

网 络 之 路

IPsec 专题

ROUTER TO NETWORK

CONTENTS

■ IPsec和IKE

IPsec 概述	2
IPsec 细节及报文封装	8
IPsec 安全算法介绍	12
IKE 介绍	32
初识 IKE v2	41

■ Features

DPD	52
IPsec NAT 穿越	58
XAUTH	69
Easy VPN	78
模式配置 (MODECFG)	88
IPsec 可靠性	92

■ IPsec应用

三种 VPN 技术的比较	101
IPsec VPN 应用	106
IPsec 典型组网	119

■ 测试方法

使用 Avalanche 进行 IPsec 性能测试	125
使用 IXIA VPN 进行 IPsec 测试	136

网 络 之 路

IPsec 和 IKE



IPsec 概述

◆◇□ 杜祥宇

引言

信息系统安全简单理解就是确保合法的用户可访问被授权的资源，同时抵抗外部攻击保证正常服务的可用性。网络安全则是在信息承载网络上解决信息安全问题，即包括了系统自身安全和被承载信息的安全双重意义。网络安全研究的方向包括：访问控制、认证、授权、审计、信息密码技术、攻击识别与防范等等。IPsec 作为网络安全的一个重要协议组，其实现的功能体现了网络安全的大部分需求，覆盖了网络安全领域的主要概念。IPsec 主要依赖密码技术提供认证和加密机制，它是现代密码技术在通信领域的应用范例。同时，因为 IPsec 为用户数据提供了一个安全通路，它也被理所当然视为 VPN 技术的一个分支。

IPsec 的由来

IPv4 众所周知的问题是地址即将耗尽。这是因为设计和开发 IPv4 时，任何人都未预料到 Internet 将会爆炸式增长。同样的原因，IPv4 产生时与现时 Internet 的差异也导致另外一个主要问题：在 IP 互联网上缺少明确的安全保护手段。

25 年以前的 Internet 规模小而且相对私有，如今可谓庞大而且公开。伴随着 Internet 的增长，安全的需求也在增长。作为 TCP / IP 载体的 Internet 前身是由美国 DARPA 作为研究项目开发的小型网络。所有的网络硬件被相互熟知且身份安全的人所控制。这种网络的安全性被建造到建筑物中，不需要由协议来保证。很显然，使用锁头和门卫比用加密技术更容易。毕竟，防止某人窃听或篡改网络上数据的最简单方法就是，拒绝他接触到与网络连接的任何系统。

在 Internet 发展初期，只有几十台主机时，这种方法很有效。在网络扩张的初期，也仅仅是接入一些研究人员和网络专业人士。起初新站点被连入的速度很慢，人们总是知道新加入者的身份。然而，随着 Internet 的持续增长直至完全公开化，试图以这种方式继续维持整个网络的安全已经变得不可能。如今，Internet 上有大量的不可靠用户，许多网络中间的路由器被未知的人管理。不幸的是，绝大部分通讯流量要穿越这些未知网络，所以根本无法保证在网络上发送和接收数据是安全的。

在过去的数年之间，很多方法被提出来满足这种安全需求。为了弥补 IP 的安全性缺失，大部分方法聚焦在 OSI 协议栈的高层协议。这些方法在特定的场景下是有价值的，但是却不够通用和简单，因为他们都关联到特定的应用之中。例如，我们可以用 SSL 来为 WEB 或 FTP 提供安全连接，但是这种安全手段对至少数十种的应用（如 ping、SNMP、telnet）无效。

人们真正需要的是在 IP 层提供安全性的方法，这样可以使 TCP / IP 高层的所有协议受益。当 Internet 社区决定要开发一个新版本 IP 时，这

被看作是同时解决地址短缺和 IP 安全性问题的绝佳时机。新的安全技术在 IPv6 构思框架下设计，但 IPv6 的开发与推广会耗费数年光景，而安全性的需求却迫在眉睫，所以 IP 安全协议被设计成同时支持 IPv4 和 IPv6。这样一个为 IP 通信带来安全性的协议就被称为 IP Security，缩写为 IPsec。

IPsec 服务和功能

IPsec 并非一个单独的协议，而是一系列为 IP 网络提供完整安全性的协议和服务的集合。这些服务和协议结合起来提供不同类型的保护。因为 IPsec 工作在 IP 层，所以它能为上层协议和应用提供透明的安全服务，这也是它最大的好处。

2.1 IPsec 提供的安全服务

IPsec 可以提供的安全保护包括：

用户数据加密，通过加密算法提供数据私密性；

消息完整性验证，通过摘要认证确保数据在传输路径上未经修改；

防御特定类型的攻击，通过序列号防数据重放、通过认证防中间人攻击；

提供设备之间安全算法和密钥的协商能力，提供安全的在线密钥生成机制；

提供隧道和传输两种安全模式，满足不同的网络结构需求。

值得指出的是，IPsec 的数据加密能力和在线密钥协商都依赖于密码学的发展。

2.2 IPsec 的应用场景

图 1 所展示的场景为 IPsec 的一种典型应用方式：IPsec 在企业中心与分支机构之间提供安全的隧道服务。企业各个部分通过 Internet 提供商提供的接口连入互联网实现 IP 互通，然后再通过出口部署专用网关或路由器来提供 IPsec 隧道（图中蓝色的连接表示 IPsec 隧道）。出差用户也可以通过 PC 直接发起的 IPsec 隧道接入公司总部

网关。所有的企业用户的异地互访数据流通过安全的 IPsec 来承载，虽然是在公网上传输，但都得到加密和认证保护。

了 SA 的作用范围和相关属性。

3.2 IPsec 的协议组成

IPsec 协议体系包括了两个安全处理协议和

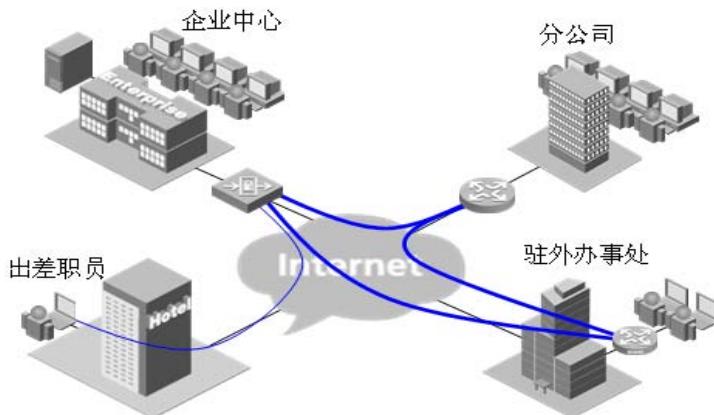


图 1 IPsec 典型应用场景

IPsec 协议框架

3.1 IPsec 关键概念

安全关联(Security Associations): 为安全目的创建的一个单向逻辑连接。所有经过同一 SA 的数据流会得到相同安全服务，AH 或 ESP。对同一个数据流同时使用 AH 和 ESP 服务时需要两个嵌套的 SA。双向的数据流需要通信实体之间维护一对出 / 入彼此呼应的 SA。

安全关联数据库(Security Associations Database): 用于存放与 SA 关联的所有状态数据的存储结构。

安全参数索引(Security Parameters Index): 一个被携带在 AH 或 ESP 报头中的 32bit 数值，用于在接收端识别数据流到 SA 的绑定关系。

安全策略数据库(Security Policy Database): 指明所有 IP 数据报文应使用何种安全服务，以及如何获得这些服务的数据结构。SPD 通常是一个有序的结构，用访问控制列表来描述数据流特性。定义数据流和安全服务方式的接口与具体实现相关。策略是 SA 创建的前提，它决定

一个密钥交换协商协议。

安全处理协议: IP 认证头协议(AH)、IP 封装安全载荷协议(ESP)。这两个协议可以独立使用，也可同时使用，前者提供数据完整性保护，后者可以提供数据私密性和完整性保护。两个协议都是将一个变长度的报文结构插入到 IP 头与上层协议之间。AH 协议使用事先协商好的算法和密钥计算整个报文不变部分的摘要值，然后作为报文完整性的证据保存在 AH 头结构中。ESP 协议则可以将原始报文加密后作为负载携带在报文中；此外 ESP 也可以实现数据完整性验证，但是与 AH 包括的字段不同。AH 和 ESP 头部共同包含的信息有 SPI 和序列号。SPI 用于标识特定的一对通信实体之间的安全关联(SA)。序列号在通信过程中维持单向递增，可以在通信实体之间提供数据抗重放服务。两个安全处理协议都支持两种模式：传输模式和隧道模式。前者在原始 IP 报文头与上层协议之间插入 AH 或 ESP 协议头，后者则是增加新 IP 头部而将原始 IP 数据包头部都作为负载。传输模式适于主机到主机方式报文的处理，后者适于转发设备上做封装处理的场景。

Internet 密钥交换协议(IKE)是一个相对独立的协议，以ISAKMP为语法表达基础。IKE最初被设计为一个通用的密钥协商协议，希望为Internet上任何需要加密和认证的通信协议提供算法和密钥协商服务。事实证明，IKE的扩展能力和严密安全的在线密钥协商能力正是IPsec得以广泛实施的关键。IKE 1.0 将交换过程分为两个阶段，两种方式：主模式和野蛮模式。第一阶段建立IKE自身的安全关联，协商IKE加密和认证算法与密钥；第二阶段为IPsec提供安全关联的算法和密钥协商。IKE 2.0 则简化了协商过程，在次协商中可直接产生IPsec的密钥。IKE可以在公有网络上协商出只有通信双方才掌握的秘密，这还要归功于DH交换——IKE中最核心的算法。通信双方可以通过使用各自生成的私有信息与相互交换的公开信息一起计算出共同秘密，而网络上的监听者却无法计算出此秘密。最后，IKE通过带外手段保证会话对方身份合法性，在协商的最后阶段通过认证确认前面交换的信息是完整可靠的。目前主要采用的认证方式是预共享密钥和PKI证书，扩展的认证方式为XAUTH或EAP。

IPsec的安全处理协议和IKE协议考虑都很完善，但是要工作起来还需要确定的加密算法套件和语法表达方式。这些附加的措施和算法都是在IPsec统一框架下定义的。

3.3 IPsec 基本工作原理

IPsec提供安全服务是基于SPD定义的策略规则，而这些规则是由管理员或应用程序添加的。数据包经过IPsec实体时有三种处理方式：IPsec保护、丢弃或旁路。协议做处理决策的判断依据被称为选择符(Selectors)，它包括数据包的IP和下一层头信息。SPD中的每个策略都有定义好的选择符，以保证策略得到正确使用。IPsec在IP层提供安全服务，通过选择不同的安全协议、密码算法和密钥实现。IPsec能够在一对主机间产生多个路径，也可以对两组主机群相互通信提供单个路径，这都是通过对选择符的设置来实施的。

图2所示的IPsec处理模型反应一个最高层次的概括。IPsec在网络或主机中创建了一个虚拟的受保护区与不受保护区的边界。数据流由受保护区穿越边界进入不受保护区时，需要IPsec为数据流提供安全服务(AH/ESP)。不需要安全服务的数据流在穿越边界时可以选择旁路(自由穿越，不受保护)或者丢弃。这两个区域是逻辑上的划分，实际主机或设备可有多个物理接口属于受保护或不受保护区域。由不受保护区进入受保护区的数据流向被称为入方向(inbound)，由受保护区进入不受保护区的数据流向被称为出方向(outbound)。

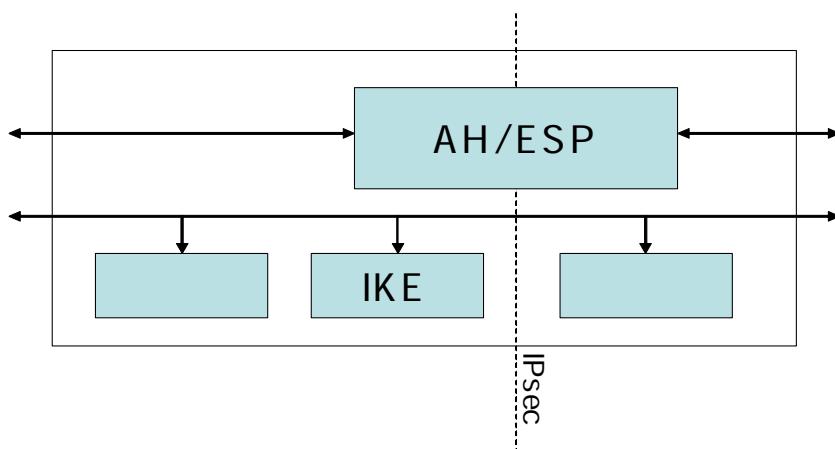


图2 IPsec 处理模型

3.4 IPsec 标准

因为 IPsec 是一系列技术与协议的集合，IETF 使用了一系列的 RFC 来定义 IPsec 的总体结构、服务和各协议标准。IPsec 最初的模型在 1995 年 RFC1825 至 1829 中被制定，1998 年 IPsec 框架被修订，IETF 发布了 RFC 编号从 2401 到 2412 对标准重新定义。2005 年底，RFC4301-4309 对大部分协议进行了再次更新。目前大部分设备厂商的实现主要是使用 1998 年标准，同时包括了新的加密算法支持、IKE 扩展功能和 NAT 穿越等特性。下表列出了 IPsec 的主要文档编号及 1998 年和 2005 年文档的对照关系。IPsec 最初开发时就意识到加密与认证算法可能随着时间推移而被破解，因此算法被设计成可以从核心协议中分离。这样的好处是，新发明算法时只要补充新文档即可，不需要对协议框架进行改动。

规范名称	1998 年文 档编号	2005 年 更新编号	描述
Security Architecture for the Internet Protocol	2401	4301	定义 IPsec 体系框架
IP Authentication Header	2402	4302	AH 协议
IP Encapsulating Security Payload	2406	4303	ESP 协议
The Internet IP Security Domain of Interpretation for ISAKMP	2407	4306	IPsec DOI 定义
Internet Security Association and Key Management Protocol	2408		ISAKMP 定义
The Internet Key Exchange	2409		IKE
The OAKLEY Key Determination Protocol	2412		OAKLEY 群

3.5 IPsec 实现的结构

RFC 标准定义了三种不同的 IPsec 结构：集成结构、BITS 结构和 BITW 结构。

集成结构：IPsec 协议被集成入 IP 协议内部提供所有安全特性，这种方式是最理想的情况，但是对于所有主机实现的 IP 协议栈要进行修改。在 IPv6 系统中，IPsec 已经作为一个基本部分，因此使用集成结构比较适宜。

BITS(Bump In The Stack)结构：IPsec 作为一个外部特性被插入到 IP 与链路层协议之间，

这种方式保持原有 IP 协议栈不需修改，而 IPsec 作为一个外挂的特性。这种方式的优点是，任何系统都可以以这种方式被翻新，而不需要对原有 IP 协议进行改动。缺点是，这种方式相对于集成结构会有冗余的数据报文分析和处理过程。

BITW(Bump In The Wire)结构：将 IPsec 的实现加入到传输路径之中，主机系统完全不需要额外的改动。传输路径上添加硬件设备负责对数据包的安全封装和认证，在到达目的时再以同样的设备还原为初始报文。这里的硬件设备可以是专用的安全网关或路由器。这种结构的优点很显然，可以在完全不支持 IPsec 的主机之间提供安全措施。缺点是实现复杂，成本相对昂贵。

BITS 和 BITW 从实现上看差异明显，但是技术上却非常相似，都是需要在 IP 协议之外分析数据报内容，并采取安全措施。只是其中一个被置于主机系统，另一个是在中间转发设备上实现。要实现 BITS 或 BITW 都需要主机或设备提供防火墙功能的最小子集，即报文过滤功能来区分不同特性的报文，并决定实施加解密、旁路透传或丢弃。而集成结构中，不需要通过以报文为单位的检查来获取这些信息，可以将 IPsec 的策略与 Socket 进行绑定，以减少协议处理的工作量。

IPsec 协议的发展

IPsec 协议自提出以来经历了长期的应用，在部署和实施中遇到的很多问题陆续被提交为补充建议。

4.1 NAT 穿越问题

NAT 穿越问题是 IPsec 遇到的一个比较复杂的问题。NAT 是为解决 IPv4 地址缺乏引入的一个临时解决措施，它在 Internet 发展中起到了很大的作用。然而，NAT 破坏了端到端对等的 IP 连接可能性，为很多 IP 应用带来麻烦。因为 NAT 修改 IP 首部的信息，会破坏 IP 完整性；同时，基于端口转换的 NAT 设备无法识别 AH 和 ESP 协议，也给

IPsec 带来灾难。IPsec 使用外层 UDP 封装 AH / ESP 来穿透 NAT 设备，同时修改 IKE 对 NAT 进行探测，同时增加保活消息避免 NAT 转换表超时。

4.2 IPsec 的简化配置问题

作为一个安全协议，IPsec 的复杂规则导致了其应用上的限制。通常要维护一个 IPsec 连接需要用户添加大量配置才能实现。在 IPsec 发展过程中各实现厂商提出了很多优化方式来提高 IPsec 的易用性。

4.3 IPsec 的热备份问题

随着网络规模的扩大，IPsec 和其他 VPN 协议一样需要解决高可用性的问题。传统的 IPsec 将协议相关的复杂信息保存在单个设备上，一旦设备失效必然造成连接中断。用户需要中心节点提供热备份能力，在主设备发生故障时可以倒换到另外一台设备而不中断网络连接。目前，还没有统一的标准来解决这个问题。

2005 年 12 月，随着 IETF 发布一系列更新 RFC，IPsec 工作组活动宣布终止。作为一个通用的，可提供数据完整性和机密性服务的安全协议，IPsec 已经被纳入 IPv6 基本协议范畴，其地位无可替代。似乎一切都已经完美了。事实上，IPsec 新框架和 IKE 2.0 并没有得到普遍实现，很多需求和限制尚未非常明晰。然而，部署灵活性、高可靠性和高强度加密 / 认证算法一直是比较活跃的话题，至少在一段时间内仍然是 IPsec 的主要发展方向。



IPsec 细节及报文封装

◆◇□ 刘倩

IPsec 的协议组成

AH 和 ESP 是 IPsec 的两个主要协议。Authentication Header (AH) 协议为 IP 通信提供数据源认证、数据完整性和反重播保证，它能保护通信免受篡改，但不能防止窃听，适合用于传输非机密数据。Encapsulating Security Payload (ESP) 为 IP 数据包提供完整性检查、认证和加密。

1.1 AH

IP Authentication Header (AH) 为 IP 报提供无连接完整性和数据源认证，并且提供重放 (replay) 攻击保护。AH 不为数据包提供数据机密性 (加密)，因此，适合用于传输非机密数据。RFC2402 中详细描述了 AH。

AH 的工作原理是在每一个数据包上添加一个身份验证报头。AH 通过将一个带有密钥的单向散列函数应用到数据包来生成一个散列或消息摘要的方法实现认证。接收方收到带有散列值的数据包后，执行同样的散列函数并和发送方提供的摘要值相比较，在传输过程中对数据的任何更改将致使散列无效，这样就提供了完整性保护。AH 协议目前支持 MD5 和 SHA1 算法，这两种算法的密钥长度分别是 128 bit 和 160 bit。

AH 可以单独使用，也可以与 ESP 协议结合使用。这样的安全服务可以在两个通信的主机之间、两个通信安全网关之间，也可以在安全网关和主机之间使用。

AH 报头位置在 IP 报头和传输层协议报头之间。AH 在 IPv4、IPv6、或扩展 IPv6 的 IP 报头中的协议号为 51，该值包含在 AH 报头之前的协议报头中，如 IP 报头。

1.1.1 AH 报文头格式

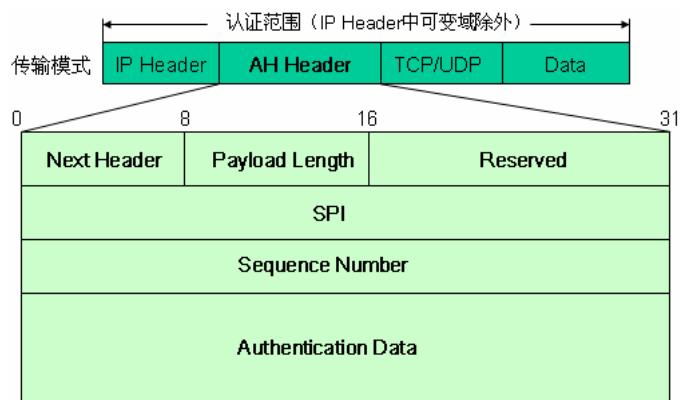


图 1 AH 报头格式

Next Header: 8bit，描述在 AH 认证头的下一个载荷的类型。例如，值为 6 代表 TCP。

Payload Length: 8bit，AH 报头长度，以 32-bit 为单位计算结果再减 2。典型值为 4。

Reserved: 16bit，用于以后的扩展。发送方必须置该位为 0，并且接收者应该忽略该字段。

SPI (Security Parameters Index 安全参数索引): 32bit，接收方根据报文中的 SPI 查找 SA 数据库，找到对应的 SA。SPI 值 0 被保留来表明“没有安全关联存在”。

Sequence Number (序列号): 从 1 开始的 32 位单增序列号，不允许重复，唯一地标识了每一个发送数据包，为安全关联提供反重播保护。接收端校验序列号为该字段值的数据包是否已经被接收过，若校验为已接受过，则拒收该数据包。

Authentication Data (AD, 认证数据): 是一个长度可变的域，包含完整性检查值 (ICV)。长度为 32bit 的整数倍，通常为 96bit。接收端接收数据包后，首先执行带密钥的 hash 计算，再与发送端所计算的该字段值比较，若两者相等，表示数据完整，若在传输过程中数据遭修改，两个

计算结果不一致，则丢弃该数据包。

1.1.2 AH 提供数据包完整性检查

在传输模式下，AH 报头插在 IP 报头之后，TCP、UDP 或者 ICMP 等上层协议报头之前。在隧道模式下，AH 头被插入到新 IP 头与原始报文之间。一般 AH 为整个数据包提供完整性检查，因外层 IP 报头（传输模式是原 IP 头，隧道模式是新 IP 头）中包含的“生存期（Time To Live）”、“服务类型（Type of Service）”等值在报文传输过程中可能发生变化，在进行完整性检查值计算时应将这些值可变字段用 0 值替代。



在使用 AH 协议时，AH 协议首先在原数据前生成一个 AH 报文头，报文头中包括一个递增的序列号（Sequence number）与验证字段（空）、安全参数索引（SPI）等。AH 协议将对新的数据包进行 hash 运算，生成一个验证字段（authentication data），填入 AH 头的验证字段。AH 协议使用 32 比特序列号结合防重放窗口（一般为 64）和报文验证来防御重放攻击。当收到了一个经过认证的数据以后，防重放窗口会滑动一次，如果该数据报文被重放，由于其顺序号码和原来的相同，因此这个数据会落到窗口之外，数据就会被丢弃。

1.2 ESP

封装安全有效载荷头在 IPv4 和 IPv6 中提供一种混合的安全服务。ESP 提供机密性、数据源验证、无连接的完整性、防重放服务等。RFC 2406 中详细描述了 ESP。

ESP 通过加密有效载荷来提供数据机密性，它

支持多种对称加密算法。一般 ESP 不对整个数据包加密，而是只加密 IP 包的有效载荷部分，不包括 IP 头。但在端对端的隧道通信中，ESP 需要对整个原始 IP 数据包进行加密。

ESP 在 IPv4、IPv6、或扩展 IPv6 的 IP 报头中的协议号为 50。

1.2.1 ESP 报文封装模式

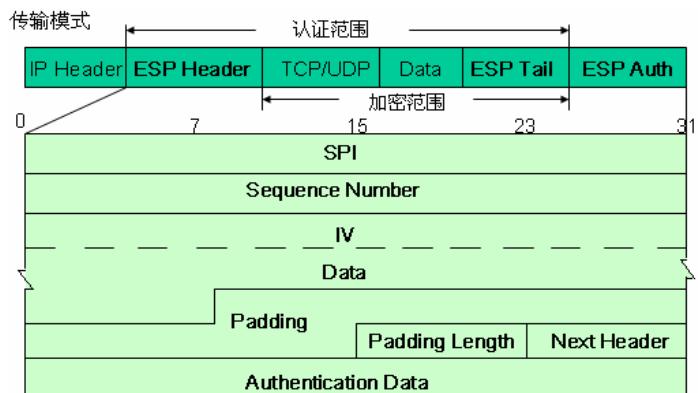


图 3 ESP 报文头、尾结构

ESP 报头字段包括：

Security Parameters Index (SPI, 安全参数索引): 32 bit, 为数据包识别安全关联。

Sequence Number (序列号): 32bit, 从 1 开始的 32 位单增序列号，不允许重复，唯一地标识了每一个发送数据包，为安全关联提供反重播保护。接收端校验序列号为该字段值的数据包是否已经被接收过，若是，则拒收该数据包。

IV (初始化向量): 某些加密算法需要使用 IV，这种特殊加密算法必须定义 IV 的位置，IV 是受保护数据的第一个 8 位组。对 IV 进行验证但不进行加密。

ESP 报尾字段包括：

Padding (填充项): 0-255 个字节。可以使用填充字段填充明文，以达到采用的加密算法要求的长度。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是 4 字节的整数倍。填充字段还可以用于隐藏有效载荷实际长度，支持

支持信息流机密性。

Padding Length (填充项长度): 接收端根据该字段长度去除数据中扩展位。

Next Header (下一个报头): 如果在通道模式下使用 ESP, 这个值就会是 4, 表示 IP-in-IP。如果在传送模式下使用 ESP, 这个值表示的就是它背后的上一级协议的类型, 比如 TCP 对应的就是 6。

ESP Auth (验证数据): 验证数据字段用于容纳数据完整性的检验结果 --- 通常是一个经过密钥处理的散列函数输出值。

这里需要注意 ESP Auth 属于 ESP 报尾字段。而验证数据是不能被加密的, 所以图 3 中示意将 ESP Auth 独立于 ESP Tail, 是为了对加密与非加密部分进行区分。

1.2.2 ESP 提供加密及完整性检查

传输模式, ESP 报头位于 IP 报头和传输层协议报头之间, 并在数据后面增加 ESP 尾。

隧道模式, ESP 报头位于新 IP 头和初始报文之间, 并在数据后面增加 ESP 尾。

认证:

ESP 认证报文的完整性检查部分包括 ESP 报头、传输层协议报头, 应用数据和 ESP 报尾, 但

不包括 IP 报头, 因此 ESP 不能保证 IP 报头不被篡改。

加密:

ESP 加密部分包括上层传输协议信息、数据和 ESP 报尾。

当 ESP 认证和加密都被选择了的时候, 由于验证总是在加密后进行的, 因此收到分组后先检查数据的有效性, 然后再解密。

1.3 AH 与 ESP 的比较

从功能上, AH 只提供认证, 不提供加密功能。ESP 可以提供认证和加密。

从认证强度, AH 认证强于 ESP, AH 可以对报头和有效载荷进行认证, ESP 只对有效载荷进行认证。AH 和 ESP 在完整性校验的主要区别在于覆盖的长度上, ESP 不对 IP 头进行认证, 除非 IP 头被封装在 ESP 内部 (例如, 隧道模式)。

1.4 AH 与 ESP 的组合使用

由于 AH 需要对整个 IP 数据包进行认证, 如果先采用 AH 再使用 ESP, ESP 的头和尾会改变数据包的长度, 另外, ESP 的填充字段也会改变数据包的长度, 造成 AH 认证的失败。因此 AH 和 ESP 同时使用时首先应该应用 ESP。AH 对整个报文进行验证。ESP 只对 IP 层以上信息进行验证。

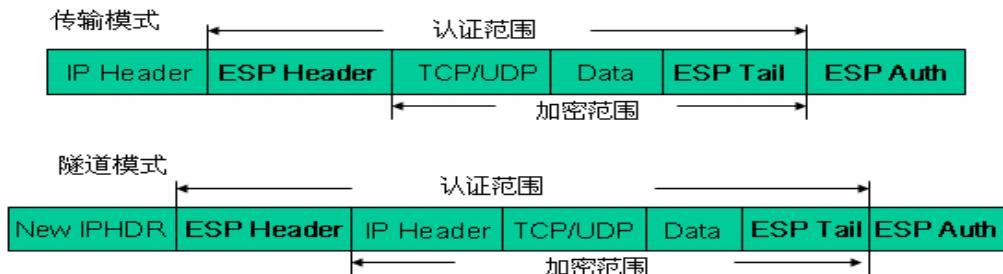


图 4 ESP 的加密部分和完整性检查部分

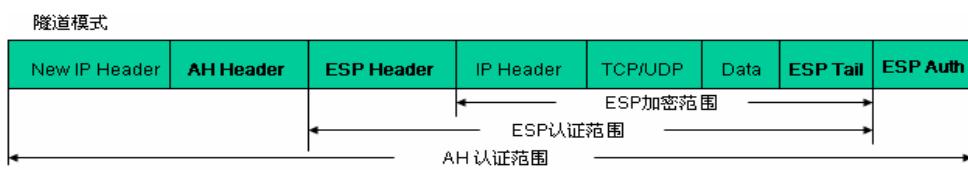


图 5 AH 与 ESP 组合使用实例

IPsec 的使用模式

上文涉及到了传输模式和隧道模式，这两种模式就是将ESP 和AH 应用到IP 数据包中的两种不同的方法。

2.1 IPsec 传输模式

在传输模式下，在IP 报头和高层协议报头之间插入一个IPsec 报头（AH 或ESP）。在这种模式下，IP 报头与原始IP 分组中的IP 报头相同，只是IP 协议字段被改为AH（51）或ESP（50），并重新计算IP 报头校验和。

传输模式保护数据包的有效负载、高层协议，IPsec 源端点不会修改IP 报头中的目标IP 地址，原来的IP 地址保持明文。传输模式只为高层协议提供安全。这种模式常应用在需要保护的两台主机之间的端到端的连接，而不是有多台主机的两个网关之间的数据流。

传输模式中的AH：

通过将原来的IP 头移到左边，并插入AH 头来实现传输模式。并对整个数据包进行散列运算来提供认证。这些新的数据包不支持NAT，因为在包头改变IP 地址会导致认证失败。可以在IPsec AH 封装前使用NAT，而不能在封装之后。

传输模式中的ESP：

通过将原来的IP 头移到左边，并在数据字段前插入ESP 头，在数据端后插入ESP 尾，并在新的数据包的末尾添加一些控制信息来实现ESP 传输模式。如果要求加密，则对数据和ESP 尾进行加密。认证使用新的数据包中的ESP 报头、数据部分和ESP 报尾，而不使用IP 头，允许在这些数据包上使用NAT。

2.2 IPsec 隧道模式

与传输模式不同，在隧道模式下，原始IP 分组被封装成一个新的IP 数据报，并在内部报头和外部报头之间插入一个IPsec 报头（AH 或ESP），原IP 地址被当作有效载荷的一部分受到IPsec 的安全保护，另外，通过对数据加密，还可以将数据包目的地址隐藏起来，这样更有助于保护端对端隧道通信中数据的安全性。隧道模式提供原始整个IP 数据包的安全性。

AH 隧道模式为整个数据包提供完整性检查和认证，认证功能优于ESP。但在隧道技术中，AH 协议很少单独实现，通常与ESP 协议组合使用。

ESP 隧道模式中完整性检查、认证部分和加密部分分别如图4 所示。ESP 的认证不包括新IP 头。

网络之路

IPsec 安全算法介绍

◆◇□ 毛 昱

IPsec (IP Security) 是 Internet 工程任务组 (IETF) 制定的一个开放的 IP 层安全框架协议。IPsec 协议不是一个单独的协议，它给出了应用于 IP 层上网络数据安全的一整套体系结构，包括网络认证协议 Authentication Header (AH)、封装安全载荷协议 Encapsulating Security Payload (ESP) 和密钥管理协议 Internet Key Exchange (IKE) 等。IPsec 规定了如何在对等层之间选择安全协议、确定安全算法和密钥交换，向上提供了访问控制、数据源认证、数据加密等网络安全服务。IPsec 能够为传输敏感数据提供安全保护，对参与 IPsec 的设备之间传输的 IP 数据包进行保护和认证。IPsec 协议涵盖了几乎所有基本安全算法，广泛的应用了密码学的基本原理和基本特性。

这篇文章将从密码学的基本分类讲起，重点介绍 IPsec 协议所使用的加密算法、认证算法以及密钥协商算法。

密码学介绍

密码学的英文单词 **cryptography** 来源于词根 **crypto** (隐藏或秘密) 以及 **graphy** (写)，所以密码学是一门用秘密方式写东西的艺术。一般来讲，人们通常认为密码学是一种将信息表述为不可读的方式，并且有一种秘密方法可以将信息恢复出来，密码学提供的最基本的服务就是使得通信者能够互相发送消息，并且避免其它人员读取消息的内容。对数据进行加密，其加密或解密变换是由密钥控制实现的。密钥 (Keyword) 由用户按照一种密码体制随机选取，它通常是一随机字符串，是控制明文和密文变换的唯一参数。

根据密钥类型不同将现代密码技术分为两类一类是对称加密（共享密钥加密）系统，另一类是不对称加密（公开密钥加密）系统。

1.1 对称加密系统

对称加密算法 (symmetric algorithm) 的加密密钥和解密密钥是相同的，它要求发送者和接收者在安全通信之前，商定一个共享密钥。对称算法的安全性依赖于密钥，泄漏密钥就意味着任何人都能对消息进行加 / 解密，密钥必须特殊保管。对称密钥算法的优点是加密强度高，计算开销小，处理速度快；缺点是密钥管理困难，在多方通信时因为需要保存很多密钥而变得复杂，为了保证较高的安全性，一般密钥都有有效期，需要进行不断地更换，而更换的密钥在传递过程中又可能造成泄漏。对称密钥算法实现速度极快，从各种对称算法的测试结果看，软件实现的速度都达到了每秒数兆或数十兆比特。

对称算法的加 / 解密示意图如下：

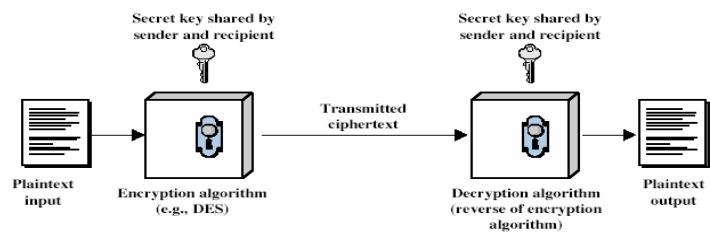


图 1.1：对称加密算法

对称加密系统最大的问题是密钥的分发和管理非常复杂、代价高昂。比如对于具有 n 个用户的网络，需要 $n(n - 1)/2$ 个密钥，在用户群不是很大的情况下，对称加密系统是有效的。但是对于大型网络，当用户群很大，分布很广时，密钥的分配和保存就成了大问题。对称加密算法另一个缺点是不能实现数字签名。

对称密钥算法可分为两类：流加密算法和块加密算法。流加密算法一次只对明文中的单个位（或单个字节）进行运算。块加密算法是对明文的一组位进行运算，这些位称为数据块。现代计算机密码算法的典型数据块的长度为 64 位——这个位数大到足以防止分析破译，但又小到足以方便使用。典型的对称密钥算法有 DES、IDEA、3DES 以及 AES 等。

1.2 不对称加密系统

不对称加密的一个通俗叫法是公开密钥加密 (public-key algorithm), 1976 年 Diffie 和 Hellman 提出了公钥密码学的思想, 给现代密码学带来了历史性的革命。公钥密码学的概念是为了解决传统密码学中最困难的两个问题: 密钥分配问题和数字签名问题。对称密码进行密钥分配的时候要求 (1) 通信双方已经共享一个密钥, 而且该密钥已经通过某种方法分配给通信双方, 或者 (2) 利用密钥分配中心。这个过程有悖于密码学的精髓, 即在通信中完全保持保密性。公钥密码学的设计者 Diffie 在谈到对称密码学时说道, “如果必须要求用户之间共享他们的密钥, 这些密钥可能因为盗窃或者索取而被泄露, 那么设计不可靠的密码体制究竟还有什么意义? ”。

他们是可读信息或者是数据。(2) 加密算法: 加密算法对明文进行各种转换。(3) 公钥和私钥: 算法的输入。这对密钥中一个用于加密, 一个用于解密。加密算法执行的变换依赖于公钥和私钥。(4) 密文: 算法的输出。它依赖于明文和密钥, 对于给定的消息, 不同的密钥产生的密文不同。(5) 解密算法: 该算法接收密文和相应的密钥, 并产生原始的明文。

公开密钥算法的加 / 解密示意图 1.2 如下: 公开密钥密码体系的应用可以分为三类: (1) 加密和解密。(2) 数字签名。(3) 密钥交换。在实际应用中, 公开密钥加密系统并没有完全取代对称密钥加密系统, 这是因为公开密钥加密系统是基于尖端的数学难题, 计算非常复杂, 它的安全性更高, 但它实现速度却远赶不上对

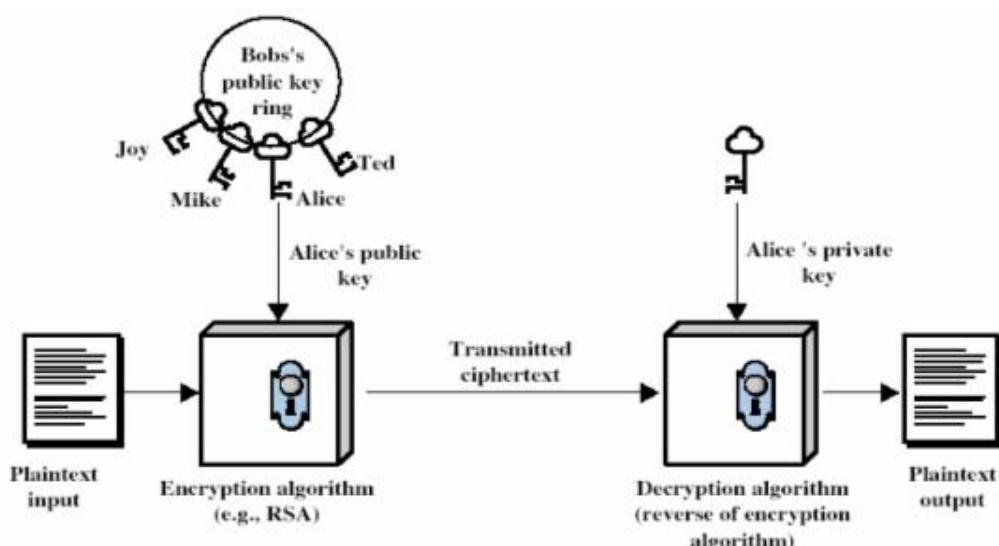


图 1.2 公开密钥加密算法

公开密钥算法依赖于一个加密密钥和一个与之相关的不同的解密密钥, 这些算法具有下述的重要特点: (1) 用作的加密的密钥不同于解密的密钥。仅根据密钥算法和加密密钥来确定解密密钥在计算上是不可行的。(2) 两个密钥中的任何一个都可以用来加密, 另一个用来解密。公钥密码体制由以下组成部分: (1) 明文: 算法的输入。

称密钥加密系统。在实际应用中可利用二者的各自优点, 采用对称加密系统加密文件, 采用公开密钥加密系统加密“加密文件”的密钥 (会话密钥), 这就是混合加密系统, 它较好地解决了运算速度问题和密钥分配管理问题。因此, 公钥密码体制通常被用来加密关键性的、核心的机密数据, 而对称密码体制通常被用来

加密关键性的、核心的机密数据，而对称密码体制通常被用来加密大量的数据。

公开密钥算法是现代密码学的核心，它的优点是便于密钥管理、分发、可以实现数字签名和数字鉴别；缺点是计算开销大，处理速度慢。在实际的应用中，对称密钥算法与公开密钥算法经常结合使用。自公开密钥加密算法问世以来，学者们提出了许多种公开密钥加密方法，它们的安全性都是基于复杂的数学难题。根据所基于的数学难题来分类，有以下三类系统目前被认为是安全和有效的：大整数因子分解系统（代表性的有RSA）、椭圆曲线离散对数系统（ECC）和离散对数系统（代表性的有DSA）。当前最著名、应用最广泛的公开密钥加密系统RSA是由Rivest、Shamir、Adleman提出的（简称为RSA系统），它的安全性是基于大整数素因子分解的困难性，而大整数因子分解问题是数学上的著名难题，至今没有有效的方法予以解决，因此可以确保RSA算法的安全性。RSA系统是公开密钥加密系统的最具有典型意义的方法，大多数使用公开密钥算法进行加密和数字签名的产品和标准使用的都是RSA算法。

