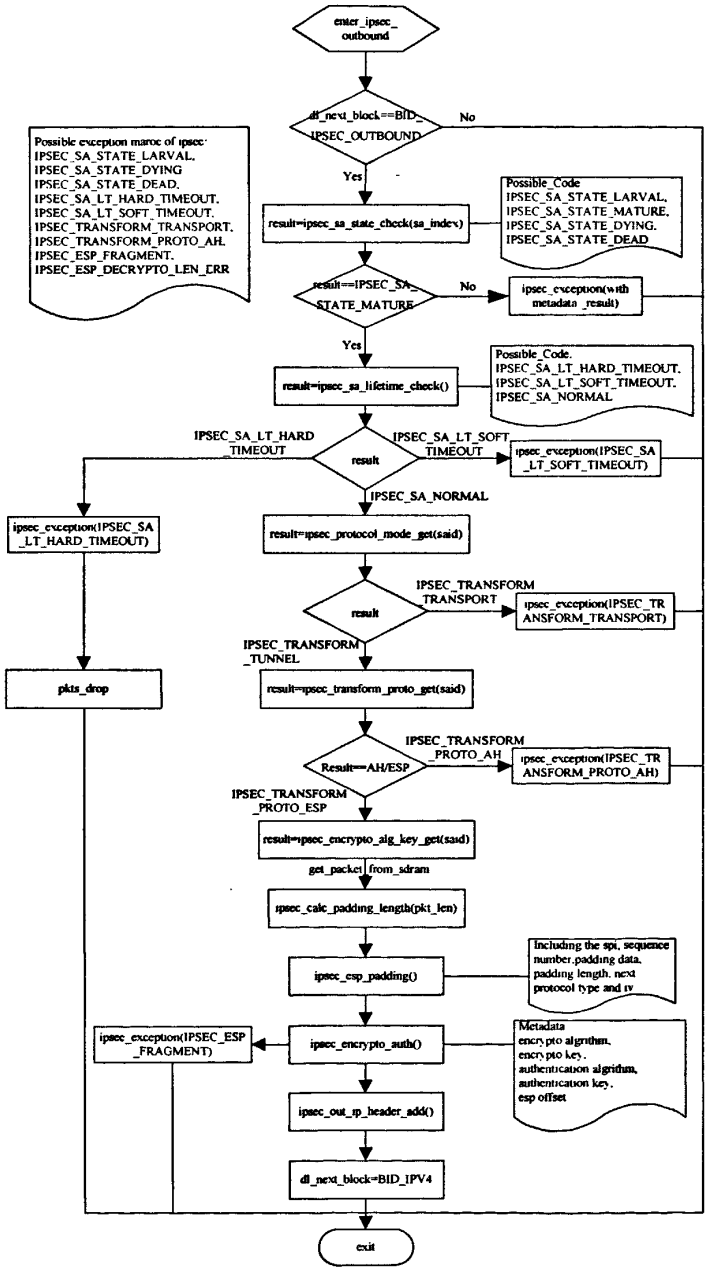


图 4-6 IPsec 出站模块流程图

实现的具体过程为：

1）若指定的 dl_next_block 不是为 IPsec 出站模块，则退出；

2）根据策略处理模块查找的 SA 索引，调用 ipsec_sa_state_check(sa_index)和 ipsec_sa_lifetime_check（）函数检查 SA 的有效性，若 SA 不存在或者 SA 存在，但硬生存期到期，则丢弃该包并通知 IKE 重新协商一个 SA，若 SA 软生存期到期，使用该 SA 处理数据包并通知 IKE 协商一个 SA 替代该 SA；

3）调用函数 ipsec_protocol_mode_get(said)获取 SA 中对该数据包应用的传输方式，若为传输模式则交由 CC 层处理；

4）调用函数 ipsec_transform_proto_get(said)获取 SA 中对该数据包应用的传输协议，若为 AH 则交由 CC 层处理；

5）调用函数 ipsec_encrypto_alg_key_get(said)获取 SA 中对数据包采用的加密算法和密钥，若加密算法为空则对数据包不进行加密处理；

6）调用函数 ipsec_calc_padding_length(pkt_len)和 ipsec_esp_padding（）添加填充数据，确定填充长度；

7）若加密密钥非空，则调用加密模块的函数 ipsec_encrypto_auth()对数据包进行加密处理，本文没有实现对 ESP 协议的认证处理；

8）封装 ESP 头和尾：从 SA 中取出 SPI，递增序列号加载在加密数据的前面，并在加密数据（包括填充数据）后添加填充长度字段和下一协议字段（若为 IPv4，该字段为 4）；

9）填充新的 IP 头：源地址和目的地址取自 SA 中的隧道地址，协议类型填 ESP（50），总长度为新 IP 包头部长度、上一个 IP 包长度、ESP 头部长度、尾部长度的总和，重新计算校验和并送至转发模块

10） 将 dl_next_block 设置为 IPv4 转发模块的 ID 号，进行下一步的转发处理。

4.2.2.3. IPsec 入站微模块的实现

基于 NP 的 ForCES 架构 IPsec VPN 入站核模块由 ipsec_inbound_process()函数来完成，流程图如图 4-7：

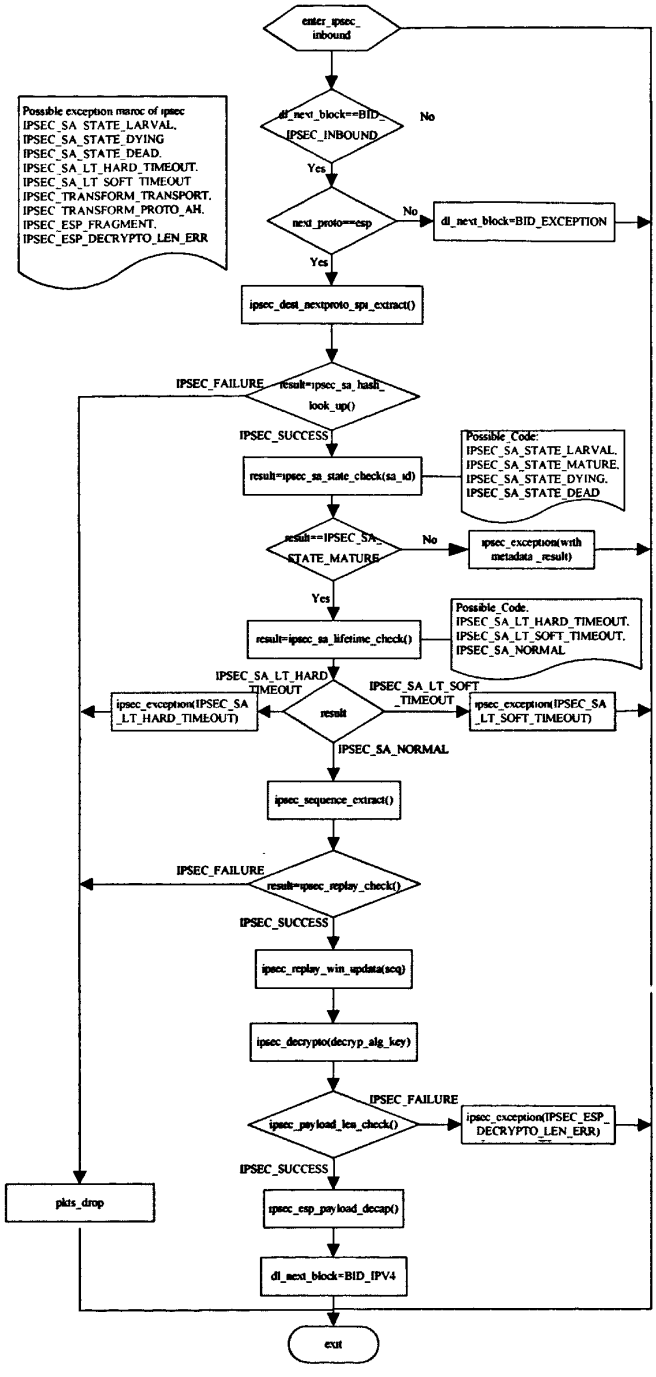


图 4-7 IPsec 入站模块流程图

实现的具体过程为：

1）若指定的 dl_next_block 不是为 IPsec 入站模块，则退出；

2）若数据包不是 ESP 类型，则将 dl_next_block 设置为异常处理模块的 ID 进行异常处理；

3）调用函数 ipsec_dest_nextproto_spi_extract()提取数据包头的目的 IP 地址，IPsec 协议以及 SPI 三元组，调用函数 ipsec_sa_hash_lookup()进行哈希算法查找，得到 SA 的索引值。若 SA 不存在则丢弃数据包并记录日志；

4）调用 ipsec_sa_state_check(sa_index)和 ipsec_sa_lifetime_check()函数检查 SA 的有效性；

5）提取数据包的序列号，进行抗重播检查。检查时将判断到达的 IPSec 包的序列号是否小于该网关所记录的处理过的最大 IPSec 包的序列号。如果小于，则再检查序号的差值是否超过了窗口的大小。如果没有超过，则检查窗口，判断该 IPSec 包是否已经到达过，如果已经到达过，则将该包丢弃；如果没到达，则接收。如果到达的 IPSec 包的序列号大于处理过的最大序列号，检查该序列号与记录的最大序列号差值，若超过 replaywin_maxdiff 则丢弃，否则接收处理；

6）解密：采用 SA 指定的解密算法和密钥将受保护的数据解密，包括有效载荷，填充长度，下一协议类型；

7）调用函数 ipsec_payload_len_check()检查解密后的数据包的填充字节长度，若与 ESP 尾部填充的字节长度不匹配则将数据包交给异常处理模块；

8）ESP 协议解封装，调用 ipsec_esp_payload_decap()函数去掉 ESP 协议的头部和尾部；

9）将 dl_next_block 设置为 IPv4 转发模块的 ID 号，进行下一步的转发处理。

4.3. 用户空间与内核空间 IPsec SA 通信的具体实现

4.3.1. IPsec 字符设备驱动注册

Linux 系统里，设备模块初始化函数 init_module()通过调用 register_chrdev()向系统注册字符型设备驱动程序。本文定义的 register_chrdev()参数为：

register_chrdev(g_IPsec_MajorNumber, DEV_NAME_IPSECCONFIG, &ipsec_config_fops);

其中，g_IPsec_MajorNumber 是为 IPsec 设备驱动程序向系统申请的主设备号，如果为 0 则系统为此驱动程序动态地分配一个主设备号，本文将其定义为

248。DEV_NAME_IPSECCONFIG 是设备名，为了便于修改，本文将设备名用一个宏来表示。&ipsec_config_fops 是对各个调用的入口点的说明，它的结构体为：

```
static struct file_operations ipsec_config_fops = {
    ioctl: ipsec_config_ioctl,
    open: ipsec_config_open,
    release: ipsec_config_close,
};
```