2 加密算法

由于对称加密算法效率很高，因此成为网络传送大量数据中最常用的一类加密方法。IPsec协议在数据转发中广泛使用了DES，3DES和AES等对称加密算法。

2.1 DES (Data Encryption Standard) 数据加密标准算法

2.1.1 DES 算法介绍

美国国家标准局1973年开始研究除国防部外的其它部门的计算机系统的数据加密标准，于1973年5月15日和1974年8月27日先后两次向公众发出了征求加密算法的公告。加密算法要达到的目的（通常称为DES密码算法要求）主要为

以下四点：提供高质量的数据保护，防止数据未经授权的泄露和未被察觉的修改；具有相当高的复杂性，使得破译的开销超过可能获得的利益，同时又要便于理解和掌握；DES密码体制的安全性应该不依赖于算法的保密，其安全性仅以加密密钥的保密为基础；实现经济，运行有效，并且适用于多种完全不同的应用。1977年1月，美国政府颁布：采纳IBM公司设计的方案作为非机密数据的正式数据加密标准（DES：Data Encryption Standard）。

1.1.2 DES 算法分析

DES算法的入口参数有三个：Key、Data、Mode。其中Key为8个字节共64位，是DES算法的工作密钥；Data也为8个字节64位，是要被加密或被解密的数据；Mode为DES的工作方式，有两种：加密或解密。DES算法是这样工作的：如Mode为加密，则用Key去把数据Data进行加密，生成Data的密码形式（64位）作为DES的输出结果；如Mode为解密，则用Key去把密码形式的数据Data解密，还原为Data的明码形式（64位）作为DES的输出结果。在通信网络的两端，双方约定一致的Key，在通信的源点用Key对核心数据进行DES加密，然后以密码形式在公共通信网中传输到通信网络的终点，数据到达目的地后，用同样的Key对密码数据进行解密，便再现了明码形式的核心数据。这样，便保证了核心数据在公共通信网中传输的安全性和可靠性。

DES算法把64位的明文输入块变为64位的密文输出块，它所使用的密钥也是64位，整个算法的主流程如2.1图：

- 首先对64比特的输入进行一次初始置换。

其功能是把输入的64位数据块按位重新组合，并把输出分为L0、R0两部分，每部分各长32位，其初始置换规则见下表：

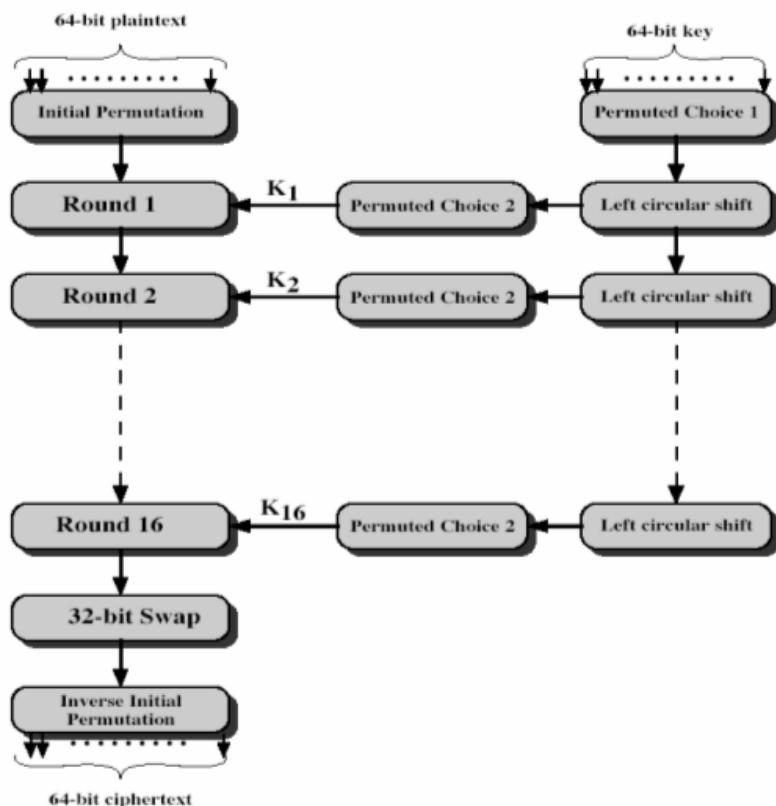


图 2.1: DES 算法流程

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

表 2.1 : DES 算法的初始置换 (IP)

即将输入的第 58 位换到第一位, 第 50 位换到第 2 位, . . . , 依此类推, 最后一位是原来的第 7 位。L₀、R₀ 则是换位输出后的两部分, L₀ 是输出的左 32 位, R₀ 是右 32 位, 例: 设置换前的输入值为 D₁D₂D₃. D₆₄, 则经过初始置换后的结果为: L₀=D₅₈D₅₀...D₈; R₀=D₅₇D₄₉...D₇。

b) 子密钥生成。

初始密钥为 64 比特, 但 DES 算法规定, 其中

第 8、16、. 64 位是奇偶校验位, 不参与 DES 运算。故初始密钥实际可用位数便只有 56 比特。56 比特的密钥用于为 16 轮运算分别产生一个 48 比特的子密钥, 称为 K₁, K₂, K₃ . . . K₁₆。子密钥 K_i (i=1..16) 的生成如下

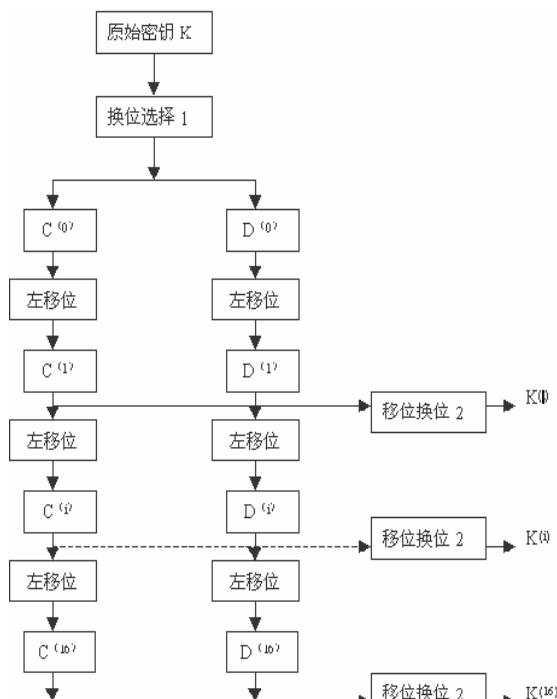


图 2.2: 子密钥的生成

(a) 密钥的初始置换

把密钥每个字节的最后一一位扔掉, 然后对 56 个有效的比特做置换 (置换不是随机的, 按照如下规律置换), 得到 56 比特的输出。把这 56 比特的输出分为左 28 位和右 28 位, 分别称为 C₀ 和 D₀。如下所示, 这样得到密钥的初始置换:

C₀

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36

D0

63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

表 2.2 置换选择 1 (PC-1)

(2)、循环左移

对 C0 和 D0 进行左移 (从最左边移出的比特又从最右边移入)，左移的位数由第几轮来决定。在第一轮，第二轮，第九轮和第十六轮进行一个比特的循环左移，其他轮左移两个比特。将第 i 轮的 C0 和 D0 称为 Ci 和 Di，上一轮产生的 Ci 和 Di 供下轮左移使用。

(3)、Ci 和 Di 缩小换位产生 Ki

用于产生 Ki 左半部分的 Ci 置换如下，需要注意，第 9 比特，第 18 比特，第 22 比特和第 25 比特被丢弃。这样 28 比特的 Ci 经过缩小换位成为了 24 比特。置换关系如下：

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2

表 2.3：置换选择 2 (PC-2) Ci

用于产生 Ki 右半部分的 Di 置换如下，Di 移位的结果记为第 29 比特，第 30 比特，...，第 56 比特。其中第 35 比特，第 38 比特，第 43 比特和第 54 比特被丢弃。置换关系如下：

41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

表 2.4：置换选择 2 (PC-2) Di

移位后的 Ci 和 Di 构成了 Ki，由于 Ki 的每一半都是 24 比特，所以 Ki 的长度为 48 比特。

c) 加密内容的处理。

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

表 2.5：扩充置换

将这 32 位变成 48 位，然后和刚刚生成的子密钥 Ki 进行一次异或运算，经过 S-box 这样一个选择运算将这 48 位变为 32 位。再用一个 p 置换函数产生 32 位的输出。如下图所示：

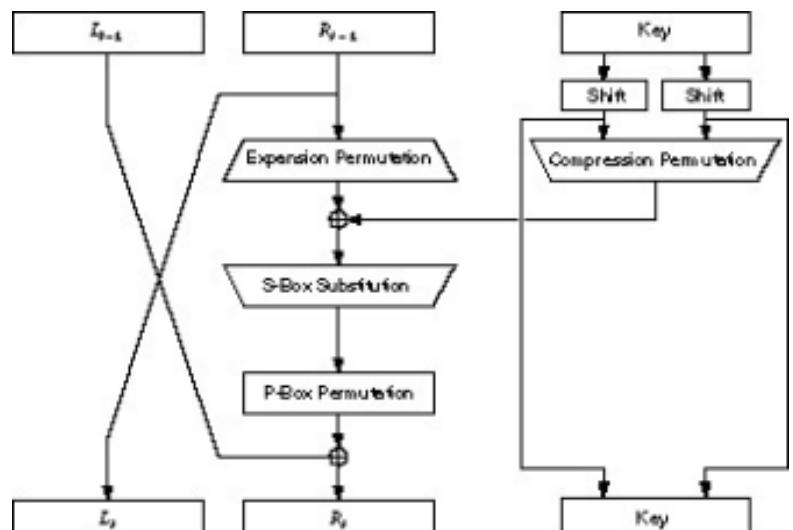


图 2.3：DES 算法一轮迭代的过程

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

表 2.6：置换函数 (P)

d) S 盒的操作

S-box 的过程如下： 将输入的 48 位分为 8 组，每组对应一个 S-box 表，每个表里有 4 行 16 列。这样每组有 6 位，取 2、3、4、5 位为列数，1、6 位为行数，在所对应的表中寻找这个数字，将找到的数字转换为 4 位二进制数输出。这样一共 8 组，就将 48 位转换为 32 位了。如下图所示：

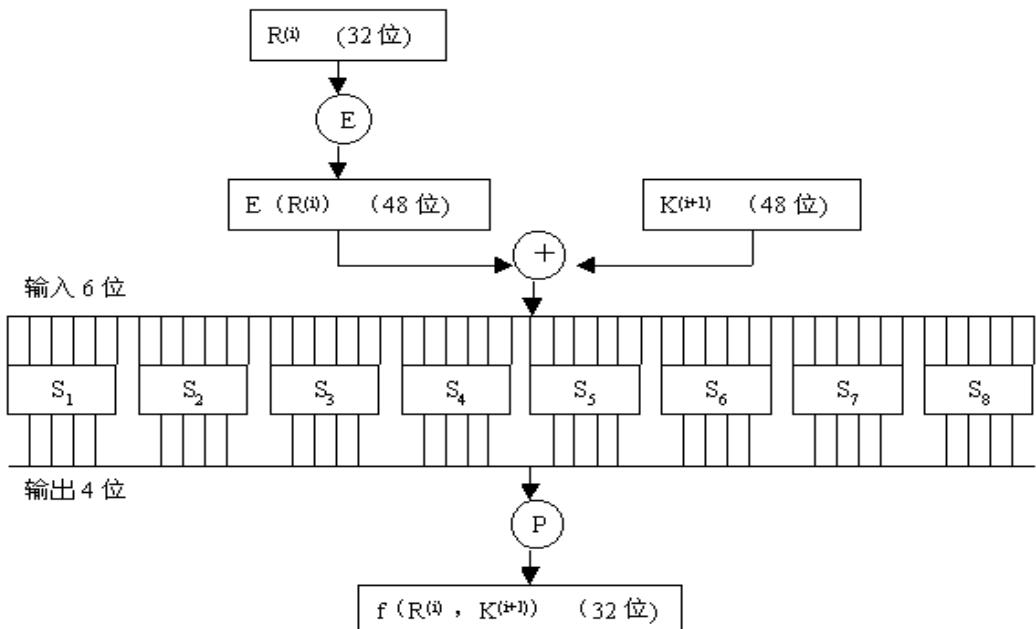


图 2.4: $F(R, K)$ 的计算

S1:

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S2:

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S3:

10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S4:

7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S5:

7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S6:

2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S7:

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S8:

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

表 2.7 DES 的 S 盒定义

c) 经过 16 轮运算后，得到 L16、R16，将此作为输入，进行逆置换，即得到密文输出。

逆置换正好是初始置的逆运算，例如，第 1 位经过初始置换后，处于第 40 位，而通过逆置换，又将第 40 位换回到第 1 位，其逆置换规则如下表所示：

40	8 48	16	56	24	64	32
39	7 47	15	55	23	63	31
38	6 46	14	54	22	62	30
37	5 45	13	53	21	61	29
36	4 44	12	52	20	60	28
35	3 43	11	51	19	59	27
34	2 42	10	50	18	58	26
33	1 41	9	49	17	57	25

表 2.8：逆初始置换 (IP⁻¹)

由上可以看出似乎DES算法非常简单，给人的感觉是任何人都可以设计出类似的密钥加密算法，只要把几个比特移动来移动去就可以得到一个算法。实际上，从密码学产生至今，几乎所有的对称密码学都是基于替换和置换这些初等算法。与之相比较的是，公开密钥算法基于数学函数而不是替换和置换。

2.2 3DES (Triple DES) 算法。

2.2.1 3DES 算法介绍

用DES算法实施多次加密以使得DES算法更为安全的常见方法为EDE (encrypt-decrypt-encrypt) 或3DES。事实上，任何一种加密算法都可以通过多次施加加密以提高安全性，但是它对于DES算法来说更为重要，因为DES的密钥太短了。

2.2.2 3DES 算法分析

3DES算法可以描述如下：设 $e_k(x)$ 和 $d_k(x)$ 表示用DES算法对64位的位串的加密和解密，密钥为K；则64位的密文c是通过执行下面的运算得到的：

$$c = e_{K_3}(d_{K_2}(e_{K_1}(x)))$$

其中 K_1, K_2, K_3 是56位的DES密钥。从密文c导出明文x的3DES的解密过程是加密过程的反过程，其描述如下：

$$x = d_{K_1}(d_{K_2}(d_{K_3}(c)))$$

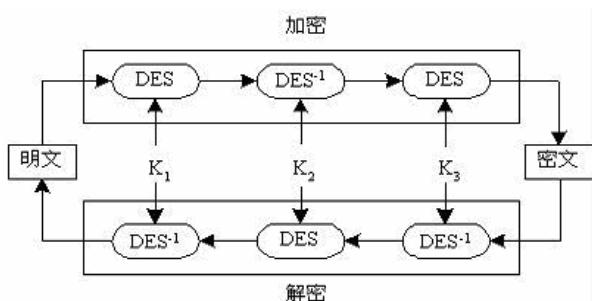


图 2.6: 3DES 算法结构

为了获得更高的安全性，三个密钥应该是互不相同的。这样，本质上就相当于用一个长为168位的密钥进行加密。多年来，它在对付强力攻击时是比较安全的。对安全性需要不那么高的数据， K_1 可以等于 K_3 。在这种情况下，密钥的有效长度为112位。在通常使用的所有64位的分组密码中，3DES是最安全的；但是，如果用软件来实现，它也是这些分组密码中最慢的。

2.3 分组密码的工作模式

DES算法是提供数据安全的基本构件。为了将DES应用于实际，人们定义了四种工作模式。这四种模式实际上覆盖了所有使用DES的应用程序。由于新的应用和要求已经出现，在特别公布的800-38A中NIST已经将推荐模式扩展为5个。这些模式可以用于包括3DES和AES在内的任何分组密码。表2.9对这些模型做了一个总结。

模式	描述	典型应用
电码本 (ECB)	用相同的密钥分别对明文组加密	单个数据的加密传输 (一个加密密钥)
密码分组链接 (CBC)	加密算法的输入是上一个密文组和下一个明文组的异或	普通目的的面向分组的传输 认证
密码反馈 (CFB)	一次处理J位。上一个分组密文作为产生一个伪随机数输出的加密算法的输入，该输出与明文异或，作为下一个分组的输入	普通目的的面向分组的传输 认证
输出反馈 (OFB)	与CFB基本相同，只是加密算法的输入是上一次DES的输出	噪声通道上的数据流的传输（如卫星通信）
计数器 (CTR)	每个明文组是与加密的计数器的异或。对于每个后续的组，计数累加的	普通目的的面向分组的传输 用于高速需求器

表 2.9.: DES 的工作模式

按照 RFC 2405 的规定，ESP 协议采用 CBC 方式对数据报文进行加密操作。下面详细介绍一下 C B C 密码分组链接模式。由于 C B C 模式能够克服 E C B 模式的缺陷，首先介绍 E C B 模式。

2.3.1 ECB (Electronic Codebook Book) 电码本模式

分组密码工作模式中最简单的是电码本模式。它一次处理 64 位明文（假设为 D E S 算法，如果为 A E S，则为 128 位），每次使用相同的密钥加密。使用电码本这个词是因为给定的密钥，任何 64 位的明文组只有唯一的密文与之对应。所以可以想象存在很厚的一个密码本，根据任意 64 位明文都可以查到相应的密文。

明文若长于 64 位，则可以简单的将其分为 64 位分组，必要时可对最后一个分组进行填充。解密也是一次执行一个分组，且使用相同的密钥。

使用 E C B 模式特别适合数据较少的情况，如果

果说加密密钥。因此若想安全的传输一个 D E S 密钥，选择这种方式是合适的。E C B 最重要的特征是一段消息中若有几个相同的明文组，那个密文就会出现几个相同的片断。

基于以上特征，我们可以发现对于很长的消息，E C B 模式可能是不安全的。如果消息是非常结构化的，比如说 I P 报文，密码分析者可以利用其特征结构来破译。

2.3.2 CBC (Cipher Block Chaining) 密码分组链接模式

为了克服 E C B 的弱点，我们需要将重复的明文组加密成不同的密文组。C B C 模式能满足这个要求。这个模式下加密算法的输入是当前的明文组和上一个密文组的异或，而使用的密钥是相同的。加密算法的每次输入与本明文组没有固定的关系。因此，若有重复的 64 位明文组，加密后就看不来了。

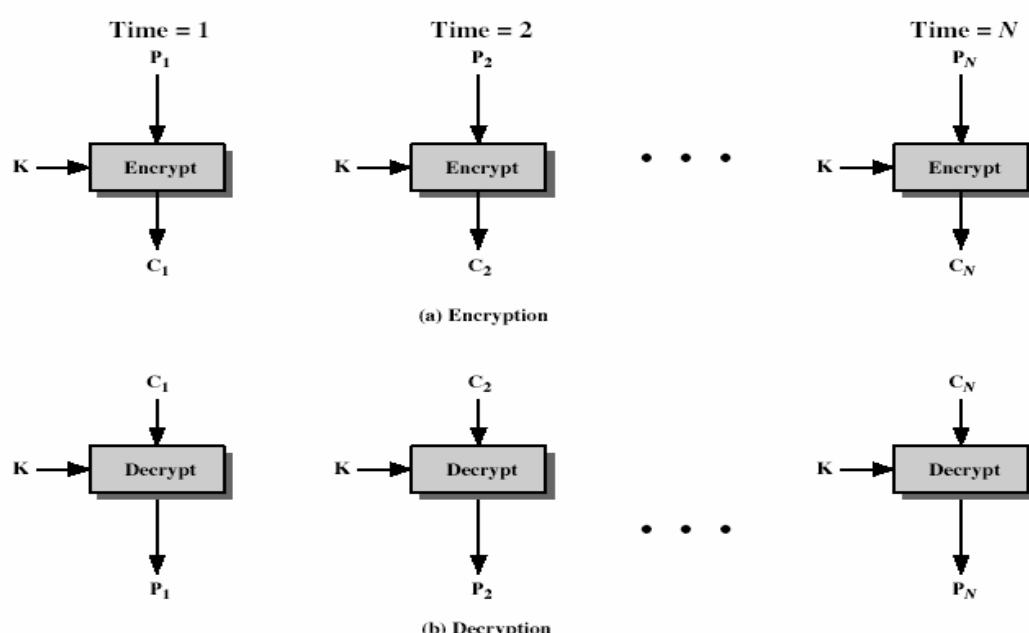


图 2.7：电码本模式

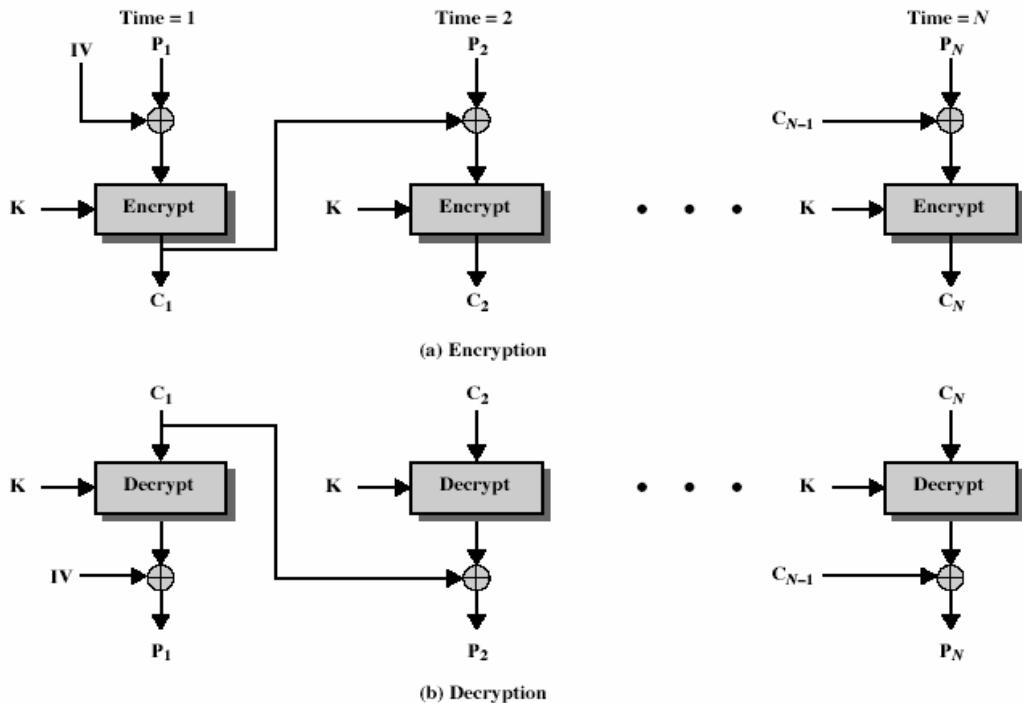


图 2.8 : 密码分组链接模式

第一块明文可以与一个初始向量(IV) 异或后
再加密。解密时将第一块密文解密的结果与 IV 异
或而恢复出第一块明文。 IV 必须为收发方共享。
为了增加安全性, IV 应该和密钥一样加以保护。
比如说用 ECB 加密来保护 IV。

生认证符的函数算法, 认证符是一个用来认
证消息的值; 上层协议中将函数算法作为原语使
接受方可以验证消息的真实性。对于消息认证算
法, 可以分为如下三类: 消息加密, 消息认证码
(MAC) 和 hash 函数(即单向散列函数)。IPSec 协
议使用了 hash 函数和 MAC 的结合体 HMAC 对报文
进行认证, 在 RFC 2403 (HMAC-MD5) 和 RFC 2404
(HMAC-SHA1) 进行了详细定义。

认证算法

在网络通信环境中, 可能存在伪装, 内容修
改, 顺序修改, 计时修改等攻击, 我们可以采用

消息认证或数字签名的方式来防范此类攻击。消
息认证就是验证所收到的消息确实是来自于真正
的发送方且未被修改的消息, 它也可以验证消息
的顺序和及时性。同样, 数字签名也可以达到相
同的功能, 这里不再赘述。任何消息认证或数字
签名的方法在功能上可以看做两层。下层中一定
有某种产生认证符的函数算法, 认证符是一个用
来认证消息的值; 上层协议中将函数算法作为原
语使接受方可以验证消息的真实性。对于消息认
证算法, 可以分为如下三类: 消息加密, 消息认
证码 (MAC) 和 hash 函数(即单向散列函数)。IPSec
协议使用了 hash 函数和 MAC 的结合体 HMAC 对报
文进行认证, 在 RFC 2403 (HMAC-MD5) 和 RFC 2404
(HMAC-SHA1) 进行了详细定义。

3.1 HASH 算法

单向散列函数 (one-way hash function) 是
在一个方向上工作的散列函数, 它可以把可变输入
长度串(叫做预映射)转换成固定长度的输入
串(叫做散列值), 单向散列函数还具有单向函数

的特征，计算起来相对容易，但求逆却很难。也就是说，已知 x ，我们很容易计算 $f(x)$ ；但已知 $f(x)$ ，却难于计算出 x 。单向散列函数是公开的，对处理过程不用保密，它的安全性是它的单向性，其输出不依赖于输入。平均而言，预映射值的单个位的改变，将引起散列值中一半的位改变。已知一个散列值，要找到预映射的值在计算上是不可行的。用单向散列函数计算得到的散列（hash）值，通常称为数据摘要（MD），单向散列函数常用于信息鉴别和数字签名。典型的单向散列函数有 md5, sha1 等。

3.1.1 MD5 (Message-Digest Algorithm 5) 算法

3.1.1.1 MD5 算法介绍

MD5 的全称是 Message-Digest Algorithm 5 (信息 - 摘要算法)，在 90 年代初由 M I T Laboratory for Computer Science 和 RSA Data Security Inc 的 Ronald L. Rivest 开发出来，经 MD2、MD3 和 MD4 发展而来。它的作用是让大容量信息在用数字签名软件签署私人密匙前被“压缩”成一种保密的格式（就是把一个任意长度的字节串转换成一定长的大整数）。不管是 MD2、MD4 还是 MD5，它们都是输入一个随机长度的信息并产生一个 128 位的信息摘要。这三个算法的描述和 C 语言源代码在 Internet RFCs 1321 中有详细的描述。

1989 年，Rivest 开发出 MD2 算法。在这个算法中，首先对信息进行数据补位，使信息的字节长度是 16 的倍数。然后，以一个 16 位的检验和追加到信息末尾。并且根据这个新产生的信息计算出散列值。后来，Rogier 和 Chauvaud 发现如果忽略了检验和将产生 MD2 冲突。MD2 算法的加密后结果是唯一的 - 既没有重复。

1990 年，为了加强算法的安全性，Rivest 又开发出 MD4 算法。MD4 算法同样需要填补信息以确保信息的字节长度加上 448 后能被 512 整除（信

息字节长度 mod 512 = 448）。然后，一个以 64 位二进制表示的信息的最初长度被添加进来。信息被处理成 512 位 D/M 迭代结构的区块，而且每个区块要通过三个不同步骤的处理。Den Boer 和 Bosselaers 以及其他很快的发现了攻击 MD4 版本中第一步和第三步的漏洞。Dobbertin 向大家演示了如何利用一部普通的个人电脑在几分钟内找到 MD4 完整版本中的冲突（这个冲突实际上是一种漏洞，它将导致对不同的内容进行加密却可能得到相同的加密后结果）。MD4 算法虽然在安全上有个这么大的漏洞，但它对其后的安全加密算法 MD5, SHA-1、RIPE-MD 以及 HAVAL 的出现却有着不可忽视的引导作用。

1991 年，Rivest 开发出技术上更为趋近成熟的 MD5 算法。与 MD4 算法相比，MD5 有以下不同点：

- ◆ 加入了第四轮，MD4 只有三轮计算；
- ◆ 每一步都有唯一的加法常数；
- ◆ 第二轮中的 G 函数从 $((X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z))$ 变为 $((X \wedge Z) \vee (Y \wedge \sim Z))$ 以减小其对称性；
- ◆ 每一步都加入了前一步的结果，以加快“雪崩效应”；
- ◆ 改变了第 2 轮和第 3 轮中访问输入子分组的顺序，减小了形式的相似程度；
- ◆ 近似优化了每轮的循环左移位移量，以期加快“雪崩效应”，各轮的循环左移都不同。

尽管 MD5 比 MD4 来得复杂，并且速度较之要慢一点，但更安全，在抗分析和抗差分方面表现更好。

3.1.1.2 MD5 算法应用

MD5 的典型应用是对一段信息（Message）产生信息摘要（Message-Digest），以防止被篡改。比如，在 UNIX 下有很多软件在下载的时候都有一个文件名相同，文件扩展名为 .md5 的文件，在这个文件中通常只有一行文本，大致结构如：

```
MD5 (tanajiyat.tar.gz) =  
0ca175b9c0f726a831d895e269332461
```

这就是 `tanajiya.tar.gz` 文件的数字签名。MD5 将整个文件当作一个大文本信息，通过其不可逆的字符串变换算法，产生了这个唯一的 MD5 信息摘要。

如果在以后传播这个文件的过程中，无论文件的内容发生了任何形式的改变（包括人为修改或者下载过程中线路不稳定引起的传输错误等），只要你对这个文件重新计算 MD5 时就会发现信息摘要不相同，由此可以确定你得到的只是一个不正确的文件。

MD5 还广泛用于加密和解密技术上。比如在 UNIX 系统中用户的密码就是以 MD5（或其它类似的算法）经加密后存储在文件系统中。当用户登录的时候，系统把用户输入的密码计算成 MD5 值，然后再去和保存在文件系统中的 MD5 值进行比较，进而确定输入的密码是否正确。通过这样的步骤，系统在并不知道用户密码的明码的情况下就可以确定用户登录系统的合法性。这不但可以避免用户的密码被具有系统管理员权限的用户知道，而且还在一定程度上增加了密码被破解的难度。

3.1.1.3 MD5 算法分析

对 MD5 算法简要的叙述可以为：MD5 以 512 位分组来处理输入的信息，且每一分组又被划分为 16 个 32 位子分组，经过了一系列的处理后，算法的输出由四个 32 位分组组成，将这四个 32 位分组级联后将生成一个 128 位散列值。

◆ 步骤一：增加填充位

在 MD5 算法中，首先需要对信息进行填充，使其字节长度对 512 求余的结果等于 448。因此，信息的字节长度 (Bits Length) 将被扩展至 $N * 512 + 448$ ，即 $N * 64 + 56$ 个字节 (Bytes)， N 为一个正整数。填充的方法如下，在信息的后面填充一个 1 和无数个 0，直到满足上面的条件时才停止用 0 对信息的填充。

◆ 步骤二：填充长度

然后，在这个结果后面附加一个以 64 位二进制表示的填充前信息长度。经过这两步的处理，现在

的信息字节长度 = $N * 512 + 448 + 64 = (N+1) * 512$ ，即长度恰好是 512 的整数倍。这样做的原因是为满足后面处理中对信息长度的要求。

◆ 步骤三：初始化 MD 缓冲区

MD5 中的中间结果和最终结果保存于 128 位的缓冲区中，缓冲区用四个 32 位的寄存器 (A, B, C, D) 表示。这四个 32 位被称作链接变量 (Chaining Variable) 寄存器初始化成为：A=0x01234567, B=0x89abcdef, C=0xfedcba98, D=0x76543210。

◆ 步骤四：以 512 位的分组 (16 个字节) 为单位处理消息

当设置好这四个链接变量后，就开始进入算法的四轮循环运算。循环的次数是信息中 512 位信息分组的数目。将上面四个链接变量复制到另外四个变量中：A 到 a, B 到 b, C 到 c, D 到 d。

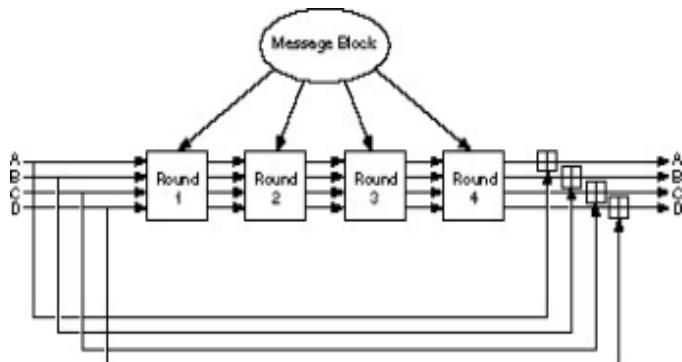


图 3.1：MD5 算法流程

主循环有四轮，每轮循环都很相似。每一轮进行 16 次操作，一共是 64 次操作。四轮运算结构相同，但是各轮使用不同的基本逻辑函数，我们称之为 F, G, H 和 I。每轮的输入是当前要处理的 512 位的分组和 128 位缓冲区 ABCD 的内容。每次操作对 a、b、c 和 d 进行 16 步迭代，每步迭代公式为：

$$a = b + ((a + g(b, c, d) + X[k] + T[I]) \ll\ll s).$$

其中：

a, b, c, d：缓冲区中的四个字，它按

照一定次序随迭代步变化

g : 基本的逻辑函数 F, G, H, I 之一, 每轮均不同。公式如下, 其中 & 是与, | 是或, - 是非, ^ 是异或。函数归纳如下:

$$F(X,Y,Z) = (X \& Y) | ((\sim X) \& Z)$$

$$G(X,Y,Z) = (X \& Z) | (Y \& (\sim Z))$$

$$H(X,Y,Z) = X \wedge Y \wedge Z$$

$$I(X,Y,Z) = Y \wedge (X | (\sim Z))$$

<<< s : 32 位的变量循环左移 s 位

X [K] : 消息的第 q 个子分组 (从 0 到 15)

的第 K 个 32 位字

T [I] : 4294967296 * abs(sin (i)) 的整数部分, i 的单位是弧度。(4294967296 等于 2 的 32 次方)。

按照以上表达式, MD5 所作的 64 次, 四轮运算如下所示, 其中表达式含义如下所示, m j 表示消息的第 j 个子分组 (从 0 到 15), <<< 表示左移计算。

$$FF(a,b,c,d,M_i,s,t_i) \leftarrow sa = b + ((a + F(b,c,d) + M_i + t_i) <<< s)$$

$$GG(a,b,c,d,M_i,s,t_i) \leftarrow sa = b + ((a + G(b,c,d) + M_i + t_i) <<< s)$$

$$HH(a,b,c,d,M_i,s,t_i) \leftarrow sa = b + ((a + H(b,c,d) + M_i + t_i) <<< s)$$

$$II(a,b,c,d,M_i,s,t_i) \leftarrow sa = b + ((a + I(b,c,d) + M_i + t_i) <<< s)$$

第一轮:

```

FF(a,b,c,d,M0.7,0xd76aaa478)
FF(d,a,b,c,M1.12,0xe8c7b756)
FF(c,d,a,b,M2.17,0x242070db)
FF(b,c,d,a,M3.22,0xc1bdceee)
FF(a,b,c,d,M4.7,0xf57c0faf)
FF(d,a,b,c,M5.12,0x4787c62a)
FF(c,d,a,b,M6.17,0xa8304613)
FF(b,c,d,a,M7.22,0xfd469501)
FF(a,b,c,d,M8.7,0x698098d8)
FF(d,a,b,c,M9.12,0x8b44f7af)
FF(c,d,a,b,M10.17,0xfffff5bb1)
FF(b,c,d,a,M11.22,0x895cd7be)
FF(a,b,c,d,M12.7,0xb901122)
FF(d,a,b,c,M13.12,0xfd987193)
FF(c,d,a,b,M14.17,0xa679438e)
FF(b,c,d,a,M15.22,0x49b40821)

```

第二轮:

```

GG(a,b,c,d,M1.5,0xf61e2562)
GG(d,a,b,c,M6.9,0xc040b340)
GG(c,d,a,b,M11.14,0x265e5a51)

```

GG(b,c,d,a,M0.20,0xe9b6c7aa)

GG(a,b,c,d,M5.5,0xd62f105d)

GG(d,a,b,c,M10.9,0x02441453)

GG(c,d,a,b,M15.14,0xd8a1e681)

GG(b,c,d,a,M4.20,0xe7d3fbcb8)

GG(a,b,c,d,M9.5,0x21e1cde6)

GG(d,a,b,c,M14.9,0xc33707d6)

GG(c,d,a,b,M3.14,0xf4d50d87)

GG(b,c,d,a,M8.20,0x455a14ed)

GG(a,b,c,d,M13.5,0xa9e3e905)

GG(d,a,b,c,M2.9,0xfcfa3f8)

GG(c,d,a,b,M7.14,0x676f02d9)

GG(b,c,d,a,M12,20,0x8d2a4c8a)

第三轮:

HH(a,b,c,d,M5.4,0xffffa3942)

HH(d,a,b,c,M8.11,0x8771f681)

HH(c,d,a,b,M11.16,0x6d9d6122)

HH(b,c,d,a,M14.23,0xfde5380c)

HH(a,b,c,d,M1.4,0xa4beea44)

HH(d,a,b,c,M4.11,0x4bdecfa9)

HH(c,d,a,b,M7.16,0xf6bb4b60)

HH(b,c,d,a,M10.23,0xebfb7c70)

HH(a,b,c,d,M13.4,0x289b7ec6)

HH(d,a,b,c,M0.11,0xea127fa)

HH(c,d,a,b,M3.16,0xd4ef3085)

HH(b,c,d,a,M6.23,0x04881d05)

HH(a,b,c,d,M9.4,0xd9d4d039)

HH(d,a,b,c,M12.11,0xe6db99e5)

HH(c,d,a,b,M15.16,0x1fa27cf8)

HH(b,c,d,a,M2.23,0xc4ac5665)

第四轮:

II(a,b,c,d,M0.6,0xf4292244)

II(d,a,b,c,M7.10,0x432aff97)

II(c,d,a,b,M14.15,0xab9423a7)

II(b,c,d,a,M5.21,0xfc93a039)

II(a,b,c,d,M12.6,0x655b59c3)

II(d,a,b,c,M3.10,0x8f0ccc92)

II(c,d,a,b,M10.15,0xffeff47d)

II(b,c,d,a,M1.21,0x85845dd1)

II(a,b,c,d,M8.6,0x6fa87e4f)

II(d,a,b,c,M15.10,0xfe2ce6e0)

II(c,d,a,b,M6.15,0xa3014314)

II(b,c,d,a,M13.21,0x4e0811a1)

II(a,b,c,d,M4.6,0xf7537e82)

II(d,a,b,c,M11.10,0xbd3af235)

II(c,d,a,b,M2.15,0x2ad7d2bb)

II(b,c,d,a,M9.21,0xeb86d391)

◆ 步骤五: 输出

在所有的 L 个分组处理完后, 第 L 个分组的输出既是 128 位的消息摘要。

综上，我们可以看到，MD5 算法有这么一个显著特点，即 hash 函数的每一位都是输出的每一位的函数。基本逻辑函数 (F, G, H, I) 的复杂迭代使得输出对输入的依赖非常小，也就是说，随即选择的两条消息，即使他们具有相同的规律性，也不可能产生相同的 hash 码。

3.1.2 SHA1 (Secure Hash Algorithm) 算法

3.1.2.1 SHA1 算法介绍

SHA (Secure Hash Algorithm) 是由美国国家标准技术研究院 (NIST) 设计并于 1993 年作为联邦信息处标准 FIPS PUB 180 发布，修订版于 1995 年发布 (FIPS 180-1)，通常称之为 SHA-1。SHA 算法建立在 MD4 算法之上，其基本框架与 MD4 类似。在协议 RFC 3174 中有完整定义。

3.1.2.2 SHA1 算法分析

SHA-1 算法的输入是长度小于 2^{64} 位的消息，输出的是 160 位的消息摘要，输入消息要以 512 位的分组为单位进行处理。SHA1 的处理过程与 MD5 相似，只是 hash 值和链接变量长为 160 位。SHA1 算法包含下列步骤：