此函数返回 0 表示成功。返回-EINVAL 表示申请的主设备号非法，一般来说是主设备号大于系统所允许的最大设备号。返回-EBUSY 表示所申请的主设备号正在被其它设备驱动程序使用。如果是动态分配主设备号成功，此函数将返回所分配的主设备号。如果 register_chrdev()操作成功，设备名就会出现在/proc/devices 文件里。初始化部分一般还负责给设备驱动程序申请系统资源，包括内存、中断、时 钟、I/O 端口等，这些资源也可以在 open 子程序或别的地方申请。在这些资源不用的时候，应该释放它们，以利于资源的共享。

4.3.2. 系统调用流程

模块被加载后，用户进程调用 open("/dev/IPsecConfig', O_CREAT | O_RDWR)时，IPsec 字符设备即被打开，调用 ioctl 进行用户空间和内核空间的双向数据通信以及调用 close 关闭 IPsec 字符设备。

ioctl 的参数包括打开 IPsec 返回的描述符、IPsec 配置命令以及 IPsec 配置参数，操作系统内核接收到用户空间调用 ioctl 函数后使用 copy_from_user 拷贝用户传递的参数，并调用 IPsec 设备驱动程序将用户的配置参数采用异步方式发送给 IPsec 的核心组件。这两个过程均采用回调函数的机制异步处理 IPsec 配置的返回值。IPsec 核心组件基于 Intel IXA 可移植性框架的软件开发套件（SDK）将用户传递的命令和参数直接对硬件设备进行操作，本文的设计中，将 IPsec SADB 存放在 SRAM 中，即 IXA SDK 可直接对 SRAM 进行读写操作，将用户需要的数据写入 SRAM 或者从 SRAM 读取到缓冲区内。

IPsec 配置的系统调用序列图如图 4-8 所示：

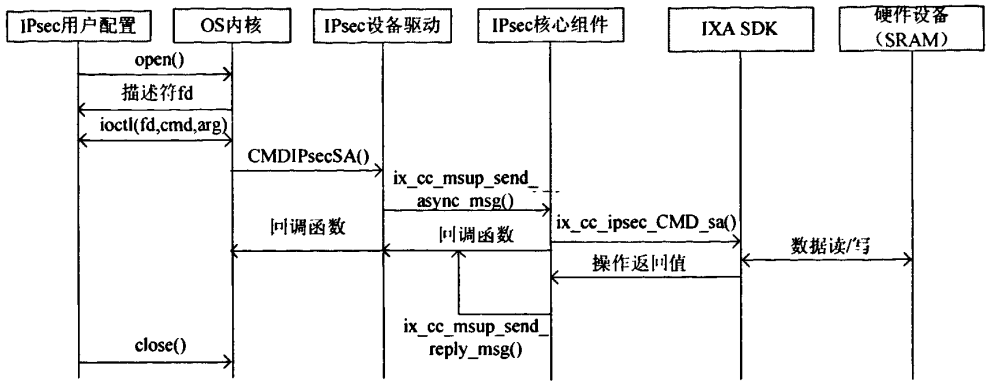


图 4-8 IPsec 配置系统调用序列图

4.3.3. IPsec 配置协议的具体实现

IPsec SA 配置的具体实现包括 IPsec SA 的添加、删除、查询、打印和清空，本文以下内容将逐一对这些操作进行详细介绍，并同时描述配置协议的使用和实现。

1、添加的实现

基于 NP 的 IPsec 安全关联数据表项添加是 IPsec 安全关联数据库管理的操作中用户空间向内核空间传递数据最大的操作，其中 Data 部分包括了 IPsec SA 的全部内容。

IPsec SA 的添加采用 IOCTL 命令：ioctl(fd, cmd,cmdArgStr)，其中：

fd = open("/dev/IPsecConfig", O_CREAT | O_RDWR);

cmd = ADD_IPSEC_SA;

cmdArgStr = (char *)& ix_user_ipsec_sa_add_sa_data ;

ix_user_ipsec_sa_add_sa_data 的结构体类型为：

```
typedef struct ix_s_user_ipsec_sa_add_sa
{
    uint16   length;
    uint16   flag ;
    uint32   said ;
    uint32   ret_code ;
    ix_user_ipsec_sa_info *pSAInfo;
```

} ix_user_ipsec_sa_add_sa;

said 为需要添加的 SA 索引号，指针 pSAInfo 为用户输入的经过正确性验证的数据的结构体指针，flag 和 ret_code 初始化为 0，留作内核空间返回的数据填充。内核空间调用 copy_from_user 函数将用户空间传递的数据拷贝至内核处理后，再使用 copy_to_user 将数据库操作的结果返回给用户空间。IPsec SA 的添加需要返回的该 SA ID 的返回代码，flag 字段有两种结果：IPSEC_SUCCESS 和 IPSEC_FAILURE，当标志位为 IPSEC_FAILURE 时，查询 Return Value 字段的错误代码，IPsec SA 错误代码有：

```
typedef enum e_ipsec_add_sa_error_code
{
    IPSEC_ERROR_INVALID_SAID,
    IPSEC_ERROR_INVALID_SPI,
    IPSEC_ERROR_INVALID_IP_ADDRESS,
    IPSEC_ERROR_INVALID_ENCRYP_ALG,
    IPSEC_ERROR_INVALID_ENCRYP_KEY,
    IPSEC_ERROR_INVALID_KEY_NUM,
    IPSEC_ERROR_INVALID_AUTH_ALG,
    IPSEC_ERROR_INVALID_AUTH_KEY,
    IPSEC_ERROR_INVALID_START,
    IPSEC_ERROR_INVALID_TRANS_MODE,
    IPSEC_ERROR_INVALID_TRANSP_RROTO,
    IPSEC_ERROR_INVALID_SA_STATUS,
    IPSEC_ERROR_INVALID_SA_LIFETIME_TYPE,
    IPSEC_ERROR_INVALID_REPLAY_WIN_SIZE,
    IPSEC_ERROR_ENCRYP_KEY_LENGTH_UNMATCH,
    IPSEC_ERROR_AUTH_KEY_LENGTH_UNMATCH,
    IPSEC_ERROR_PARAMETER_NULL,
    IPSEC_ERROR_PARAMETER_NOT_ENOUGH
}ipsec_add_sa_error_code;
```

用户空间可以根据错误代码分析用户输入的数据的非法性，并将结果呈现给用户。

2、删除的实现

基于 NP 的 IPsec 安全关联数据表项的删除是 IPsec 安全关联数据库管理的操作中用户空间与内核空间数据交互最简单的操作，在用户空间传出与返回的指针中所指向的结构体均只有一个协议头，即协议体 Data 为零字节，协议头部 Length 字段为 0。这是因为对 IPsec SA 的删除只需要通过 SA ID 即可，传入的结构体不需要其他内容，且返回的结构体中只需要返回此次操作的结果正确与否即可，若是操作失败，则将错误代码填充至 Return Value 字段中。IPsec SA 删除的错误代码为：

```
typedef enum e_ipsec_del_sa_error_code
{
    IPSEC_DEL_SUCCESS,
    IPSEC_DEL_SA_NOT_EXIST,
    IPSEC_DEL_SA_IS_INUSED
}ipsec_del_sa_error_code;
```

注意，在 IPsec 安全关联数据库的管理中，若 IPsec SA 表项的标志位 ipsec_sa_in_use 被设置为 1，即表示 SA 正在被 NP 使用，则不能删除此 SA，数据库管理单元返回用户一个错误代码：IPSEC_DEL_SA_IS_INUSED，并将 Flag 字段设为 IPSEC_FAILURE。

3、查询的实现

当用户需要对 IPsec SADB 中的某条 SA 表项进行查看时需要用到 IPsec 安全关联数据表项查询的操作。IPsec SA 的查询只需要通过传输参数 IPsec SAID 即可。在 SA 查询操作的过程中需要返回整条 SA 表项的内容，所以在用户空间使用 IOCTL 命令时，必须将查询后获得的 SA 的结构体指针传递到内核空间，待内核将数据通过 copy_to_user( )函数拷贝至该结构体指针后返回到用户空间打印显示完毕后释放指针。

查询操作也有两种返回值，当 Flag 标志位为 IPSEC_FAILURE 时 Return Value 中附带的返回参数为 IPSEC_SA_NOT_EXIST，用户空间函数获得此参数后将不

会显示 SA 的信息，并给用户打印一条提示信息，让用户检查输入的 SA ID 是否有效或存在。

4、打印的实现

IPsec 安全关联数据的打印操作是内核与用户空间交互数据量最大的操作，如果安全关联数据库非常庞大，譬如说 IPsec SA 数据库中有一千条 SA 表项，而每条 SA 的结构体大小为 100 bytes 左右，但是内核空间与用户空间采用 copy_from_user()和 copy_to_user()函数拷贝的最大值不同的操作系统也只有 512/1024 bytes，远远不能满足用户空间打印显示数据库中所有 SA 的需要，因此，这也成为本文设计 IPsec 安全关联数据库用户空间与内核空间的通信协议的出发点和目标。实验证明，本文的设计方法很好的解决了不能将数据库的庞大数据通过内核空间的库函数交互到用户空间的难题。

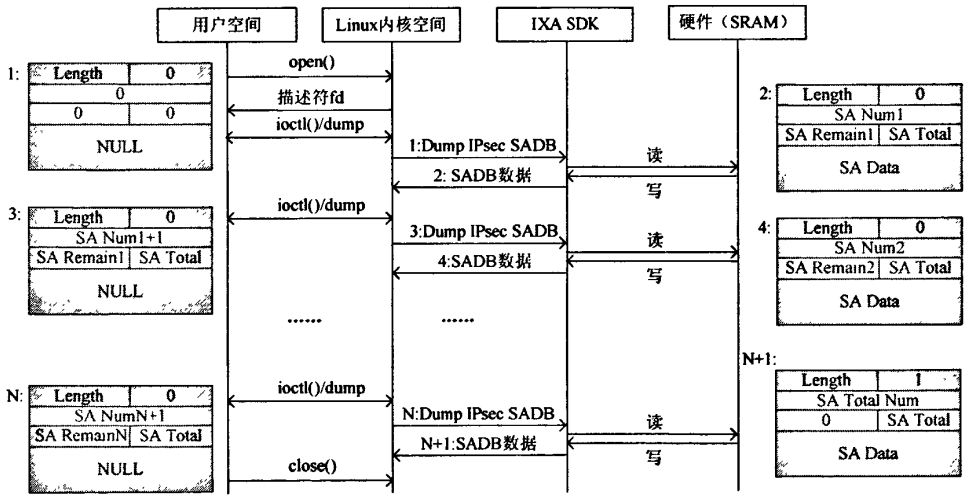


图 4-9 IPsec SA dump 用户空间与内核空间交互序列图

本协议的设计思想是多次调用 ioctl 命令将 IPsec 安全关联数据库的内容分次返回给用户空间打印显示到屏幕（如图 4-9），每次返回的数据量通过协议头中 Flag、SA ID 和 Return Value 传输。Flag 字段的作用是标志 IPsec SADB 是否打印完毕，用户空间检查 Flag 等于 0，则表示 SADB 内容没有打印完，将会重复调用 ioctl 命令；SA ID 的作用是保存上次打印的 SA ID，第一次打印时从 0 开始，以后每次当数据库单元取数据时从该 ID 号的下一 ID 开始；而 32 位的 Return