◆ 步骤 1：添加填充位 (一个 1 和若干个 0)。

在 SHA1 算法中，首先需要对信息进行填充，使其字节长度对 512 求余的结果等于 448。因此，信息的字节长度 (Bits Length) 将被扩展至

$N * 512 + 448$ ，即 $N * 64 + 56$ 个字节 (Bytes)， N 为一个正整数。填充的方法如下，在信息的后面填充一个 1 和无数个 0，直到满足上面的条件时才停止用 0 对信息的填充。

◆ 步骤 2：添加填充长度。

然后，在这个结果后面附加一个以 64 位二进制表示的填充前信息长度。经过这两步的处理，现在的信息字节长度 $= N * 512 + 448 + 64 = (N+1) * 512$ ，即长度恰好是 512 的整数倍。这样做的原因是为满足后面处理中对信息长度的要求。

前两步的处理与 MD5 算法完全相同。

◆ 步骤 3：初始化 MD 缓冲区。

一个 160 位 MD 缓冲区用以保存中间和最终 Hash 函数的结果。它可以表示为 5 个 32 位的寄存器 (A, B, C, D, E)。初始化为以下数值，前四个与 MD5 相同，但存储为大数在前的形式。

A = 67452301

B = EFCDAB89

C = 98BADCFE

D = 10325476

E = C3D2E1F0

◆ 步骤 4：以 512 位数据块为单位处理消息。

算法的核心是具有四轮运算的模块，每轮执行 20 步迭代，其处理过程如下图所示。四轮运算结构相同，但是各轮使用不同的基本逻辑函数，分别称为 f_1, f_2, f_3 和 f_4 。

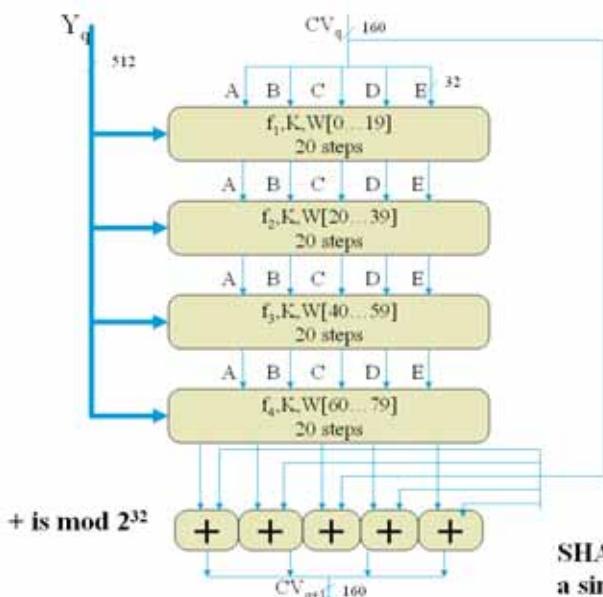


图 3.2 SHA1 算法处理 512 位分组的过程

SHA-1 Processing of a single 512-bit block

步数， $0 \leq t \leq 79$ ，但实际上使用了四个不同的加法常量，分别为：

$$\begin{array}{ll} 0 \leq t \leq 19 & : K_t = 5A827999 \\ 20 \leq t \leq 39 & : K_t = 6ED9EBA1 \\ 40 \leq t \leq 59 & : K_t = 8F1BBCDC \\ 60 \leq t \leq 79 & : K_t = CA62C1D6 \end{array}$$

处理每个 512 分组要执行 80 步，每步处理具有下述形式：

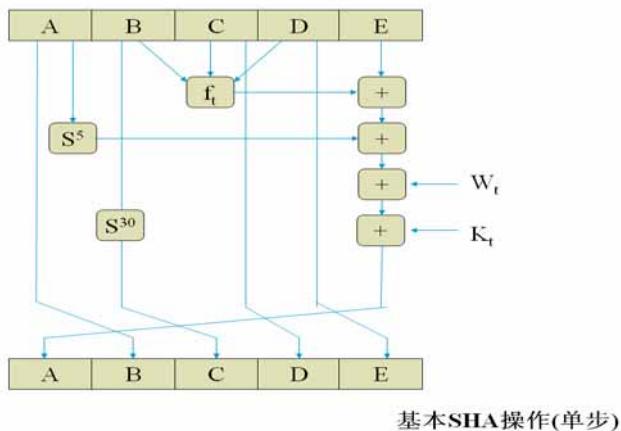


图 3.3: SHA 算法的基本操作

$$A, B, C, D, E \leftarrow (E + f(t, B, C, D) + S^5(A) + W_t + K_t), A, S^{30}(B), C, D$$

其中：

$$\begin{aligned} A, B, C, D, E &= \text{缓冲区的 5 个字;} \\ t &= \text{步数, } 0 \leq t \leq 79; \\ f(t, B, C, D) &= \text{步 } t \text{ 的基本逻辑函数;} \\ S^k &= \text{循环左移 } k \text{ 位给定的 32 位字;} \\ W_t &= \text{一个从当前 512 数据块导出的} \\ 32 \text{ 位字;} \\ K_t &= \text{一个用于加法的常量, 四个} \\ \text{不同的值, 如前所述} \\ + &= \text{模 } 2^{32} \text{ 加法.} \end{aligned}$$

每个基本逻辑函数的输入是三个 32 位字，输出是一个 32 位字。每个函数执行位逻辑运算，即输出的第 n 位是三个输入的第 n 位的函数。

Step	Function Name	Function Value
(0 ≤ t ≤ 19)	f1	= $f(t, B, C, D) = (B \wedge C) \vee (B \wedge D)$
(20 ≤ t ≤ 39)	f2	= $f(t, B, C, D) = B \oplus C \oplus D$
(40 ≤ t ≤ 59)	f3	= $f(t, B, C, D) = (B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$

$$(60 \leq t \leq 79) f4 = f(t, B, C, D) = B \oplus C \oplus D$$

◆ 步骤 5：输出。

全部 L 个 512 位数据块处理完毕后，输出 160 位摘要消息：

3.1.3 MD5 与 SHA1 比较

由于 SHA1 和 MD5 都基于 MD4，所以他们的性质非常相似。但是他们在下面一些地方存在不同。

1) 算法实现

	MD5	SHA-1
摘要长度	128 位	160 位
基本处理单位	512 位	512 位
步数	64 (4 of 16)	80 (4 of 20)
基本逻辑函数	4	4
最大消息长度	无限	$2^{64}-1$ 位
加法常数	64	4
低位 / 高位结构	低位在前	高位在前

表 3.1 MD5 算法与 SHA1 算法比较

2) 抗穷举攻击的能力

SHA-1 的消息摘要比 MD5 的摘要要长 32 位。对 MD5 用穷举攻击方法产生具有给定摘要的消息，其代价是 2^{128} 数量级，而对于 SHA-1，代价为 2^{160} 数量级，所以 SHA-1 的抗穷举攻击的能力要强的多。

3) 速度

由于这两个算法都是基于模 2^{32} 加法，所以在 32 位机上这些算法的执行速度较快。SHA-1 要执行 80 步迭代且缓冲区为 160 位，而 MD5 要执行 64 步迭代且缓冲区为 128 位，因此同样的环境下，SHA-1 的执行速度要比 MD5 慢很多。

4) 简单和简洁性

两个算法都易于描述和实现，不需要大的程序或置换表。

5) 低位结构和高位结构

MD5 使用低位在前的方式将消息映射为 32 位

32位字的序列，而SHA-1使用高位在前的方式将消息映射为32位字的序列，这两个结构没有本质的区别，也许是仅仅因为SHA-1的设计者采用sun计算机来实现的。

3.2 消息认证码

MAC (message authentication codes)消息认证码，利用密钥来生成一个固定长度的短数据块，并且将该数据块附在消息之后。在这种方法中假定通信双方，比如说A和B，共享密钥K。若A向B发送消息时，则A计算MAC，它是消息和密钥的函数，即 $MAC = C_k(M)$ 。其中：M = 输入消息；C = MAC函数；K = 共享的密钥；MAC = 消息认证码。消息和MAC一起将被发送给接收方。接收方对收到的消息用相同的密钥K进行相同的计算得出新的MAC，并将接收到的MAC与其计算出的MAC进行比较。如果我们假定只有收发双方知道该密钥，那么若接收到的MAC与计算出的MAC相等，则有：

1. 接收方可以相信消息未被修改。如果攻击者改变了消息，但他无法改变相应的MAC。
2. 接收方可以相信消息来自真正的发送方。因为其他各方均不知道密钥，因此他们不能产生具有正确MAC的消息。
3. 如果消息中含有序列号（如HDLC，X.25和TCP中使用的序列号），那么接受方可以相信消息顺序是正确的。

3.3 HMAC算法

诸如MD5这样的hash函数并不是专门为MAC而设计的，由于hash函数不依赖于密钥，所以它们不能直接用于MAC。目前，将密钥加到现有的hash函数中已经提出了很多方案，HMAC（RFC 2104）是最受欢迎的方案之一，它是IPSec必须使用的方法，在其他Internet协议中（如SSL）也使用了HMAC。

在RFC 2104中给出了HMAC的设计目标：

- ◆ 不必修改而直接使用现有的hash函数。
- ◆ 如果找到或者需要更快或更安全的hash函数，应能很容易替代原有嵌入的hash函数。
- ◆ 应保持hash函数原有的性能，不能降低其性能。
- ◆ 对密钥的使用和处理应该简单。
- ◆ 如果已知嵌入的hash函数的强度，则完全可以知道认证机制抗密码分析的强度。

3.3.1 HMAC算法介绍

在图3.4中给出了HMAC的总体结构，定义下列符号：

H	= 嵌入Hash函数 (MD5, SHA-1, RIPEMD-160)
M	= 消息（包括Hash函数所需填充位）
Y _i	= M的第i个数据块，0 ≤ i ≤ L-1
L	= M的数据块数
b	= 数据块的位数
n	= 嵌入Hash函数产生的Hash码长度位数
K	= 保密密钥。如果密钥长度大于b，则密钥送入Hash函数，形成一个n位的密钥；推荐程度大于等于n
K ⁺	= K在左部添加0使得其长度为b位
ipad	= 00110110重复 b/8次
opad	= 01011010重复 b/8次
n	= 嵌入Hash函数产生的Hash码长度位数
K	= 保密密钥。如果密钥长度大于b，则密钥送入Hash函数，形成一个n位的密钥；推荐程度大于等于n
K ⁺	= K在左部添加0使得其长度为b位
ipad	= 00110110重复 b/8次
opad	= 01011010重复 b/8次

HMAC可描述为如下：

$$HMACK = H[K^+ \oplus opad] || H(K^+ \oplus ipad) || M]$$

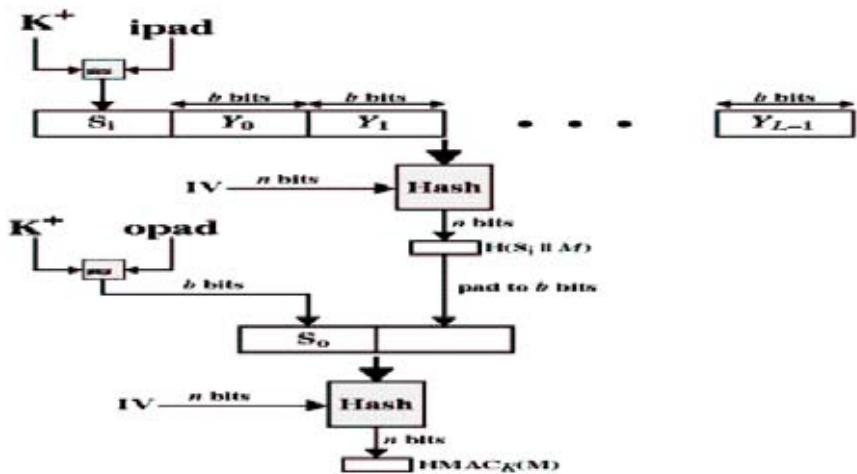


图 3.4 HMAC 运算过程

具体步骤如下：

1. 在 K 左边填充 0，得到 b 位的 K^+ （例如，若 K 是 120 位， $b = 512$ ，则在 K 中加入 44 个 0 字节 $0x00$ ）。
2. K^+ 与 $i\text{pad}$ 执行异或运算（位异或）产生 b 位的分组 S_i 。
3. 将 M 附于 S_i 后。
4. 将 H 作用于步骤三所得出的结果。
5. K^+ 与 $o\text{pad}$ 执行异或运算（位异或）产生 b 位的分组 S_o 。
6. 将步骤 4 中的 hash 值附于 S_o 后。

将 H 作用于步骤 6 所得的结果，并输出该函数值。

IPsec 协议中密钥的管理可以通过两种方式进行，一种是手工方式，另外一种是 IKE (Internet Key Exchange) 方式。手工方式由于简单的密钥管理以及其较弱的安全性，我们并不推荐使用。IKE 的精髓在于它永远不在不安全的网络上直接传送密钥，而是通过一系列数据的交换，通信双方最终计算出共享的密钥，并且即使第三方（如黑客）截获了双方用于计算密钥的所有交换数据，也不足以计算出真正的密钥。其中的核心技术就是 DH (Diffie Hellman) 交换技术。

密钥管理算法

4.1 DH 组算法

4.1.1 DH 算法介绍

Diffie-Hellman 是发明的第一个公开密钥算法，以其发明者 Whitfield Diffie 和 Martin Hellman 的名字命名。Diffie-Hellman 算法是用于密钥交换的最早最安全的算法之一。DH 算法的基本工作原理是：通信双方公开或半公开交换一些准备用来生成密钥的数据，而在彼此交换过密钥生成数据后，两端可以各自生成出完全一样的共享密钥。在任何时候，双方都绝不交换真正的密钥。

4.1.2 DH 算法分析

Diffie-Hellman 密钥交换算法的有效性依赖于计算离散对数的难度。简言之，可以如下定义离散对数：首先定义一个素数 p 的原根，为其各次幂产生从 1 到 $p-1$ 的所有整数根，也就是说，如果 a 是素数 p 的一个原根，那么数值 $a \bmod p, a^2 \bmod p, \dots, a^{p-1} \bmod p$ 是各不相同的整数，并且以某种排列方式组成了从 1 到 $p-1$ 的所有整数。对于一个整数 b 和素数 p 的一个原根 a ，可以找到唯一的指数 i ，使得 $b = a^i \bmod p$ 其中 $0 \leq i \leq (p-1)$

指数 i 称为 b 的以 a 为基数的模 p 的离散对数或者指数。该值被记为 $\text{inda}_{p(b)}$ 。基于此

背景知识，可以定义 Diffie-Hellman 密钥交换算法。该算法描述如下：

- ◆ 有两个全局公开的参数，一个素数 q 和一个整数 a ， a 是 q 的一个原根。
- ◆ 假设用户 A 和 B 希望交换一个密钥，用户 A 选择一个作为私有密钥的随机数 $X^A < q$ ，并计算公开密钥 $Y^A = aX^A \bmod q$ 。A 对 X^A 的值保密存放而使 Y^A 能被 B 公开获得。类似地，用户 B 选择一个私有的随机数 $X^B < q$ ，并计算公开密钥 $Y^B = aX^B \bmod q$ 。B 对 X^B 的值保密存放而使 Y^B 能被 A 公开获得。
- ◆ 用户 A 产生共享秘密密钥的计算方式是 $K = (Y^B)X^A \bmod q$ 。同样，用户 B 产生共享秘密密钥的计算是 $K = (Y^A)X^B \bmod q$ 。这两个计算产生相同的结果：

$$\begin{aligned}
 K &= (Y^B)X^A \bmod q \\
 &= (aX^B \bmod q)X^A \bmod q \\
 &= (aX^B)X^A \bmod q \quad (\text{根据取模运算规则得到}) \\
 &= aX^BX^A \bmod q \\
 &= (aX^A)X^B \bmod q \\
 &= (aX^A \bmod q)X^B \bmod q \\
 &= (Y^A)X^B \bmod q
 \end{aligned}$$

因此相当于双方已经交换了一个相同的秘密密钥。

- ◆ 因为 X^A 和 X^B 是保密的，一个敌对方可以利用的参数只有 q 、 a 、 Y^A 和 Y^B 。因而敌对方被迫取离散对数来确定密钥。这在数学上几乎是不太可能的。

Diffie-Hellman 密钥交换算法的安全性依赖于这样一个事实：虽然计算以一个素数为模的指数相对容易，但计算离散对数却很困难。对于大的素数，计算出离散对数几乎是不可能的。

下面给出例子。密钥交换基于素数 $q = 97$ 和 97 的一个原根 $a = 5$ 。A 和 B 分别选择私有密钥 $X^A = 36$ 和 $X^B = 58$ 。每人计算其公开密钥

$$Y^A = 536 = 50 \bmod 97$$

$$Y^B = 558 = 44 \bmod 97$$

在他们相互获取了公开密钥之后，各自通过计算得到双方共享的秘密密钥如下：

$$K = (Y^B)X^A \bmod 97 = 4436 = 75 \bmod 97$$

$$K = (Y^A)X^B \bmod 97 = 5058 = 75 \bmod 97$$

从 $|50, 44|$ 出发，攻击者要计算出 75 很不容易。

4.1.3 DH 算法的优劣

Diffie-Hellman 算法具有两个吸引力的特征：

- ◆ 仅当需要时才生成密钥，减小了将密钥存储很长一段时间而致使遭受攻击的机会。
- ◆ 除对全局参数的约定外，密钥交换不需要事先存在的基础结构。

Diffie-Hellman 算法的不足：

- ◆ 没有提供双方身份的任何信息。
- ◆ 它是计算密集性的，因此容易遭受阻塞性攻击，即对手请求大量的密钥。受攻击者花费了相对多的计算资源来求解无用的幂系数而不是在做真正的工作。
- ◆ 没办法防止重演攻击。

容易遭受中间人的攻击。第三方 C 在和 A 通信时扮演 B；和 B 通信时扮演 A。A 和 B 都与 C 协商了一个密钥，然后 C 就可以监听和传递通信量。中间人的攻击按如下进行：

- (1) B 在给 A 的报文中发送他的公开密钥。
- (2) C 截获并解析该报文。C 将 B 的公开密钥保存下来并给 A 发送报文，该报文具有 B 的用户 ID 但使用 C 的公开密钥 Y_C ，仍按照好像是来自 B 的样子被发送出去。A 收到 C 的报文后，将 Y_C 和 B 的用户 ID 存储在一块。类似地，C 使用 Y_C 向 B 发送好像来自 A 的报文。
- (3) B 基于私有密钥 X_B 和 Y_C 计算秘密密钥 K_1 。A 基于私有密钥 X_A 和 Y_C 计算秘密密钥 K_2 。C 使用私有密钥 X_C 和 Y_B 计算 K_1 ，并使用 X_C 和 Y_A 计算 K_2 。
- (4) 从现在开始，C 就可以转发 A 发给 B 的报文或

转发 B 发给 A 的报文；在途中根据需要修改它们的密文。使得 A 和 B 都不知道他们在和 C 共享通信。

4.1.4 IKE 协议中的 DH 组交换

IKE 协议在使用 DH 算法时，为了增强其安全性，避免上述 DH 交换的缺陷，使用了 Oakley 协议。IKE 协议本身是 ISAKMP、Oakley 和 SKEME 这三个协议构混合而成。Oakley 算法是对 Diffie-Hellman 密钥交换算法的优化，它保留了后者的优点，同时克服了其弱点。Oakley 算法具有五个重要特征：

- ◆ 它采用称为 cookie 程序的机制来对抗阻塞攻击。
- ◆ 它使用了限时来保证抵抗重演攻击。
- ◆ 它能够交换 Diffie-Hellman 公开密钥。
- ◆ 它对 Diffie-Hellman 交换进行鉴别以对抗中间人的攻击。

Oakley 可以使用三个不同的鉴别方法：

- ◆ 数字签名：通过签署一个相互可以获得的散列代码来对交换进行鉴别；每一方都使用自己的私钥对散列代码加密。散列代码是在一些重要参数上生成的，如用户 ID 和现时。
- ◆ 公开密钥加密：通过使用发送者的私钥对诸如 ID 和现时等参数进行加密来鉴别交换。
- ◆ 对称密钥加密：通过使用某种共享密钥对交换参数进行对称加密，实现交换的鉴别。

—— 本文完 ——

IKE 介绍

◆◇□ 李丹

概述

用 IPsec 保护一个 IP 包之前，必须先建立一个安全联盟（SA），SA 可以手工创建或动态建立。Internet 密钥交换（The Internet Key Exchange，既 IKE）用于动态建立 SA。

由 RFC2409 文件描述的 IKE 属于一种混合型协议。它建立在由 Internet 安全联盟和密钥管理协议（ISAKMP）定义的一个框架上。同时，IKE 还实现了两种密钥管理协议的一部分根 Oakley 和 SKEME。IKE 沿用了 ISAKMP 的基础、Oakley 的模式以及 SKEME 的共享和密钥更新技术，从而定义出自己独一无二的验证加密材料生成技术，以及协商共享策略。IKE 并非由 IPsec 专用，IKE 利用 ISAKMP 语言定义密钥交换，是对安全服务进行协商的手段。事实上，其他协议也可以使用它来协商安全服务。



1 ISAKMP

ISAKMP 是 IKE 的基础，IKE 使用 ISAKMP 协议定义密钥交换的过程，下面先简要介绍一下 ISAKMP 的消息格式。

ISAKMP 定义了通信双方如何沟通，如何构建彼此间用以沟通的消息，还定义了保障通信安全所需的状态变换。ISAKMP 提供了对对等体的身份进行验证的方法，密钥交换时交换信息的方法，以及对安全服务进行协商的方法。

下图为 ISAKMP 的消息头格式：

发起者 Cookie				
响应者 Cookie				
下一个载荷	主版本	副版本	交换类型	旗 标
消息 ID				
消息 长度				

图 1 ISAKMP 消息头

Cookie: 发起者 Cookie 和响应者 Cookie 是由通信双方创建的，并随消息 ID 一道，用来标识状态，以便对进行中的一次 ISAKMP 交换进行定义。

下一个载荷：该字段被用来指出在各个 ISAKMP 载荷中，哪一个紧随在这个头之后。

主版本和副版本： ISAKMP 版本的标识是用主版本和副版本字段中的主 / 副编号来进行的。

交换类型：标识 ISAKMP 交换的具体类型。

消息长度：标识 ISAKMP 消息的全长（包括头自身的长度）。

旗标：为接收者提供与消息有关的特殊信息，旗标是由一个位掩码来表示的，其中，每个位对应一个具体的选项。目前总共定义了三个旗标位，其他位数留待以后扩展，分别为：加密、委托和纯验证。加密旗标指出跟随在这个头之后的载荷已经加密；委托旗标指出通信的某一方在交换完成之后收到通知；纯验证旗标主要由那些希望为 ISAKMP 引入密钥恢复机制的人使用。

在消息头中的“下一个载荷”字段已经标明了消息头后面跟随的相应的载荷数据，所有已定义的载荷数据都是以同样的消息头开始的，其格式是：

下一个载荷	保 留	载荷 长度
-------	-----	-------

图 3 载荷数据头格式

跟随在当前载荷之后的下一个载荷的类型由“下一个载荷”字段来指定。最后一个载荷的该字

段填 0，表示后面无载荷数据。载荷的总长度由“载荷长度”字段来表示。至于保留字段，则未使用，必须设为 0。

至于载荷内数据的格式分为以下两种：

1	属性 类型	属性 值
---	-------	------

图 2 基本属性数据格式

0	属性 类型	值 长
属性 值		

图 1 变长属性数据格式

两种属性的数据格式通过第一位的值来区分：第一位为 0，表示为变长属性；第一位为 1，表示为基本属性。

IKE 简介

IKE 的精髓在于它永远不在不安全的网络上直接传送密钥，而是通过一系列数据的交换，通信双方最终计算出共享的密钥，并且即使第三方（如黑客）截获了双方用于计算密钥的所有交换数据，也不足以计算出真正的密钥。其中的核心技术就是 DH (Diffie Hellman) 交换技术。

IKE 使用了两个阶段的 ISAKMP。第一阶段建立 IKE 安全联盟，第二阶段利用这个既定的安全联盟，为 IPsec 协商具体的安全联盟。

IKE 第一阶段交换定义了两种模式，第二阶段交换只有一种模式，另外还有两个额外的交换（用于对安全联盟进行正确的维护）。对第一阶段交换来说，IKE 采用的是身份保护交换，以及根据基本 ISAKMP 文档制订的野蛮交换法。对此，我们分别叫作“主模式”和“野蛮模式”。对第二阶段来说，IKE 则定义了一种快速模式交换。它的作用是为 IKE 之外的其他协议协商安全 – 例如 IPsec。IKE 定义的另外两种交换均属信息方面的交换。在这种交换中，IKE 通信双方可相互间传达有关错误和状态的资讯，而且一种新的组交换模式可使各方协商如何在它们之中使用一个新的 Diffie-Hellman 组。

IKE SA 提供了各种各样的参数，它们是由通信双方协商制定的。由于有些 IKE 消息需要加密处理和验证，所以通信双方必须协商拟定对消息进行加密和验证的方式。针对所有协商好的参数，IKE 规定了它们可能用到的一系列属性以及取值范围。我们将所有参数称为一个“保护套件” – 包括加密算法、散列算法、验证方法以及 Diffie-Hellman 组。保护套件将作为一个整体进行协商，通过 ISAKMP SA 载荷进行交换。保护套件中的每一种属性都包含在该载荷中。除了这些强制使用的属性之外，还有一些可选属性，它们可作为保护套件的一部份进行协商。

IKE 第一阶段

对 IKE 第一阶段交换来说，无论是主模式还是野蛮模式，它们做的都是相同的事情：建立一个保密和验证无误的通信信道 (IKE SA)，以及建立验证过的密钥，为双方的 IKE 通信提供机密性、消息完整性以及消息源验证服务。IKE 中定义的其他所有交换都要求一个验证过的 IKE SA 作为首要条件。所以在一个完整的 IKE 交换过程中，第一阶段交换 – 无论主模式还是野蛮模式 – 必须在进行其他任何交换（如第二阶段交换，消息交换等）之前完成，其他交换需要在第一阶段交换后建立的安全联盟的保护下进行。

参与通信的双方会生成四种秘密：SKEYID，后续的所有秘密都建立在它的基础上；SKEYID_d，用于为 IPsec（和其他）衍生出加密的材料；SKEYID_a，用来为 IKE 消息保障数据的完整性以及对数据源的身份进行验证；以及 SKEYID_e，用于对 IKE 消息进行加密。SKEYID 的生成取决于协商好的是何种验证方法。其他所有以 SKEYID 为基础的秘密都以相同的方式衍生出来 – 无论岩镇方法是什么。验证方法决定了如何交换载荷，以及在什么时候交换。目前使用较为普遍的验证方法为：

- 1) 预共享密钥；
- 2) 使用“数字签名算法 (DES)”得到的数字签名；

对于预共享密钥验证来说：

SKEYID = PRF (预共享密钥, Ni | Nr);

对于签名验证来说：

SKEYID=PRF(Ni | Nr, qxv) ;

其中，PRF 为伪随机函数。

其他秘密的生成方法为：

SKEYID d = PRF(SKEYID, qxv| CKY-I|CKY-R|0)

SKEYID a = PRF(SKEYID, SKEYID d | qxv| CKY-I|CKY-R|1)

SKEYID e = PRF(SKEYID, SKEYID a | qxv| CKY-I|CKY-R|2)

在上面的公式中，有几项在此简单说明一下，CKY 表示 cookie；N 代表 nonce，为一个伪随机数。

CKY 表示 cookie; N 代表 nonce, 为一个伪随机数; 后面的 I 表示为发起者的数据, R 表示为相应者的数据; 双方的 cookie 和 nonce 需要在主模式过程中进行交换; $q_{x,y}$ 为双方进行 DH 交换之后的计算出来的共享秘密。

3.1 主模式交换

主模式分为三次交换, 总共用到了六条消息, 最终建立了 IKE SA。这三次交换是:

1. 策略协商;
2. Diffie-Hellman 和 nonce 交换;
3. 对对方身份的验证。

正如前面所提到的, 验证方法决定了如何交换载荷, 以及在什么时候交换。下面针对两种验证方法分别介绍主模式的消息交换情况。

3.1.2 预共享密钥

下图为预共享密钥情况下, 主模式的消息交换情况:

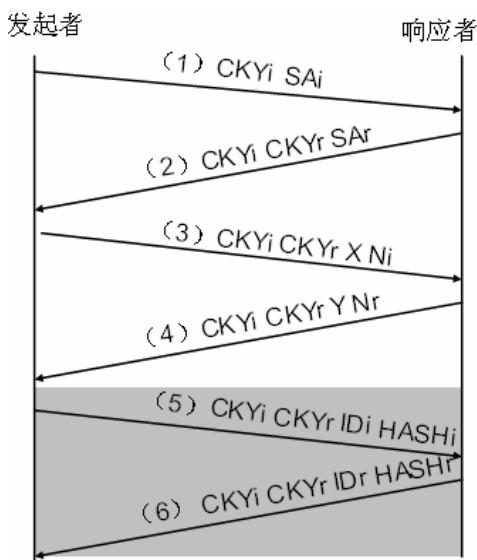


图 5 预共享密钥主模式

简单的说, 第一次交换(消息1、2)用于IKE提议和转换方式的协商; 第二次交换(消息3、4)用于IKE DH 和伪随机值 nonce 的交换; 第三次交换(消息5、6)用于通信双方的身份认证。

在消息1、2发送之前, 协商发起者和响应者必须计算产生 cookie, 用于唯一的标识每个单独的协商交换, cookie 使用源 / 目的 IP 地址、随机数字、日期和时间进行 MD5 运算得出, 并且将其放入消息1的 ISAKMP 头中, 用以标识一个单独的协商交换。

在第一次交换中, 需要交换双方的 cookie 和 SA 载荷, 在 SA 载荷中携带需要协商 IKE SA 的各项参数, 主要包括 IKE 的散列类型、加密算法、认证方法、IKE SA 协商的时间限制。

第一次交换使用的消息情:

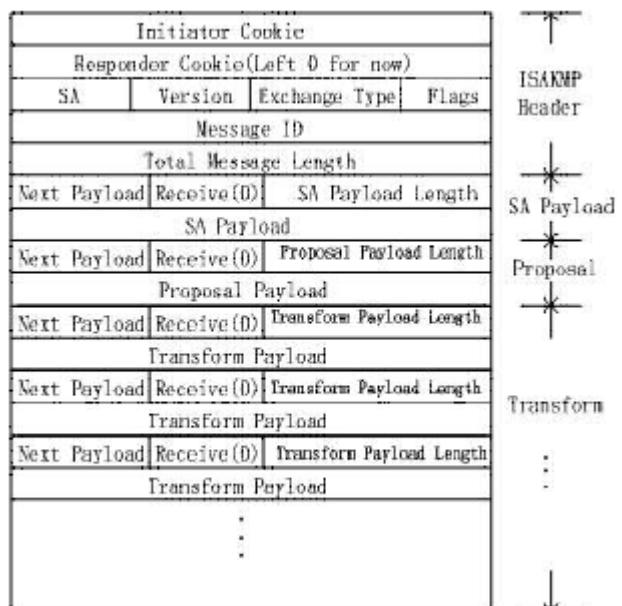


图 6 主模式消息 1 结构

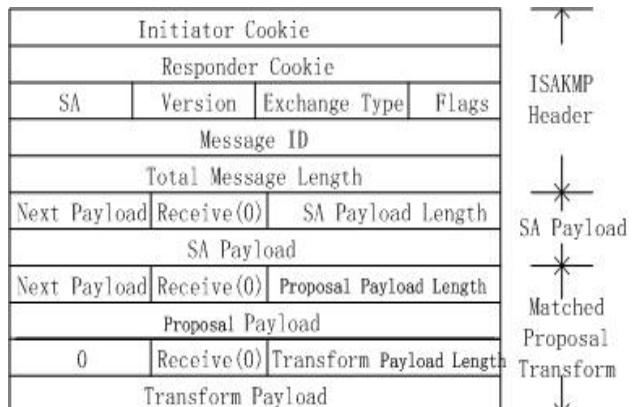


图 7 主模式消息 2 结构式结构

第一次交换之后，通信双方需要生成用于产生Diffie-Hellman共享密钥的DH值，生成方法是双方各自使用一个随机数字，通过DH算法对随机数字进行运算得出一个DH值X_a和X_b(X_a是发起方的DH值，X_b是响应方的DH值)，然后双方再根据DH算法各得出一个临时值N_i和N_r，双方交换这DH值之后通过自己计算的DH值和交换得到的DH值进行运算就可以产生一个只有双方知道的共享秘密，此共享秘密并不进行传输，传输的是DH值，即使第三方得到了这个DH值也无法计算出共享秘密。

第二次交换对密钥交换载荷和临时值载荷进行交换，其中密钥交换载荷包含了X_a或者X_b，临时值载荷包含了N_i或者N_r。

第二次交换中用到的消息结构如下：

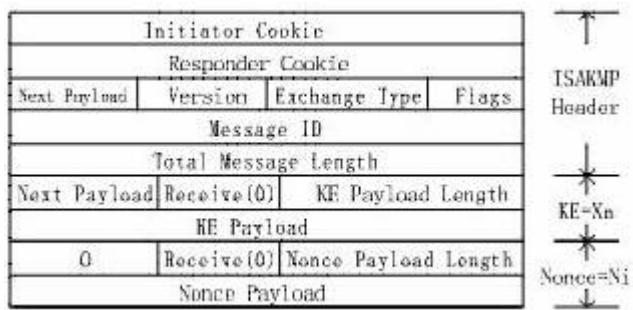


图8 主模式消息3结构

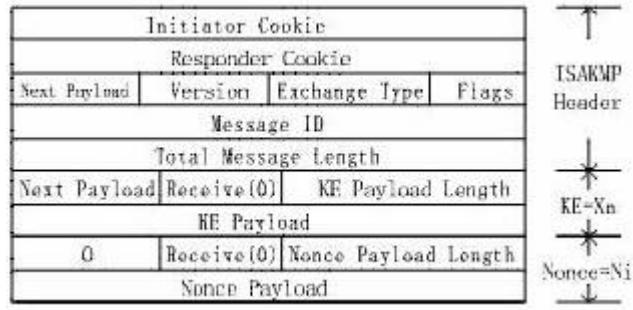


图9 主模式消息4结构

在第二次交换完成之后，按照前面介绍过的密钥生成公式，其所需的所有计算材料都已交换完毕，此时已经可以将所有密钥计算出来，并使用计算得到的密钥对后续的IKE消息提供安全

服务。

第三次交换对标识载荷和散列载荷进行交换，标识载荷包含了发起者的标识信息，IP地址或者主机名。散列载荷包含对上一过程中产生的三组密钥进行Hash运算得出的值。这两个负载使用SK EY I D _ e 进行加密。如果通信双方散列载荷中的hash值相同，那么双方的认证成功。IKE第一阶段主模式预共享密钥方式交换也就完成了。

第三次交换中用到的消息的结构如下：

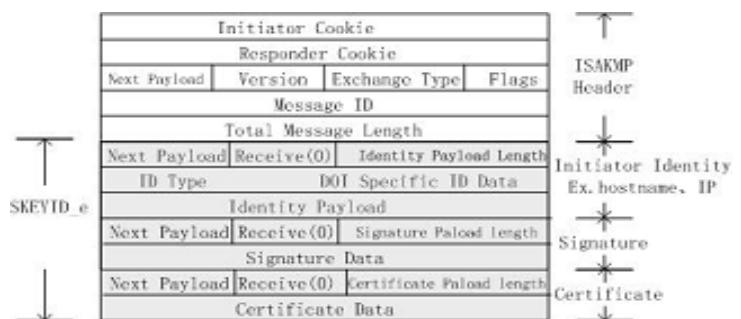


图10 主模式预共享密钥验证方式消息5结构

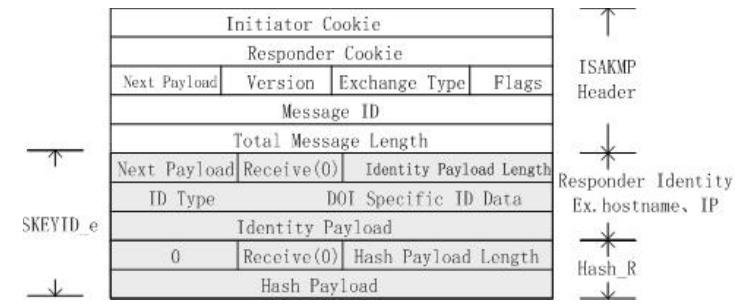


图11 主模式预共享密钥验证方式消息6结构

3.1.3 数字证书

对于预共享密钥验证方式来说，当有1个对等体对应多个对等体时，需要为每个对等体配置预共享的密钥，这也是预共享密钥验证方式的一个缺点；对于对等体IP地址为动态的情况下，IKE主模式使用预共享密钥无法通过IP地址来标识对方，查找预共享密钥。

使用数字证书方式则没有以上的问题，但是需要CA服务器来颁发证书。

证书验证和预共享密钥验证的主要区别在于

SKEYID 的计算和第三次交换中消息的负载的发送，其他的交换与计算过程和预共享密钥验证方式相同。关于 SKEYID 的计算公式前面已经针对不同的验证方式介绍过了，下面说一下主模式第三次交换证书方式的负载情况：证书方式下第三次交换的消息比预共享密钥的消息多两个负载：签名负载和证书负载。签名负载：使用自己的私钥加密部分消息的 Hash 值，然后使用 SKEYID_e 加密发送给对方；签名的计算方法：

$$\text{Signature} = \text{Private Key}_I \{ \text{PRF}(\text{SKEYID}, \text{Cookie-I}, \text{Cookie-R}, \text{SAPayload}, \text{Proposals+Transforms}, \text{ID-I}) \}$$
；证书负载：将自己的证书使用 SKEYID_e 加密发送给对方。

下图以图示为例，说明在证书验证方式下第三次交换的负载情况：

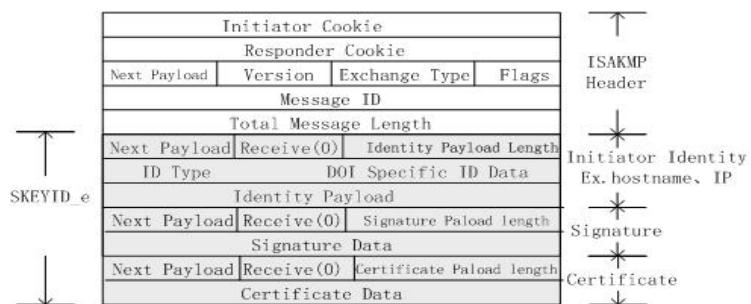


图 12 主模式证书验证方式下消息 5 结构

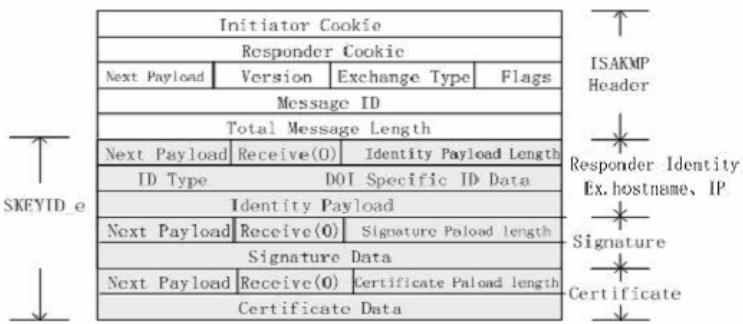


图 13 主模式证书验证方式下消息 6 结构

3.2 野蛮模式交换

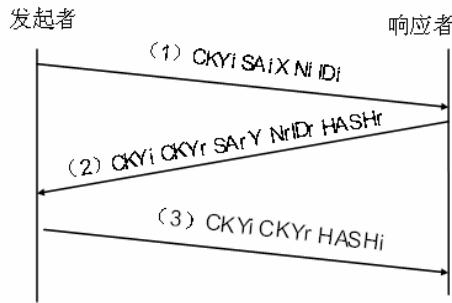
从上述对主模式协商的叙述中可以看到，在主模式完成第二次交换之后即要生成会话密钥，