Value 被分解成两个 16 位的字段，高 16 位保存当前打印完的 SA 条目，低 16 位保存数据库中 SA 的总条目数。当用户空间判断 Flag 为 1 时将调用 close()关闭 IPsec 设备，结束本次 dump 操作。

5、清空的实现

基于 NP 的 IPsec 安全关联数据库的清空操作是 IPsec SADB 管理中最简单的操作，只需要在 ioctl 命令中传输一条 cmd 为 PURGE_IPSEC_SAD 的有效参数即可，保留字符串指针 cmdArgStr 为空。通常情况下，purge 命令带有强制性，无论 SADB 中的 SA 是否正在被使用，内核模块都将 SADB 的数据清空为 0。用户空间的程序不对 purge 命令的返回值作检查，默认操作为成功。

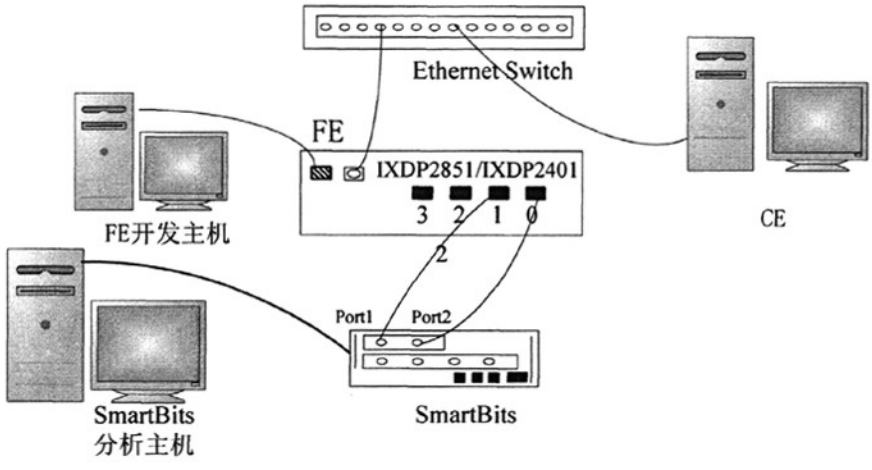
4.4. 小结

本章主要是对基于 NP 的 ForCES 架构 VPN 的一些具体实现，包括 IPSec VPN 入站和出站模块的实现，安全关联数据库的哈希管理的 CC 层和微码层的实现。本文只针对 IPsec ESP 协议的实现，且 ESP 协议中未实现认证处理。本章采用字符设备驱动的方式实现了用户空间与内核空间的通信，并实现了基于 NP 的 IPsec 安全关联数据库的各种管理操作。

5. 测试与分析

本章主要是对本文实现的 VPN 系统进行测试，以验证方案的正确性和可行性。测试内容分为：IPsec 安全关联配置的测试、IPsec 封装/解封装 LFB 测试、IPsec VPN 拓扑结构的测试、以及整体功能的测试和性能分析。

如图 5-1 所示，本文的 VPN 开发与测试环境主要由 CE、FE、SmartBits、以太网交换机(Ethernet Switch)和一台 FE 开发主机（Development Host）组成。其中开发主机上安装了 MontaVista。SmartBits 是测试仪，具有多项综合的网络测试功能，是我们主要的测试设备。



端口 MAC 地址说明：

FE	Port1: 00:01:02:03:00:01	Port2: 00:01:02:03:00:02
	Port3: 00:01:02:03:00:03	Port4: 00:01:02:03:00:04
SmartBits	Port1: 00:00:00:00:00:01	Port2: 00:00:00:00:00:02

图 5-1 VPN 开发与测试图[9]

5.1. IPsec 安全关联内核通信功能测试

本测试主要是对基于 NP 的 IPsec SA 配置时采用字符设备驱动的方式进行用户空间与内核空间通信的测试。

5.1.1. 测试方案

接受用户输入的字符串，用户空间的函数调用 IOCTL 命令将用户的参数传递到内核空间。

关于 IPsec SA 配置的一些基本命令如图 5-2 所示：

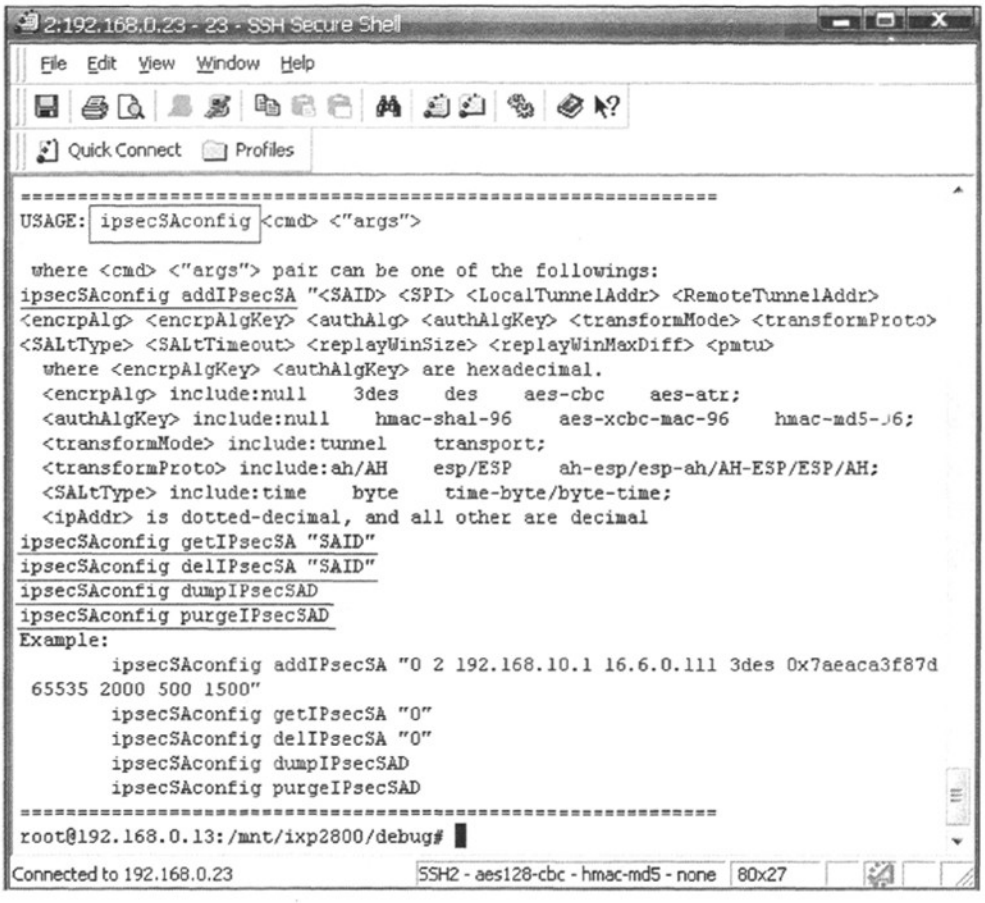


图 5-2 IPsec SA 的一些配置命令

5.1.2. 测试结果

以 IPsec SA 的批量添加命令和 dump 命令为例，本文实现了 IPsec SA 的配置，完成了用户空间和内核空间的字符设备驱动方式的通信。

如图 5-3 所示，本文将 IPsec SA 的配置以 shell 脚本的形式进行批量添加。

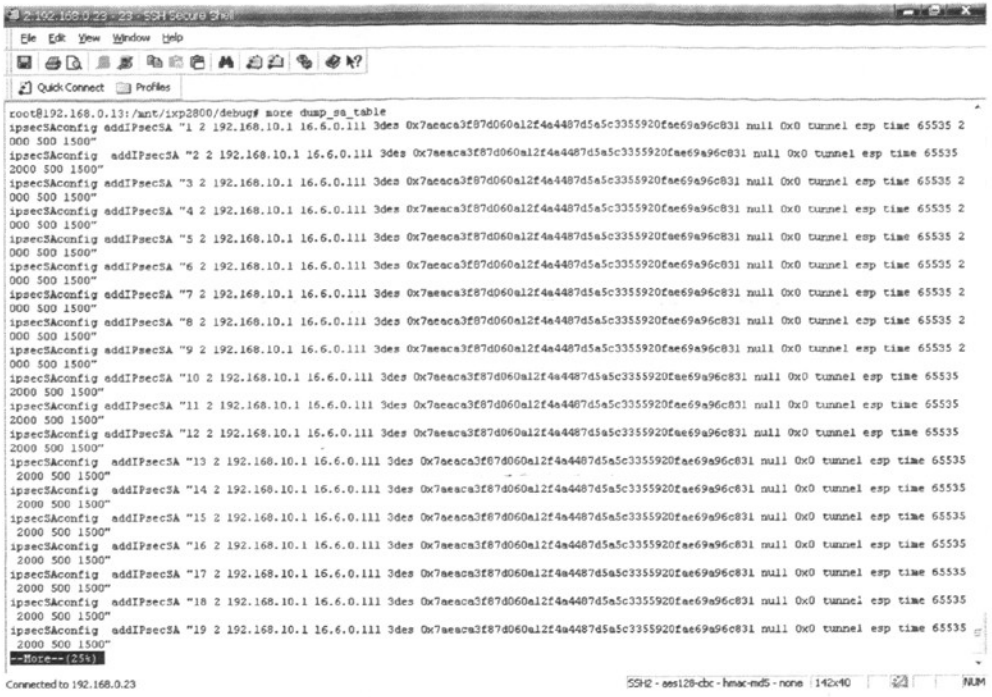
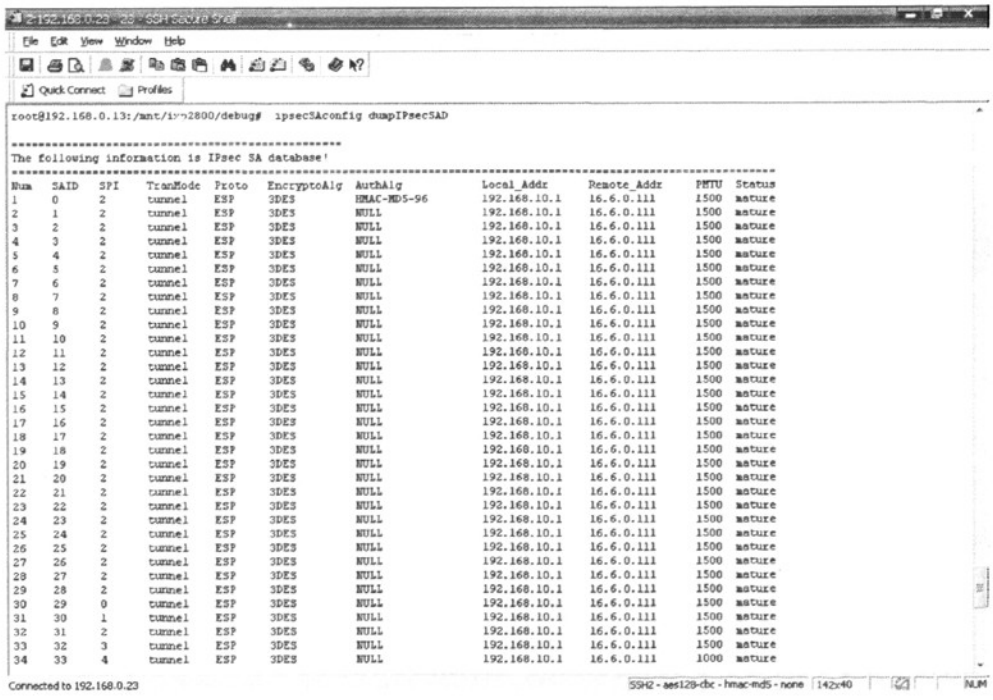


图 5-3 IPsec SA 批量添加的 shell 脚本

由于 SA 的条目过多，结构体过大，用户空间将 SADB 的信息以调用多次 IOCTL 的方式从内核空间中取出 SA 信息，并显示在屏幕上，如图 5-4 所示。



Num	SAID	SPI	TranMode	Proto	EncryptoAlg	AuthAlg	Local_Addr	Remote_Addr	PMTU	Status
1	0	2	tunnel	ESP	3DES	HMAC-MD5-96	192.168.10.1	16.6.0.111	1500	mature
2	1	2	tunnel	ESP	3DES	NULL	192.168.10.1	16.6.0.111	1500	mature
3	2	2	tunnel	ESP	3DES	NULL	192.168.10.1	16.6.0.111	1500	mature
4	3	2	tunnel	ESP	3DES	NULL	192.168.10.1	16.6.0.111	1500	mature
5	4	2	tunnel	ESP	3DES	NULL	192.168.10.1	16.6.0.111	1500	mature
6	5	2	tunnel	ESP	3DES	NULL	192.168.10.1	16.6.0.111	1500	mature
7	6	2	tunnel	ESP	3DES	NULL	192.168.10.1	16.6.0.111	1500	mature
8	7	2	tunnel	ESP	3DES	NULL	192.168.10.1	16.6.0.111	1500	mature
9	8	2	tunnel	ESP	3DES	NULL	192.168.10.1	16.6.0.111	1500	mature
10	9	2	tunnel	ESP	3DES	NULL	192.168.10.1	16.6.0.111	1500	mature
11	10	2	tunnel	ESP	3DES	NULL	192.168.10.1	16.6.0.111	1500	mature
12	11	2	tunnel	ESP	3DES	NULL	192.168.10.1	16.6.0.111	1500	mature
13	12	2	tunnel	ESP	3DES	NULL	192.168.10.1	16.6.0.111	1500	mature
14	13	2	tunnel	ESP	3DES	NULL	192.168.10.1	16.6.0.111	1500	mature
15	14	2	tunnel	ESP	3DES	NULL	192.168.10.1	16.6.0.111	1500	mature
16	15	2	tunnel	ESP	3DES	NULL	192.168.10.1	16.6.0.111	1500	mature
17	16	2	tunnel	ESP	3DES	NULL	192.168.10.1	16.6.0.111	1500	mature
18	17	2	tunnel	ESP	3DES	NULL	192.168.10.1	16.6.0.111	1500	mature
19	18	2	tunnel	ESP	3DES	NULL	192.168.10.1	16.6.0.111	1500	mature
20	19	2	tunnel	ESP	3DES	NULL	192.168.10.1	16.6.0.111	1500	mature
21	20	2	tunnel	ESP	3DES	NULL	192.168.10.1	16.6.0.111	1500	mature
22	21	2	tunnel	ESP	3DES	NULL	192.168.10.1	16.6.0.111	1500	mature
23	22	2	tunnel	ESP	3DES	NULL	192.168.10.1	16.6.0.111	1500	mature
24	23	2	tunnel	ESP	3DES	NULL	192.168.10.1	16.6.0.111	1500	mature
25	24	2	tunnel	ESP	3DES	NULL	192.168.10.1	16.6.0.111	1500	mature
26	25	2	tunnel	ESP	3DES	NULL	192.168.10.1	16.6.0.111	1500	mature
27	26	2	tunnel	ESP	3DES	NULL	192.168.10.1	16.6.0.111	1500	mature
28	27	2	tunnel	ESP	3DES	NULL	192.168.10.1	16.6.0.111	1500	mature
29	28	2	tunnel	ESP	3DES	NULL	192.168.10.1	16.6.0.111	1500	mature
30	29	0	tunnel	ESP	3DES	NULL	192.168.10.1	16.6.0.111	1500	mature
31	30	1	tunnel	ESP	3DES	NULL	192.168.10.1	16.6.0.111	1500	mature
32	31	2	tunnel	ESP	3DES	NULL	192.168.10.1	16.6.0.111	1500	mature
33	32	3	tunnel	ESP	3DES	NULL	192.168.10.1	16.6.0.111	1500	mature
34	33	4	tunnel	ESP	3DES	NULL	192.168.10.1	16.6.0.111	1000	mature

图 5-4 IPsec SADB dump 结果

5.2. IPsec 封装/解封装 LFB 测试

本测试主要是对 IPsec ESP 协议的封装和解封装 LFB 进行测试。

5.2.1. 测试方案

对完成上述的 IPsec SA 内核通信的配置功能后，我们利用 SmartBits 构造相关类型的数据包， 对 IPsec ESP 协议的封装和解封装 LFB 进行测试。在此先对一些配置命令进行介绍并检查对这些 LFB 的配置是否生效。