会话密钥的生成材料中包含了预共享密钥；而另有说明在一个通信对等体和多个对等体通信是需要为每个对等体设置一个预共享密钥；那么它如何在多个密钥中进行选择呢（对等体的身份 ID 信息在第三次交换中才会发送，此时对等体并不知道对端的身份信息）？答案是：根据前面的交换的消息的 IP 地址来区分不同的对等体。

那另一问题又出现了，在需要进行远程访问的时候（即发起者的 IP 地址是动态分配的情况），由于发起者的地址不可能被响应者提前知道，而且双方都打算使用预共享密钥验证方法，如何建立 IKE SA 呢？野蛮模式被用于处理这种情况。

野蛮模式交换的用途与主模式交换相同—建立一个验证的安全联盟和密钥，随后可用 IKE SA 为其他安全协议建立安全联盟，野蛮模式仅交换 3 个消息便完成 IKE SA 的建立。由于对消息的数量进行了限制，野蛮模式同时也限制了它的协商能力，而且不会提供身份保护。

野蛮模式的消息交换情况是：



在野蛮模式交换过程中，发起者会提供一个保护套件列表、Diffie-Hellman 公共值、nonce 以及身份资料。所有这些信息，都是随第一条消息传送的。作为响应者，则需要回应一个选定的保护套件、Diffie-Hellman 公共值、nonce、身份资料以及一个验证载荷 – 对于预共享密钥以及加密的 nonce 验证来说，是一个散列载荷；而对基于签名的验证来说，则是一个签名载荷。发起者将它的验证载荷作为最后一条消息来传送。

野蛮模式各消息的结构如下：

Initiator Cookie			
KE	Responder Cookie	Version	Exchange Type
Flags			
Message ID			
Total Message Length			
Next Payload	Receive(1)	SA Payload Length	SA Payload
Next Payload	Receive(1)	Proposal Payload Length	Proposal Payload
Next Payload	Receive(1)	Transform Payload Length	Transform Payload
Next Payload	Receive(0)	KE Payload Length	KE Payload
0	Receive(0)	Nonce Payload Length	Nonce Payload
Next Payload	Receive(0)	Identity Payload Length	Identity Payload
ID Type	DOI Specific ID Data		
0	0	Hash Payload Length	Hash Payload
Hash Payload			

图 15 野蛮模式消息 1 结构

Initiator Cookie			
KE	Responder Cookie	Version	Exchange Type
Flags			
Message ID			
Total Message Length			
Next Payload	Receive(1)	SA Payload Length	SA Payload
Next Payload	Receive(1)	Proposal Payload Length	Proposal Payload
Next Payload	Receive(1)	Transform Payload Length	Transform Payload
Next Payload	Receive(0)	KE Payload Length	KE Payload
0	Receive(0)	Nonce Payload Length	Nonce Payload
Next Payload	Receive(0)	Identity Payload Length	Identity Payload
ID Type	DOI Specific ID Data		
0	0	Hash Payload Length	Hash Payload
Hash Payload			

图 16 野蛮模式消息 2 结构

Initiator Cookie			
KE	Responder Cookie	Version	Exchange Type
Flags			
Message ID			
Total Message Length			
0	0	Hash Payload Length	Hash Payload

图 17 野蛮模式消息 3 结构

野蛮模式由于其在第一个消息中就携带了身份信息，因此其本身无法对身份信息进行加密保护，这降低了协商的安全性。但也因此使其不依赖于 IP 地址标识身份，在野蛮模式下有了更多灵活的应用：

1、对等体标识：主模式只能采用 IP 地址方式标识对等体；而野蛮模式可以采用 IP 地址方式或者 Name 方式标识对等体；

2、NAT 支持：主模式不支持 NAT 转换，而野蛮模式支持；

3、野蛮模式传输的消息更少，效率更高。

野蛮模式除了没有对身份信息进行加密外，还有一个缺点：由于野蛮模式交换次数的限制，因此野蛮模式的协商能力低于主模式。在野蛮模式中，由于第一个消息就需要交换 DH 消息，而 DH 消息本身就决定了采用哪个 DH 组，这样在提议转换对中就确定了使用哪个 DH 组，如果第一个消息中包含多个提议转换对，那么这多个转换对的 DH 组必须相同（和 DH 消息确定的 DH 组一致），否则消息 1 中只能携带和确定 DH 组相同的提议转换对。

IKE 第二阶段

无论第一阶段使用主模式还是野蛮模式交换建立 IKE SA，其目的都是进行身份认证并为第二阶段的交换提供保护。第二阶段交换的目的是为其他协议（如 IPsec）生成 SA，这一阶段是通过快速模式交换来实现的。通过一次主模式或野蛮模式交换，许多快速模式都可以完成。事实上，在单独一 IKE SA 的保护下，甚至可以并发地执行多个快速模式交换！

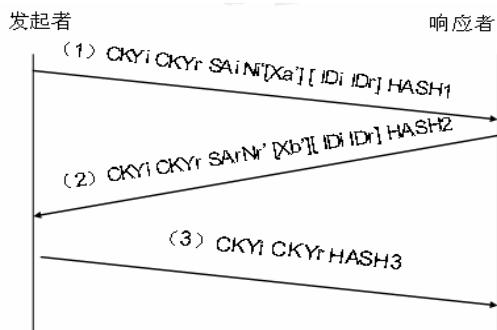
由于快速模式交换使用 SKEYID_a 对数据完整性和数据源身份进行验证，使用 SKEYID_e 进行加密，保障交换的机密性。

在一次快速交换模式中，通信双方需要协商拟定 IPsec 安全联盟的各项特征，并为其生成密

钥。快速模式需要从 SKEYID_d 状态中衍生出用于 IPsec SA 的密钥。随同交换的 nonce 以及来自 IPsec SA 的 SPI 及协议一道，这个密钥将在伪随机函数中使用，这样便可确保每个 SA 都有自己独一无二的密钥。所有 IPsec 密钥都是自相同的来源衍生的，所以相互间都有关联。假如一名攻击者能够根据 IKE SA 判断出 SKEYID_d 的值，那么就能非常容易地掌握自那个 SKEYID_d 衍生出来的任何 IPsec SA 的任何密钥。

快速模式为此专门提供了一个 PFS（完美向前保密）选项，来解决密钥之间相互无关性的需求。PFS 为可选项，用户可根据自己的安全需要选择是否启用 PFS。为了在快速模式交换中实现 PFS，需要执行一次额外的 Diffie-Hellman 交换，最终生成的共享秘密将在为 IPsec 生成密钥的过程中用到。显然，一旦交换完成，这个秘密便不复存在。一旦完成，它所驻留的那个内存位置必须清零和释放。从而保证了各个 SA 的密钥之间没有衍生关系，一个 SA 的密钥被破解不影响其他 SA 的密钥安全性。

快速模式的消息交换过程：



注：SA 负载用于协商 IPsec Proposal；当启用 PFS 时，才交换 Xa' 和 Xb'，并且在计算 Hash 和密钥时添加 Xa' 和 Xb' 的信息。ID 负载是可选的，如果接受方不支持 ID 负载，会回应一个 INVALID-ID-INFORMATION 的 Notify 错误。最后是 HASH 值交换用于认证。

下面以 PFS 情况为例，介绍一下快速模式

的消息结构：

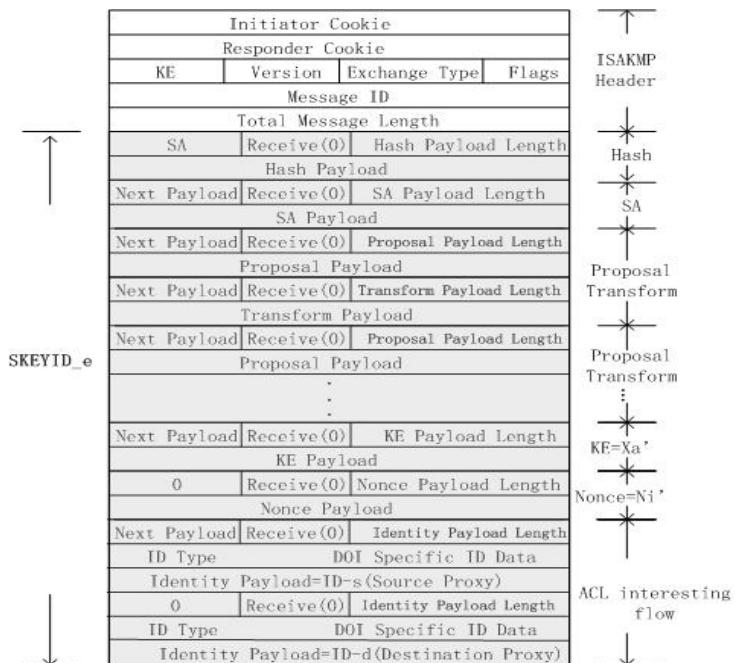


图 19 快速模式 PFS 下的消息 1 结构

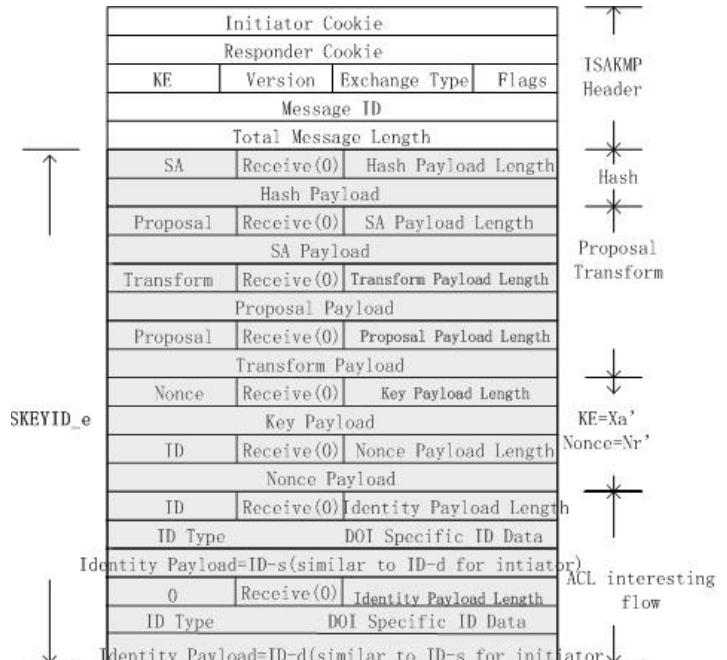
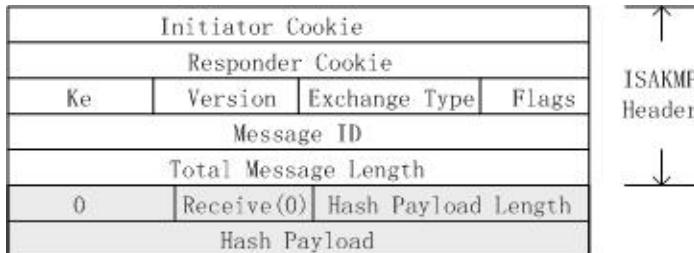


图 20 快速模式 PFS 下的消息 2 结构

以上消息交换完成后，对等体双方通过交换的 Xa' 和 Xb' 生成一个新的 DH 共享密钥，将其和

其和 IKE 第一阶段生成的 SKEYID_d 共享密钥以及 Ni' 和 Nr 还有 SPI 等信息生成最终用于 IPsec 加密的密钥。

发起者发送消息 3，用于验证响应者是否可以通信，相当于确认信息，响应者收到该信息以后就知道发起者已经接收到它在 IKE 第二阶段发送的消息 2，然后 IKE 第二阶段结束。



其他 IKE 交换

我们前面介绍的 IKE 交换都用于 SA 的创建，第一阶段交换创建的是 IKE SA，第二阶段交换创建的是 IPsec SA（或其他协议的安全联盟）。除以上的交换形式外，IKE 还定义了另外两种交换，一种用于 IKE SA 的维护，另一种用于通信双方的 Diffie-Hellman 组的协商。

IKE 通过信息交换，可以向对等体发送状态及错误提示消息。这实际上只是单独一条消息，它不需要确认，而且由于 IKE 要以 UDP 为基础，所以也不能担保肯定能够到达目的地。

已定义的另一种 IKE 交换是一种新式的组交换。IKE 总共定义了五个 Diffie-Hellman 组，通信各方均可使用。新的组交换允许各方协商拟定自己私有的组，并定义了组标识符，以便在将来的交换中对组进行指定。这种交换必须受到 IKE SA 的保护，因为要交换的都是敏感数据。新的组交换属于一种请求 / 响应交换。在这个过程中，发起者需要送出一个 SA 载荷，其中包含了提案的那个组的特征；以及一个标识符，编号在为私人专用保留的范围之外。举例来说，要想提议新组，它在一个质数模的基础上，使用乘幂运算，那么作为发起者，就必须送出相应的质数和底数。如果响应者能够接受这个组，便需用完全一样的信息作出回应。



初识 IKEv2

◆◇□ 徐庆伟

引言

IPsec 隧道两端通过共享密钥对 IP 报文提供私密性、完整性、访问控制以及数据源认证服务。共享密钥的建立可以手工建立，也可以通过协商方式自动建立。在 RFC2407、RFC2408 和 RFC2409 中定义了一种密钥协商机制棗IKE（Internet Key Exchange），为了与本文描述的 IKEv2 区别，称为 IKEv1。

IKEv2 保留了 IKEv1 的大部分特性，IKEv1 的一部分扩展特性被引入到 IKEv2 框架中，例如模式配置 (MODECFG)、扩展认证 (X-AUTH) 以及 NAT 穿越等，作为 IKEv2 协议的一个组成部分。另外 IKEv2 新引入了一些新的特性。

IKEv2 目前在 RFC4306 中定义，目前处于标准跟踪状态。本文主要介绍了 IKEv2 的协商交换过程以及 IKEv2 中的一些新特性。

IKEv2 的协商过程

要建立一对 IPsec SA，IKEv1 需要经历两个阶段：“主模式 + 快速模式”或者“野蛮模式 + 快速模式”。前者需要交换至少 9 条消息，后者也至少需要 6 条消息。而 IKEv2 如果只建立一对 IPsec SA，那么需要交换的报文数量大大减少，正常情况使用两次交换 4 条消息就可以完成一个 IKE SA 和一对 IPsec SA 的协商建立，如果要求建立的 IPsec SA 大于一对时，每一对 SA 只需要额外增加一次交换，也就是两个消息就可以完成。这比 IKEv1 要简化很多。下面简单介绍一下 IKEv2 的协商过程。

下面简单介绍一下 IKEv2 的协商过程。IKEv2 定义了三种交换：初始交换 (Initial Exchanges)、创建子 SA 交换 (CREATE_CHILD_SA Exchange) 以及通知交换 (INFORMATIONAL Exchange)。

2.1 初始交换 (Initial Exchanges)

初始交换至少应该包括两个交换四条消息。IKE_SA_INIT 交换 (2 个消息) 和 IKE_AUTH 交换 (2 个消息)。IKE_SA_INIT 交换和 IKE_AUTH 交换顺序完成后可以建立一个 IKE SA 和一对 IPsec SA。

1、IKE_SA_INIT 交换

1.1.1.1 1.1.a. 图 1 为 IKE_SA_INIT 交换中的第一个消息和第二个消息。这两个消息完成 IKE 安全联盟参数协商以及 KE 交换。这两个消息是通过明文方式交换的。

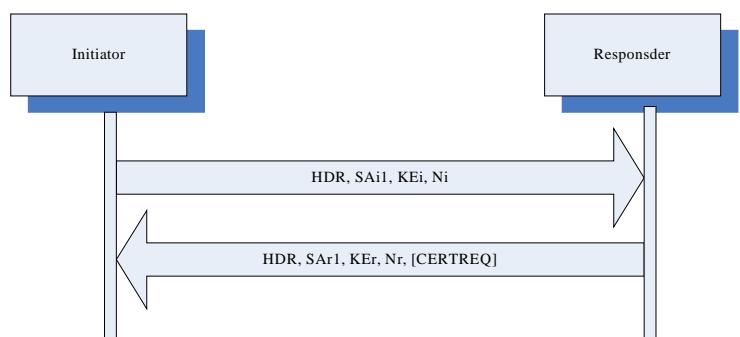


图 1 IKE_SA_INIT 交换

通过 IKE_SA_INIT 交换后可以生成一个共享密钥材料 SKEYSEED。通过 SKEYSEED 可以得到其他七个相关密钥。这七个密钥（材料）分别是：

SK_d：为子安全联盟 (CHILD_SA) 生成密

钥的密钥材料；

SK_ ai: 后续 IKE 交换进行用于发起者完整性保护的认证密钥

SK_ ar: 后续 IKE 交换进行用于响应者完整性保护的认证密钥

SK_ ei: 后续发起者 IKE 交换报文加解密密钥

SK_ er: 后续响应者 IKE 交换报文加解密密钥

SK_ pi: 发起者生成 AUTH 载荷的材料

SK_ pr: 响应者生成 AUTH 载荷的材料

各密钥材料计算公式如下：

$$\begin{aligned} \text{SKEYSEED} &= \text{prf}(\text{Ni} \mid \text{Nr}, g^{\wedge}ir) \\ \{\text{SK}_d \mid \text{SK}_ai \mid \text{SK}_ar \mid \text{SK}_ei \mid \text{SK}_er \mid \text{SK}_pi \mid \text{SK}_pr\} &= \text{prf} + (\text{SKEYSEED}, \text{Ni} \mid \text{Nr} \mid \text{SPIi} \mid \text{SPIr}) \end{aligned}$$

prf+ 的定义：

$$\text{prf+}(\text{K}, \text{S}) = \text{T1} \mid \text{T2} \mid \text{T3} \mid \text{T4} \mid \dots$$

其中：

$$\text{T1} = \text{prf}(\text{K}, \text{S} \mid 0x01)$$

$$\text{T2} = \text{prf}(\text{K}, \text{T1} \mid \text{S} \mid 0x02)$$

$$\text{T3} = \text{prf}(\text{K}, \text{T2} \mid \text{S} \mid 0x03)$$

$$\text{T4} = \text{prf}(\text{K}, \text{T3} \mid \text{S} \mid 0x04)$$

从上面的交换可以看出，在第一个消息中发起者就携带了 KE 载荷，但是在 D-H 交换中，公共值的计算是基于 D-H 组的。不同的 D-H 组计算的 KE 是不同的。既然如此，在第一个消息中 D-H 组选择哪一个呢？在 IKEv2 中采用的“猜”的办法，也就是发起者猜一个响应者最可能使用的 D-H 组。响应者根据发起者“猜”的组来响应发起者，这样就可以通过两条消息完成密钥材料交换。

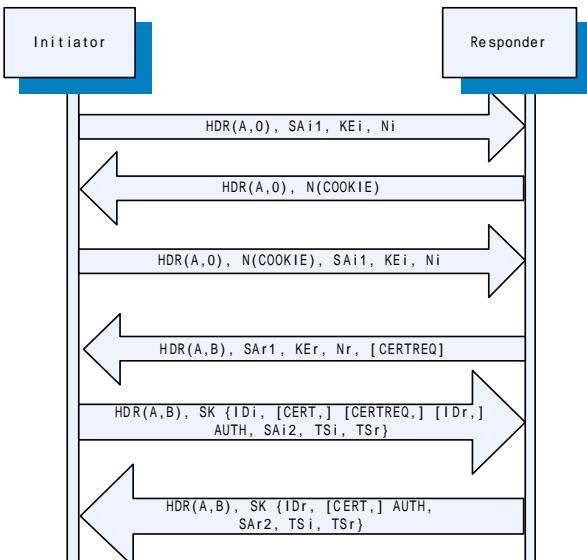
但是这个“猜”出来的 D-H 组也必须告诉响应者，在 IKEv1 中我们知道单独 KE 载荷是不能完成这个工作的，在 IKEv2 中，KE 载荷进行了扩展，增加了一个 D-H 组的字段，通过 KE 载荷可以通告发起者计算 KE 使用的 D-H 组。

还有一个问题需要解决，就是如果发起者“猜”的 D-H 组响应者不支持怎么办呢？在 IKEv2 中规定，如果响应者不支持发起者使用的 D-H 组，则响应者必须拒绝发起者的这个消息，并且向发

起者回应一个带有 INVALID_KE_PAYLOAD 载荷的消息。发起者一旦收到这个拒绝消息，则需要选择另一个 D-H 组进行重新计算 KE 值，然后重新发起一个请求。直到响应者接受请求或者发起者所有的 D-H 组响应者都不支持。前者继续进行下面协商，后者协商终止。

在 IKE_INIT_SA 交换中消息是明文传输的，特别是响应者接收到第一个消息后无法确认发送者是否是一个地址欺骗的报文。如果此时一个网络攻击者伪造大量地址向响应者发送大量 IKE_INIT_SA 请求，根据 IKEv1 协议，响应者需要维护这些半开的 IKE 会话信息，从而耗费大量响应者的资源，造成对响应者的 DoS 攻击。为了解决 DoS 攻击问题，IKEv2 提出了使用 COOKIE 类型 Notify 载荷的解决方案。具体方法如下：

当响应者发现存在过多的半开 IKE_SA 的时候，就启用这种 COOKIE 解决方案。首先对于新的 IKE_INIT_SA 请求，响应者将使用明文向请求地址发送一个包含 COOKIE 类型 Notify 载荷的响应消息，其中包含一个响应者期望发起者回送的 COOKIE 值。如果对端是一个真正的 IKE 发起者，那么它需要构造一个 Notify 载荷，包含从响应者接收到的 COOKIE，并且放到原来消息第一个载荷前，重新向响应者发起请求。当响应者接收到了包含期望 COOKIE 的请求后就可以继续进行其他消息交换了。如下图所示



以上机制是否可以拒绝 D o S 攻击呢？在响应者发出 COOKIE 后它必须保留 COOKIE 与发起者之间的对应关系。如果这种对应关系是一个有状态的，那么 D o S 攻击实际上还是发生了。在 IKEv2 中要求实现无状态的 COOKIE 方法与发起者进行对应，响应者在向发起者发送期望 COOKIE 后不需要维护与发起者的有状态对应关系，而是通过一种计算公式实现无状态的对应关系。具体采用什么样的计算方法，应该是实现的问题，不影响互通。在 IKEv2 中建议了一种计算方法：

```
COOKIE = <VersionIDofSecret> | Hash(Ni | IPi | SPIi | <secret>)
```

以上计算公式中使用 (Ni, IPi, SPIi) 来作为发起者的标识，使用 (VersionIDofSecret, secret) 来保证 COOKIE 的不可猜测性。secret 是只有响应者知道的秘密，并且是定期变化的，一旦 secret 发生变化，VersionIDofSecret 也要跟随变化，这样就保证了 COOKIE 的唯一性和不可猜测性。

2、IKE-AUTH 交换

从 IKE-AUTH 交换开始，所有报文都必须加密交换。IKE-AUTH 交换至少两个消息。发起者发送的消息采用 SK_ei 加密和 SK_ai 认证；响应者发送的消息采用 SK_er 和 SK_ar 进行加密和认证。图 3 中 SK { ... } 表明消息使用了各自方向上密钥进行了消息加密和认证。

从图图 3 中可以看出，交换报文中还包含了

自己的 ID 载荷、认证载荷，说明该交换用于身份认证。

另外除了身份认证信息外，该交换报文中还包含了 SA 载荷，流量选择符载荷，这些信息可以用于协商 IPsec SA。也就是在 IKE-AUTH 两个报文的交互中完成了身份认证以及一个 IPsec SA 的创建过程。在 IKEv2 中认证方式只支持公钥签名认证和预共享密钥认证。

AUTH 载荷的生成：如果使用数字签名，对于响应者来说，AUTH 载荷签名的数据为 IKE-SA-INIT 交换中第二条消息（响应者发送的第一个消息）的从 SPI 开始（包含 SPI）到最后一个载荷结束的所有字节，另外加上发起者在第一个消息中发送的 Ni（只是 nonce 值，不包含载荷头），以及 prf(SK_pr, IDr') 的值 (IDr' 仅为响应者的 ID，不包含 ID 载荷头固定信息)。类似对于发起者，AUTH 载荷签名的数据为 IKE-SA-INIT 交换中第一条消息（发送者发送的第一个消息）的从 SPI 开始（包含 SPI）到最后一个载荷结束的所有字节，另外加上响应者在第二个消息中发送的 Ni（只是 nonce 值，不包含载荷头），以及 prf(SK_pr, IDi') 的值 (IDi' 仅为发送者的 ID，不包含 ID 载荷头固定信息)。特别需要注意的是，IKE-SA-INIT 交换发起者发送的第一个消息可能会发送多次（其中一个原因前面已经讲到，DH 组猜错的情况），那么对于发起者来说实施签名的报文将不是第一个消息，而是最后发送成功的一个请求消息。

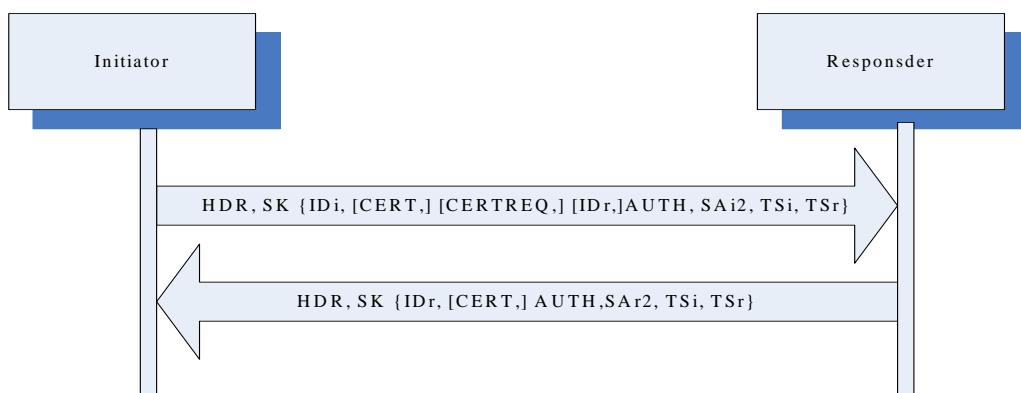


图 3 IKE-AUTH 交换

如果使用预共享密钥方式，A U T H 载荷的计算方法如下：

$$\text{AUTH} = \text{prf}(\text{prf}(\text{Shared Secret}, \text{"Kev Pad for IKEv2"}), \text{<msa octets>})$$

在 I K E _ A U T H 交换中由于消息已经被保护，因此理论上在这个交换中就可以进行 I Psec S A 的协商了，实际上确实在这个阶段产生了一个 I P s e c S A。在这个阶段 I Psec S A 的密钥材料生成公式为：

$$\text{KEYMAT} = \text{prf}(\text{SK d}, \text{Ni} | \text{Nr})$$

一旦生成这个密钥材料后，一个 I Psec S A 的所有密钥将从这个密钥材料中按照一定的规则衍生出来。

2.2 创建子 S A 交换 (CREATE_CHILD_SA Exchange)

当一个 I K E S A 需要创建多个 I Psec S A 的时候，需要使用 C R E A T E _ C H I L D _ S A 交换来协商多于一个的 S A。另外 C R E A T E _ C H I L D _ S A 还可以用于进行 I K E S A 的重协商。

C R E A T E _ C H I L D _ S A 包含一个交换两个消息。在 I K E v 1 中这个交换称为阶段 2 交换（快速模式交换）。这个交换必须在 I K E 初始交换完成之后才能进行，交换的发起者可以是 I K E 初始交换的发起者，也可以是 I K E 初始交换的响应者。在交换中的两个消息需要由 I K E 初始交换协商的密钥进行保护。如图 4 所示。

最简单情况下交换包含一个 S A 载荷和一个 N i 载荷。在这种情况下密钥材料计算公式为：

$$\text{KEYMAT} = \text{prf}(\text{SK d}, \text{Ni} | \text{Nr})$$

类似 I K E v 1 中的 P F S，C R E A T E _ C H I L D _ S A 可以重新进行一次 D - H 交换，交换新的 K E 值，生成新的 D H 公共值 ($g^{\wedge}ir$ (new))。在这种情况下密钥材料计算公式为：

$$\text{KEYMAT} = \text{prf}(\text{SK d}, g^{\wedge}ir(\text{new}) | \text{Ni} | \text{Nr})$$

在 I K E v 1 的 P F S 中，D H 组是不能协商的，只能使用 I K E 协商出的 D H 值。但是在 I K E v 2 中 C R E A T E _ C H I L D _ S A 在 S A 载荷中可以包含新的 D H 组，K E 载荷由新的 D H 组计算出来。这样同样存在一个新的 D H 组响应者不接受的问题，这种情况下需要与初始交换一样的处理。

生成密钥材料后，子 S A 的所有密钥都从 K E Y M A T 中按照一定规则衍生出来。

在图 4 中的消息报文存在一个 N 载荷，这个载荷的位置在所有载荷之前，该载荷表明这个交换是一个重新 I K E S A 的交换，而不是创建子 S A 交换。关于使用 C R E A T E _ C H I L D _ S A 交换进行 I K E S A 重协商稍后介绍。

2.3 通知交换 (INFORMATIONAL Exchange)

运行 I K E 协议的两端有时候会传递一些控制信息，例如错误信息或者通告信息，这些信息在 I K E v 2 中是通过 I N F O R M A T I O N A L 交换完成的。

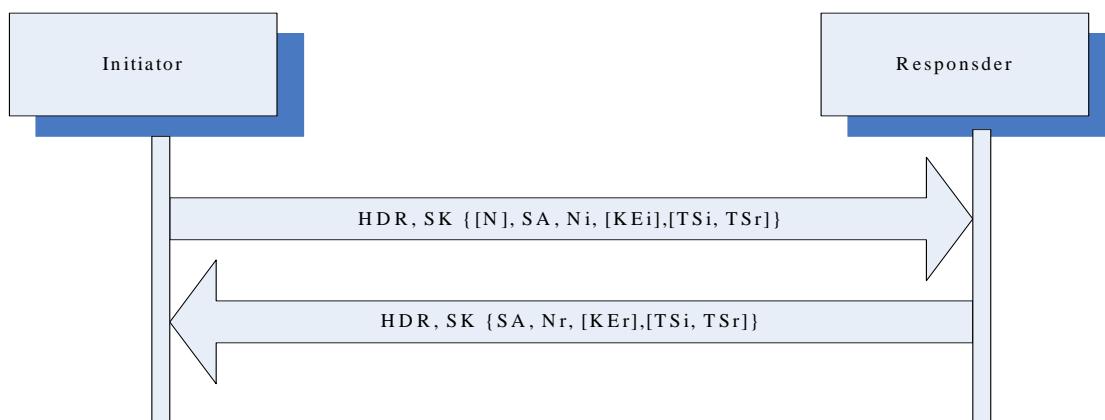


图 4 C R E A T E _ C H I L D _ S A 交换

INFORMATIONAL 交换必须在 IKE SA 保护下进行，也就是说 INFORMATIONAL 交换只能发生在初始交换之后。INFORMATIONAL 可能是 IKE_SA 的控制消息，那么此时 INFORMATIONAL 交换必须由该 SA 来保护进行；如果 INFORMATIONAL 是某子 SA (CHILD_SA) 的控制消息，那么该 INFORMATIONAL 交换必须在生成该子 SA 的 IKE SA 的保护下进行。

INFORMATIONAL 交换如图 5 所示。在 INFORMATIONAL 交换中可以包含通知载荷 N (Notification)，配置载荷 CP (Configuration)、删除载荷 D (Delete) 等。这些载荷在 INFORMATIONAL 交换中都是可选的。也就是说在 INFORMATIONAL 交换中可以不包含任何载荷。具体包含哪些载荷，要由交换的功能来定。

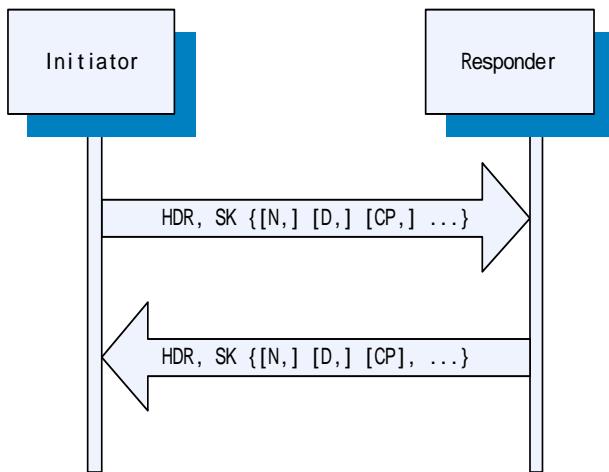


图 5 INFORMATIONAL 交换

INFORMATIONAL 交换在 IKE 中主要用于传递控制信息，下面列举的是有限的应用。

2.3.1 SA 的删除

SA 删除包括删除 IKE_SA 和 CHILD_SA (IPsec SA)。在 IKEv2 中使用 INFORMATIONAL 交换包含 delete 载荷删除 SA。Delete 载荷格式如图 6 所示：

Delete 载荷类型为 42。在 Delete 载荷中，协议标识 (Protocol ID) 用于标识该载荷所删除的 SA 的协议类型。取值可以是 1 (IKE_SA)，2 (AH)

01234567890123456789012345678901

Next Payload	C	RESERVED	Payload Length
Protocol ID		SPI Size	# of SPIs
Security Parameter Index(es) (SPI)			

图 6 Delete 载荷格式

或 3 (ESP)。SPI Size 字段用于标识 SPI 的长度，对于 IKE 来说 “SPI Size” 必须为 0，对于 AH 和 ESP 必须为 4。“# of SPIs” 为在载荷中包含的 SPI 的数量，这个值和 “SPI Size” 共同决定了下面字段 SPI 的长度。Security Parameter Index(es) (SPI) 字段决定需要删除的 SA，该字段是变长的，长度由 “SPI Size” 和 “# of SPIs” 域决定。

SA 的删除过程如下：

1、删除 SA 发起者选择要删除的 SA。对于 AH 和 ESP 来说，SA 是有方向的，删除 SA 的发起者只能选择入方向的 SA。

2、创建 delete 载荷。对于一个载荷，协议 ID 必须是唯一的，因此在同一个 delete 载荷中，只能存在一个协议的 SPI。不同的协议需要创建多个不同的 delete 载荷。

3、将 delete 载荷封装在 INFORMATIONAL 交换消息中发送给对端接收者。并等待接收者的响应消息。

4、接收者接收到包含 delete 载荷的 INFORMATIONAL 消息后，必须关闭 delete 载荷中指定的 SA，同时需要给发起者回应一个响应消息，如果是删除的成对的 SA，必须删除本地入方向的 SA (发起者出方向的 SA)，并在响应报文中的 delete 载荷包含本地入方向的 SA (发起者出方向的 SA) 的 SPI。

5、发起者接收到接收者的响应报文后关闭本地出方向的SA。

以上删除过程中接收者接收到发起者发送的消息后必须发送相应消息进行响应，这一点和IKE v1不同。如果在一定时间内发送者没有收到响应者的相应消息，那么发送者认为消息发送失败，会进行消息重传。

2.3.2 通过INFORMATIONAL交换请求对端版本信息

在一些情况下，通信的一方可能需要知道对端的软件版本信息。在IKE v2中增加了一种机制，可以通过INFORMATIONAL交换请求对端版本信息。在INFORMATIONAL交换中携带配置载荷(CP)来请求对端版本信息。有些系统可能因为安全考虑不愿意发布自己的版本信息，这种情况下可以通过INFORMATIONAL交换消息回应请求者一个空字符串或者不携带CP载荷的消息，但是无论如何都要给请求者以响应，因为IKE v2所有报文都是要求确认的。下面图7是一个版本请求的例子。

2.4 SA重协商

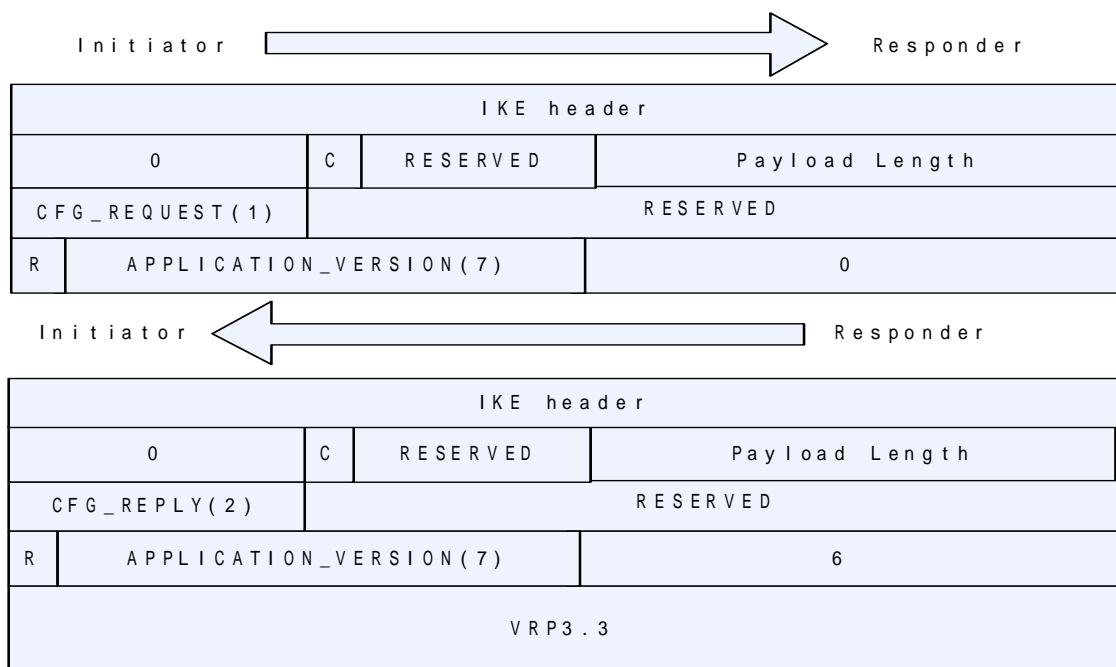


图7 版本请求举例

在IKEv2中使用`CREATE_CHILD_SA`交换来重协商SA。“CREATE_CHILD_SA”交换如图4所示。需要注意的是在重协商中流量选择符(TS)负载是不需要的。重协商中密钥材料计算公式为：

$$\text{SKEYSEED} = \text{prf}(\text{SK_d (old)}, [\text{g}^{\wedge} \text{ir (new)}] | \text{Ni} | \text{Nr})$$

其中`SK_d (old)`是当前使用的IKE SA的`SK_d`; `g^ir (new)`是重协商DH交换协商的公共值,该参数可选; `Ni`和`Nr`是重协商中交换的随机值。

在图4中重协商发起者发起请求后,响应者应该首先创建能够接收消息的新SA,之后才能回一个响应消息给发起者。这样对于发起者来说,一旦接收到响应者的响应报文,就可以知道对端SA已经建立起来,可以使用新的SA发送报文了。对于响应者来说是否发送了响应消息后立即可以使用新的SA保护发送报文呢?IKEv2规定了之后在下面两种情况有其中一种发生后响应者才能知道发起者的新SA已经就绪。一是响应者接收到了发送者使用新SA发送的报文;其二是响应者接到了关闭老SA的删除消息。当以上两个条件满足其一,响应者就可以使用新的SA发送报文了。但是如果以上两个条件不满足,此时响应者需要等待一个超时时间,当超时后响应者可以使用新的SA发送报文。以上两种情况都不满足的情况,可能是发起者没有要发送的报文,或者网络发生丢包现象。为了避免前者发生,IKEv2重协商发起者可以在接收到响应报文后发送一个哑元(`dummy`)报文给对端;解决后者需要在响应端设置一个超时时间,一旦超过超时时间,响应者可以使用新的SA发送报文。但是对于第二种情况可能存在一个问题,就是如果响应者发送的对`CREATE_CHILD_SA`请求的响应报文在网络传输中丢失,那么发起者在接收到响应之前是不会发送任何报文的,因此此时响应者可能等待超时后就向发起者发送了使用新的SA保护的报文,一旦发起者在这种情况下接到了使用新的SA保护的报文,那么发起者要认为回应消息丢失,需要重新发送`CREATE_CHILD_SA`请求消息。