用户执行添加命令以后，在 IPsec SADB 中将会有一条 SA 信息，在此以查询命令为例，对配置的 IPsec SA 进行查看。查看结果如图 5-5 所示。

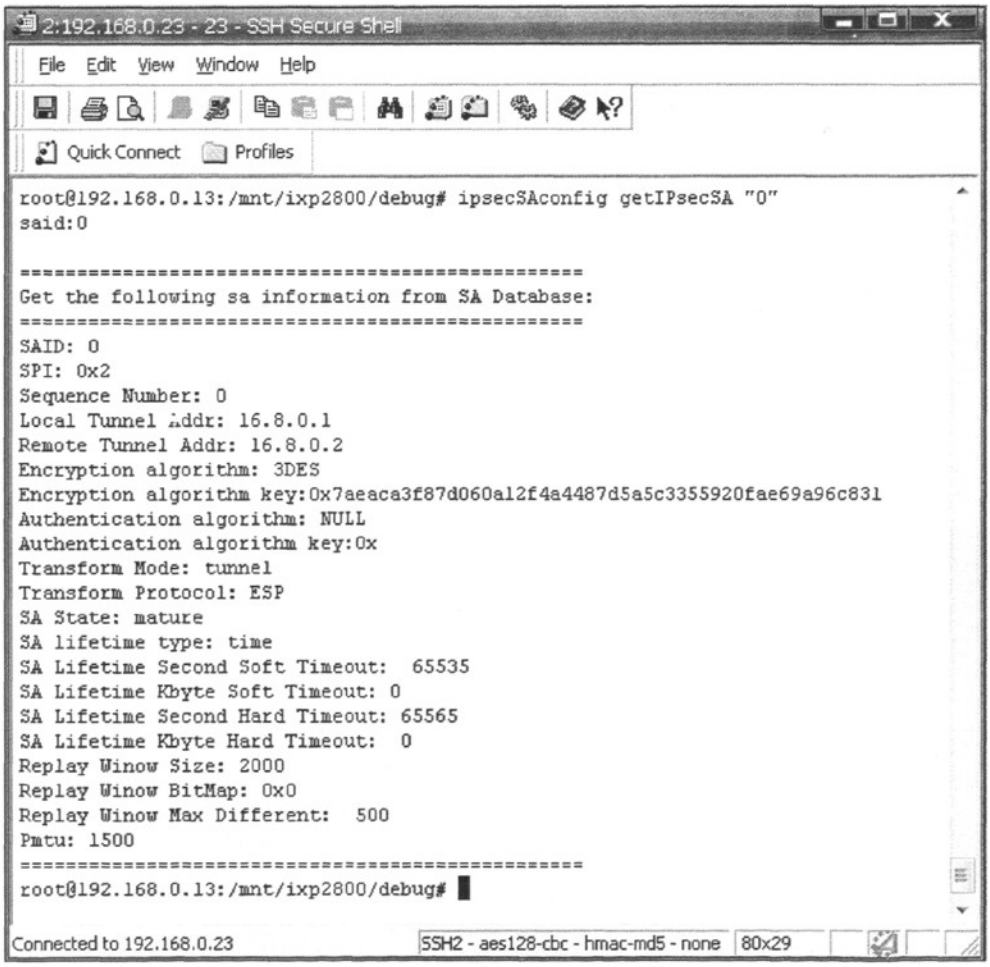


图 5-5 IPsec SA 查询结果

5.2.2. 测试结果

首先，构建需要进行 ESP 封装的数据包，利用 SmartWindow 软件构建如图 5-6 所示的数据流。

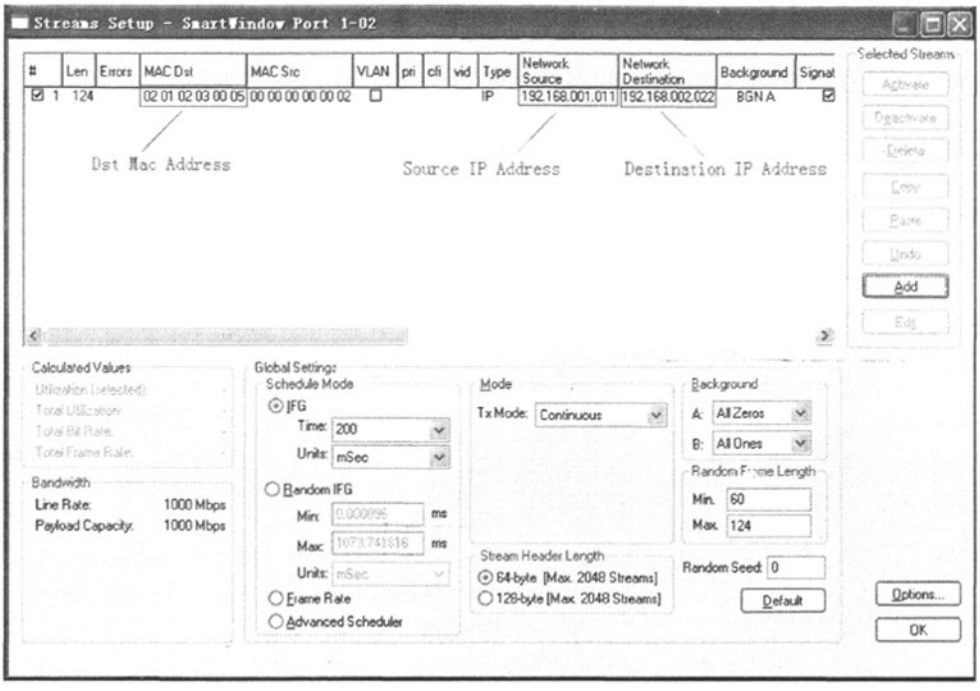


图 5-6 数据包构造图

然后在 SmartBits 的 1 号端口上捕捉数据包，其数据格式如图 5-7 所示：

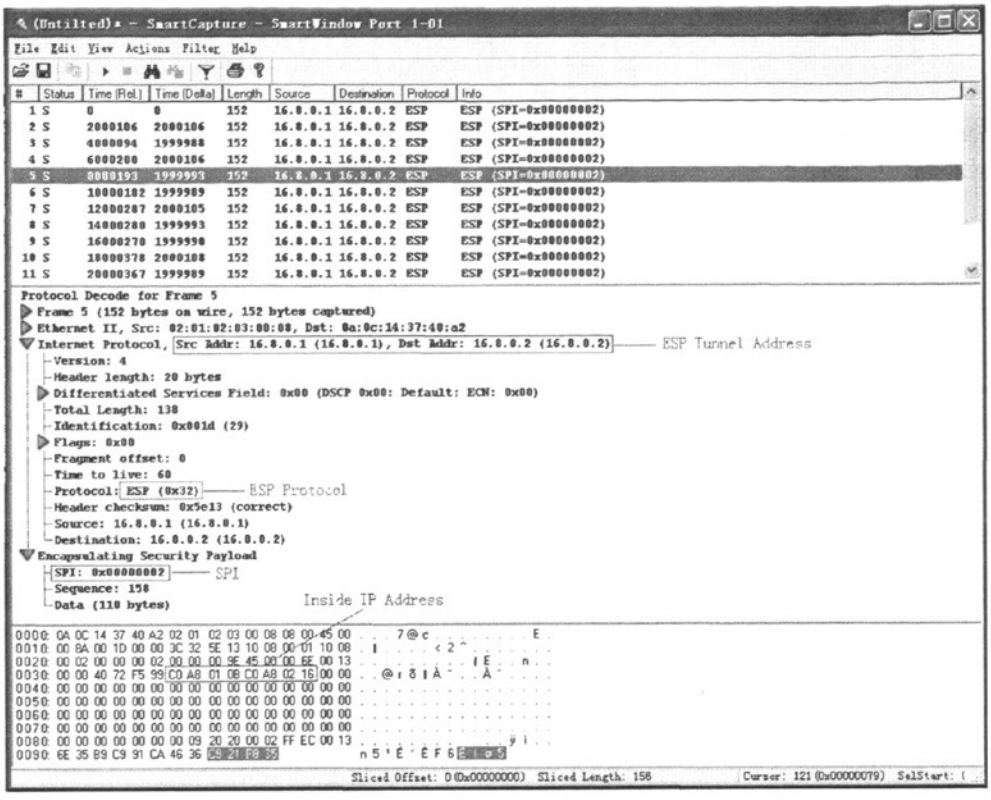


图 5-7 ESP 封装后的数据包格式

由图 5-7 可知，经过了基于 ForCES 架构的 IPsec VPN 之后，整个数据被封装成 ESP 格式的数据包被发送出去。

然后，对解封装进行验证。首先，也是构造一个 ESP 格式的数据包类型，如图 5-8 所示。

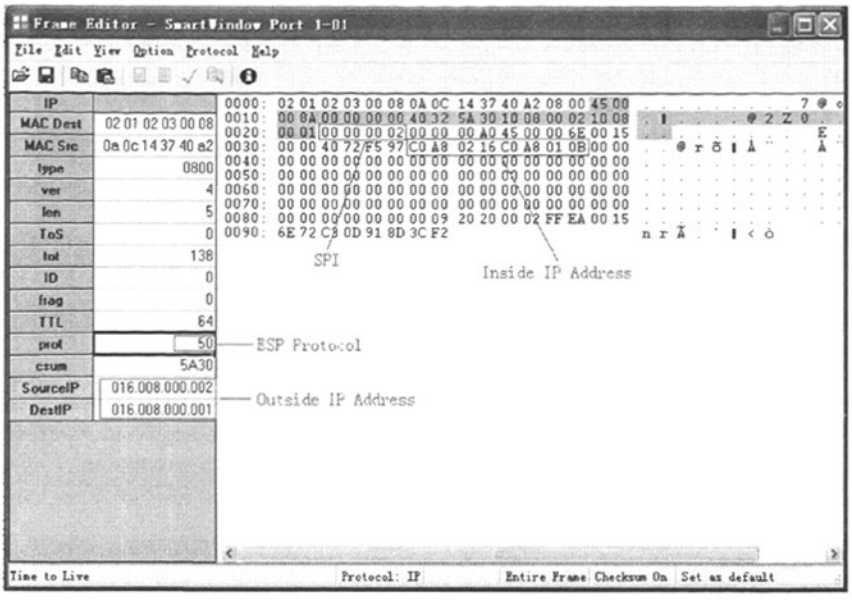


图 5-8 构造 ESP 数据包的格式

同样可以发现在 SmartBits 的 1 号端口有数据流出，对数据包进行捕捉，格式如图 5-9 所示：

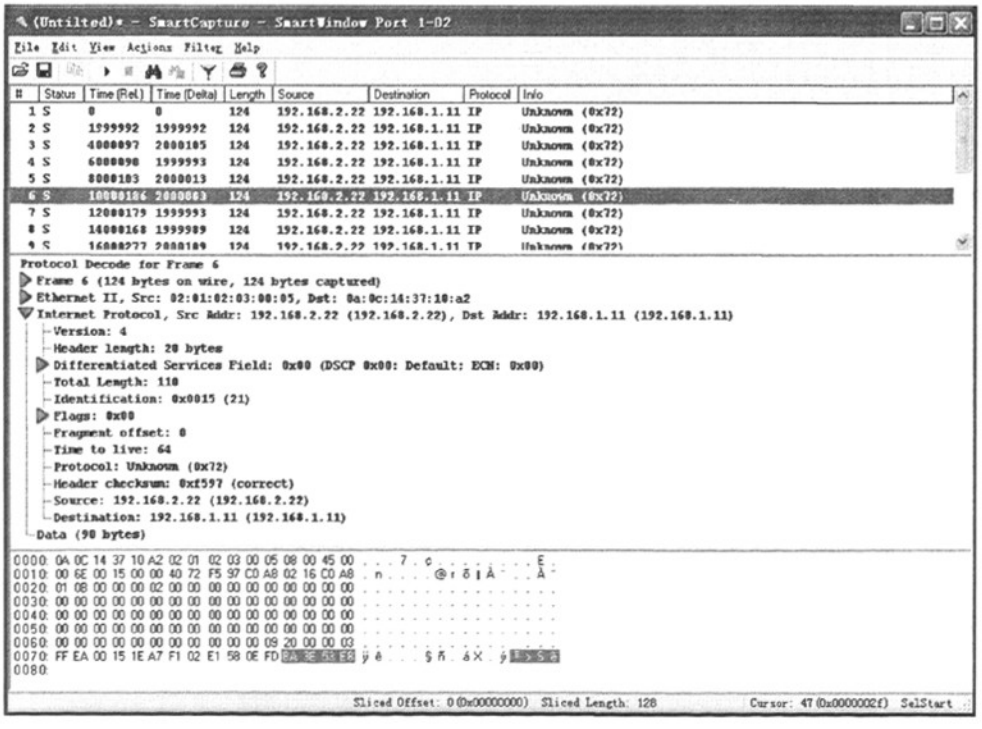


图 5-9 ESP 解封装之后的数据包格式

由图可知，ESP 数据包经过基于 ForCES 架构的 IPsec VPN 之后可以正常的

转发出去。

5.3. IPsec VPN 拓扑结构测试

5.3.1. 测试方案

在 CE 端，我们采用 WEB 形式的用户管理界面，只要在 CE 端对 LFB 拓扑结构进行查询即可。主要检查 CE 查询的拓扑结构和我们设计的拓扑结构是否一致。

5.3.2. 测试结果

测试结果如图 5-10 所示。

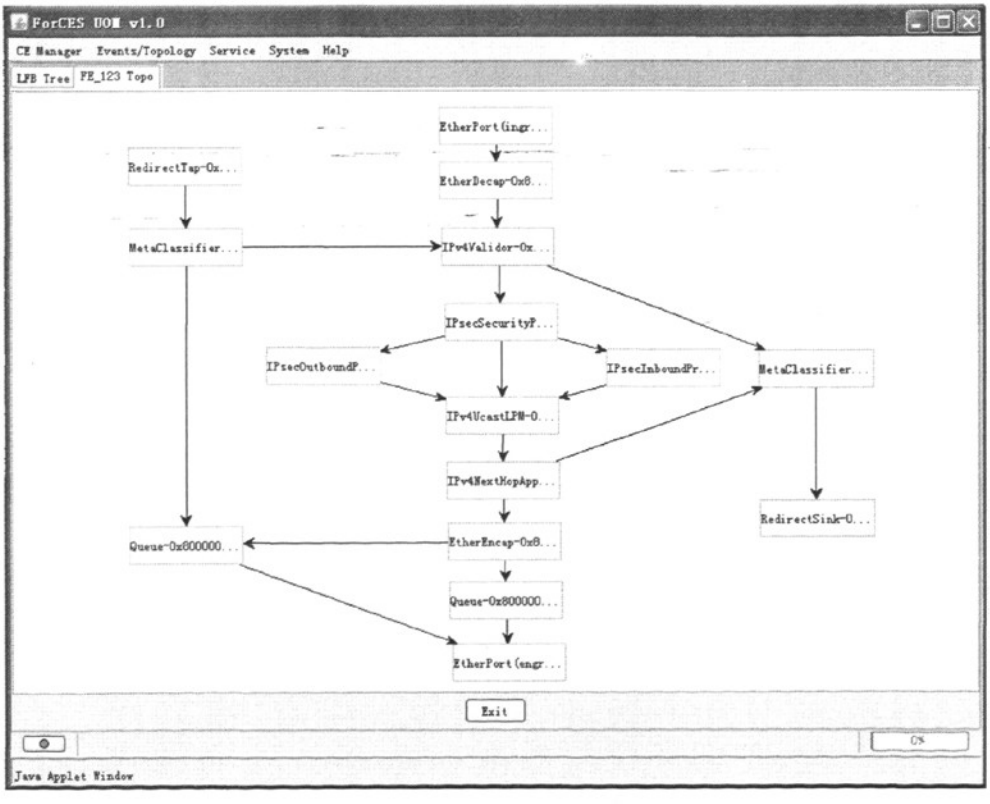


图 5-10 LFB 的拓扑结构图

由上图可知，CE 查询获得的 LFB 拓扑结构图和我们设计的拓扑结构一致。

5.4. IPsec VPN 整体功能测试

5.4.1. 测试方案

本测试方案主要是针对 VPN/LAN_to_LAN 的拓扑结构进行设计的，这种拓扑结构运用于公司的办事处之间以及不同的客户/厂商之间，两个站点必须建立隧道，在此我们就用 IPsec 隧道来完成。

测试环境主要有 ForCES 路由器原型机（作为 VPN 网关）和一台 Linux VPN 网关以及多台 Windows 或者 Linux 主机组成。平台主要是搭建网关对网关的 VPN 测试环境。其中，具体的配置在图中已经详细描述，这些配置的具体含义在上述的一些测试中已经给出说明。

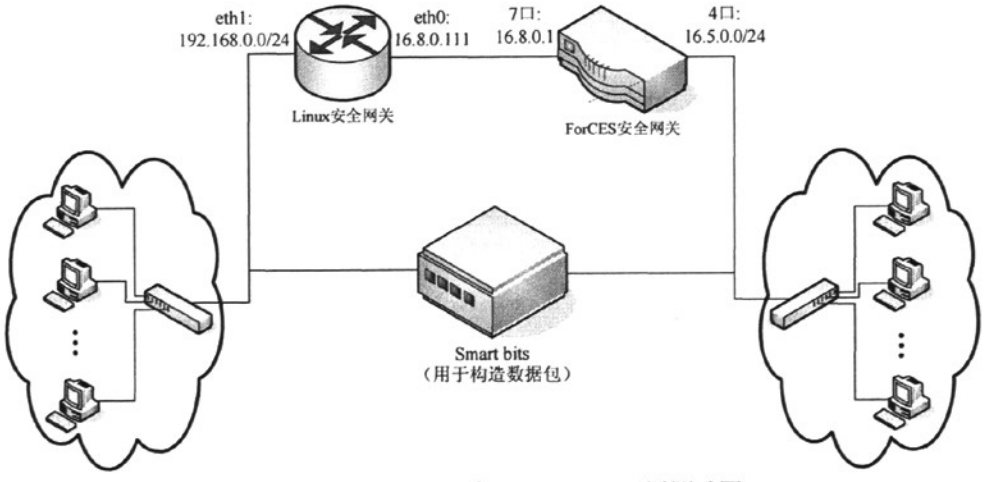


图 5-11 ForCES VPN 和 Linux VPN 互通测试图

1、ForCES VPN 侧配置：

在 ForCES VPN 开机完成后，使用命令行接口进行配置。

① 添加路由：./rtm_config_linux.sh

② 安全策略配置：

spconfig addIpsecSP “16.5.0.0 24 192.168.0.0 24 65535 65535 65535 4 OUT PROTECT 1”

③ 安全关联配置：

ipsecSAconfig addIPsecSA “1 2 16.8.0.1 16.8.0.111 3des 0x7aeaca3f87d060a12f4a4487d5a5c3355920fae69a96c831 null 0x0 tunnel esp time

65535 2000 500 1500”

我们也可以通过添加 SA 为加密和解密配置不同的 spi 和密钥。ipsecSAconfig addIPsecSA “2 3 16.8.0.1 16.8.0.111 3des 0x11111111222222223333333344444444555555557777777 7 null 0x0 tunnel esp time 65535 2000 500 1500”

2、Linux VPN 侧配置：

① 网卡 IP 地址配置

eth0 为隧道端的端口，其 IP 地址应与 ForCES VPN 隧道另一端的端口 IP 地址为同一网段，通过 ifconfig eth0 16.8.0.111 netmask 255.255.255.0 修改。同样的，eth2 的 IP 地址通过 ifconfig eth1 192.168.0.1 netmask 255.255.255.0 配置为属于 SP 中子网段的地址。

输入上述两条命令后，需要重启网络：service network restart，修改后的 IP 地址才能应用。

② 修改 eth0 的 MAC 地址

因为实验中将 7 口与 eth0 直连，故须将 eth0 的 MAC 地址设为 7 口的目的 MAC 地址。先将端口 down 掉：ifconfig eth0 down；然后修改 MAC 地址：ifconfig eth0 hw ether 0A:0C:14:37:40:A2；现将端口 up：ifconfig eth0 up。