2.5 IKEv2报文确认重传机制

与IKEv1不同, IKEv2中所有消息都是以“请求-响应”形式成对出现的,发起者发送的报文都响应者都要进行确认,例如建立一个IKE SA一般需要两个“请求-响应”对(四条消息)。如果在规定时间内没有接收到确认报文,需要对报文进行重传处理。

IKEv2重传只能由发起者发起,响应者在接收到发起者的重传请求之前不能发送重传响应报文。也就是说在发起者在接收到响应报文之前必须保留请求报文,响应者在接收到新的请求报文之前必须保留响应报文。



图8 IKEv2报文格式

如何判断响应者接收到了新的请求呢?图8是IKEv2的报文格式,从报文格式可以看到存在一个Message ID字段,该字段用于标识一个“请求-响应”对。Message ID字段占四个字节,最大值为 $2^{32}-1$,第一个请求报文总是从0开始。因此IKEv2初始交换这个值总是0和1。当达到最大值的时候需要进行重新进行IKE-SA的协商,并且将该值重置为0。

为了增加IKEv2协商的吞吐量,引入了请求序列窗口的概念。响应者可以通过“SET_WINDOW_SIZE”通知消息通知发起者它可以接收处理多个请求,并在消息中携带所允许的序列号窗口大小,此时发起者可以在不等待响应的情况下发送多个请求消息,但是发送的数量不能超过窗口规定大小。此时由于网络原因可能会导致请求到达响应者顺序与发送顺序不一致,因此

要求响应者要能处理请求乱序的情况，这样对实现的要求比较高，因此这个特性并不要求必须实现。

IKEv2 继承的 IKEv1 的扩展特性

3.1 EAP 认证

在 IKEv1 中有一个扩展认证特性“X-AUTH”，主要用于远程用户通过 IPsec 接入时的客户端认证。X-AUTH 认证机制在基于用户的接入认证方面

提供了较好的安全机制。它可以借助其他认证服务器完成认证，例如 Radius Server、TACACS + Server 等。在 IKEv1 中 X-AUTH 发生在 IKE 协商第一阶段和第二阶段之间，因此有时候说 X-AUTH 发生在 IKE 1.5 阶段。

图 9 中显示了一个实际远程客户端接入 IPsec 的报文交互过程。在报文 6~9 即为 X-AUTH 交换。图 10 为报文交互过程示意图，X-AUTH 使用模式配置报文进行协商交互，图 11 为交互报文格式。关于 X-AUTH 本刊有专题讨论，此处不再详细解释。

No.	Time	Source	Destination	Protocol	Info
3	0.999819	10.1.1.11	10.1.1.2	ISAKMP	Aggressive
4	1.113247	10.1.1.2	10.1.1.11	ISAKMP	Aggressive
5	1.119294	10.1.1.11	10.1.1.2	ISAKMP	Aggressive
6	1.128093	10.1.1.2	10.1.1.11	ISAKMP	Transaction (Config Mode)
7	3.704616	10.1.1.11	10.1.1.2	ISAKMP	Transaction (Config Mode)
8	4.008021	10.1.1.2	10.1.1.11	ISAKMP	Transaction (Config Mode)
9	4.008540	10.1.1.11	10.1.1.2	ISAKMP	Transaction (Config Mode)
10	4.030016	10.1.1.11	10.1.1.2	ISAKMP	Transaction (Config Mode)
11	4.040187	10.1.1.2	10.1.1.11	ISAKMP	Transaction (Config Mode)
12	4.072906	10.1.1.11	10.1.1.2	ISAKMP	Quick Mode
13	4.086716	10.1.1.2	10.1.1.11	ISAKMP	Informational
14	4.090007	10.1.1.2	10.1.1.11	ISAKMP	Quick Mode
15	4.090479	10.1.1.11	10.1.1.2	ISAKMP	Quick Mode

图 9 远程客户端使用 IPsec 接入时 IKE 协商过程

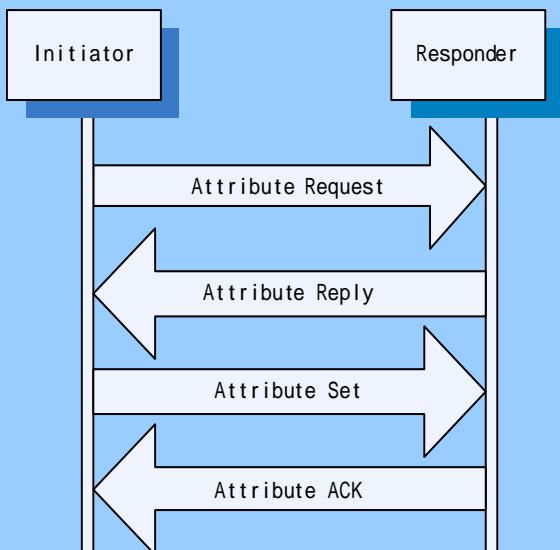


图 10 X-AUTH 交换

ISAKMP HEADER		
Next payload	Reserved	Attribute len
Type	Reserved	Identifier
Attributes		
0	0	Hash payload len
Hash		

图 11 Config Mode 报文格式

IKEv1 的 X-AUTH 扩展特性在 IKEv2 中得到了保留，但是以 EAP 认证的方式进行了定义。在报文交互以及机制上与 X-AUTH 有不同，IKEv2 中基于 RFC3748 的 EAP 认证，EAP 报文被封装在 EAP 载荷中，通过在 IKE 报文中传递 EAP 载荷的实现 EAP 认证。

EAP 载荷是在通用载荷头后追加一个 EAP 消息构成。EAP 载荷如图 12 所示。

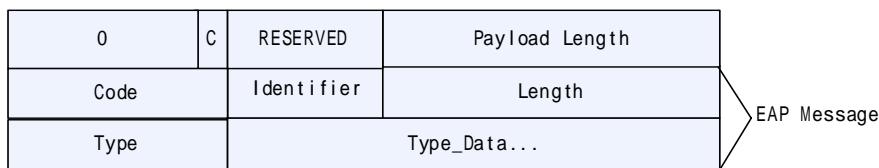


图 12 EAP 载荷

当 IKE 协商发起者在发送第三条消息的时候如果包含 ID 载荷但是却不包含 AUTH 载荷，即发

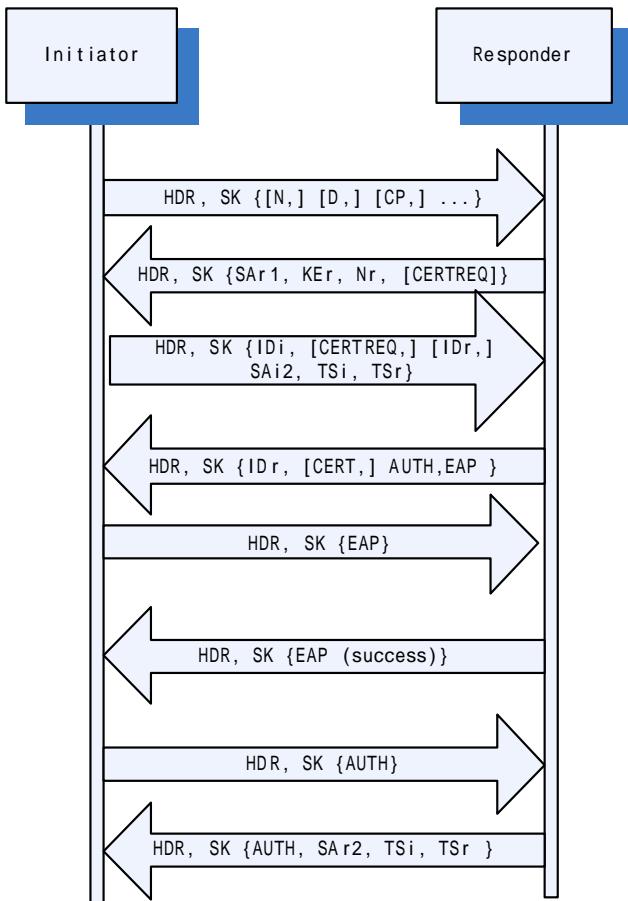


图 13 EAP 方法认证报文交互过程

起者向响应者发送了一个没有经过验证的 ID，此时说明发送者要求启用 EAP 方法认证。如果响应者此时愿意以 EAP 方法认证，那么响应者在第四个响应消息中会包含一个 EAP 载荷，图 13 是使用 EAP 方法认证的报文交互过程。

从图中的报文交互过程可以看出，第三个报文中有 ID 载荷，但是没有 AUTH 载荷，说明发起者希望进行 EAP 方法认证。响应者接收此报文后

发现 ID 没有通过认证，因此返回一个包含 EAP 载荷（EAP 请求）的 IKE 报文（第四个消息）。此消息同时还包含 ID 载荷、AUTH 载荷。此处 AUTH 载荷的计算与 IKE-AUTH 认证中载荷计算方法相同，注意这个消息与只有 IKE-AUTH 认证的第四个消息的区别是没有包含 SA 载荷和 TS 载荷，这些载荷要在最后消息中携带。消息 5 ~ 消息 6 完成了 EAP 认证过程。消息 7 ~ 消息 8 类似 IKE-AUTH 中消息 3 ~ 消息 4。但是在 EAP 方法中 AUTH 计算方法有所要求。AUTH 载荷的计算方法为：

对于能够生成共享密钥的 EAP 方法，在 EAP 中这个共享密钥是 MSK（参考 RFC3748）。AUTH 的计算方法为：

$$\text{AUTH} = \text{prf}(\text{prf}(\text{Shared Secret}, \text{"Key Pad for IKEv2"}), \langle \text{msg octets} \rangle)$$

对于不能够生成共享密钥的 EAP 方法，IKEv2 不建议使用。如果一定要使用，那么需要分别使用 SK_pi 和 SK_Pr 生成 AUTH 载荷。

3.2 通过配置载荷获取内部地址

在图 14 所示的情况下，移动用户如果要以 IPsec 方式接入企业内部网络，常用的解决方案有 L2TP+IPsec，例如华为-3COM 远程用户接入方

案。这种方案实际上是首先在移动用户与企业网关之间建立一个穿越公网的私网隧道，然后对隧道数据进行加密的一种解决方案。不难想到 IPsec tunnel 模式实际上也可以完成以上功能，唯一不同是 L2TP+IPsec 方案使用了 PPP 地址协商解决对移动用户的地址分配问题，如果能够在 IPsec SA 建立之前能够为 IPsec 隧道移动端点分配一个私网地址，那么这样也能解决远程用户接入的问题。CISCO 的远程接入方案就采用了这种方式，在 IPsec SA 创建之前由 VPN 网关将分配的私网地址“推”给客户端，这种方式使用一种被称为模式配置 (MODE - CONFIG) 的技术，实际上是对 IKEv1 的一种扩展技术。当然模式配置不仅可以实现隧道地址分配，还可以实现 DHCP、DNS 等 Server 地址的分配。

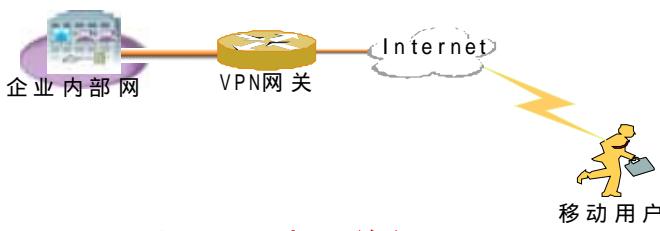


图 4 远程用户访问模型

图 9 中消息 10 ~ 11 是一个模式配置的报文交互实例。图 15 是模式配置报文交换示意图。关于报文格式，请参考图 1-1。第一个消息是“Attribute Request”消息，类型为 ISAKMP_CFG_REQUEST(1)，属性部分 (Attributes) 为空。第二个消息为“Attribute Reply”，类型为 ISAKMP_CFG_REPLY(2)，属性部分 (Attributes) 可以是 INTERNAL_IP4_ADDRESS(1)、INTERNAL_IP4_NETMASK(2)、INTERNAL_IP4_DNS(3)、INTERNAL_IP4_NBNS(4)、INTERNAL_IP4_ADDRESS_EXPRY(5) 等。关于模式配置细节，请参考 [3]。

IKEv2 中对模式配置技术进行了继承，只是形式进行了一些改变。下面介绍通过配置载荷 (CP 载荷) 进行地址请求的实现。

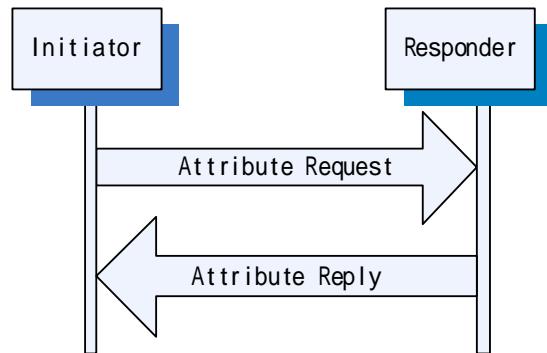


图 5 INFORMATION 交换

地址请求一定要在 IPsec SA 创建之前完成，由于在 IKE-AUTH 交换中同时创建一个 IKE-SA 和一个 IPsec SA，因此地址请求的 CP 载荷必须包含在 IKE-AUTH 交换消息中。关于 CP 载荷的位置，IKEv2 规定在 IKE-AUTH 交换中，CP 载荷必须包含在含有 SA 载荷的消息中，并且在同一消息中必须位于 SA 载荷之前。根据 CP 载荷包含属性不同可以请求不同的地址信息，例如隧道端点地址、DNS 服务器地址、DHCP 服务器地址等，但是在一个 CP 中 INTERNAL_ADDRESS 必须是唯一的，要么 IPv4，要么 IPv6。图 16 是在 IKE-AUTH 交换中包含地址请求的情况。

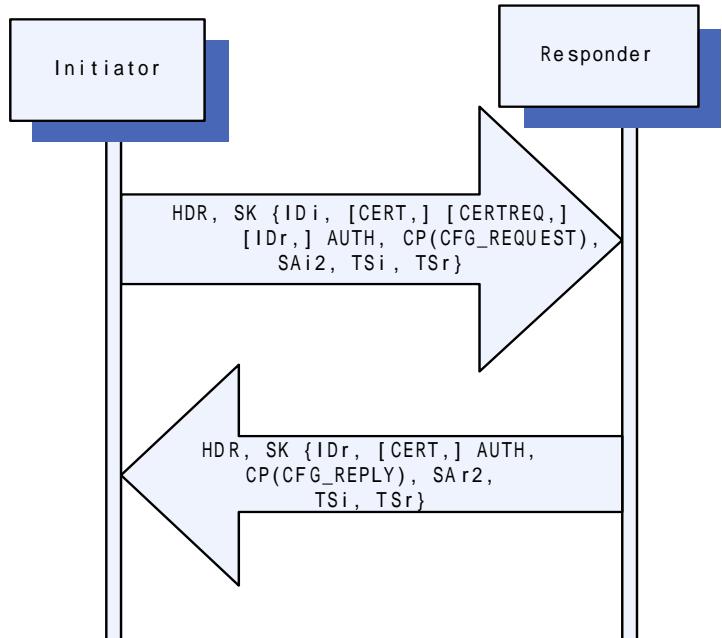


图 6 IKE_AUTH 中地址请求报文交互

在地址请求交互中，响应者的配置响应必须是对配置请求的应答，响应者不能在无请求的时候给发起者发送一个包含配置响应的 CP 载荷。响应者如果将配置载荷与某一个 ID 绑定，那么一个非法的配置请求 CP 载荷将导致整个 IKE 交换失败。下面图 17 是一个地址交换的例子：

在图 17 的例子中发起者向响应者请求了 INTERNAL_IP4_ADDRESS、INTERNAL_IP4_NETMASK 和 INTERNAL_IP4_DNS，但是响应者却回应了请求中 INTERNAL_IP4_ADDRESS、INTERNAL_IP4_NETMASK，以及请求中没有的一个属性 INTERNAL_IP4_SUBNET，没有回应 INTERNAL_IP4_DNS 请求属性。在 IKEv2 中请求属性与响应属性是独立的，响应可以包含请求的属性，也可以忽略请求中的非必选属性，例如 INTERNAL_IP4_DNS 在本例中被响应者忽略，甚至可以包含请求中没有的其他属性，例如本例中的 INTERNAL_IP4_SUBNET。

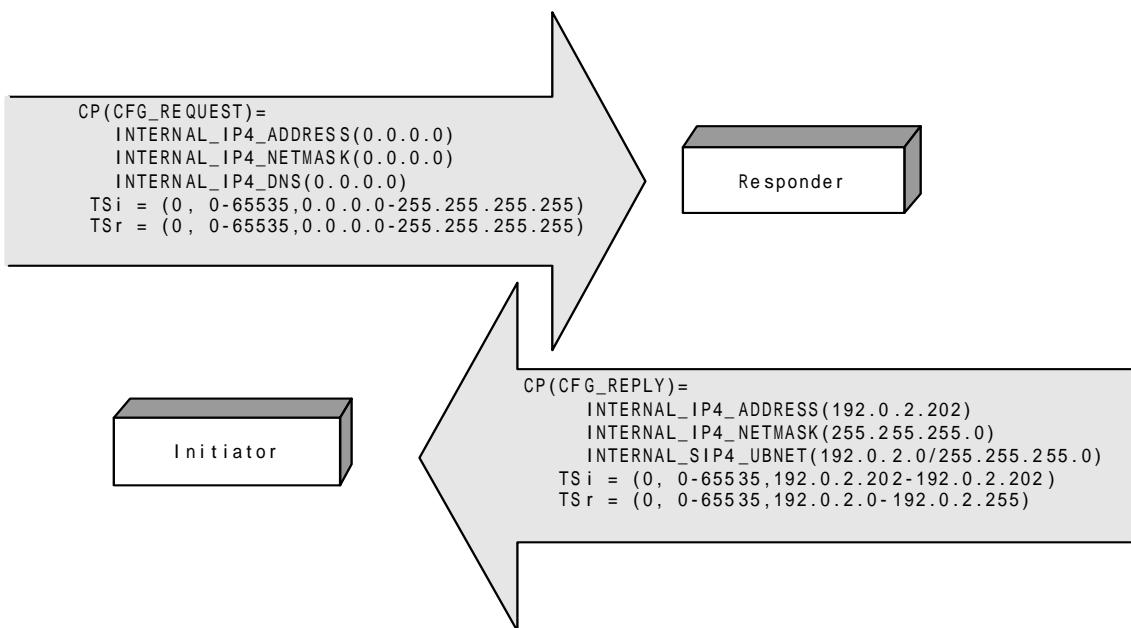


图 17 内部地址请求实例

参考资料

- [1] Ramakrishnan, K., “Internet Key Exchange (IKEv2) Protocol”, RFC4306, December 2005.
- [2] Saadat Malik, “Press.Network.Security.Principles.and.Practices”, November 15, 2002
- [3] Vijay Bollapragada, Mohamed Khalid, Scott Wainner, “IPSec VPN Design”, April 07, 2005



Features

DPD

◆ ◇ □ 刘胜伟

概述

描述目前被广大设备生产商使用的 IKE Peer 状态检测方法棗 Dead Peer Detection。这种方法使用 IPsec 流量来最小化 peer 状态检测所需消息报文的数量。

IPsec 是一种对等体到对等体的技术，要在 IPsec 对等体之间建立 IPsec 会话，它们之间必须有 IP 连接性。由于路由选择问题、对等体重启或其它原因，对等体之间可能失去 IP 连接性，IKE 和 IPsec 通常无法知道这一点。IKE 是基于 UDP 的，因此是无连接的；在生存期到期之前，对等体之间的 IKE 和 IPsec 安全关联 (SA) 将一直存在。IPsec 会话的中断将引发“黑洞”，导致数据流丢失。迫切需要尽早地发现这种“黑洞”，这主要是因为会话的一方继续对发往不可达对等体的数据流进行加密，这将占用宝贵的 CPU 资源。其次，由于无法检测到对等体故障，使得无法激活备用对等体。最后，到对等体的可达性恢复后，对等体可能没有 IKE 或 IPsec SA，导致加密的分组到达对等体后由于没有有效的 SA 而被丢弃。另外，旧的 IKE 和 IPsec SA 可能仍在，但却不是有效的，从而妨碍了创建新的 IPsec SA。

存活机制可帮助解决这种问题。它是对等体通过周期性交换 HELLO / ACK 消息来告知对方自己处于活动状态。这种方法有单向和双向之分。所谓单向是指只发 HELLO 包，而双向是指使用 HELLO 和 ACK 对。通常提到的“heartbeat”可以认为是单向消息，而“keepalive”是双向消息。

IKE 存活机制在检测失效 IKE 对等体、防范黑洞和节省昂贵的 CPU 资源方面很有效；但其局限性之一是可扩展性太低。在端接几千个场点到场点的 IPsec 会话的 IPsec 路由器上，如果启用 IKE 存活机制，将消耗昂贵的 CPU 周期来处理 IKE 存活消息，这限制了该路由器可端接的 IPsec 会话的数量。

最后，许多厂商都采用了自己的方法来检测 peer 的状态，而不使用周期发送消息的机制。失效对等体检测 (Dead Peer Detection, DPD) 就是 IKE 存活机制的一种替代机制。

存活机制

2.1 基本原理

下面介绍一下通过周期性交换消息检测 peer 状态这种机制的基本原理：

IKE 协议本身没有提供检测 peer 状态的机制，一旦发生 peer 不可达的情况，除了等待 IKE SA 超时外别无他法。这样在 SA 的剩余生存周期内，peer 之间一直处于非连通状态。所以一个能够随时检验 peer 状态的方法是必需的。所以使用 IKE 通告来查询 peer 状态的方法出现了。这些方法依赖双向的“keepalive”或者是单向的“heartbeat”消息交换来实现。

2.2 Keepalives 和 Heartbeats

2.2.1 Keepalives

使用 keepalives 存活机制的 peer A 和 peer B 之间需要有对方在线的定期规律确认。这种确认消息以认证通告负载方式进行交换。Peer 之间必须在 keepalives 报文的发送时间间隔上统一，这就意味着在第一阶段的一些参数协商是必须的。

假设 peer A 和 peer B

1.1 Keepalives 和 Heartbeats

1.1.1 Keepalives

1.1 Keepalives 和 Heartbeats

1.1.1 Keepalives

使用 keepalives 存活机制的 peer A 和 peer B 之间需要有对方在线的定期规律确认

2.2 Keepalives 和 Heartbeats

2.2.1 Keepalives

使用 keepalives 存活机制的 peer A 和 peer B 之间需要有对方在线的定期规律确认。这种确认消息以认证通告负载方式进行交换。Peer 之间必须在 keepalives 报文的发送时间间隔上统一，这就意味着在第一阶段的一些参数协商是必须的。

假设 peer A 和 peer B 协商每 10 秒交换一次“keepalives”信息。这样每隔 10 秒每一方会发

一个 hello 包到另一端。这个 hello 包的发送证明了发送方的在线。接着，另一方必须对每个 hello 报文进行确认 (ACK)。如果 10 秒后一方还没有收到 hello 报文，它自己将发送 hello 消息，若收到对端的 ACK 消息就能证明对方的在线。Hello 和 Ack 消息的接收都能使 keepalives 计时器复位。在特定时间内不能收到 ACK 消息将标识一个连通的失败。

具体解释如下：

假定 1：Peer A 有维护了一个 10 秒的 keepalives 计时器，且 A 给 B 发 hello，B 给 A 回 ACK。

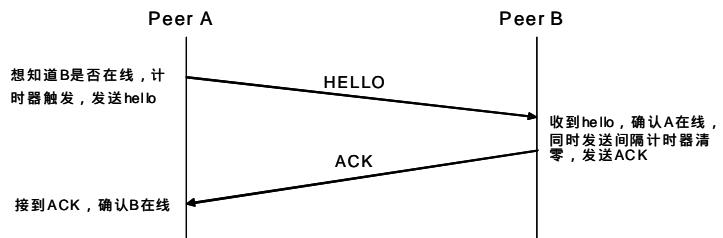


图 1 Keepalives 机制消息交互 1

假定 2：Peer A 维护一个 10 秒的计时器，且它发 hello 给 B。A 的 hello 包丢失，然后 B 没有回复，A 重发 hello。这种情况描述了 peer A 如何检测它的对端是否已下线。

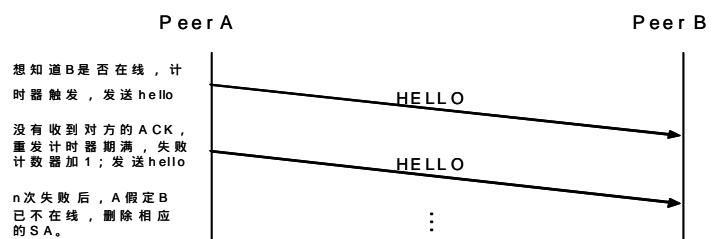


图 2 Keepalives 机制消息交互 2

这种方式的优点是仅由对对端是否在线感兴趣的发送者开始消息交换。在假定 1 中，peer A 对 peer B 是否在线感兴趣，随后，peer A 发送 hello。而 peer B 始终不对 peer A 是否在线感兴趣，所以它不需要发送 hello 报文。

2.2.2 Heartbeats

作为对比，下面解释一下另一种使用单向消息（仅有 hello），证明在线的方案。对对方的在线感兴趣的 peer，依靠对方发送的周期消息来证明在线，这种方案中，消息交换如下：

假定 1：Peer A 和 Peer B 对对方的在线都感兴趣。双方都依靠对方的周期 hello 消息来达到目的。

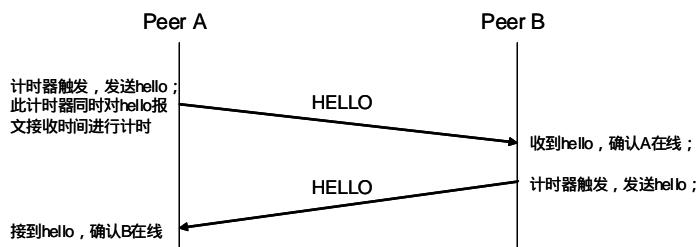


图 3 Heartbeats 机制消息交互 1

假定 2：Peer A 几次没有收到 peer B 的 hello 消息则标识 peer B 不在线。

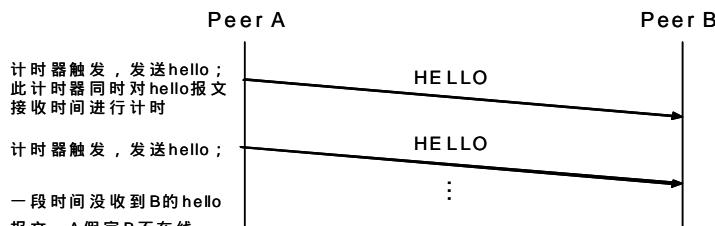


图 4 Heartbeats 机制消息交互 2

这种方式的缺点在于它依靠对端的 hello 来证明对端的在线。在以上假定中 B 可能从不对 A 的是否在线感兴趣，虽然这样，如果 A 对 B 的在线感兴趣，B 必须要意识到这点，发送周期 hello 消息给 A。这种方案的缺点在远程接入中更加明显。假设 VPN 组网中维持有 50000 个进程，每个 peer 都需要相当快的响应，因此大约每 10 秒钟共需要发送 50000 个 hello 包。

在这两种方案（keepalives 和 heartbeats）中，消息发送间隔协商必须要执行，以使双方都能知道对端多长时间需要收到一次 hello 消息。这就增加了复杂度。另外，周期性发送消息也增加了计算开销。

DPD 协议

DPD 致力解决 IKE keepalives 和 heartbeats 方式的缺点，通过引入一个更合逻辑的控制消息交换来实现。本质上，keepalives 和 heartbeats 要求和对端周期性交换 hello 消息。而 DPD 规定每个 peer 的状态和对端状态是完全独立的。当 peer 想知道对方是否在线时，随时请求，不需要等待间隔时间的到来。这种 DPD 交换可以发送更少的消息，所以 DPD 能够达到更大的扩展性。

下面详细介绍 DPD 的细节。当 peer 之间有正常的 IPsec 流量时，没有太大必要去发送额外的消息来证明对端的在线。因为 IPsec 流量本身就能证明 peer 的在线。若在一段时间内没有流量发生，即没有数据报交换，则 peer 的活动状态是值得怀疑的。而 peer 是否在线，实际上是仅仅在有数据报要发送时才知道的。所以只有在这种情况下 peer 应该发送 DPD 消息来检测对端的状态。

每个 peer 可能对空闲时长有不同的需求。比如 peer A 可能需要快速检测，而 peer B 则可能是相对长时间的。当然每端都能单独定义它的 DPD 交换消息时间间隔。

例如：peer A 定义间隔时间为 10 秒，那么如果 peer A 在发出 IPsec 流的 10s 内没有收到回应流，那么 peer A 就能发起一个 DPD 交换。同理，peer B 也能定义它的间隔时间为 5 分钟，在 IPsec 通道空闲 5 分钟后，B 也能发起 DPD 交换。

3.1 DPD vendor ID

IKE peer 两端在 DPD 交换开始前必须发送 DPD vendor ID，DPD vendor ID 的格式为：

0	HASHED_VENDOR_ID	14 15
	M J R	M N

图 5 DPD vendor ID 格式

此处的 HASHED_VENDOR_ID 取值为 0xAF, 0xCA, 0xD7, 0x13, 0x68, 0xA1, 0xF1；

`0xC9, 0x6B, 0x86, 0x96, 0xFC, 0x77, 0x57`。这些值标识了这个 vendor ID 为 DPD。其中的 MJR 和 MNR 字段分别对应当前 DPD 协议的主次版本。

3.2 消息交换

DPD 消息是一个双向交换的宣告消息，交换定义如下：

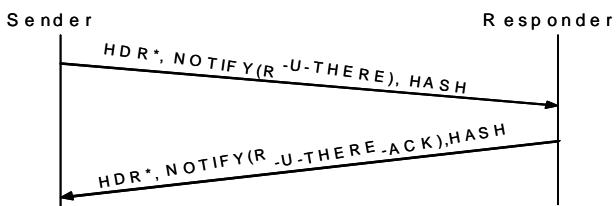


图 6 DPD 消息交互

R-U-THERE 消息相当于一个 HELLO，而 R-U-THERE-ACK 相当于一个 ACK，这两个消息只是 ISAKMP 的宣告负载。DPD 协议中定义两种新的 ISAKMP 宣告消息种类。

Notify	Message
R-U-THERE	36136
R-U-THERE-ACK	36137

发送了 DPD Vendor ID 的实体必须回应 R-U-THERE 请求。此外，实体必须拒绝不加密的 R-U-THERE 和 R-U-THERE-ACK 消息。

3.3 宣告消息的格式

R-U-THERE 消息的报文结构应该为：

0	8	16	31
Next Payload	RESERVED	Payload Length	
Domain of Interpretation (DOI)			
Protocol-ID	SPI Size	Notify Message Type	
Security Parameter Index (SPI)			
Notification Data			

图 7 R-U-THERE 消息的报文结构

因为这消息是一个 ISAKMP 宣告，所以 Next

Payload, RESERVED, Payload Length 域应该被置零。其余的 DOI 域应该被置为 IPsec-DoI；Protocol ID 必须被设为 ISAKMP 的协议 ID；SPI Size 域应置为 16，是指 ISAKMP cookies 的长度为 2 个字节；Notify 消息类型必须被设为 R-U-THERE；SPI 域应为发起者或响应者的 IKE SA 的 cookies；Notification 数据是指消息对应的序列号。

R-U-THERE-ACK 的格式和 R-U-THERE 相同，不同的是宣告类型为 R-U-THERE-ACK。注意 R-U-THERE-ACK 宣告数据必须是接收到对应的 R-U-THERE 消息序列号。

3.4 DPD 执行

DPD 不强制要求以协商时间间隔发送交换消息。仅仅当它对对端的状态感兴趣时，即希望和对端建立连接，这时 IKE peer 应发送 R-U-THERE，查询对端 peer 的活动状态。当 peer 间流量正常交互时，每个 peer 应该使用流量来证明本端在线，两端都不应发起 DPD 交换。但 peer 应跟踪收到的 DPD 交换消息。例如 peer 发送一个 R-U-THERE 请求消息后，希望得到对端 peer 的回复，若在定义时间周期内，不能得到 ACK 的话，它将重传 R-U-THERE 请求。一定次数后，它将认为对端 peer 已不可达或者是 IPsec IKE SA 已被删除。

至于消息交换的时机，因为 IKE peer 的是否在线，仅仅在没有流量交换的时候是可疑的，所以一个 DPD 交换的执行可能从监视空闲时长开始。根据前面所述了解一个 peer 的在线状态仅仅当有外出流量发送的时候是重要的。所以希望发送流量时，可以开始一个 DPD 执行（例如发送 R-U-THERE 消息）。另外当实体发送了外出 IPsec 流量后，没有接收到应答 IPsec 包时，应开始一个 DPD 交换。

3.5 与存活的机制对比

DPD 和传统的 keepalives、heartbeats 相比，优势是不需要定期发送消息。如 keepalives

方式需要维护一个计时器来标识何时应发 hello 消息，还需一个计时器，对“ACK”消息的返回进行计时（这个也是重发计时器）。同样的，heartbeats 方式也需要维护一个计时器，来标识何时发送 hello 消息，也有另一个计时器来标识从对端收到 hello 的间隔。

相比之下，DPD 方式需要保持一个记录了从对端 peer 接收到的最后数据包的时间戳。用来标识这个流量空闲的开始。一旦一个 R-U-THERE 消息发出，DPD 只需要维护一个计时器来标识超时重传，因此，维护计时器状态的资源开支少了，可扩展得到了改进（假定维护一个时间戳比维护活动计时器的系统开支少）。另外，因为 DPD 交换仅仅发生在实体近来没有从对方 peer 收到流量时，发送的消息数量也有很大下降。因此，DPD 的扩展性要比 keepalives 和 heartbeats 好得多。

3.6 DPD 消息的序列号

为了防止消息重放攻击和产生虚假在线状态，DPD 规定必须有一个 32 bit 的序列号对应每个 R-U-THERE 消息，而 R-U-THERE 的回应都必须发送一个具有相同序列号的 R-U-THERE-ACK 消息。当接收到 ACK 消息后，初始发送者应检查序列号的有效性。发送者应拒绝和 R-U-THERE 序列号不同的 R-U-THERE-ACK 消息。

另外，R-U-THERE 和 R-U-THERE-ACK 消息的接收者都应检查发送者和响应者在 payload 的 SPI 域中的 cookies 的有效性。

3.6.1 序列号的选择和维护

DPD 的两端都能触发 DPD 交换（发送 R-U-THERE），每个端点必须维护它自己的 R-U-THERE 消息的序列号。一个进程中的第一个 R-U-THERE 消息，必须有一个随机的序列号。为了防止 32 bit

边界溢出，序列号高位最初应置零，后序 R-U-THERE 消息的序列号必须逐一增加。序列号会在 IKE SA 期满时，被重置，重新选择一个随机数。每个实体也应维护它对端 peer 的 R-U-THERE 消息序列号，并应拒绝不匹配期望的序列号的 R-U-THERE 消息。

实现中可能会维护一个可接受的序列号“窗”，本文档不予详述。

3.7 安全考虑

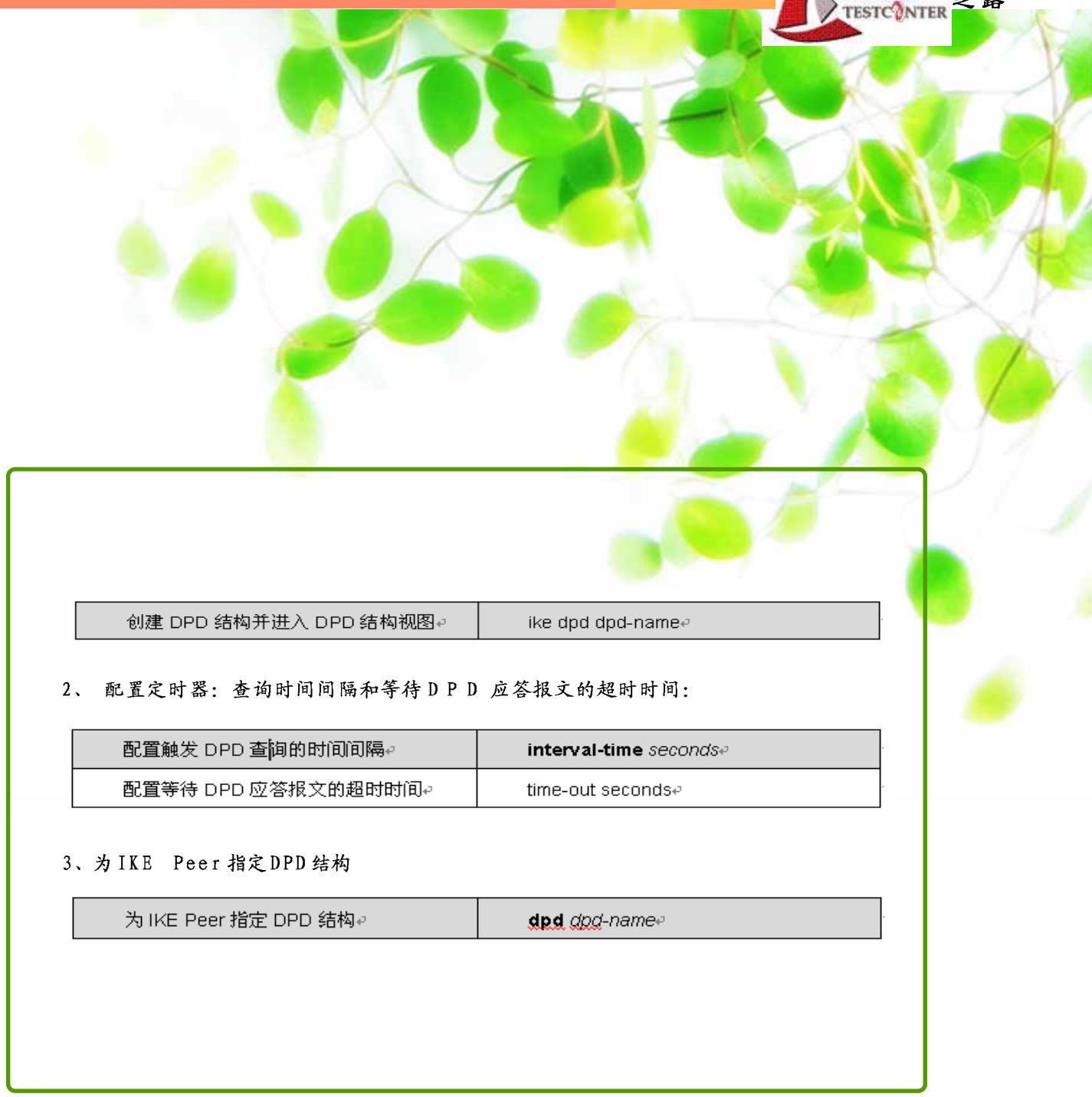
下面介绍使用序列号比使用随机值的优越性。当使用消息序列号来保持 peer 端到端状态时，DPD 也提供一个附加的防止重放攻击的方法。例如 peer A 发送给 peer B 一个有效的 R-U-THERE 消息，攻击者 C 截取此消息并用多个消息拷贝泛洪给 peer B，B 将不得不对每个包解密、处理。（若不论是使用随机值还是序列号）。假如使用序列号，B 就能检测到这个包是重放包，因为序列号不能匹配 B 希望从 A 收到的递增序列号。这样就防止了 B 要对接收到的所有报文进行处理回复。相反，如果 DPD 协议使用随机值，它将不能提供检测消息是重放消息的方法。（除非 B 维护一个最近接收到包的 NONCE 值的列表）。

另一个使用序列号的好处是它提供一个额外的 peer 活动状态确认。只要接收者验证 DPD R-U-THERE 消息的有效性，就能确认对端 peer 活动状态。随机数方法则不能。