③ 开启转发功能并添加路由

echo “1” > /proc/sys/net/ipv4/ip_forward

route add –net 16.5.0.0 netmask 255.255.255.0 gw 16.8.0.1

④ 添加 arp

arp –i eth0 –n –s 16.8.0.1 02:01:02:03:00:08

arp –i eth1 –n –s 192.168.0.100 01:02:03:04:05:06

⑤IPsec VPN 配置

Linux 下我们使用 setkey 命令来添加 SAD 和 SPD。

完成上述操作后，我们就可以用 Smart bits 构建并发送 IP 包。

5.4.2. 测试结果

在此主要看 LAN 与 LAN 之间的主机能否完成通信，测试方法可以用 Ping

主机或者直接用一些应用程序进行访问。

1、在主机 192.168.1.4 上 Ping 主机 192.168.2.222，测试结果如图 5-12 所示。

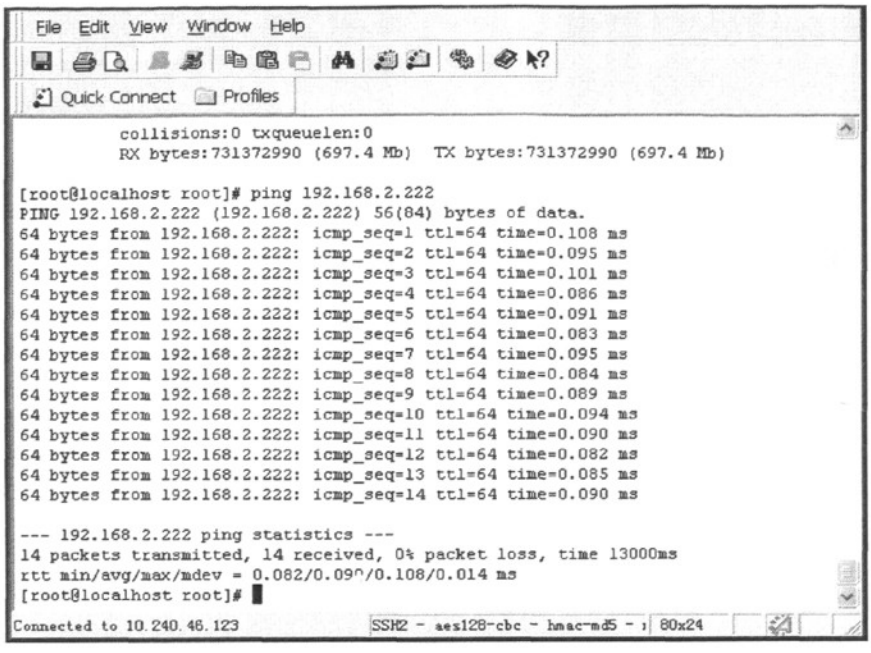


图 5-12 主机之间的通信

由图 5-12 可知，测试结果表明主机之间能够进行通信。然后在 Linux VPN 网关上进行数据包的捕获测试，测试结果如图 5-13 所示。

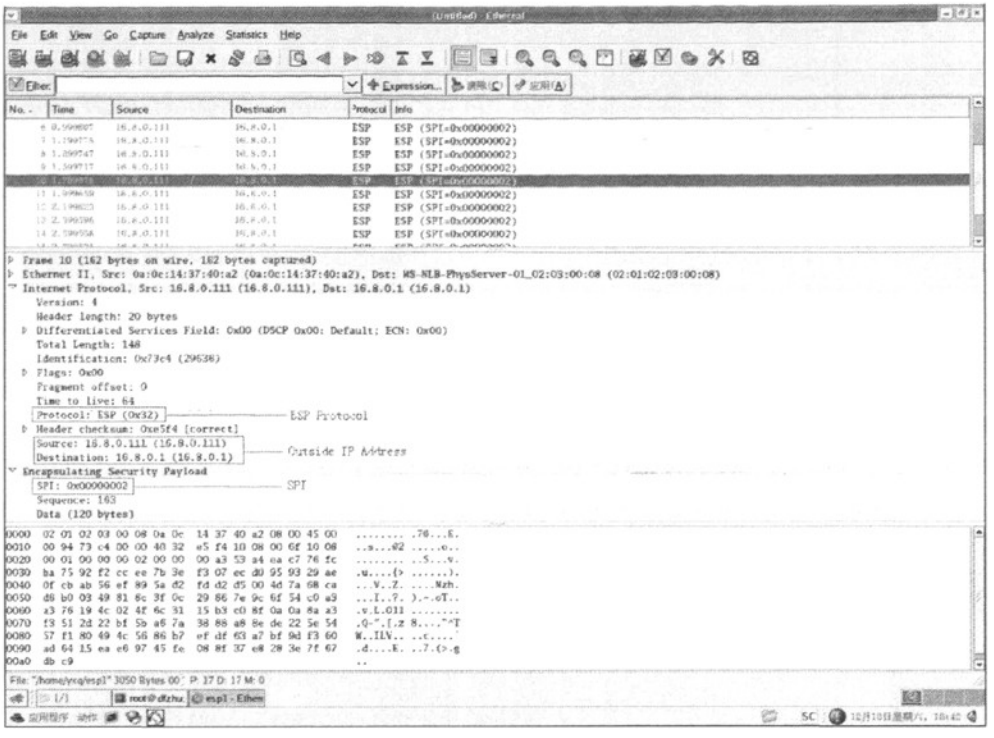


图 5-13 ESP 数据包解析

由截获的数据包分析可知整个数据的通信是通过 IPsec 隧道进行的。

2、用主机 192.168.1.4 Telnet 登录到主机 192.168.2.222 上，测试结果如图 5-14 所示：

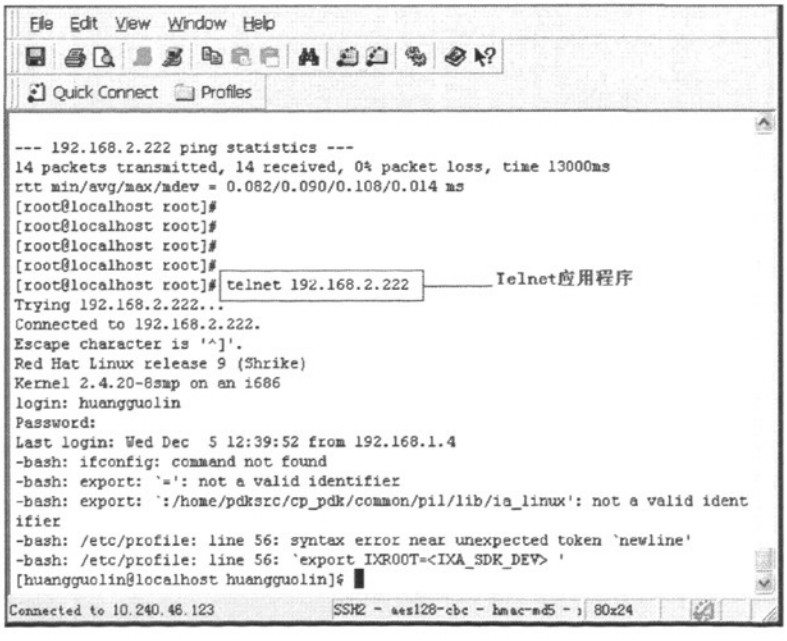


图 5-14 Telnet 登录测试图

由图可知，主机的应用程序也能够进行正常的通信。

5.5. IPsec VPN 数据包处理平台性能分析与测试

通过前几章的论述，我们对基于 NP 的 ForCES 架构下实现 IPsec VPN 的关键技术及网络平台作了详细讨论。将 NP 技术应用于 VPN，构建低成本，高性能的 IPsec VPN 网关是本文的创新点。那么，本文所构建的 IPsec VPN 网关性能究竟如何呢？本章将对 IPsec VPN 网络数据包处理平台的性能做出分析。

对网络安全平台进行性能评估时，常用到以下参数：数据包吞吐量、数据包处理资源。数据包处理资源包括微引擎、存储器等资源，通常用总线带宽和指令条数来衡量。本节将使用这几个参数对系统进行性能分析，最后对系统性能进行了测试。

5.5.1. 系统性能分析

考虑最坏的情况，当 IXP2850 的时钟频率为 1.4GHz[33]，数据包为最小的 64bytes 以太网包，由于广域网物理层的数据率要比 1Gbps 略低一些，为了使 10/100M 以太网的帧能够插入到千兆以太网帧的有效载荷中，就必须设法降低进入物理层的数据率。具体做法是，在 MAC 子层的以太网帧之间插入一些附加帧间隔(IPG，Inter-Packet Gap)。每一个 IPG 的字节数与前一帧的长度成正比。本文的帧间隔为 20bytes，当线速为 1 Gbits/sec 时，数据包的到达间隔时间为 945 个时钟周期，每个微引擎上有 8 个线程，数据包的存储器访问时延门限为 945*8 个时钟周期。因此，每一个数据包处理模块的处理周期不应该超过这个门限值，否则就会出现丢包现象。具体的性能分析如表 5-1 所示：

表 5-1 千兆以太网 VPN 性能分析

速率	1 Gigabits/sec
最小以太网包字节	64 bytes + 20 bytes 帧间隔
最小 IPv4 包吞吐量	$1.48*10^6$ pkt/s= (1Gb/s / (84*8))
时钟周期	1.4 GHz

最小数据包到达间隔	1.4GHz/1.48*10^6 pkt/s = 945 cycles
单个微引擎的指令周期门限	945 cycles
单个微引擎的存储器时延门限	945* 8 cycles
N 个微引擎的指令周期门限	945* N cycles
N 个微引擎的存储器时延门限	945* N * 8 cycles

5.5.2. 微引擎性能分析

如表 5-2 所示，本文对 SDK 上的微码指令条数和最坏情况下的以太网包处理时微引擎的资源利用情况做了分析。每条指令执行的时间为一个微引擎计算周期，称指令周期。当 SDK 中的微码实际指令周期小于微引擎资源的理论阈值评估时，网络处理器便可以对数据包进行线速处理，即不存在缓存和丢包的情况。本文将 IPv4 转发和 IPsec 入站以及 IPsec 出站处理微块都分配到 8 个相同的微引擎上，以满足网络处理器的线速处理需求。

表 5-2 微引擎性能分析

微块	微码实际指令周期（cycles）	以太网包处理指令理论阈值评估（cycles）
包接收	720	945*1
包发送	810	945*2=1890
IPv4 转发	1350	945*8=7560
IPsec 入站处理	2120	7560
IPsec 出站处理	1960	7560
队列管理	530	945*1
调度	340	945*1

在进行微引擎资源的分配、将数据包处理任务映射到微引擎时，不仅要考虑

到 NNR 和线程信号等的使用效率，也要考虑 Command Bus，S Push/Pull Bus 的利用情况。因为 ME3~ME6 属于 ME Cluster0，ME12~ME15 属于 ME Cluster1，所以在本文 3.2 节讨论的微引擎的分配中，考虑到了两个 ME Cluster 的 Command Bus 和两组 Push/Pull Bus 的利用率的均衡。

5.5.3. 存储器性能分析

根据如表 3-1 给出的 IXP2850 上的三大存储器 SRAM、DRAM、Scratch 以及 Local Memory 的访问时延，本文粗略计算了在最坏情况下的存储器访问时延如表 5-3 所示，当数据包内容读写到存储器时，由于 NP 上包含 8 个微引擎资，且每个微引擎上有 8 个线程可并行处理，数据包不存在访存阻塞。在实际的处理过程中，执行代码会利用微引擎上线程及存储器的特点，当某一线程在访问存储器时，该线程将被挂起，并将微引擎的控制权交给其他线程，当存储器访问结束后唤醒线程继续执行，这个过程被称为 I/O 时延隐藏。充分利用 I/O 时延隐藏将会大大提高系统的处理效率。

表 5-3 存储器性能分析

微块	微码实际时延(cycles)	以太网包处理时延阈值评估(cycles)
包接收	296	945*8=7560
包发送	N/A	945*8*2=15120
IPv4 转发	5800	945*8*8=60480
IPsec 入站处理	9480	60480
IPsec 出站处理	9300	60480
队列管理	849	7560
调度	N/A	7560

综上所述，本节分析的微引擎性能和存储器性能满足了数据包千兆线速处理的性能要求。

5.5.4. 测试方案

本测试方案基于 NP 的实施 IPsec 功能模块的性能测试。数据包由 SmartBits 产生，从网络处理器的一个千兆以太网端口流进，从另一个千兆以太网端口流出，逐渐增大 SmartBits 发送数据包的速率，统计数据包的转发速率，以此来测试 ForCES VPN 的性能。本文构造的数据包长度固定为最坏情况下的 64 bytes，IPsec 模块对数据包进行字节填充和 ESP 协议封装，不对数据包进行加密处理，如果 64 字节的数据包能被线速处理，那么对于字节更长的数据包就能达到线速处理。

5.5.5. 测试结果

测试结果如图 5-15 所示，在 SmartBits 发送速率比较小时，系统完全可以线速处理，所以本文仅测试了几个小速率的情况，在原坐标附近的斜率比较小。当逐渐增大端口发送速率时，ForCES VPN 的处理速率呈线性增长。当速率慢慢接近 1Gbit/s 时，系统维持在一个平滑的水平线上，若继续增大端口的发送速率，由于 VPN 系统的流量控制作用，将控制数据包的接收速率，所以在如图 5-15 所示中，横坐标的右半部分将维持在一个稳定的速率上下浮动。

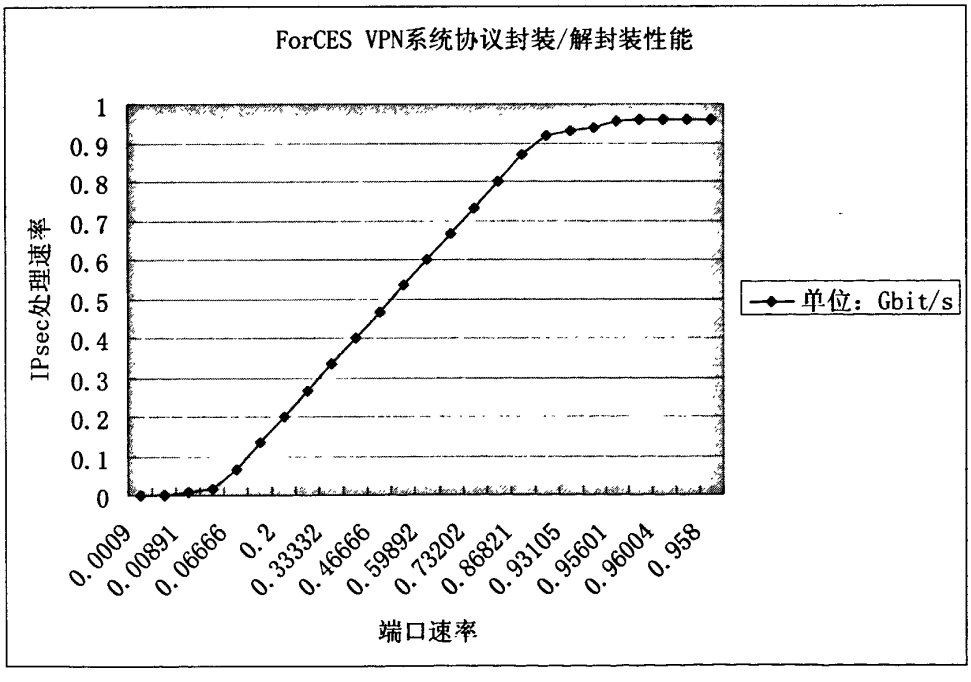


图 5-15 ForCES VPN 系统的封装解封装性能

本测试不包括 IPsec 加密模块，测试结果表明，本文设计的 ForCES VPN 系统可以用作对数据包进行 IPsec 的线速处理。

6. 总结与展望

6.1. 论文总结

本文的研究内容是基于 ForCES 架构路由器中实现 VPN 的功能。

根据 ForCES 路由器的 FE 建模要求和 VPN 的特点以及三层 VPN 的实现要求，我们采用了 IPSec 协议来实现基于 ForCES 架构的 VPN。首先根据 ForCES 思想实现了基于 ForCES 架构的 IPSec VPN 的整体架构的设计，然后在此基础上，对数据处理流程中的关键模块进行抽象，形成各自的 LFB。本文主要针对 VPN 的 LFB 进行研究和设计，提出了 IPSec_Inbound_Process LFB 和 IPSec_Outbound_Porcess LFB 等。设计完 VPN 相关 LFB 之后，根据与其它 LFB 的连接关系，构建基于 ForCES 架构的 VPN FE 端 LFB 拓扑结构。