3.8 我司设备 DPD 配置

我司设备的 DPD 实现配置很简单：

- 1、首先定义一个 DPD 数据结构用于配置 DPD 查询参数。包括 DPD 查询时间间隔及等待 DPD 应答报文超时时间间隔。该数据结构可以被多个 IKE Peer 引用：



创建 DPD 结构并进入 DPD 结构视图。 `ike dpd dpd-name`

2、配置定时器：查询时间间隔和等待 D P D 应答报文的超时时间：

配置触发 DPD 查询的时间间隔	interval-time seconds
配置等待 DPD 应答报文的超时时间	time-out seconds

3、为 IKE Peer 指定 DPD 结构

为 IKE Peer 指定 DPD 结构。 **dpd *dpd-name***

-- 本文完 --

IPsec NAT 穿越

◆ ◇ □ 徐庆伟

NAT（本文如无特殊说明，NAT 包括端口转换 PAT 情况）主要用于解决 IPv4 地址紧缺问题，在目前网络中 NAT 应用非常广泛，特别是在企业网出口网关大都使用了 NAT 技术解决公网地址不足的问题。同时为了实现同一企业集团不同地域分支之间的低成本安全互连，VPN 技术，特别是 IPsec 技术得到了广泛的应用。但是 NAT 技术与 IPsec 技术存在一些不兼容的问题，使 NAT 与 IPsec 共存成为一个 IPsec 实施中需要重点考虑的因素。本文主要介绍 NAT 与 IPsec 共存可能引起的主要问题以及如何解决这些问题。并利用实例对 NAT 穿越进行了说明。

NAT 与 IPsec 共存可能引起的问题

NAT 与 IPsec 共存问题涉及到很多方面，在 RFC 3715 中进行了详细描述。这里介绍一些主要问题，包括 NAT 本身机制与 IPsec 兼容问题，NAT 设备实现问题以及为了实现 IPsec 穿越在 NAT 上支持 helper 功能引入的兼容性问题。

1.1 NAT 本身机制与 IPsec 兼容性问题

这类不兼容问题是直接由 NAT 与 IPsec 协议本身不兼容造成的。NAT 协议固有的不兼容性包括 IPsec AH 和 NAT 不兼容、NAT 与校验和不兼容、IPsec SPI 选择和 NAT 不兼容、IKE 地址标识符和 NAT 不兼容、固定 IKE 目的端口和 NAPT 之间不兼容、重叠(Overlap) SPD 条目和 NAT 间不兼容、嵌套 IP 地址和 NAT 不兼容以及 NAT 隐含的方向性问题等。

1.1.1 AH 与 NAT 共存问题

由于 AH 主要用于保护消息的完整性，其验证范围涵盖了整个 IP 报文，对于 IP 报文头中参与验证字段的修改将导致 AH 检查失败。因此使用 AH 保护的 IPsec 隧道是不能穿越 NAT 网关的。但是 ESP 协议保护的报文却不存在该问题，因为 ESP 保护的部分不包含 IP 报文头（对隧道方式而言是外层 IP 头）。

1.1.2 校验和与 NAT 共存问题

由于在 TCP 和 UDP 中校验和的计算是依赖于 IP 报文头的源地址和目的地址的，因此如果在 NAT 网关处不修改 TCP 或 UDP 的校验和，会导致校验和检查失败。在没有 IPsec 的情况下，报文是明文传输的，因此 NAT 网关可以对 TCP 或 UDP 的校验和进行修改，但是对于 IPsec 加密报文 NAT 网关是没有办法修改的，这样就出现共存问题。UDP 校验在 IPv4 中是可选的，TCP 是必选的，在 IPv6 中 UDP 和 TCP 的校验都是必选的。校验和与 NAT 共存问题一般只发生在传输模式情况，并且加密部分是 TCP 或者 UDP 的情况。

1.1.3 IKE 的 ID 载荷中包含地址与 NAT 共存问题

在 IKE 中第一阶段和第二阶段协商中某些验证方式会使用 IP 地址作为 ID 负载的内容。IKE 报文穿越 NAT 后由于地址的改变会导致出现不一致的问题。

1.1.4 IKE 固定端口号与 NAT 共存问题

IKE 使用的源端口与目的端口号都是 500。如果存在地址端口转换的情况，IKE 报文的源端口将被改变，这样如果响应者判断源端口不是 500，可能会存在协商问题。因此如果存在 NAPT 网关，要支持 NAT 穿越，必须支持非 500 端口发起的 IKE 报文，并且能够正确回应到这个非 500 端口。

1.1.5 重叠 SPD 入口和与 NAT 共存问题

缺省在传输模式下，IKE 协商的是两点之间通信的隧道，因此可以使用报文的源地址来确定进行流量选择符，不同的发起者与同一个响应者建立隧道的时候可以使用报文的源地址协商多个 SPD 入口（请参考 RFC 2409, 5.5）。但是在 NAT 后的主机由于经过 NAT，所有报文的源地址都相同，因此会使 SPD 重叠，这样将导致响应者无法确定一个报文到底使用那个 SA。

1.1.6 内部地址封装与 NAT 共存问题

在报文内部封装 IP 地址进行一些地址信息协商的时候，如果报文通过明文方式传输，可以在 NAT 设备上启用 ALG 进行检测修改这些报文内部地址，例如 FTP 协议、H. 323 协议等。但是一旦启用 IPsec，报文进行 ESP 加密后 NAT 网关无法检测经过的报文，因此对报文内部地址也就无法修改，这样就存在共存问题。

1.1.7 入口 SA 对选择符检查与 NAT 共存问题

对于入接口报文进行 IPsec 处理的时候，需要对报文源地址根据 SA 中的流量选择符（感兴趣流）进行一致性检查。如果报文源地址与流量选择符不一致，需要对报文丢弃处理。

1.2 NAT 设备实现问题

在有些 NAT 设备上只支持 TCP 和 UDP 流量，对于非 TCP 和 UDP 的流量不能处理，这样由于 AH 和 ESP 协议号为 50、51，因此在这种情况下 IPsec 流量穿越这样的 NAT 网关将存在问题。

另外由于在 NAT 网关上建立的 NAT 会话是有超时时间的，一旦 IKE 报文的 UDP 会话在 NAT 上老化，那么 NAT 外的 IKE Peer 不能与内部 IKE Peer 进行通信。

另外 NAT 网关对分片报文的处理存在的一些问题对 IPsec 穿越 NAT 也存在一些问题。

1.3 NAT Helper 功能引入的问题

有些 NAT 网关为了实现 NAT 穿越，在 NAT 网关实现的时候做了一些特殊处理，例如使用 cookie 进行流量区分导致重协商问题，对 UDP 500 端口号特殊处理以及对 IKE 载荷进行检测等导致的问题。由于 NAT Helper 设备的这些“多余”的处理，可能导致一些问题。

IPsec 穿越 NAT 的兼容性要求

2.1 可部署性

IPsec-NAT 兼容性解决方案必须比 IPv6 易于部署，还应满足只需修改主机，无需改变路由器的要求。为了在短时间内实现穿越方案的部署，必须要求兼容性解决方案能与现存的路由器和 NAT 产品协同工作。

2.2 协议兼容性

IPsec-NAT 穿越方案不解决某些协议与 NAT 的兼容性问题。这些协议是指用 IPsec 协议不能进行安全保护且无法穿越 NAT 的协议。因此，即使有了 IPsec-NAT 穿越方案，也不能解决诸如 H. 323 协议穿越 NAT 的问题，仍然需要 ALG 支持。

2.3 方向性

NAT 的方向性也是一种安全功能，所以 IPsec 穿越方案不应允许 NAT 后面的主机接收来自任意 IP 地址随意发送的 IPsec 或 IKE 通信流。一旦双向 IKE 和 IPsec 通信已经建立，则地址转换的映射即告连接。

2.4 远程访问

IPsec 的一个重要应用是远程访问公司的内部网络。NAT 穿越方案必须支持通过 IPsec 隧道模式或者 L2TP over IPsec 的 NAT 穿越，故要求穿越方案必须考虑远程客户端与 VPN 网关之间存在多个 NAT 的情况。

2.5 防火墙兼容性

目前，防火墙已经广为应用，IPsec-NAT 兼容性方案必须能使防火墙管理员创建简单的静态访问规则，以决定是否允许 IKE 及 IPsec-NAT 的穿越。原则上，应该避免 IKE 或者 IPsec 目的端口的动态分配。

2.6 可扩展性

IPsec-NAT 兼容性方案应具有良好的扩展性，可部署在大规模远程访问的环境中。在大量远程接入的环境下，不可能在同一时间段内只有一个主机使用同一个给定的地址进行通信。因此，在兼容性方案中，必须解决 SPD 条目重叠的问题。

2.7 模式支持

IPsec-NAT 方案必须支持 IPsec ESP 模式的穿越。例如 IPsec 安全网关必须支持 ESP 隧道模式的 NAT 穿越，IPsec 主机必须支持 IPsec 传输模式的 NAT 穿越。

AH 的目的是保护 IP 头部中不变的区域（包

括地址域），而 NAT 必须转换地址，从而使 AH 完整性检验失效。因此，NAT 和 AH 从根本上就是不兼容的。在 IPsec-NAT 兼容性方案中，没有必要支持 AH 传输或隧道模式。

2.8 后向兼容和互操作性

IPsec-NAT 兼容性方案中必须能与已有的 IKE/IPsec 实现互操作，与不经过 NAT 的 IKE/IPsec 进行通信，即 IPsec-NAT 穿越方案必须能后向兼容 RFC2401 定义的 IPsec 和 RFC2409 定义的 IKE。穿越方案应该能自动检测是否存在 NAT，使通信双方只在必要时才使用 NAT 穿越支持。兼容方案应能判断通信对方的 IKE 实现是否支持 NAT 穿越，判断双方是否只进行标准的 IKE 会话。也就是说，虽然 IKE 在发起协商时，目的端口只能使用 500 端口，但并没有对源端口提出特殊要求，因此 UDP 源端口可以使用 500 或非 500 的端口。

2.9 安全性

IPsec-NAT 兼容性解决方案的引入不得对 IKE 或 IPsec 的安全带来影响。例如，一个可行的方案必须能证明，它没有引入新的拒绝服务攻击和欺骗攻击。IKE 必须允许双向方式的密钥能够重生成。

IKE 中 NAT 穿越 (NAT-T) 的协商

3.1 NAT-T 能力检测以及 NAT 网关发现

NAT-T 能力检测是指检查通信双方对 NAT-T 能力是否支持；NAT 网关发现是指在欲建立 IPsec 隧道的两个 peer 之间是否存在 NAT 网关。NAT-T 能力检测以及 NAT 网关发现是发生在 IKE 协商的第一阶段。NAT-T 能力检测在 IKE 协商第一阶段的前两个消息交换完成，通过在消息中插入一个标识 NAT-T 能力的 vendor ID 载荷 (VID) 来告诉对方对该能力的支持。如果双方都在各自的消息中包含了该载荷，说明双方对 NAT-T 都是支持的。只有双方同时支持 NAT-T 能力，才能继续进行相关 NAT-T 协商。载荷的内容是对特定字串进行 MD5 运算得出的散列值，在 RFC3947 中规定标识 NAT-T 能力的 vendor ID 内容为 “RFC 3947” 的 MD5 HASH 运算值，具体运

算后的结果 16 进制表达为

“4A131C81070358455C5728f20E95452F”。

IKE 中可以使用 NAT-D (NAT Discovery) 载荷用于探测两个 IKE 实体之间是否存在 NAT。由于 Keepalive 消息必须从位于 NAT 后面的实体发出，因此 NAT 位置的检测也非常重要，NAT-D 载荷也能用于探测 NAT 所处的位置。为了探测出两台主机之间的 NAT，需要检查 IP 地址和端口是否沿着传输路径发生改变。协商双方只需各自向对端发送源方和目的方的 IP 地址与端口的散列值，就可以检测地址和端口在传输过程中是否发生改变。如果协商双方计算出的散列值与其收到的散列值相同，则表示它们之间没有 NAT。反之，则是在传输中对地址或端口进行了转换，说明所通过的 IPsec 报文进行了 NAT 穿越的处理。如果发送者不能确定自己的 IP 地址（比如拥有多个网络接口，并且不能确定包路由选择到哪一个接口），它可以在报文中包含多个本地 IP 地址的散列值。在这种情况下，仅当所有的散列值均不匹配时，才表明 NAT 的存在。

图 1、图 2 分别是是 IKE 主模式和野蛮模式协商时使用 VID 载荷进行能力检测以及使用 NAT-D 载荷实现 NAT 网关发现的过程（注：图中“#”表示检测到 NAT 后该消息发送到新的端口，关于端口的改变后面介绍）。

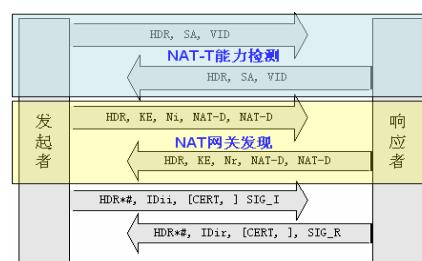


图 1 主模式 NAT 能力与 NAT 网关发现过程

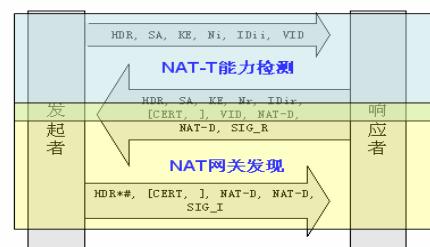


图 2 野蛮模式 NAT 能力与 NAT 网关发现过程

一般情况下一个消息中只包含两个 NAT-D 载荷。第一个 NAT-D 载荷内容为对端 IP 地址和端口的 Hash 值，第二个为本端 IP 地址和端口的 Hash 值。但是如果发起者不知道发送 IKE 报文的源地址，那么需要为每一个可能的源地址计算一个 NAT-D 载荷，因此在一个消息中可能存在多个 NAT-D 载荷的情况。无论在什么情况下第一个 NAT-D 载荷内容永远是对端的 IP 地址与端口的 Hash 值。

有些 NAT 设备可能会对 IKE 报文进行一些特殊处理，他们可能在不改变 500 的源端口号情况下，也可以通过使用 IKE Cookie 值映射不同的通信流，而不必使用源端口。对 IKE 来说，很难发现 NAT 网关是否具有上述能力。对 NAT 的透明性而言，这些方法均存在弊端。最好的办法是使 IKE 通信流简单地不使用 500 端口，采用一个新的端口号 4500。在图 1、图 2 所示的 IKE 交换中，一旦发起者检测到 NAT-T 能力以及 NAT 网关的存在，就将 IKE 报文端口号修改为 4500。

3.2 NAT 穿越的启用协商

在阶段 1 完成后，协商双方都已明确，在它们之间是否存在 NAT 以及谁在 NAT 后面。至于是否使用 NAT 穿越，则由快速模式协商决定。

NAT 穿越的启用协商在快速模式的 SA 载荷中进行，协商双方可向对端发送 IPsec 报文的原始地址（传输模式情况下），从而使对端有可能在 NAT 转换之后，对 TCP/IP 进行校验和修正。

在 IKE 中增加了两种 IPsec 报文传输模式：UDP-Encapsulated-Tunnel（3）和 UDP-Encapsulated-Transport（4），前者用于 UDP 封装 tunnel 模式 IPsec 报文，后者使用 UDP 封装 Transport 模式 IPsec 报文。在快速模式交换中通过安全提议携带报文封装载荷来协商使用的报文封装模式。如果没有 NAT 网关，就没有必要采用 UDP 封装。

在 Transport 模式时，为了执行 TCP 校验和修正，协商双方可能需要知道对端在构造报文时所使用的原始 IP 地址。在 IKE 快速模式中使用 NAT-OA 载荷承载原始 IP 地址。对于发起方，其原始发起方地址（NAT-OAi）定义为发起方的 IP 地址，而原始响应方地址（NAT-OAr）定义为当前所知道的对端的 IP 地址。对于响应方，原始发起方地址（NAT-OAi）定义为当前所知道的对端 IP 地址，原始响应方地址（NAT-OAr）定义为

为响应方的 IP 地址。下面图 3 中，10.1.1.1 作为发起者，向公网中 202.102.10.2 发起 IKE 协商。那么 NAT-OA 定义为：

10.1.1.1: NAT-OAi = 10.1.1.1; NAT-OAr = 202.102.10.2

202.102.10.2 : NAT-OAi = 202.102.10.1; NAT-OAr = 202.102.10.2

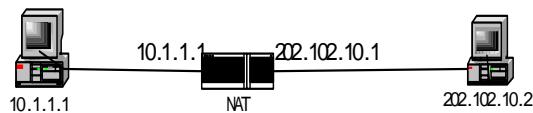


图 3 发起者向公网响应者发起 IEK 请求

如果发起者在其 IPsec 提议中含有 UDP-Encapsulated-Transport 模式，必须同时提供 NAT-OA 载荷。如果响应者同意发起者的 UDP-Encapsulated-Transport 模式提议，那么响应者也必须同时在回应消息中提供 NAT-OA 载荷

使用 UDP 封装 IPsec ESP 报文穿越 NAT 网关

本文前面介绍了在 IKE 协商中检测双方 NAT 穿越支持能力以及检测 NAT 网关的存在。并且在快速模式中可以协商 IPsec 报文的封装模式使用 UDP 封装传输模式或者隧道模式。由于 AH 协议本身不能进行 NAT 穿越，因此使用 UDP 封装只适合于 ESP 报文。

4.1 UDP 封装 IPsec ESP 报文格式

UDP 封装 IPsec ESP 报文格式如图 4：

Source port	Destination Port
Length	Checksum
SPI	
Sequence Number	
... ...	

图 4 UDP 封装 ESP 报文格式

从图 4 可以看到，在 UDP 报头后面直接跟了一个 ESP 报文头。在 UDP 报文头中源端口号以及目的端口号采用和 IKE 协议一致的端口号。这样做的好处是可以减少 NAT 网关上 NAT 会话的数量，另外不用单独考虑 IKE 的 keepalive 报文。并且在配置和使用都非常简单。但是共用端口号的同时带的另一个问题是上层实现如何区分一个报文是 IKE 报文还是 UDP 封装的 ESP 报文呢？为了区分这两种报文，RFC 3948 规定采用 UDP 封装方式的 ESP 报文的 SPI 一定不能为 0，同时规定使用启用 NAT 穿越的 IKE 协商报文在 UDP 报文头后插入 4 个值为 0 的字节，作为非 ESP 报文的标识。具体报文格式图 5 所示。

Source port	Destination Port
Length	Checksum
Non-ESP Marker	
.....	

图 5 启用 NAT 穿越后 IKE 封装

4.2 报文处理过程

4.2.1 解封装过程中辅助处理过程

对于穿越 NAT 后的 UDP 封装 ESP 报文，在接收端接收到报文后需要进行一些辅助处理过程，根据封装模式不同处理方式也不同。

当使用隧道模式传输报文的时候，内部 IP 头中会包含与接收设备所在网络不一致的地址。以下说明将其转换成适合当前网络地址的处理方法。根据本地策略，必须完成下列任务之一：

- 如果在策略中，已为对端的封装报文定义了一个有效的源 IP 地址空间，则应根据策略检查在内部报文中的 IP 源地址是否属于有效范围。
- 如果已经为远程对端分配了一个地址，则应检查内部报文中的 IP 源地址是否与该地址一致。对报文执行 NAT 转换，使其适合在本地网络中传输。

当使用传输模式传送报文时，如果在传输中

IP 头部发生变化，TCP 或 UDP 头部将包含错误的校验和。根据本地策略必须完成以下任务之一：

- 如果在 ESP 头部之后的协议头部是一个 TCP/UDP 头，并且已经获得对端的真实源 / 目的 IP 地址，则应增量计算 TCP/UDP 校验和，包括：
 - 从校验和中减去接收包的 IP 源地址；
 - 在校验和中增加通过 IKE 获得的真实的 IP 源地址（从 NAT-OA 中获得）；
 - 从校验和中减去接收包的 IP 目的地地址；
 - 在校验和中增加通过 IKE 获得的真实的 IP 目的地地址（从 NAT-OA 中获得）。
- 如果接收到的地址和真实地址是相同的，则取消相关操作。
- 如果在 ESP 头后面的协议是 TCP/UDP，则应重新计算 TCP/UDP 头中的校验和字段。
- 在传输模式情况下，如果 ESP 头后面的协议是 UDP，将 UDP 头中的校验和字段置 0。如果在 ESP 头后面的协议头是 TCP 头，并且存在一个选项指示协议栈不用检查 TCP 效验和，则可以不检验 TCP 校验和。但是对隧道模式情况下，TCP 校验和是必须进行验证。因为校验和由发送方产生并由接收方验证，该校验和是对整个 IPsec 处理的报文的完整性检验。

4.2.2 传输模式 UDP 封装与解封装过程

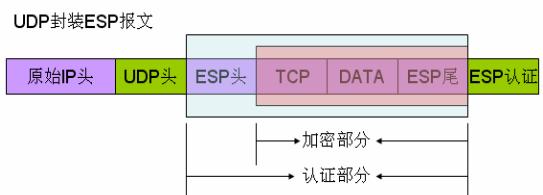


图 6 UDP-Encapsulated-Transport 封装报文前后格式

报文封装过程：

首先原始报文按照普通 ESP 封装过程进行 ESP 封装；然后在如图 6 所示的位置插入一个 UDP 的报文头，最后计算最终报文的 IP 头中的可变

变字段，例如总长度、协议 ID 以及校验和等。
报文解封装过程：

首先将图 6 中 UDP 头去掉，重新计算 IP 头中的可变字段；再进行普通 ESP 解封装过程；最后进行 4.2.1 传输模式的辅助处理过程。

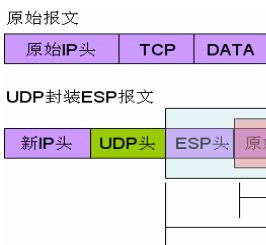


图 7 UDP-Encapsulated-Tunnel 封装报文前后格式

4.2.3 隧道模式 UDP 封装与解封装过程

报文封装过程：

首先原始报文按照普通 ESP 封装过程进行 ESP 封装；然后在如图 7 所示的位置插入一个 UDP 的报文头，最后插入一个新的 IP 报文头，计算最终报文的 IP 头中的可变字段，例如总长度、协议 ID 以及校验和等。

报文解封装过程：

首先将图 7 中 UDP 头去掉，重新计算 IP 头中的可变字段；再进行普通 ESP 解封装过程；最后进行 4.2.1 隧道模式的辅助处理过程。

4.3 NAT 保活过程

由于在 NAT 网关上 NAT 映射会话是有一定存活时间的。因此，隧道建立后如果中间长时间没有报文穿越，就会导致 NAT 会话表被删除。这样将导致无法通过隧道继续传输数据。解决这一问题的办法是，在 NAT 会话超时前发送 NAT-keepalive 报文，以维持 NAT 会话的存活。

NAT-keepalive 报文格式如图 8 所示：

Source port	Destination Port
Length	Checksum
0xFF	

图 8 NAT-keepalive 报文格式

为了能够实现保持 NAT 网关会话的目的，UDP 头中端口信息必须与图 4 中 UDP 封装 ESP 中的端口信息一致。Checksum 字段建议为 0，接收者接收到为 0 的 checksum 后应该忽略校验和校验。NAT-keepalive 报文最后一个字节为固定值“0xFF”。接收者接收到 NAT-keepalive 报文后不进行任何处理，直接忽略即可。

NAT-keepalive 报文仅用于 NAT 会话维持使用，不能用于判断隧道是否存在的依据。

UDP 封装 IPsec ESP 穿越 NAT 分析

使用 UDP 封装 ESP 报文穿越 NAT 的方法较好的解决了 IPsec 穿越 NAT 的问题，下面是简单总结。

NAT 无法更新上层校验和问题。该问题只发生在使用传输模式并且 IP 报文后紧跟 TCP/UDP 头的时候才出现。在上面的介绍中我们看到通过在 NAT-OA IKE 载荷中发送原始地址，接收方拥有检验解密之后的上层校验和（源和目标 IP 地址和端口）所需的所有信息，这样就可以做校验和的修正工作，从而解决该问题。

NAT 无法多路传输 IPsec 数据流问题。由于采用了 UDP 封装，因此可以使用 UDP 的端口号来区分不同的数据流。

无法改变 IKE UDP 端口号问题。一些 IKE 报文敏感的 NAT 网关对 IKE 进行特殊处理，针对该问题，采用了新的 UDP 端口号 4500。这样支持 NAT-T 的设备不仅监听 UDP 500，同时监听 UDP 4500 端口。

IKE 的 ID 载荷中包含地址与 NAT 共存问题。对于主模式和快速模式协商，每个 IPsec 对话方发送一 ID 载荷，其中包括发送方的 IP 地址。通过在负载中包含发送方的 IP 信息解决报文被 NAT 转换后地址不可用的问题。

另外在 NAT 穿越中采用了保活机制，通过使用 NAT-keepalive 报文保证 NAT 网关上会话老化的机制不会影响 IPsec 的正常通信需求。

IPsec NAT 穿越实例

我司设备目前支持基于 UDP 封装的 IPsec ESP 隧道模式下穿越 NAT 方案，同时 IKE 协商必须在野蛮模式下进行。下面介绍我司设备部分版本实现情况，以便更加深刻了解 IPsec NAT 穿越。图 9 中分支通过 NAT 网关后才能访问总部。分支与总部流量需要 IPsec 保护。在这种环境下需要在 RT1 和 RT2 之间的 IPsec 实现上启用 NAT 穿越功能，关键配置如图所示。

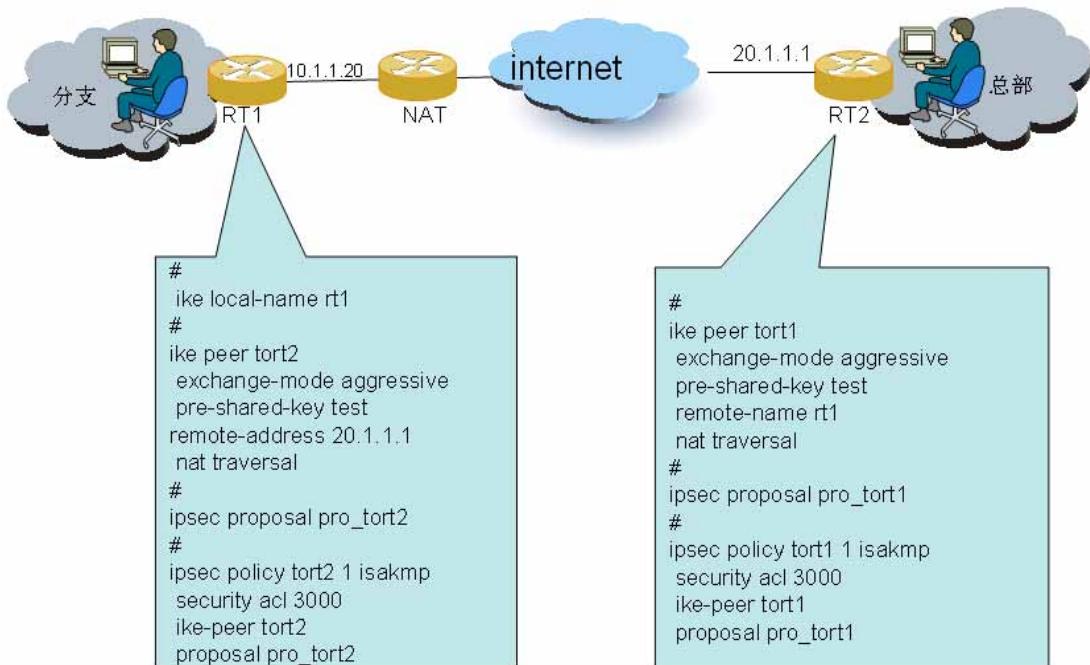


图 9 IPsec 穿越 NAT 应用实例

No.	Time	Source	Destination	Protocol	Info
2 1.549556	10.1.1.20	20.1.1.1		ISAKMP	Aggressive
3 2.783788	20.1.1.1	10.1.1.20		ISAKMP	Aggressive
4 2.864380	10.1.1.20	20.1.1.1		ISAKMP	Aggressive
5 2.865561	10.1.1.20	20.1.1.1		ISAKMP	Quick Mode
6 2.886625	20.1.1.1	10.1.1.20		ISAKMP	Quick Mode
7 2.887610	10.1.1.20	20.1.1.1		ISAKMP	Quick Mode
8 3.526875	10.1.1.20	20.1.1.1		ESP	ESP (SPI=0xb4e3b341)
9 3.534359	20.1.1.1	10.1.1.20		ESP	ESP (SPI=0xd16e0711)
10 3.626806	10.1.1.20	20.1.1.1		ESP	ESP (SPI=0xb4e3b341)
11 3.634384	20.1.1.1	10.1.1.20		ESP	ESP (SPI=0xd16e0711)
12 3.726804	10.1.1.20	20.1.1.1		ESP	ESP (SPI=0xb4e3b341)
13 3.734248	20.1.1.1	10.1.1.20		ESP	ESP (SPI=0xd16e0711)
14 3.826831	10.1.1.20	20.1.1.1		ESP	ESP (SPI=0xb4e3b341)
15 3.834394	20.1.1.1	10.1.1.20		ESP	ESP (SPI=0xd16e0711)
16 4.876760	10.1.1.20	20.1.1.1		ISAKMP	NAT Keepalive

图 10 报文交互过程

图 10 是 IKE 协商过程以及 ESP 报文、NAT-keepalive 报文发送过程。下面具体介绍这些报文中关于 NAT 穿越协商以及报文封装。

6.1 NAT-T 能力协商与 NAT 网关位置确定

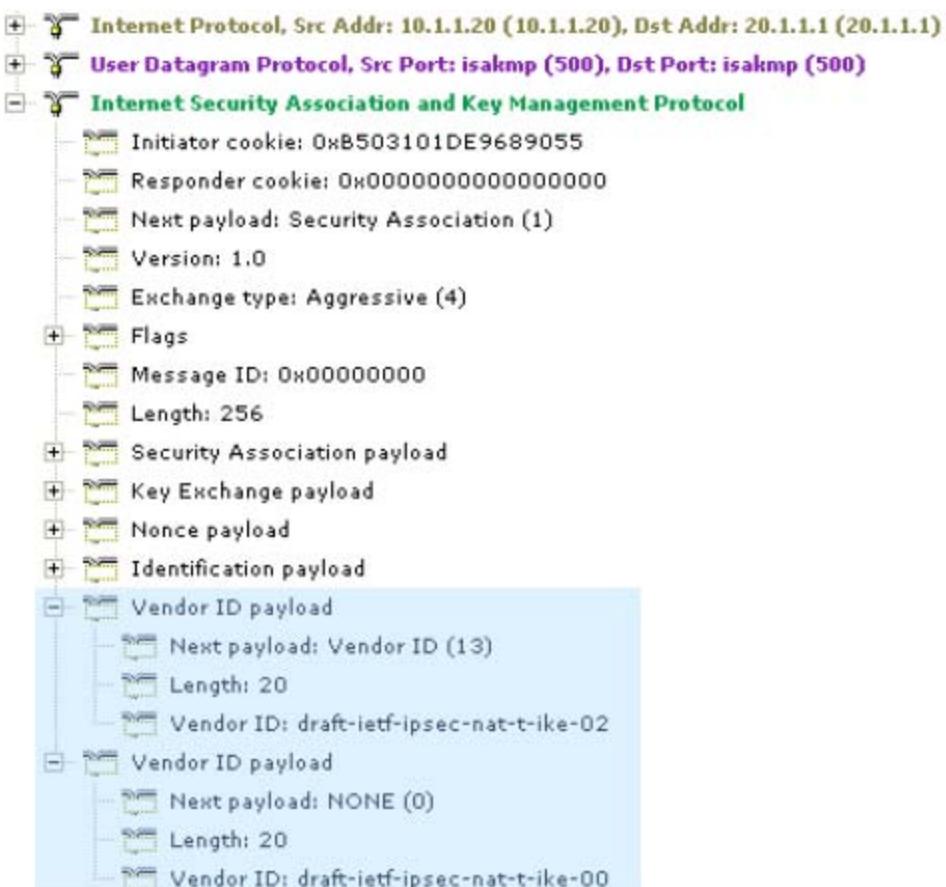
RT1 通过在第一个报文中携带 Vendor ID 载荷来通知 RT2 本身支持 NAT 穿越功能。图 11 是 RT1 发起的第一个 IKE 报文。从图中可以看出 RT1f 发

送了关于NAT穿越能力的两个Vendor ID，这是由于该试验版本实现较早，能够支持基于“draft-ietf-ipsec-nat-t-ike-00”和“draft-ietf-ipsec-nat-t-ike-02”两个草案，同时说明这个版本是不能与本文介绍的RFC 3947进行互通的。但是这个问题并不影响对NAT穿越的理解，因此本文仍以此为例介绍。

图11是RT2的响应消息，除了包含支持的NAT穿越能力Vendor ID之外，还包含两个NAT-D载荷，用于NAT网关发现以及设备相对于NAT网关的位置。关于NAT-D载荷请参考3.1节。

RT1发送第三条消息使用了新的端口号4500，如图13所示。从这个消息开始后续的所有报文都是用该端口号，包括快速模式和UDP封装的ESP报文。从图中可以看到在这个报文中增加了一个None-ESP Marker标识，具体含义请参考4.1节描述。

3.1节中介绍在野蛮模式下第三条消息中包含了发起者计算的NAT-D载荷，用于接收者确定NAT网关的存在与位置。由于第三条消息是加密消息，此处没有给出报文分析图。



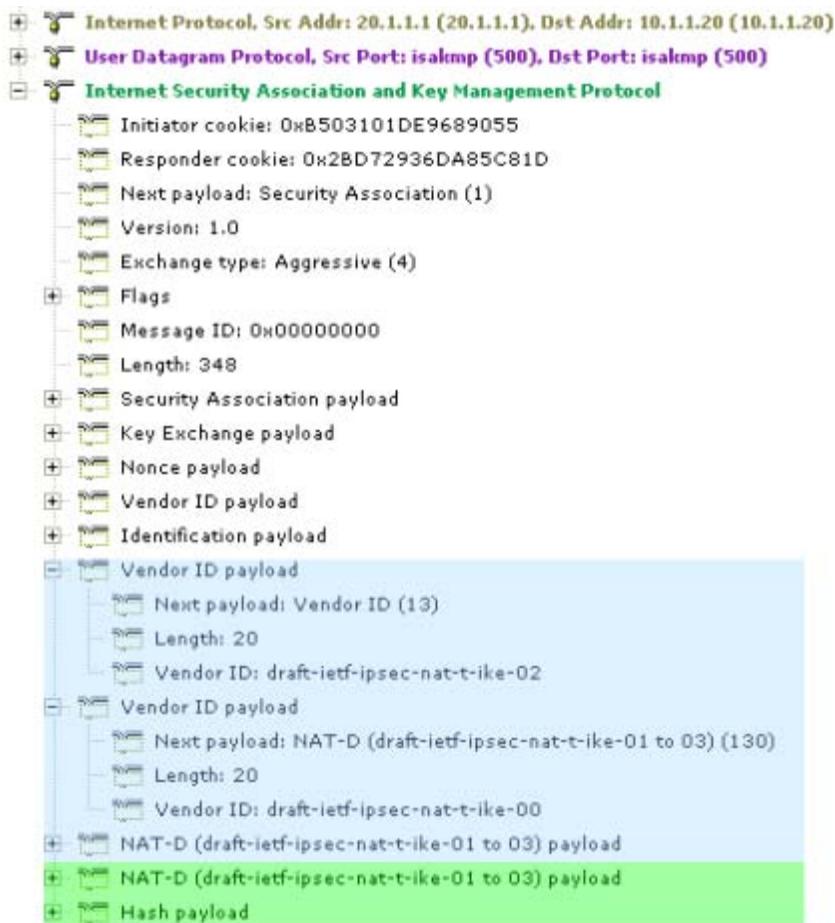


图 12 RT2 发送的第二条消息

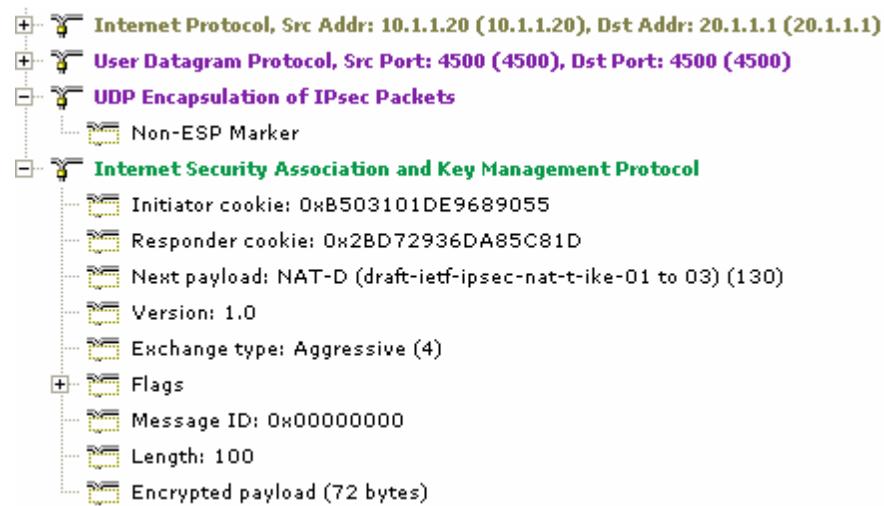


图 13 RT1 发送的第三条消息

以上野蛮模式三条消息交互确定了 RT1 和 RT2 的 NAT 穿越支持能力，以及相互之间相对 NAT 网关的位置。

6.2 快速模式协商中进行NAT-T 功能启用 协商

在我司设备上由于只支持隧道模式的NAT 穿越功能，因此会在快速模式协商中携带 UDP-Encapsulated-Tunnel (3) 载荷用于完成NAT-T 功能启用的协商。由于在快速模式协商中报文都是加密的，因此无法通过抓包来分析报文载荷。下

面是从RT2 上查看到的调试信息。

6.3 穿越NAT 的IPsec 报文封装

图15 是从RT1 ping RT2，在NAT 转换之前抓到的报文情况。从图中看到在IP 报文后紧接着的是UDP 报文头，使用端口号为4500。在UDP 报文中封装了IPsec ESP 报文，这样IPsec ESP 报文就可以穿越NAT 网关了。

```
*0.1332647975 Quidway IKE/8/DEBUG: message_recv: message 2adea04
*0.1332648026 Quidway IKE/8/DEBUG: ICOOKIE: 0x351d3252f35bd936
*0.1332648127 Quidway IKE/8/DEBUG: RCOOKIE: 0x4a6a875b0c23735f
*0.1332648177 Quidway IKE/8/DEBUG: NEXT_PAYLOAD: HASH
*0.1332648278 Quidway IKE/8/DEBUG: VERSION: 16
*0.1332648329 Quidway IKE/8/DEBUG: EXCH_TYPE: QUICK_MODE
*0.1332648379 Quidway IKE/8/DEBUG: FLAGS: [ ENC ]
*0.1332648480 Quidway IKE/8/DEBUG: MESSAGE_ID: 0x8be9b755
*0.1332648531 Quidway IKE/8/DEBUG: LENGTH: 172
*0.1332648582 Quidway IKE/8/DEBUG: message parse payloads: payload HASH
*0.1332648645 Quidway IKE/8/DEBUG: message parse payloads: payload SA
*0.1332648695 Quidway IKE/8/DEBUG: message parse payloads: payload NONCE
*0.1332648746 Quidway IKE/8/DEBUG: message parse payloads: payload ID
*0.1332648797 Quidway IKE/8/DEBUG: message parse payloads: payload ID
*0.1332648898 Quidway IKE/8/DEBUG: validate payload SA of message 2adea04
*0.1332648948 Quidway IKE/8/DEBUG: DOI: 1
*0.1332648999 Quidway IKE/8/DEBUG: message parse payloads: payload PROPOSAL
*0.1332649100 Quidway IKE/8/DEBUG: message parse payloads: payload TRANSFORM
*0.1332649150 Quidway IKE/8/DEBUG: validate payload PROPOSAL of message 2adea04
*0.1332649201 Quidway IKE/8/DEBUG: NO: 1
*0.1332649263 Quidway IKE/8/DEBUG: PROTO: IPSEC_ESP
*0.1332649314 Quidway IKE/8/DEBUG: SPI_SZ: 4
*0.1332649365 Quidway IKE/8/DEBUG: NTRANSFORMS: 1
*0.1332649416 Quidway IKE/8/DEBUG: validate payload TRANSFORM of message 2adea04
*0.1332649516 Quidway IKE/8/DEBUG: NO: 1
*0.1332649567 Quidway IKE/8/DEBUG: ID: 2
*0.1332649668 Quidway IKE/8/DEBUG: Transform 1's attributes
*0.1332649718 Quidway IKE/8/DEBUG: Attribute SA_LIFE_TYPE: SECONDS
*0.1332649819 Quidway IKE/8/DEBUG: Attribute SA_LIFE_DURATION: 3600
*0.1332649870 Quidway IKE/8/DEBUG: Attribute SA_LIFE_TYPE: KILOBYTES
*0.1332649982 Quidway IKE/8/DEBUG: Attribute SA_LIFE_DURATION: 1843200
*0.1332650033 Quidway IKE/8/DEBUG: Attribute ENCAPSULATION_MODE: TUNNEL_UDP_ENCAPSULATE
*0.1332650084 Quidway IKE/8/DEBUG: Attribute AUTHENTICATION_ALGORITHM: HMAC_MD5
```

图14 RT2 上查看快速模式第一个报文信息



图15 使用 UDP 封装的 ESP 报文

6.4 维持NAT会话的NAT-keepalive报文

4.3介绍了使用NAT-keepalive报文保持NAT网关上的会话信息。图16是在RT1发送给RT2的一个NAT-keepalive报文，用于保证NAT网关上

NAT session不被老化掉。

6.5 NAT网关上NAT会话信息

对应上述例子，在NAT网关上NAT会话信息如图17所示。

```

# Frame 16 (60 bytes on wire, 60 bytes captured)
# Ethernet II, Src: 10.1.1.20 (00:e0:fc:2f:9d:4f), Dst: 10.1.1.1 (00:e0:fc:00:00:01)
# Internet Protocol, Src: 10.1.1.20 (10.1.1.20), Dst: 20.1.1.1 (20.1.1.1)
    version: 4
    Header length: 20 bytes
# Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 33
    Identification: 0x0037 (55)
# Flags: 0x00
    Fragment offset: 0
    Time to Live: 255
    Protocol: UDP (0x11)
# Header checksum: 0x9b7e [correct]
    Source: 10.1.1.20 (10.1.1.20)
    Destination: 20.1.1.1 (20.1.1.1)
# User Datagram Protocol, Src Port: 4500 (4500), Dst Port: 4500 (4500)
# UDP Encapsulation of IPsec Packets
    Non-ESP Marker
# Internet Security Association and Key Management Protocol
    NAT Keepalive

0000  00 e0 fc 00 00 01 00 e0 fc 2f 9d 4f 08 00 45 00  ::.... ./.O..E.
0010  00 21 00 37 00 00 ff 11 9b 7e 0a 01 01 14 14 01  ::!7.... ~.....
0020  01 01 11 94 11 94 00 0d bd 94 00 00 00 00 00 00  ....:.....
0030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....


```

图16 NAT-keepalive报文

```

<NAT>display nat session
There are currently 1 NAT session:
Protocol      GlobalAddr Port      InsideAddr Port      DestAddr Port
17            20.1.1.2 12290      10.1.1.20 4500      20.1.1.1 4500
status: 11,     TTL: 00:00:40,   Left: 00:00:28

```

图17 NAT网关上NAT会话信息

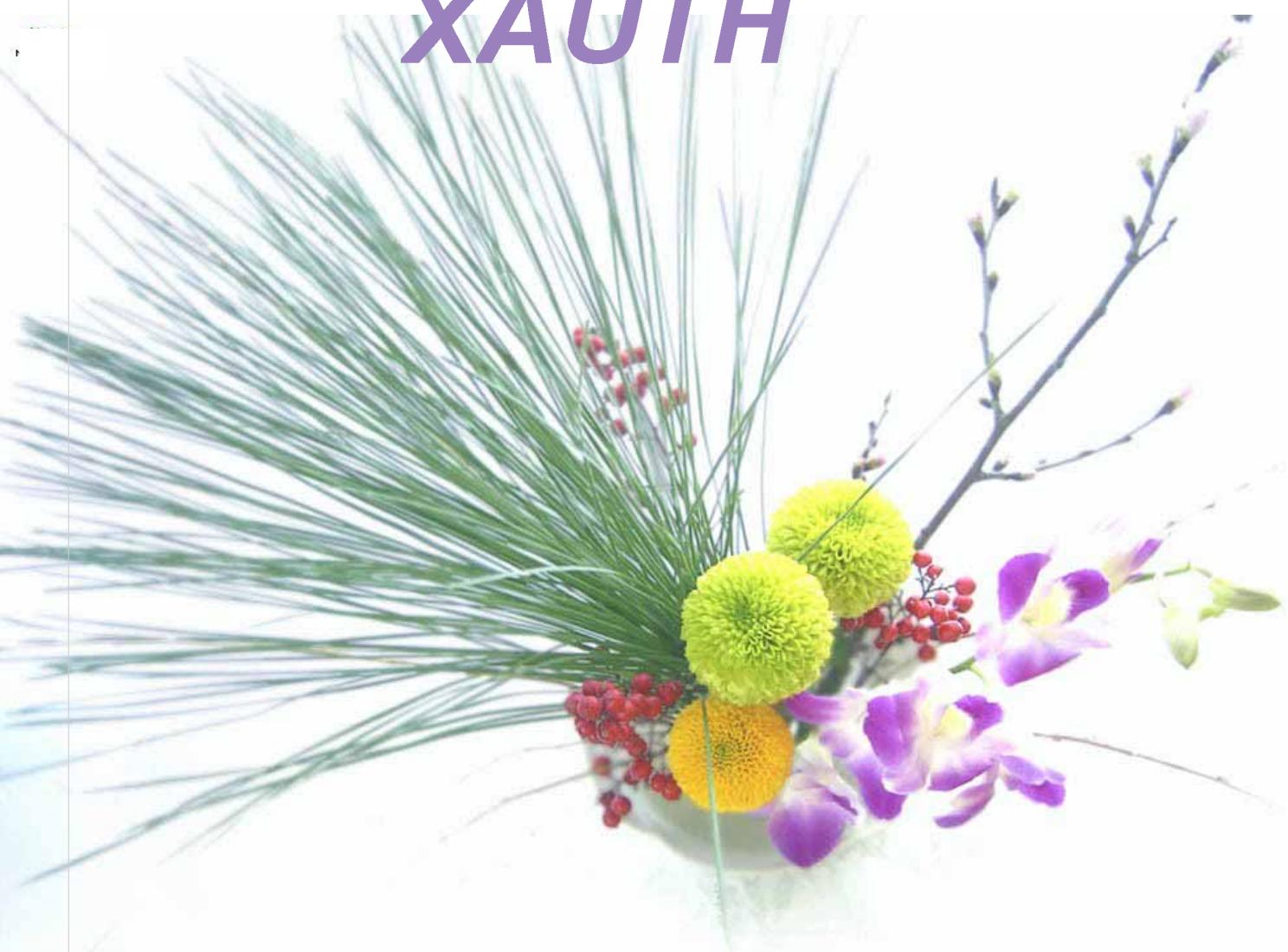
后记

IPsec本意是要保证数据在传输过程中的安全性，但是NAT却是要改变报文的某些字段，因此出现了矛盾。IPsec的NAT穿越经历了多年发展，逐步形成了一个统一的标准。但是在一些较早的实现中，包括我司部分设备，是按照早期draft实现的，可能存在与本文介绍不一致的地方。例如“[draft-ietf-ipsec-udp-encaps-01](#)”介绍的内容与本文介绍的封装方式差别很大，感兴趣的读者可以参考相关文档。

参考资料：

- [1] Aboba, B. and W. Dixon, “IPsec-Network Address Translation(NAT) Compatibility Requirements”, RFC 3715, March 2004.
- [2] T. Kivinen., “Negotiation of NAT-Traversal in the IKE”, RFC3947, January 2005.
- [3] A. Huttunen., “UDP Encapsulation of IPsec ESP Packets”, RFC3948, January 2005.
- [4] T. Kivinen., “Negotiation of NAT-Traversal in the IKE”, draft-ietf-ipsec-nat-t-ike-01, October 2001.

XAUTH



◆◆□ 李红霞

背景

目前在远程用户接入VPN方面，主要采用的是PPTP、L2F和L2TP等协议，它们同IPsec协议相比都存在安全加密能力低的弱点，并且不支持公钥基础设施PKI，但是IPsec协议本身对远程用户接入的支持还不够，特别是对于远程用户的认证存在较大的局限性。由于远程接入用户可能在任何地方，因此对这些用户使用VPN接入进行认证和授权成为一个重要的问题。

在IKE第一阶段中，IKE对通信双方的身份验证主要采用了预共享密钥认证、数字签名认证和公共密钥加密认证三种方法，这三种方法都要求通信双方双向对等认证。这样的认证方式比较适合局域网和局域网之间的VPN部署，然而目前许多企业采用的是客户/服务器模式的VPN部署，在这种情况下远程用户作为客户访问服务器时便遇到了困难。在部署此类远程访问的IPsec VPN应用时，通常是要设置多个客户端连接到VPN中心网络，网管人员的通常做法是为每一个用户设置不同VPN策略和预置密码用以区分每一个用户，此工作量是巨大的，管理也不方便。

为此，IETF又提出了IKE的扩展认证方式(Extended Authentication with IKE, XAUTH)。XAUTH在IKE的基础上提供了单向认证的机制，例如RADIUS、SecurID和OTP等，而这些机制所使用的传统认证方法如Challenge/Response, Two-factor authentication等已经被广泛地运用于客户/服务器模式的网络中。这样在VPN网关中只要配置一条VPN的策略，就可允许多个如

高达1000个远程客户端的同时接入，但要求每个远程客户在接入时需要提供不同的用户名和口令的身份认证，VPN网关设备可以集中管理远程用户的合法信息。这样就大大减少了网络管理人员的工作负担和保证远程客户接入的安全性，提高了企业的整体工作效率。

简介

XAUTH作为IKEv1的扩展认证特性，定义了一种单向认证的方法，主要用于远程用户通过IPsec接入时的客户端认证。XAUTH与AAA认证结合使用，可以使用本地认证，也可以借助其他远程认证服务器完成认证。

XAUTH在IKE第一阶段协商完成后进行，此时IKE SA已经生成，IKE Peer认证成功；XAUTH是对IKE认证的补充，通过对用户进行认证，进一步强化了安全性。只有两次认证都通过了才能够实现VPN的接入。

XAUTH原理

3.1 概述

由IPsec网关向远程用户发出扩展认证请求后，强制要求用户响应，通过XAUTH的认证后才能接入VPN。值得说明的是XAUTH首先是在IKE第一阶段的认证并建立IKE SA后，在IKE第二阶段之前附加的用户身份验证。由IPsec网关决定是否使用XAUTH认证。

下图的消息25-28属于XAUTH的认证过程。消息29-30属于MODECFG的过程，MODECFG主要是

□ 22	10.1.1.65	10.1.1.2	ISAKMP	Aggressive
□ 23	10.1.1.2	10.1.1.65	ISAKMP	Aggressive
□ 24	10.1.1.65	10.1.1.2	ISAKMP	Aggressive
□ 25	10.1.1.2	10.1.1.65	ISAKMP	Transaction (Config Mode)
□ 26	10.1.1.65	10.1.1.2	ISAKMP	Transaction (Config Mode)
□ 27	10.1.1.2	10.1.1.65	ISAKMP	Transaction (Config Mode)
□ 28	10.1.1.65	10.1.1.2	ISAKMP	Transaction (Config Mode)
□ 29	10.1.1.65	10.1.1.2	ISAKMP	Transaction (Config Mode)
□ 30	10.1.1.2	10.1.1.65	ISAKMP	Transaction (Config Mode)
□ 31	10.1.1.65	10.1.1.2	ISAKMP	Quick Mode
□ 32	10.1.1.2	10.1.1.65	ISAKMP	Informational
□ 33	10.1.1.2	10.1.1.65	ISAKMP	Quick Mode
□ 34	10.1.1.65	10.1.1.2	ISAKMP	Quick Mode

图1 XAUTH认证报文

3.2 IKE 第一阶段协商过程

VPN 客户端的私网地址可以自己配置，也可以由网关分配；远程拨号用户通常需要由网关来分配地址。远程拨号用户的地址变动较大，IPsec 网关一般无法预先得知对端远程拨号用户的地址，因此，IKE 第一阶段的协商无法使用主模式 + 预共享密钥的方式，只能使用野蛮模式 (Name) + 预共享密钥，或者主模式 + 其他认证方式（如数字证书）。

下面说一下野蛮模式下的协商过程：

消息 1：由客户端发起 IKE 第一阶段野蛮模式的协商，客户端携带所有支持的 IKE 策略（包括加密算法、Hash 算法和 DH 组）；CISCO VPN 客户端携带 14 个策略；

消息 2：由网关选定合适的策略，通知客户端；

消息 3：对客户端进行认证，加密的确认消息。

需要说明的是 CISCO VPN 客户端均使用 DH 组 2，因此，在消息 1 中已经携带 NONCE 值。建议配置网关前首先通过抓包或者察看 VPN Client 配置说明了解客户端所携带的 IKE 策略的内容。

3.3 XAUTH 认证

3.3.1 XAUTH 消息类型

不同于 IKE 两个阶段消息个数固定的特点（第一阶段共六条消息，第二阶段共三条消息），XAUTH 的消息个数要根据具体的认证方法以及设计的安全强度来决定，而最基本的消息传送形式就是 REQUEST->REPLY->SET->ACK，下面具体说明远程用户和 IPsec 网关交互的四种基本消息类型。这些消息携带各种不同的属性。这四个消息分别是：

ISAKMP_CFG_REQUEST：由 IPsec 网关发出，请求 IPsec 客户端进行扩展认证。

ISAKMP_CFG_REPLY：客户端回应消息，必须填充由网关要求的认证属性，如果客户端没有合适的认证属性可以填充，必须在 XAUTH_STATUS 属性中填充 FAIL。

ISAKMP_CFG_SET：由网关发出，用来表示认证成功或者失败。

ISAKMP_CFG_ACK：由客户端发出，用来确认收到认证结果。

3.3.2 认证过程

简要说明一下本地用户名 / 密码认证的过程：

1. 首先由 IPsec 网关发送 REQUEST 请求，Attributes 的内容为空等待 Client 端填充，Type=1；
2. 客户端收到 REQUEST 请求后，发送 REPLY 报文，按照网关的要求填充 Attributes，一般填入用户输入的用户名 / 密码，否则在 Attributes 中填充 XAUTH_STATUS=FAIL，Type=2；
3. 网关认证后发送 SET 报文，在 Attributes 中填入 SUCCESS 或者 FAIL，Type=3；
4. 客户端发送确认报文。

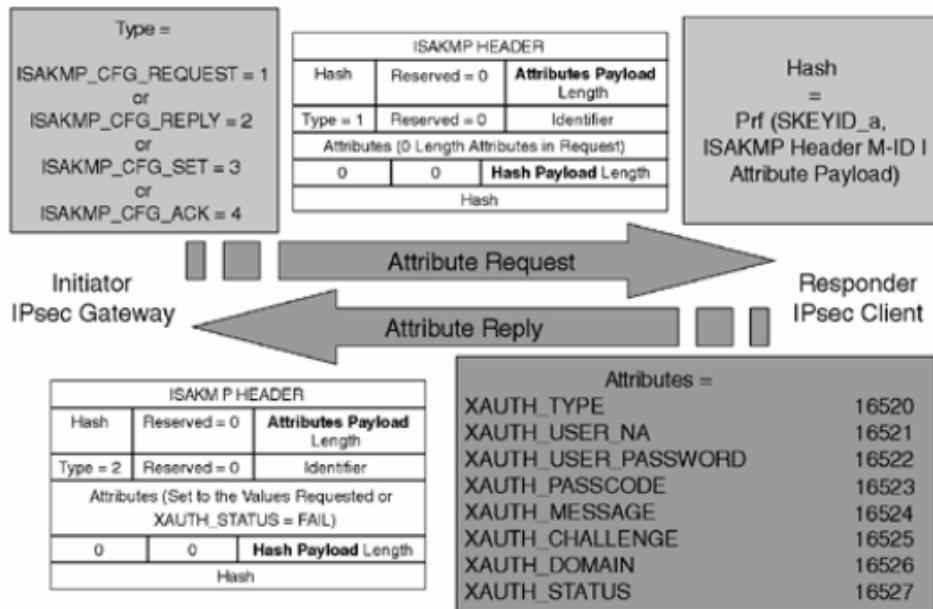


图 2 XAUTH 认证第一、二报文

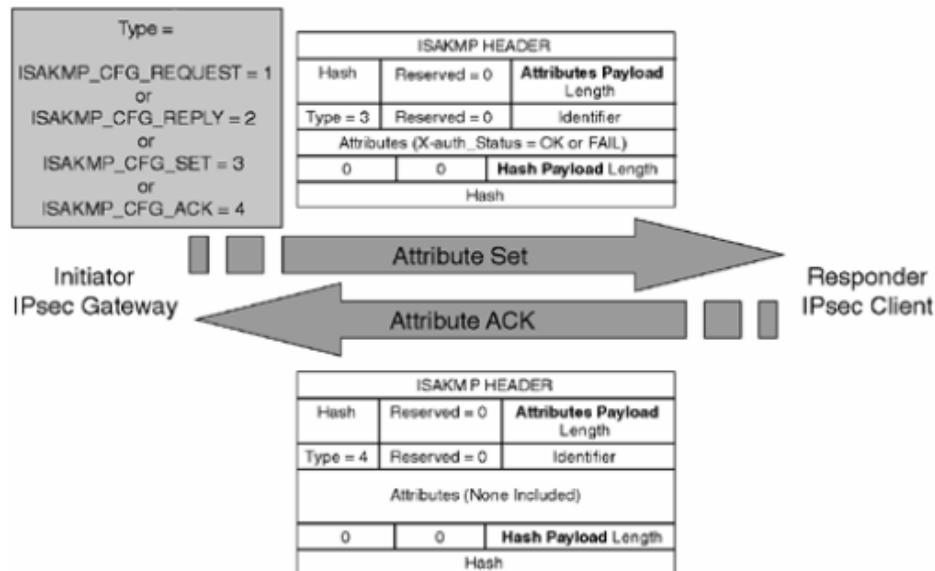


图 3 XAUTH 认证第三、四报文

在这里需要注意的是由于用户的密码需要在线传递给边界设备，所以必须确保该密码的安全。为此，可以用通信双方的共享密钥对该密码加密。

XAUTH发生在IKE第一阶段后，因此报文是加密的：

4 0.131485	10.2.1.1	10.2.1.3	ISAKMP Transaction (Config Mode)
5 4.271686	10.2.1.3	10.2.1.1	ISAKMP Transaction (Config Mode)
6 4.580407	10.2.1.1	10.2.1.3	ISAKMP Transaction (Config Mode)
Frame 4 (142 bytes on wire, 142 bytes captured)			
Ethernet II, Src: Cisco_8a:6f:15 (00:03:a0:8a:6f:15), Dst: vmware_62:32:29 (00:0c:29:62)			
Internet Protocol, Src: 10.2.1.1 (10.2.1.1), Dst: 10.2.1.3 (10.2.1.3)			
User Datagram Protocol, Src Port: isakmp (500), Dst Port: isakmp (500)			
Internet Security Association and Key Management Protocol			
Initiator cookie: 0x6894AE29CF4A5537			
Responder cookie: 0x203A9165FF2C5996			
Next payload: Hash (8)			
Version: 1.0			
Exchange type: Transaction (Config Mode) (6)			
Flags			
.... .1 = Encrypted			
.... .0. = No commit			
.... .0.. = No authentication			
Message ID: 0xFB947C82			
Length: 100			
Encrypted payload (72 bytes)			

图4 XAUTH认证加密报文

对于远程接入IPsec VPN，一种常见的部署是使用IKE预共享密钥和野蛮模式，因为网关通常不知道远程客户端的IP地址（IP地址在第一阶段协商完成后，由网关分配，或者由L2TP分配）。在大多数使用共享密钥的部署中，由于多个共享密钥的管理复杂，通常所有VPN用户都使用同一组预共享密钥，这意味着如果不采取额外的用户认证，将无法验证使用该VPN客户端连接的人是否合法用户。

如果VPN客户的笔记本电脑被盗，由于该VPN客户端已经配置了一组有效的密钥，那么使用该设备的人将在没有进一步的认证要求的情况下，毫无障碍的进入企业的内部网络。这将是非常危险的，XAUTH被用于解决这样的安全漏洞。

由此可以看出，XAUTH解决了服务器端对远程用户的身份认证，而IKE本身则可以完成远程用户对服务器的认证（通过验证签名或证书），这样IKE和XAUTH的结合便可以实现客户/服务器的双向非对等认证。

XAUTH也被称为“双因素认证”。密码可以是一次性密码（来自于SecureID卡），这将进一步增强这种部署的安全性。XAUTH被广泛使用，适合用于使用预共享密钥+野蛮模式（Name），也可用于主模式+其他认证方式（如数字证书）。需要指出的是，虽然XAUTH得到广泛应用，但IETF IPsec工作组并没有将其作为一种标准，这意味着在不同厂商的实现之间可能存在互操作性问题。

应用

XAUTH的主要应用场景在前面也有所提及，它主要是为远程办公人员从任何地方连接到Internet的认证和授权（VPN Client-LAN），但同时适用于两个设备之间（LAN-LAN）建立的IPsec隧道。远程用户的接入通常也需要分配私网IP地址，因此XAUTH通常与模式配置、EasyVPN结合使用。Cisco对XAUTH的实现是必须和模式配置、EasyVPN结合使用，否则IPsec隧道无法建立。目前我司设备不支持XAUTH。

XAUTH在EasyVPN的使用中属于可选项，由IPsec网关决定是否启用XAUTH。

配置

支持在PC（客户端）和设备建立IPsec通道时使用XAUTH认证，并支持在两个设备之间建立IPsec通道时使用XAUTH。

这里仅说明使用客户端的情况。

配置准备：需要在PC上安装支持XAUTH的VPN Client。

5.1 客户端配置

5.1.1 客户端支持的CISCO产品的版本

CISCO VPN 3000 Concentrator Software Version 3.0 and later

CISCO IOS® Software Release 12.2(8)T and later

CISCO PIX® Security Appliance Software Version 6.0

and later

- CISCO ASA 5500 Series Software Version 7.0 and later

5.1.2 配置举例

使用 VPN Client 版本为 4.0.3 作为配置举例。

1. 先创建一个连接，使用 pre-shared key; group name 和 password 要和设备中的认证组的名字和密码一致（参见设备配置 6 和 7）；

2. 在 VPN Client 进行 IKE 协商后会有一个窗口弹出，要输入用户名和密码，这是因为 CISCO 的 IPsec 网关使用了 XAUTH，要再验证一次 AAA 服务器中设置的帐号，输入配置的用户名和密码（参见设备配置 4）。

注意：在客户端不需要输入 IKE 协商所需要的算法和 DH 组的信息，协商时客户端将携带所支持的转换（Transport Payload），由网关选择与配置相符的算法。

5.2 设备端的配置

XAUTH 需要和 Easy VPN 结合使用，必须要配置模式配置以及 Easy VPN 的相关命令。下面仅说明了与 XAUTH 相关的命令，配置模式配置以及 Easy VPN 在其他文档中说明。

5.2.1 CISCO IOS® Software Release 12.2(8)

T and later

说明不同颜色标出的部分对应为必须相同的项。

1. 配置用户认证列表

语法: aaa authentication login **name**
local

举例: aaa authentication login **userlist**
local

2. 使能 XAUTH，在 IKE 第一阶段和第二阶段之间触发 XAUTH，并引用用户认证列表

语法: crypto map **map-name** client authentication list **list-name**

举例: crypto map **dynmap** client authentication list **userlist** (红色标出为相同的部分)

说明：如果没有配置该项，不进行 xauth 认证，仍然能够成功建立 IPsec 隧道。

3. 将 IPsec 策略绑到接口

进入接口视图后

语法: crypto map **map-name**

举例: crypto map **dynmap** (蓝色标出为相同的部分)

4. 新建认证的用户名和密码

语法: username **username** password 0
password

举例: username xauth password 0 xauth

5. 新建组的授权列表

语法: aaa authorization network **name**
local

举例: aaa authorization network
groupauth local

6. isakmp 授权使用的方式

语法: crypto map **map-name** isakmp authorization list **list-name**

举例: crypto map vpn isakmp authentication list **hw-client-groupname**

7. 新建认证的组名（在模式设置中配置）

语法: crypto isakmp client configuration group **name**

举例: crypto isakmp client configuration group **hw-client-groupname**

8. 新建认证的组的密码（在模式设置中配置）

进入到组的视图

语法: key 0 password

举例: key 0 vpngroup

注意：使用 xauth 后，IKE 第一阶段的认证的 pre-shared key 在组视图下配置，不用再配置 pre-shared key（命令为 crypto isakmp key password address ip-address）。

1. 配置不使用 xauth (在两个设备之间建立 IPsec, 例如路由器和路由器之间)

语法: crypto isakmp key password address ip-address no-xauth

举例: crypto isakmp key 123456 address 10.1.1.2 no-xauth

配置需注意的地方: IKE 第一阶段的协商需使用野蛮模式 (Name) + 预共享密钥, 或者主模式 + 其他认证方式 (如数字证书), 具体原因见 3.2。

5.2.2 CISCO VPN 3000

VPN 3000 具有图形化的界面, 默认包含 EasyVPN Server 的配置, 因此配置起来容易方便。

1、配置组:

Configuration-->System-->User Management-->Group 中填加。

2、配置 XAUTH 认证所使用的用户名和密码。

Configuration-->System-->User Management-->Users 中填加。

5.3 典型配置举例

路由器-VPN Client 之间的 IPsec:

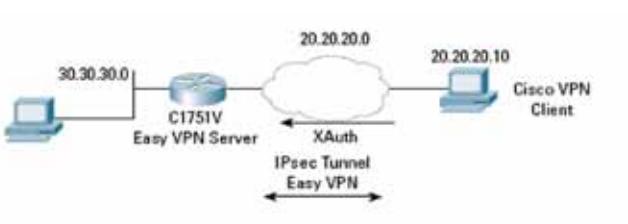


图 5 路由器-VPN Client 之间的 IPsec 拓扑图

说明: 配置中的红色字体与 XAUTH 和 MODECFG 相关。

CISCO1751VPNROUTERCONFIGURATION

version 12.2

no service pad

service timestamps debug uptime

service timestamps log uptime

no service password-encryption

service internal

!

hostname Cisco1751

!

aaa new-model

!

!-- 配置 AAA 认证、授权列表

aaa authentication login userlist local

aaa authorization network hw-client-groupname local

aaa session-id common

enable password cisco

!-- 配置 XAUTH 认证时使用的用户名和密码

username cisco password 0 cisco

memory-size iomem 15

clock timezone -0 6

ip subnet-zero

no ip source-route

!

!

ip domain-name cisco.com

!

ip audit notify lo0

ip audit po max-events 100

!

!-- 配置 IKE policy, 注意一定要使用 DH group

2

crypto isakmp policy 1

encr 3des

authentication pre-share

group 2

crypto isakmp client configuration address-pool local

dynpool

注: 与 pool dynpool 配置一条既可

!-- 配置 group 中相关内容, 主要为 MODECFG 相关配置项

!----- 和 pre-shared key 认证中的用户名 (组名) 和密码

crypto isakmp client configuration group hw-client-

```
groupname
key hw-client-password
dns 30.30.30.10 30.30.30.11
wins 30.30.30.12 30.30.30.13
domain cisco.com
pooldynpool
!
!
crypto ipsec transform-set transform-1 esp-3des esp-sha-hmac
!-- 配置 IP Sec 动态策略, Easy VPN 中必须使用动态策略
!-- 因为客户端的私网地址是由服务器端分配的, 所以需要配置反向路由, 以生成路由。
crypto dynamic-map dynmap 1
set transform-set transform-1
reverse-route
!
!-- 触发 XAUTH 认证
crypto map dynmap client authentication list userlist
crypto map dynmap isakmp authorization list hw-client-groupname
!-- 触发 MODECFG
crypto map dynmap client configuration address respond
crypto map dynmap 1 ipsec-isakmp dynamic dynmap
!
!
interface Ethernet0/0
description connected to INTERNET
ip address 20.20.20.2 255.255.255.0
half-duplex
no cdp enable
crypto map dynmap
!
interface FastEthernet0/0
description connected to HQ LAN
ip address 30.30.30.1 255.255.255.0
speed auto
no cdp enable
!
!-- MODECFG 为对端分配地址的地址池
ip local pool dynpool 30.30.30.20 30.30.30.30
ip classless
ip route 0.0.0.0 0.0.0.0 Ethernet0/0
no ip http server
ip pim bidir-enable
!
!
```



```

no cdp run
!
line con 0
line aux 0
line vty 0 4
password cisco
!
end

```

附：XAUTH 属性值描述

Table 4-1. XAUTH Attributes	
Attribute	Description
XAUTH TYPE	This attribute describes the type of extended authentication method requested. Four authentication methods are defined in the protocol: Generic, RADIUS CHAP, One Time password (OTP), and Secure ID. This is an optional attribute for the ISAKMP CFG REQUEST and ISAKMP CFG REPLY messages. The XAUTH TYPE in a REPLY must be identical to the XAUTH TYPE in the REQUEST. However, an XAUTH transaction may have multiple REQUEST/REPLY pairs with different XAUTH TYPE values in each pair.
XAUTH USER NAME	The username may be any unique identifier of the user, such as a login name, an email address, or a X.500 Distinguished Name.
XAUTH USER PASSWORD	The user's password.
XAUTH PASSCODE	A token card's passcode.
XAUTH MESSAGE	A textual message from the gateway to the IPsec client. The message may contain a textual challenge or instruction.
XAUTH CHALLENGE	A challenge string sent from the gateway to the IPsec client to be included in its calculation of a password. This attribute should be sent only in an ISAKMP CFG REQUEST message. Typically, the XAUTH TYPE attribute dictates how the receiving device should handle the challenge. For example, RADIUS-CHAP uses the challenge to hide the password.
XAUTH STATUS	A variable that is used to denote authentication success (OK=1) or failure (FAIL=0). This attribute must be sent in the ISAKMP CFG SET message, in which case it may be set to either OK or FAIL, and may be sent in a REPLY message by a remote peer, in which case it must be set to FAIL.

Easy VPN

◆◇□ 潘冰

Easy VPN 概述

IPSec VPN 技术以其高度的安全性、可靠性和性能，以及灵活多变的策略成为企业构建其 VPN 网络的首选。但是 IPSec 配置的复杂却让很多人望而却步，面对 IPSec VPN 大量相似或者重复的配置，即使对于有经验的网络技术人员也会一筹莫展。能否在保证 IPSec 的安全性和功能的前提下简化 IPSec 的配置呢？

1.1 提出

Cisco 的 Easy VPN (也称为 EzVPN) 技术可以解决这个问题。EzVPN 技术可以将管理员从 IPSec VPN 复杂繁琐的配置中解脱出来，而将更多的精力放在 IPSec VPN 的设计和优化上。

在很多 IPSec 的应用中，用户总是希望安全策略能够通过核心站点的安全设备统一分发和管理，而且通常由于技术力量的限制，企业还希望各个分支站点的配置能够尽量简单，复杂的配置集中在核心设备上完成。

Cisco 的 EzVPN 技术就是针对这些需求提出的。EzVPN 采用 Client/Server 结构，基本原理是将 IPSec 及其相应的配置集中在 EzVPN Server 端，当 EzVPN 的 Client 配置好基本参数时，由 Client 发起和 Server 建立 VPN，然后 Server 端将 IPSec 的安全策略及其相关属性“推”给 Client。

注意：EzVPN 使用的是 Cisco 的私有协议：Unity client protocol，因此 EzVPN 的 Server 端和 Client 端必须都是 Cisco 设备。

1.2 特点

EzVPN 最大的特色就是配置简单、管理方便。和标准的 IPSec VPN 相比，EzVPN 有如下的优势：

- 终端用户和远程站点配置简单；
- 策略动态分发，当 Server 策略变化时，EzVPN Client 设备只需要修改很少的配置甚至不用修改配置；
- 根据配置自动进行 NAT 或者 PAT 转换（Client 模式），不占用核心站点地址空间；
- 验证方式灵活可靠，支持预共享密钥、证书、Xauth、802.1x 等验证方式；
- Web-Based Activation 功能为 VPN 和 Internet 的访问提供了最大的便利；
- 支持 EzVPN Server 冗余；

1.3 软硬件要求

请参考 Cisco 网站的最新软硬件列表。

1.4 操作模式

EzVPN 提供了两种操作模式：

1.4.1 Client 模式

Client 模式是 Cisco 专门为小型办公室或者 SOHO 一族设计的，非常适合小型网络连接 VPN 的情况。这种网络的特点是网络规模较小，只有几台、甚至只有一台 PC。在这种情况下，当来自 EzVPN Client 的流量到达 EzVPN Server 的私网时，由 EzVPN Server 通过 Mode Config 为其分配能够和 Server 所连接私网通信的私网地址，并且根据 Client 连接的情况来决定是否使用 NAT 转换。

Client 模式有两种典型组网应用：VPN Client 软件（简称软客户端）作为 EzVPN Client；Cisco 设备（简称硬客户端）作为 EzVPN Client。

- 软客户端

这种客户端是安装了 Cisco VPN Client 的 PC，该 PC 必须能够访问 Internet，其通过 VPN 软件连接到 EzVPN Server 以后，由 Server 通过 Mode Config 过程为其分配一个私网地址，所有从该 PC 发往 VPN 的报文内层的源地址都是 Server 分配的私网地址，外层地址则是公网地址。

- 硬客户端

这种客户端是一台 Cisco 网络设备，其包含一个公网接口和一个或者多个私网接口。硬客户端和软客户端的最大区别在于硬客户端可以使自己私网的多台主机同时访问 VPN。EzVPN Client 和 EzVPN Server 建立了 VPN 连接，会通过 Mode Config 过程从 Server 获得一个或者多个私网地址，所有来自 Client 私网访问 VPN 的报文会被 Client 设备进行 NAT 转换，将其原来的私网地址转换成 EzVPN Server 为其分配的地址，并且在 Client 上记录一个 NAT Session 以便报文能够正确返回。这种 NAT 转换支持 PAT 方式，这样带来的最大优点是 Client 的多个设备可以共用 Server 端的一个私网地址，极大的节约了 Server 的私网地址。

Client 模式一个重要的特点就是地址需要

EzVPN Server 来分配，原因是如果 EzVPN Client 使用自己子网的私网地址，并且 Server 所连接的网络比较复杂时，所有的网络设备都必须有 EzVPN Client 所连接的私网路由，这样如果同时拨入的 EzVPN Client 数目较多时，需要手工在这些网络设备上配置大量路由，这样增加了配置的复杂性。

1.4.2 网络扩展模式

EzVPN 的“Easy”仅仅是体现在配置上，在组网应用上可绝对不简单。Cisco 对于 EzVPN 的定位不仅仅是中小型网络的 VPN，其完全可以胜任大型网络 VPN 的部署，这也是 EzVPN 相对于 L2TP 等其它的拨入型 VPN 存在的巨大优势。

对于大型站点的 VPN 互连，EzVPN 可以采用网络扩展模式来完成。在网络扩展模式中，不需要 Server 分配给 Client IP 地址。也没有采用 NAT/PAT，所有来自 EzVPN 内网的流量通过 VPN 直接到达 Server 端内部网。但是 Server 可以分配给 Client 其它的 TCP/IP 参数，例如 DNS 地址、WINS 地址等等。

来自 EzVPN Client 私网的报文到达 EzVPN Server 以后，EzVPN Server 并不改变报文的源地址，而是拆开报文的 VPN 封装以后直接转发，这样，对于 Server 所连接私网的其它网络设备来讲，就必须有回程路由才能使报文成功返回 Client，可以采用手工的方式完成，也可以采用 Cisco 的反向路由注入(RRI) 技术。

Cisco 的 RRI 将一条同 Client 相关的静态路由插入到路由表中，通过充分发静态路由，可以将该路由发送给其动态路由邻居。在本文后面章节有关于 RRI 的详细介绍。

对于网络扩展模式，EzVPN 支持隧道分离技术，即同一个接口既可以以加密的形式访问 VPN，也可以以明文的形式访问 Internet。

1.5 认证方式

EzVPN 的认证方式比较灵活，通常有两步认证：第一步认证实际就是 IKE Peer 认证，可以采用预共享密钥的方式或者证书的方式；第二步认证称为 Xauth，Xauth 认证是在 IKE 第一阶段 SA 生成以后进行的二次认证，是 Cisco 对 IKE 协议的补充。虽然该认证是可选的，但是强烈推荐不要关闭该认证。

1.5.1 Xauth

Xauth 认证可以控制哪些用户能够建立 EzVPN 隧道以及用户可以得到的安全策略，结合 AAA 可以灵活的定义不同的用户使用不同的 IPsec 策略。是否采用 Xauth 认证由 EzVPN 的 Server 端决定，当在 Server 端配置了 Xauth 认证后，在 Client 端有两种方法用于输入 Xauth 认证所需要的用户名和密码。

- EzVPN Client 设备本地存储用户名和密码

这种方法一般用于需要 Client 自动建立 VPN 隧道或者通过流量触发 VPN 隧道的情况。由于用户名和密码都存储在设备本地，因此用户访问 VPN 不需要输入任何信息，此时 Xauth 对用户来说是完全透明的。

- EzVPN Client 设备本地不存储用户名和密码

这种方法一般用于隧道手工建立的情况。由于用户名密码没有存储在本地，因此用户需要建立 VPN 时，EzVPN 的安全网关设备会向用户发出一个输入用户名密码的特定 Web 页面，当接收到用户名和密码信息时，安全网关设备使用这些信息和 EzVPN 的 Server 建立连接。当然，用户名密码信息也可以在 EzVPN Client 设备需要连接 Server 时手工输入，但是一般这种情况只适用于调试使用。

值得注意的是，Xauth 认证发生在 IKE 第一阶段 IKE SA 建立完成、第二阶段协商开始之前。

因此 Xauth 认证只决定了 IPsec VPN 能否建立起来，即哪些用户能建立 IPsec VPN，而不能决定哪些用户能够使用 IPsec VPN。也就是说如果一个用户通过了 Xauth 认证建立了 IPsec VPN，那么只要该 VPN 存在，其它未经过认证的用户仍然可以使用该 VPN。如果我们希望对用户使用 VPN 进行认证和授权，采用 Cisco 的 IOS Proxy 或者 802.1x 认证结合 AAA 是一个较好的选择。

当 EzVPN Client 本地不存储用户名密码时，通常采用的方式是使用 Web 接口来进行用户建立 VPN 的认证，Cisco 为这种 Web 接口的认证进行了扩展，提出了 Web-Based Activation 的认证方法。

1.6 Tunnel 建立方式

EzVPN Tunnel 的发起总是由 Client 端完成的，根据需求的不同，Tunnel 的建立有三种方式：自动建立（隧道一直存在）、流量触发、手工建立。

1.6.1 自动建立

当在 EzVPN 的 Client 外网接口配置了 Tunnel Connect Auto 选项后，Client 设备就会自动和 Server 尝试建立 VPN Tunnel。如果 VPN Tunnel 超时断开或者建立失败，Client 设备也会自动的重新连接。

默认情况下，EzVPN 的 Tunnel 建立方式就是自动建立。当 VPN Tunnel 建立起来以后，管理员可以在 Client 上采用“clear crypto ipsec client ezvpn”命令或者 Web 接口断开或者重置 VPN 连接。