本文重点研究了基于 NP 的 ForCES VPN 实现的关键技术，包括 IPsec 入站模块及出站模块的研究与设计、基于散列技术的安全数据库的管理、IPsec 用户空间及内核通信机制的研究与设计，并对微码上的数据结构进行了设计，便于高效紧凑地利用了 NP 上有限的存储空间进行软件开发。此外，本文设计了基于 NP 的 IPsec 安全关联数据库配置协议，此协议适用于内核及网络通信，对内核与用户空间之间的通信研究具有一定的理论与实践指导意义。

根据本文提出的设计方案，我们采用了网络处理器作为 FE，普通的 PC 机作为 CE，实现一个基于 NP 的 ForCES 架构 VPN 的模型。

最后，对该系统进行了功能性测试和性能测试。测试结果验证了基于 ForCES 架构的 VPN 整体设计方案的有效性，功能运行的正确性，此外，测试结果还表明本文的设计方案达到了千兆线速处理的性能。

6.2. 工作展望

随着下一代网络的发展和信息科技的迅速发展，网络安全问题已经成为信息化社会的一个焦点问题。如何实现网络安全，抵御网络风险，成为目前网络研究的一个重点和难点问题。而 ForCES 路由器作为下一代路由器，目前还处于研究

阶段。因此，在 ForCES 路由器中实现 VPN 支持是我们工作组的一个重要研究方向。

在之前的工作中，由于工作组团队规模的限制，只针对基于 ForCES 架构的 IPSec VPN 只做了初步的研究和设计，并只对其中部分功能进行了实现，离我们的目标还相差很远。其次我们对基于 ForCES 架构的 VPN 做了一些功能性的测试，在性能方面还可以继续优化。所以下一阶段需要添加和改善的地方有：继续实现基于 ForCES 架构的 IPSec VPN 的 AH 协议，并对整个系统的性能进行测试和改善。

参考文献：

[1] Doria (Ed.), et al., ForCES Protocol Specification[EB/OL], Internet Draft, March 2006.

[2] Khosravi, T. Anderson et al., Requirements for Separation of IP Control and Forwarding[EB/OL], RFC 3654, November 2003.

[3] L. Yang., R.Dantu, T. Anderson., et al., Forwarding and Control Element Separation (ForCES) Framework[EB/OL] , RFC 3746, April 2004

[4] J. Halpern, E. Deleganes et al., ForCES Forwarding Element Model[EB/OL], Internet Draft, October 2007

[5] 田春岐，王立明等. IPSec VPN 的研究和分析，计算机工程与应用，2004.4, P163-166

[6] 张尧学，赵艳标.网络安全技术.世界网络与多媒体，Vol,5,No.1(1997-01), P36-41

[7] Carlton R.Davis (美). IPSec:VPN 的安全实施[M]. 北京：清华大学出版社, 2002

[8] Vijay Bollapragada(美), Mohamed Khalid(美), Scott Wainner(美). IPSec VPN 设计[M]. 北京：人民邮电出版社,2006

[9] 黄国淋. 基于 NP 的 ForCES 架构 VPN 的研究与实现 [D].浙江:浙江工商大学信电学院,2008.

[10] 宋伟，刘卫宁.虚拟专用网(VPN)—安全灵活的电子商务网络平台.计算机应用，Vol.19,No .9 (1999)P43-450

[11] 施恩,郑爱赛,杨彬. 基于网络处理器的高性能 IPSec VPN 的设计方案[J]. 计算机应用研究,2005/08

[12] 胡宁，王勇军. 基于网络处理器的 IPSec 协议实现技术的研究[J] . 计算机工程,2006/05

[13] 文成玉,杨槐. 基于网络处理器的 IPSec 的实现方案研究[J]. 成都信息工程学院学报,2005/01

[14] 周贤伟. IPSec 解析[M]. 北京：国防工业出版社,2006

[15] Naganand Doraswamy, Dan Harkins. IPSec：新一代因特网安全标准[M]. 北京：机械工业出版社,2000.1

[16] Paul Wouters, Ken Bantoft. Building and Integrating Virtual Private Networks with OpenSwan[EB/OL]. http://www.openswan.org/

[17] Multiservice Switching Forum, http://www.msforum.org/

[18] Network Processing Forum (NPF), http://www.npforum.org/

[19] Ben Hardekopf, et al. Impact of Network Protocols on Programmable Router Architectures[EB/OL] 2003. http://www.cs.utexas.edu/users/vin/pub/pdf/jspe03.pdf

[20] Michael E. Kounavis, et al. Programming the Data Path in Network Processor-Based Routers[J], Software Practice and Experience:Special Issue on Software for Network Processors, 2004

[21] Tilman Wolf. Design and Performance of Scalable High-performance Programmable Routers High-performance Programmable Routers[EB/OL] 2002. http://citeseer.ist.psu.edu/wolf02design.html

[22] Patrick Crowley, et al. Characterizing Processor Architectures for Programmable Network Interfaces[EB/OL] May, 2000. http://www.cs.washington.edu/homes/baer/ics00.pdf

[23] 华为 3com，IPsec 入门指南，http://www.stor-age.com/itpaper/detail/2/18762.shtml

[24] S. Kent, BBN Technologies, IP Authentication Header,RFC4302, December 2005

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

[26] 张宏科，苏伟，武勇．网络处理器原理与技术[M]．北京：北京邮电大学出版社，2004

[27] Shah N. Understanding network processors [M]. Berkeley: Department of Electrical Engineering and Computer Sciences, University of California, 2001

[28] Stamatis Vassiliadis, et al. Network Processors: Issues and Prospectives[EB/OL]. http://citeseer.ist.psu.edu/476843.html

[29] Intel. Network Processors[M]. Intel Technology Journal, 2002

[30] Intel. Intel IXP2800 Network Processor Hardware Reference Manual[C]. 2004(8).

[31] Intel. Intel IXP2XXX Product Line of Network Processors Development Tools User's Guide[EB/OL], 2004

[32] Intel. Intel Exchange Architecture Software Building Blocks Applications Design Guide[EB/OL], 2004.

[33] Intel. Intel IXP2850 Network Processor Hardware Reference Manual[C]. 2004(8).

[34] Intel. Intel IXP2400 and IXP2800 Network Processor Programmer's Reference Manual[EB/OL], 2004.

[35] Muthu Venkatachalam, Prashant Chandra, RajYavatkar. A highly flexible, distributed multiprocessor architecture for network processing[EB/OL]. http://portal.acm.org/citation.cfm?id=765784.765787&dl=GUIDE&dl=ACM

[36] Intel. Intel Exchange Architecture Software Development Kit Software Framework Tutorial[EB/OL], 2004

[37] Intel. Intel Internet Exchange Architecture (Intel IXA) Software Development Kit 4.0 [EB/OL],2004

[38] Intel. Intel Internet Exchange Architecture Portability Framework Developer's Manual[EB/OL], 2004

[39] Chidamber Kulkarni. Programming Challenges in Network Processor Deployment[EB/OL], 2003

[40] 曹道刚，曹庆华，网络处理器 IXP2XXX 微引擎开发的若干关键问题[J]，沿海企业与科技，2005

（5），52

[41] 董昱，马鑫，基于 netlink 机制内核空间与用户空间通信的分析[J]，测控技术，2007，26（9），57-58.

[42] 郑伟，王钦若，吴乃优，Linux 内核空间设备驱动程序的开发，微计算机信息[J]，2003，19（12）

[43] 杨宇音，李志淮，Linux 中用户空间与内核空间的通信实现[J]，微机发展，2005，17（5），76

[44] 魏永明，耿岳，钟书毅译，Linux 设备驱动程序（第三版）[M]，中国电力出版社，2006

本文作者硕士期间参加的科研项目及发表的学术论文

一、参加的科研项目

1、国家高技术研究发展计划（863计划)项目："ForCES体系结构与关键技术研究和国际标准制定"（2007AA01Z201） — 研究中；

2、国家自然科学基金："松散耦合型分布式路由器的若干关键技术研究"（60603072)— 研究中；

3、浙江省科技计划重大专项："ForCES国际标准制定及产品研发与产业化应用"（2006C11215） — 研究中；

4、浙江省教育厅重点项目："基于ForCES结构的分布式路由器的关键技术研究"（20061070） — 研究中。

二、发表的学术论文

"Design and Implementation of State Machine based Distributed Transaction in the ForCES Router", 2009 International Conference on Communications and Mobile Computing (CMC 2009), in Kunming, China ,January 6-8, 2009 （第一作者）（EI检索）

致 谢

值此论文完成之际，谨向所有在我研究生阶段的学习生活中给予指导、帮助、关心和支持的老师、同学、亲人及朋友们表示出衷的感谢和深深的敬意。

首先，我要特别感谢我的导师王伟明教授。他严谨细致、一丝不苟的作风一直是我工作、学习的榜样；他循循善诱的教导和不拘一格的思路给予我无尽的启迪；他渊博的学识、优秀的组织和领导能力、对学科发展方向的敏锐洞察力对我今后的学习与工作将会产生深远的影响，使我终身受益。王老师不仅对我的学习、科研和生活给予了悉心的指导和无微不至的关怀，而且为我提供了一个很好的发展个人潜力的平台，让我在实践中提高独立发现问题、分析问题和解决问题的能力。正是王老师的言传身教，使我在这两年半的学习中有了长足的进步。

衷心感谢贾凤根博士，他们在本课题的研究和本论文的写作过程中给予的无私帮助和提出的许多非常有价值的建议使我受益匪浅，少走了许多弯路。

感谢董黎刚副教授、诸葛斌副教授、李传煌、俞谨、金蓉、吴晓春和高明等老师，他们对科研事业的不懈努力和追求，对工作的忘我热情，还有平易近人的处世态度都给我留下了极其深刻的印象。

感谢所有的同学和朋友，特别是杨尚大、张丽霞、钱柯、陈坷、颜小清、彭剑辉、金少丹、卢英、姚天明和张有兵，他们给我提供了一个和谐、融洽的学习与生活环境，让我愉快地度过了研究生学习生涯。

感谢我的爸爸妈妈，你们的支持与鼓励是我不断前进，并战胜所有困难的动力与源泉，我的所有成绩与殊荣都归功于你们，养育之恩，无以回报，你们永远健康快乐是我最大的心愿。谨以此文献给你们。

孙中海

2008 年 12 月

于浙江工商大学