1.6.2 流量触发

流量触发方式是当有符合条件的流时自动触发 VPN 的建立，这个流一般由 ACL 来定义。流量触发方式一般适合基于 VPN 处理的应用，用户在使用应用程序时可以不必再次输入验证信息，同

时 V P N 链路超时断开也可以节约安全设备的性能资源。

1.6.3 手工建立

手工的建立是指 V P N 隧道的建立由手工输入连接命令来完成，这样可以使 V P N 的建立和断开完全按照我们的需要来进行。手工建立 V P N 意味着 Client 设备必须等待建立 V P N 的命令完成之后才会尝试和 Server 建立 V P N 。当 V P N 超时断开或者失败，同样必须输入建立 V P N 的命令才会重试连接。

手工建立 V P N 需要在 Client 上输入“`crypto ipsec client ezvpn connect`”，断开或者重置 V P N 需要输入“`clear crypto ipsec client ezvpn`”。

1.7 反向路由注入

反向路由注入(Reverse Route Injection, RRI)是一种集成了 Ipsecpeer 状态和设备路由能力的机制。在 IPsec V P N 中，IPsec 设备必须存在到达 peer 私网的路由，这条路由可以手工静态配置，也可以由 RRI 机制自动导入到 IPsec 中。

RRI 支持下面两种 I p s e c 应用：静态 IPsecpeer、动态 IPsecpeer

1.7.1 静态 Ipsecpeer

静态 Ipsecpeer 是指 Ipsec 相关配置都是静态的情况。在静态 Ipsecpeer 中，双方都可以发起 V P N 连接。

在静态 Ipsecpeer 中，由于 ipsec peer 信息都是手工配置，本地设备会在 Ipsec SA 建立之前就了解其 ipsec Peer 网络的相关信息，因此 RRI 会在建立 Ipsec SA 之前就注入一条到达对方私网的静态路由(前提是使用了 Static 关键字)。

1.7.2 动态 Ipsecpeer

动态 Ipsecpeer 是指在 Ipsecpeer 中的其中一方配置动态加密图，只能由另一方发起 V P N 连接的情况。在动态 Ipsecpeer 中，核心节点一般作为响应方接受分支节点的 V P N 连接请求，并且根据远端设备的 Ipsec 相关参数来动态确定 V P N 的配置信息。动态 Ipsecpeer 一般用于分支办公室或者远程办公人员发起 V P N 连接和总部相连的情况。

由于在动态 ipsec peer 中，核心结点在建立 Ipsec SA 之前并不知道远端设备的相关网络信息，因此动态 ipsec peer 中只有建立了 Ipsec SA 以后，RRI 才会注入静态路由，在这种情况下，RRI 会在 IKE 第二阶段协商中的源代理、目的代理来动态创建一条静态路由。

EzVPN 就是采用动态 Ipsec Peer 的方式建立 V P N 连接。

1.8 可靠性

EzVPN 同样也继承了 IPSec V P N 高可靠性的优点。显而易见，EzVPN 的可靠性集中在 Server 端，如何实现 EzVPN Server 的冗余就是 EzVPN 可靠性设计的核心内容。

当有多个 EzVPN Server 时，EzVPN 的 Client 可以定义 Backup Server List 来完成 Server 故障的切换。根据在 EzVPN Client 端 Backup Server List 的获得方式不同，有两种备份选项：

● Backup Server List 本地配置

这种方式需要在 Client 端定义 Backup Server List 列表，只需要使用多条 Peer 命令即可。这样，当 EzVPN Client 连接到主 Server 时失败，就会自动尝试去备份 Server 建立连接，当所有的备份 Server 都协商失败时，Client 会重新尝试和主 Server 再次建立连接。

当 Client 切换到另外一个 Server 时，会将和前面 Server 所建立的 IPsec/IKE SA 都会删除，重新协商新的 IPsec/IKE 连接。

● Backup Server List 自动配置

Backup Server List 的自动配置是首先在 Server 上配置 Backup Server List，然后当 EzVPN 的 Client 和 Server 协商成功建立了 VPN 以后，Server 会将 Backup Server List 内容“推”给 Client，这样 Client 连接哪些 Server 就可以在 Server 上统一的根据需求灵活的定义，并且简化了 Client 端的配置，因为这种方式不需要 Client 进行任何额外的配置。

但是这种方式必须保证 Client 和 Server 之间第一次的 VPN 建立必须成功，这样 Server 才能将各种属性信息，包括 Backup Server List 内容下发给 Client。

另外，如果 Client 通过 Server 获得了 Backup Server List，然后和 ServerA 连接失败，client 根据 Backup Server List 连接到 ServerB，而 ServerB 也同时配置了 Backup Server List，这些配置就会覆盖 Client 原来从 ServerA 获得的 Backup Server List。当和 ServerB 连接失败时，Client 会按照新的 Backup Server List 连接其它 Server。

对于 Tunnel 自动建立的模式，Client 会自动尝试连接备份 Server。而对于 Tunnel 的手工建立模式，必须手工输入建立 VPN Tunnel 的命令，Client 才能够尝试连接备份 Server。

1.8.1 支持多子网

当 EzVPN 的内网接口下连接了多个子网的时候，可以通过支持多子网特性使这些子网也能够访问 VPN 资源，只需要在 EzVPN Client 上使用 ACL 定义这些子网即可。每个 Client 设备会自动的为 ACL 定义的子网再创建相应的 IPsec SA。

注意：Client 模式不支持多子网功能。

Easy VPN 工作原理

2.1 概述

EzVPN 是继承于 IPsec VPN 的，因此 IKE Sa 和 IPsec Sa 是必不可少的，但是 EzVPN 和 IPsec VPN 也有很大的不同。EzVPN 的 IKE 第一阶段协商过程和 IPsec VPN 是完全一致的，最大的区别是 IKE 第一阶段协商完成以后，需要进行 Xauth 和 Mode Config 过程进行二次认证和网络参数下发（Xauth 一般为可选参数，但是在 EzVPN 中一般都会使用）。

当 EzVPN 的 Client 初始化一个 VPN 连接时，首先需要和 EzVPN Server 进行 IKE 协商，完成 IKE Sa 的建立，然后和 Server 进行 Xauth 的扩展认证，认证通过以后 Server 继续和 Client 协商 IPsecSA，Client 和 Server 根据这些信息建立 IPsec Sa 完成 EzVPN 的建立过程。

2.2 IKE 协商过程

EzVPN 的 Client 首先发起 IKE 协商，有两种认证方式：预共享密钥和证书，如果采用预共享密钥认证，则 Client 必须采用野蛮模式+Name 方式协商，这是因为 EzVPN Server 并不知道 Client 的 IP 地址，无法通过 IP 地址查找到相应的预共享密钥，只能通过配置的 Vpn group 名称来查找预共享密钥，vpngroup 名称就是野蛮模式的 Name ID 信息；如果采用证书认证，则目前 Cisco 仅支持采用主模式协商，EzVPN Server 通过证书 DN 中 OU 名字来匹配 Vpn group 名称查找对应的 Vpn group。为了减少 Client 端的配置，Client 在和 Server 进行 IKE 第一阶段协商时会将自己所有支持的加密算法、认证算法以及 DH 组等信息全部发送给 Server 端，Server 会根据自己 IKE 策略的配置选择第一个匹配的 IKE 策略来响应 Client，完成 IKE 第一阶段 IKE Proposal 的协商过程。

当 IKE 第一阶段 SA 建立完成以后，如果在 EzVPN Server 上启用了 Xauth 认证，那么 Client

就会等待 Server 发送“用户名 / 密码”的挑战值，然后发送 Xauth 的认证信息。Server 可以结合 AAA 来完成认证。

2.3 IPSec SA 建立过程

如果 Xauth 认证成功，则由 EzVPN 的 Client 发送 Request 消息请求其需要的配置参数，包括为其分配的 IP 地址、DNS 地址、WINS 地址、Backup Server 等信息（这些参数都是可选的）。Server 通过其特有的 Mode-Configuration 过程将参数推给 Client。

当 Server 为 Client 分配了 IP 地址以后，RRI 的过程会确保 Server 为每个分配给 Client 的 IP 地址建立一个静态路由。一般情况下，Cisco 都推荐在动态 crypto map 中配置 RRI，除非能够明确的获得对端的路由信息。

当 Client 成功的收到了 Server “推”过来的配置参数，就启用 IKE 快速模式和 Server 协商建立 IPsec SA，协商的过程和第一阶段一致，Client 会将自己所有支持的加密算法、加密协议等信息发送给 Server，Server 选择第一个匹配的 IPsec 策略来响应 Client 从而建立 IPsec SA。

配置

3.1 Easy VPN Client

3.1.1 配置 Client 策略

1. 创建 crypto ipsec client ezvpn 策略

语法: **crypto ipsec client ezvpn name**

举例: Router (config)# crypto ipsec client ezvpn easy client remote

2. 定义预共享密钥和对应的策略组（必须和 Server 一致）

语法: **group group-name key group-key**

举例: Router (config-crypto-ezvpn)# group easy-vpn-remote-groupname key easy-vpn-remote-password

3. 定义 Server 地址（可以定义多个 Server 实现冗余）

语法: **peer [ip-address | hostname] {default}**

说明: peer 命令可以使用多次，但是只有一个可以定义为 Default（主）

举例: Router (config-crypto-ezvpn)# peer 192.185.

0.5

Router (config-crypto-ezvpn)# peer 192.185.0.6

4. 定义操作模式

语法: **mode {client | network-extension}**

举例: Router (config-crypto-ezvpn)# mode client

5. 定义 Tunnel 建立方式

语法: **connect [auto | manual]**

举例: Router (config-crypto-ezvpn)# connect auto

6. 定义本地存储 Xauth 认证信息（可选）

语法: **username name password {0 | 6} {password}**

举例: Router (config-crypto-ezvpn)# username server 1 password 0 blue

7. 定义 Client 支持多子网（允许 Client 内部有多个网段访问 VPN）

语法: **acl { acl-name | acl-number}**

举例: Router (config-crypto-ezvpn)# acl acl-list1

8. 定义 VPN Tunnel 的超时时间

语法: **idle-time idle-time**

举例: Router (config-crypto-ezvpn)# idle-time 60

3.1.2 将策略应用到接口（内部和外部接口）

语法: **crypto ipsec client ezvpn name [inside/ outside]**

举例: Router (config-if)# crypto ipsec client ezvpn easy vpn remote1 outside

3.1.3 调试配置

1. 重置 EzVPN 连接

语法: **clear crypto ipsec client ezvpn**

举例: Router# clear crypto ipsec client ezvpn

2. 手工建立 E z V P N 连接

语法: **crypto ipsec client ezvpn connect name**

举例: Router# crypto ipsec client ezvpn connect easy
vpn remote1

3.2 Easy VPN Server

3.2.1 定义本地 Xauth 认证数据库

语法: **username name password encryption-type encrypted-password**

Example: Router (config)# username server r
password 7 121F0A18

3.2.2 为 Client 定义 Group Policy

1. 创建 Group Policy

语法: **crypto isakmp client configuration group { group-name | default}**

Example: Router (config)# crypto isakmp client configuration group group1

说明: 如果没有定义 Group-name , 则采用默认名称, 用户会自动采用默认策略。

2. 定义预共享密钥

语法: **key name**

Example: Router (config-isakmp-group)# key group1

3. 定义 DNS 地址

语法: **dns primary-server secondary-server**

Example: Router (config-isakmp-group)# dns 10.2.

2.2 10.3.3.3

4. 定义 W I N S 地址

语法: **wins primary-server secondary-server**

Example: Router (config-isakmp-group)# wins

10.10.10.10 10.12.12.12

5. 定义域名

语法: **domain name**

Example: Router (config-isakmp-group)# domain
domain.com

6. 定义地址池

语法: **pool name**

Example: Router (config-isakmp-group)# pool green

7. 定义 ACL (Client 端有多子网时)

语法: **acl number**

Example: Router (config-isakmp-group)# acl 199

用法: 当 Client 有多个非直连子网也需要访问 VPN 时使用

8. 指定必须通过 V P N 隧道的私网 D N S 解析的域名

语法: **split-dns domain-name**

Example: Router (config-isakmp-group)# split-dns
green.com

9. 限制 Group Policy 的 Client 只能通过某个接口连接 V P N

语法: **access-restrict { interface-name}**

Example: Router (config-isakmp-group)#access-restrict fastethernet0/0

10. 允许 Server 端访问 Client 端子网资源

语法: **include-local-lan**

Example: Router (config-isakmp-group)#include-lo-
cal-lan

11. 定义本地存储 X a u t h 密码信息

语法: **save-password**

Example: Router (config-isakmp-group)#save-pass-
word

12. 定义备份网关

语法: **backup-gateway**

Example: Router (config-isakmp-group)# backup
gateway

13. 启用 P F S

语法: **pfs**

Example: Router (config-isakmp-group)# pfs

3.2.3 应用 Group Policy 策略

1. 定义 Server 发起或者响应 Client 的 Mode Configuration 请求

语法: **crypto map tag client configuration address [initiate | respond]**

Example: Router (config)# crypto map dyn client configuration address initiate

说明: 如果 Client 为 Cisco Secure VPN Client 1.x, 则必须使用 **initiate** 参数, 其它 Client 使用 **respond** 即可。另外这两个参数可以同时使用。

2. 当收到用户的 VPN 协商请求后启用 IKE 的 Group Policy 查询

语法: **crypto map map-name isakmp authorization list list-name**

Example: Router (config)# crypto map ikessaaamap isakmp authorization list ikessaaalist

3. 启用 Xauth 认证, 并且调用认证方法

语法: **crypto map map-name client authentication list list-name**

Example: Router (config)# crypto map xauthmap client authentication list xauthlist

4. 将 IPsec 策略绑定到接口

语法: **crypto map crypto-map-name**

Example: Router (config-if)#crypto map ezvpnolicy

3.2.4 定义 IPsec 策略

1. 定义转换集

语法: **crypto ipsec transform-set name**

Example: Router (config)#crypto ipsec transform-set vpn esp-3des esp-sha-hmac

2. 定义静态或者动态加密图

语法: **crypto dynamic map-name seq-num**

or

crypto map map-name seq-num ipsec-isakmp

Example: Router (config)# crypto dynamic mymap 10
or

Router (config)# crypto map yourmap 15 ipsec-isakmp

3. 调用转换集

语法: **set transform-set transform-set-name**

Example: Router (config-crypto-map)# set transform-set dessha

4. 定义 Peer 地址

语法: **set peer ip-address**

Example: Router (config-crypto-map)# set peer 10.20.20.20

说明: 此项为可选配置。

5. 定义 RRI

语法: **reverse-route**

Example: Router (config-crypto-map)# reverse-route

说明: 此项为可选配置。

6. 定义保护流

语法: **match address**

Example: Router (config-crypto-map)# match address

说明: 此项为可选配置。

7. 查看 EzVPN Server 的配置

show crypto map [interface interface | tag-map-name]

Example: Router# show crypto map interface ethernet 0

3.2.5 定义 DPD

语法: **crypto isakmp keepalive secs retries**

Example: Router (config)# crypto isakmp keepalive 20 10

3.2.6 定义 Server 的访问限制

1. 允许同一个用户最大同时登陆数

语法: **max-logins number-of-logins**

Example: Router (config-isakmp-group)# max-logins 10

2. 允许 Server 最大同时连接数

语法: **max-users number-of-users**

Example: Router (config)# max-users 1000

3.3 配置举例

3.3.1 Client 模式

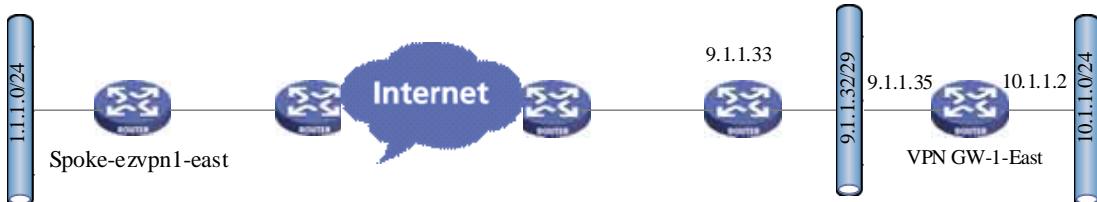


图 1 Client 模式配置举例

EzVPN Client 配置如下

```

spoke-ezvpn1-east#
!
hostname spoke-ezvpn1-east
!
crypto ipsec client ezvpn vpn
connect auto
group vpngroup key ciscoezvpn
local-address Ethernet0
mode client
peer 9.1.1.35
username ezvpn1@vpngroup password ezvpn1east
!
!
interface Ethernet0
ip address dhcp
load-interval 30
half-duplex
crypto ipsec client ezvpn vpn outside
!
interface FastEthernet0
ip address 1.1.1.1 255.255.255.0
load-interval 30
speed 100
full-duplex
no keepalive
crypto ipsec client ezvpn vpn inside
!
p route 0.0.0.0 0.0.0.0 Ethernet0
!
end
EzVPN Server 配置如下
vpn-gw1-east#
!
hostname vpn-gw1-east
!
username ezvpn password 0 east

```

```

username ezvpn1@vpngroup password 0 ezvpn1east
username ezvpn2@vpngroup password 0 ezvpn2east
aaa new-model
!
aaa authentication login vpn local
aaa authorization network vpn local
aaa session-id common
ip subnet-zero
!
crypto isakmp policy 1
encr 3des
authentication pre-share
group 2
crypto isakmp keepalive 10 10
!
crypto isakmp client configuration group vpngroup
key ciscoezvpn
dns 10.1.1.10
wins 10.1.1.11
pool vpnpool
include-local-lan
backup-gateway 9.1.1.36
!
!
crypto ipsec transform-set vpn esp-3des esp-sha-hmac
!
crypto dynamic-map dynamic 1
set transform-set vpn
reverse-route remote-peer 9.1.1.33
!
!
crypto map vpn client authentication list vpn
crypto map vpn isakmp authorization list vpn
crypto map vpn client configuration address respond
crypto map vpn 3 ipsec-isakmp dynamic dynamic
!
```

```

interface Loopback0
ip address 9.2.1.100 255.255.255.255
!
interface FastEthernet0/0
ip address 9.1.1.35 255.255.255.248
duplex full
crypto map vpn
!
interface FastEthernet2/0
ip address 100.1.1.147 255.255.255.0
duplex full
!
interface FastEthernet4/0
ip address 10.1.1.1 255.255.255.0
duplex full
!
router ospf 1
log-adjacency-changes
redistribute static subnets
network 10.1.1.0 0.0.0.255 area 0
!
ip local pool vpnpool 10.0.68.1 10.0.68.100
ip classless
ip route 0.0.0.0 0.0.0.0 9.1.1.33
!
radius-server host 100.1.1.4 auth-port 1645 acct-port
1646
radius-server key cisco
end

```

3.3.2 Network 扩展模式

要求如图 2

```

EzVPN client 配置如下
spoke-ezvpn1-east#
!
hostname spoke-ezvpn1-east
!
crypto ipsec client ezvpn vpn
connect auto
group vpngroup key ciscoezvpn
local-address Ethernet0
mode network-extension
acl 100
peer 9.1.1.35
username ezvpn1@vpngroup password ezvpn1east
!
interface Ethernet0
ip address dhcp
load-interval 30
half-duplex
crypto ipsec client ezvpn vpn outside
!
interface FastEthernet0
ip address 1.1.1.1 255.255.255.0
load-interval 30
speed 100
full-duplex
no keepalive
crypto ipsec client ezvpn vpn inside
!
access-list 100 permit 11.1.1.0 0.0.0.255 any
ip route 0.0.0.0 0.0.0.0 dhcp

```

注意：EzVPN Server 配置和 Client 模式一致。

配置模式

◆◇□ 李红霞

概述

IPsec 不会给用户分配私网地址，这样对于仅拥有公网地址的远程移动用户，无法访问 IPsec 网关保护的内部网络。通常使用 L2TP+IPsec 的方式为远程移动用户通过 IPsec 接入时分配私网地址。这种解决方法的原理为：首先在远程移动用户与企业网关之间建立 IPsec 隧道，然后在 IPsec 隧道上建立 L2TP 隧道，通过 PPP 对远程移动用户进行地址分配和对用户的验证。L2TP+IPsec 方案使用了 PPP 地址协商解决对移动用户的地址分配问题，但是这种方案需要同时使用两种隧道，配置复杂，开销较大，并且不便于维护。如果在 IPsec SA 建立之前能够为 IPsec 隧道移动端点分配一个私网地址，这样 IPsec 网关对报文解封装后，数据报可以使用私网地址，就像内网的 PC 一样访问 IPsec 网关保护的内部网络。CISCO 的远程接入方案就采用了这种方式，在 IPsec SA 创建之前由 VPN 网关将分配的私网地址“推”给客户端，这种方式使用一种被称为模式配置 (MODECONFIG) 的技术，实际上是对 IKEv1 的一种扩展技术。模式配置不仅可以实现 IP 地址分配，还可以实现 WINS、DNS 等相关 TCP/IP 参数的分配。

MODECFG 原理

2.1 概述

模式配置的消息在 IKE 第一阶段协商完成后，由客户端发出，共有两个消息：客户端发出的地址请求消息和 IPsec 网关的回应消息。

XAUTH 和 MODECFG 均属于 CISCO 的扩展属性 XAUTH 主要用于对用户的认证，关于 XAUTH 的介绍详见文档《XAUTH》。与 XAUTH 不同，模式配置属于必选项。

下图的消息 29-30 属于 MODECFG 的认证过程。

□ 22	10.1.1.65	10.1.1.2	ISAKMP	Aggressive
□ 23	10.1.1.2	10.1.1.65	ISAKMP	Aggressive
□ 24	10.1.1.65	10.1.1.2	ISAKMP	Aggressive
□ 25	10.1.1.2	10.1.1.65	ISAKMP	Transaction (Config Mode)
□ 26	10.1.1.65	10.1.1.2	ISAKMP	Transaction (Config Mode)
□ 27	10.1.1.2	10.1.1.65	ISAKMP	Transaction (Config Mode)
□ 28	10.1.1.65	10.1.1.2	ISAKMP	Transaction (Config Mode)
□ 29	10.1.1.65	10.1.1.2	ISAKMP	Transaction (Config Mode)
□ 30	10.1.1.2	10.1.1.65	ISAKMP	Transaction (Config Mode)
□ 31	10.1.1.65	10.1.1.2	ISAKMP	Quick Mode
□ 32	10.1.1.2	10.1.1.65	ISAKMP	Informational
□ 33	10.1.1.2	10.1.1.65	ISAKMP	Quick Mode
□ 34	10.1.1.65	10.1.1.2	ISAKMP	Quick Mode

图 1 MODECFG 认证报文

2.2 MODECFG 消息类型

MODECFG 共有两个消息，携带各种不同的属性：

ISAKMP-CFG-REQUEST：由 IPsec 客户端发出，请求 IPsec 网关分配地址，携带需要的 IP、DNS 等地址。

ISAKMP-CFG-REPLY：IPsec 网关回应消息，根据客户端的请求，填充分配的 IP、DNS 地址等内容。如果网关不能分配 DNS 服务器等的地址，则返回的消息中地址以 0.0.0.0 代替。该报文中必须携带为客户端分配的 IP 地址，如果网关不能为对端分配 IP 地址 (INTERNAL_IP4_ADDRESS)，则协商失败，不再进行后续的 IKE 第二阶段协商。

MODECFG 类似 PPP 的 IPCP 阶段。

2.3 MODECFG 过程

MODECFG 过程发生在 IKE 第一阶段和第二阶段之间，由 IKE 第一阶段协商的密钥进行加密，为密文传输。

简要说明一下的 MODECFG 获取地址的过程：

- 首先由 IPsec 客户端发起 REQUEST 请求，Attributes 的内容填写需要 IPsec 网关分配的地址，格式为：INTERNAL_IP4_ADDRESS (0.0.0.0)；
- IPsec 网关收到后，发送 REPLY 报文，按照网关的要求填充 Attributes，如果不能分配相应的地址则不改变属性中的内容，仍然填写 0.0.0.0。

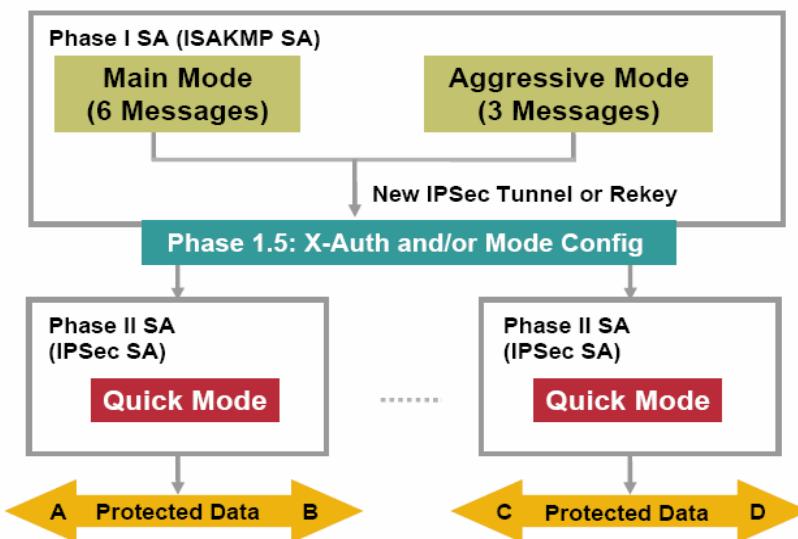


图 2 MODECFG 发生的阶段

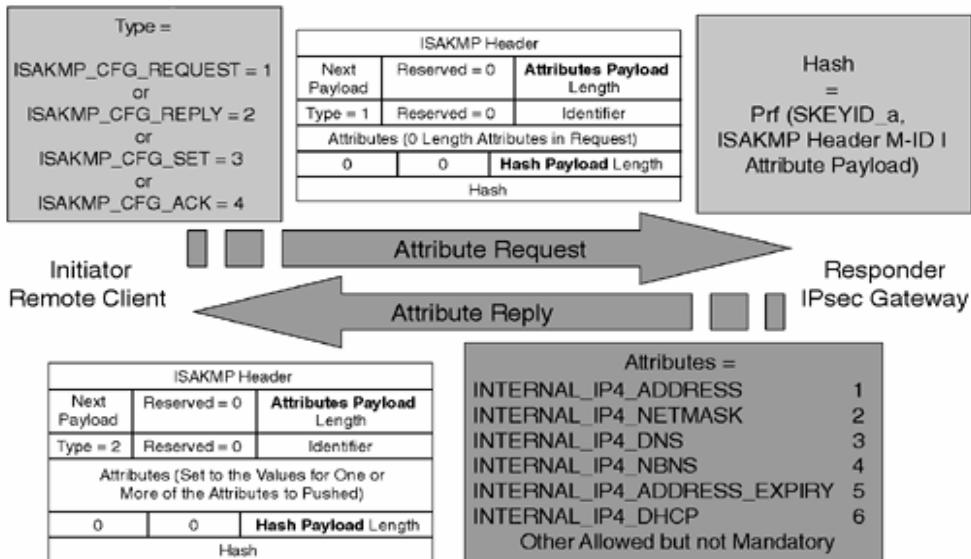


图 3 MODECFG 过程

2.4 MODECFG 重要属性

- INTERNAL_IP4_ADDRESS 和 INTERNAL_IP6_ADDRESS: 指定内部网络中的一个地址: 由 IPsec 网关分配, 客户端可以使用该地址访问 IPsec 网关保护的内部网络。用于确保请求安全的 IKE SA 过期之前, 请求的地址一直有效。如果请求与 IKE 第二阶段协商相关联, 该地址将在 IPsec SA 过期时失效。
- INTERNAL_IP4_NETMASK 和 INTERNAL_IP6_NETMASK: 内部网络的子网掩码。
- INTERNAL_IP4_DNS 和 INTERNAL_IP6_DNS: 指定网络中一台或多台 DNS 服务器的地址。应答方可能用零个、一个或多个 DNS 服务器属性进行应答。
- INTERNAL_IP4_NBNS 和 INTERNAL_IP6_NBNS: 指定网络中一台 NetBIOS 名称服务器 (NBNS) 的地址。可

能请求多个 NBNS, 应答方可能用零个、一个或多个 NBNS 属性进行应答。

和 XAUTH 一样, MODECFG 也不是 IETF IPsec 工作组标准。虽然 Cisco 定义了这种协议, 且大多数客户端实现能够同 Cisco 实现协同工作, 但鉴于这不是标准, 因此无法保证互操作性。

IKEv2 中对模式配置技术进行了继承, 只是形式进行了一些改变, 成为标准。

应用

MODECFG 的主要应用场景在前面也有所提及, 它主要是为远程办公人员通过 Internet 连接到企业内部网络时, 分配私网地址以及各服务器的地址。模式配置必须与 EasyVPN 结合使用, 也是 EasyVPN 使用中的必配项。

配置

支持在 PC (客户端) 和设备建立 IPsec 通道时使用 MODECFG 认证, 也支持在两个设备之间建

IPsec 通道时使用 MODECFG。
这里仅介绍客户端模式。
配置准备：需要在 PC 上安装支持 MODECFG 的 VPN Client 端。

4.1 客户端配置

- 4.1.1 客户端支持的 Cisco 产品的版本
 - Cisco VPN 3000 Concentrator Software Version 3.0 and later
 - Cisco IOS® Software Release 12.2(8)T and later
 - Cisco PIX® Security Appliance Software Version 6.0 and later
 - Cisco ASA 5500 Series Software Version 7.0 and later

4.1.2 配置说明

VPN Client 不需要做关于 MODECFG 的配置，其他配置在 XAUTH 的文档中给出了说明。

4.2 设备端的配置

MODECFG 需要和 Easy VPN 结合使用，必须要配置 Easy VPN 的相关命令。下面仅说明了与 MODECFG 相关的命令，配置 Easy VPN 在其他文档中说明。

4.2.1 Cisco IOS® Software Release 12.

2(8)T and later

如果不做特别的说明，配置均在系统视图下进行。

1. 使能 MODECFG，在 IKE 第一阶段和第二阶段之间触发 MODECFG

语法：**crypto map map-name client configuration address respond**

举例：crypto map dynmap client configuration address respond

2. 将 IPsec 策略绑到接口

进入接口视图后

语法：**crypto map map-name**

举例：crypto map dynmap

3. 新建认证的组名

语法：**crypto isakmp client configuration group name**

举例：crypto isakmp client configuration group hw-client-groupname

4. 新建组中设定 DNS 等服务器地址
进入到组的视图

语法：**dns dns-address1 dns-address2**

举例：dns 30.30.30.10 30.30.30.11

5. 新建组中指定给远程用户分配地址所使用的 pool

进入到组的视图

语法：**pool poolname**

举例：pool dynpool

6. 也可使用下面的命令指定给远程用户分配地址所使用的 pool

语法：**crypto isakmp client configuration address-pool local poolname**

举例：crypto isakmp client configuration address-pool local dynpool

说明：命令 5 和 6 使用一条即可。

7. 定义本地 pool

语法：**ip local pool poolname start-ip-address end-ip-address**

举例：ip local pool dynpool 30.30.30.20 30.30.30.30

4.2.2 Cisco VPN 3000

VPN 3000 具有图形化的界面，默认包含 EasyVPN Server 的配置，因此配置起来容易方便。

1. 服务器地址：

Configuration—>System—>Servers—>DNS
中填加地址。

2. 配置给远程用户分配地址的 pool：

Configuration—>System—>Address Management—>Pool 中填加。

4.3 典型配置举例

请参见 XAUTH 中典型配置举例部分。

IPsec 可靠性

◆◆◆ 李红霞



概述

我们知道，IPsec 可以为连接 Internet 的私网之间提供安全的通信。目前的实际网络中，企业总部和其位于各地分支站点之间使用安全网关提供 IPsec 服务建立 VPN 已经得到了非常广泛的应用。现代企业网络主要关心的是安全性和可靠性，IPsec 的安全性已经不容置疑，那么，如何实现 IPsec 的高可靠性就成为 IPsec 使用者最关心的问题。

IETF IPsec 工作组在 IPsec 的设计和发展过程中没有对于其可靠性进行专门的设计和定义，因此 IPsec 的可靠性是无法通过 IPsec 协议本身来实现，只能通过各个厂商对 IPsec 协议扩展定义的特性来完成，这样也带来了兼容性和易用性的问题。

和传统 IP 网络类似，IPsec 的可靠性也体现在链路和设备的冗余这两个方面，但是 IPsec 的可靠性设计又比传统网络复杂得多，回顾一下 IPsec 的典型组网，如下图 1 所示：



图 1 IPsec 基本组网图

网关 A 和网关 B 通过主干网（一般为 Internet）互联，在网关 A 和网关 B 之间建立 IPsec 隧道来封装和加密来自网关后面的私网流量。这种 IPsec 典型应用的可靠性实际上非常的脆弱，来自链路或者设备的故障都会导致 IPsec 网络的崩溃，如下图 2 所示：

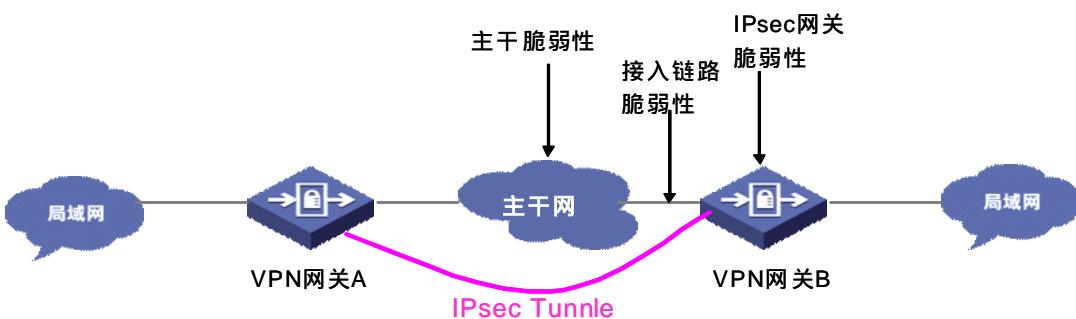


图 2 IPsec 网络组件脆弱性

显而易见，提高 IPsec 可靠性主要就是提供主干网络、接入链路和 IPsec 网关的冗余能力。主干网络一般使用 Internet 或者运营商提供的专网，而这种网络的可靠性作为用户来讲是没有能力保障的。当然，我们可以采取使用多个运营商网络作为主干网络冗余，但是这种设计应该归为接入链路的可靠性。因此主干网络的可靠性设计不应该属于 IPsec 可靠性设计的范畴。

下面分别介绍一下如何实现 IPsec 网络接入链冗余和网络设备的冗余。

接入链路的冗余

实际上，接入链路的冗余设计并不是IPsec网络的专利，这种设计在传统IP网络中使用也非常广泛。接入链路的冗余实际上就是为接入设备连接主干网提供多个出口，在某个接入链路出现故障时能够切换到其他链路，在保证冗余能力的同时往往还可以提供负载均衡。

因为IPsec VPN建立的前提条件是安全网关之间首先要协商出用于认证和加密的IKE/IPsec SA，IPsec的接入链路冗余设计和传统网络相比要更加复杂。当本地安全网关主链路失效时，启动备份链路可能会因为安全网关地址发生变化而导致远程设备和本地的IPsec协商失败。因此，对于IPsec网络的接入链路冗余要重点考虑到IKE/IPsec的协商情况和SA的变化。

目前在启用了多条接入链路的条件下通常使用单个IKE SA或者多个IKE SA两种方式来提高可靠性。

2.1 单个IKE SA

单个IKE SA是指远程安全网关和本地安全网关之间只建立一个IKE邻居关系，解决了链路切换时地址变化造成的问题。

由于连接多个冗余链路一般都会产生多个出口地址，因此远程安全网关一般不和本地网关的某个出口地址建立IKE邻居关系，而是和本地网

的环回地址建立IKE邻居关系。

在单个IKE SA模型中，链路的切换虽然不会导致IKE邻居关系的中断，但是会造成本地报文出接口发生变化，这样就必须在所有报文出接口上配置相同的IPsec策略，原因是显而易见的，如果IPsec策略不同就不能和同一个远程安全网关建立IKE/IPsec SA。

此外，这种模型有一个明显的缺点，在本地网关的主链路出现故障，由路由协议切换到备份链路时，备份链路的接口还没有和远程网关之间建立IPsec SA，如果此时是本地网关发送数据，会重新和远程网关协商新的IPsec SA，协商过程中会导致少量的报文丢失。更糟糕的情况是：切换到备份链路时，远程网关无法感知链路的切换，因此不会重新和本地网关协商新的IPsec SA，仍然使用旧的IPsec SA加密数据发往本地网关的备份接口，本地网关备份接口由于没有相应的IPsec SA会一直丢包，直到本地有流量触发新的IPsec SA协商。

一些设备厂商开发了的IPsec的新特性可以有效的解决单个IKE SA模型带来的缺点。下面就通过一个具体实例来说明这些IPsec特性，下图所示的网络就是单个IKE SA的典型应用：如图3

网关A和网关B建立IPsec连接，网关B采用冗余链路连接主干网，每个冗余链路分别有一个地址用于接入主干。

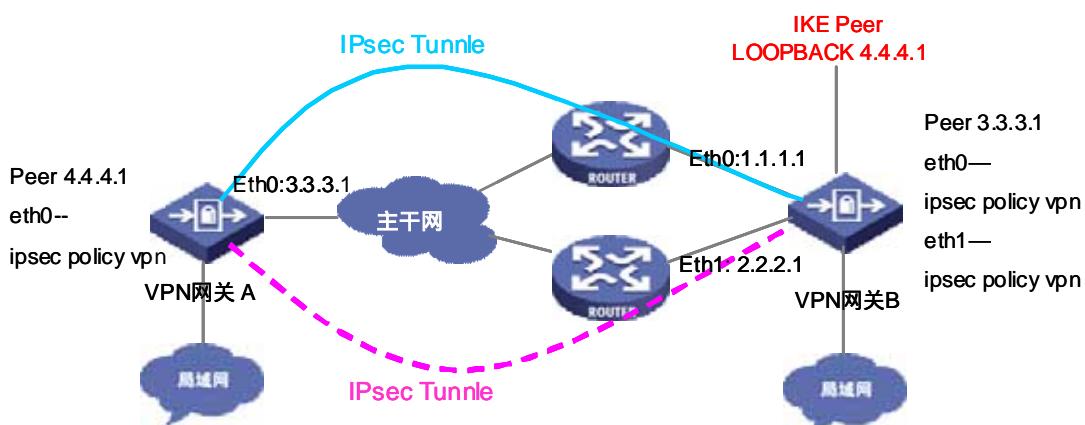


图3 VPN B仅有一个IKE SA

网关B有发往网关A的流量时触发IPsec SA协商，流量才会恢复正常。问题的关键在于备份接口没有初始IPsec SA造成的流量长时间或者短时间的中断，思科针对这个问题开发了IPsec SA复制的特性解决了这个问题。

当启用了IPsec SA复制功能，本地安全网关协商成功IPsec Sa以后，会立即将该SA向所有配置了相同IPsec策略的接口复制，使每个接口都有相同的IPsec SA。这样上面的问题就迎刃而解了：主链路失败备份链路会立即开始加密或者解密报文，而不需要再协商新的IPsec SA，这种切换是没有任何延迟和丢包的，从远程设备的角度来看，甚至并不知道本地设备进行了切换。

值得注意的是，IPsec Sa的复制并不仅仅是在刚开始协商的过程中进行，而是需要在每次IPsec重新协商时都要复制，这样才能保证备份链路接口的SA在任何时候都和主链路接口一致。

但是，这种方案也有其固有的缺点，要保证VPN网关B的环回接口的IP可达性，需要给环回接口分配公网地址，将会造成公网地址的浪费；或者在两条链路上均启用GRE封装环回接口地址并且运行静态或者动态路由，这样配置和维护将会非常复杂。

2.2 多个IKE SA

多个IKE SA模型是指在本地存在冗余链路的情况下，远程设备和本地的每个冗余链路的接口地址建立IKE连接，这样带来的优势是IKE邻居的可达性可以很容易实现，但是同时带来了链路切换时IKE SA切换的复杂性。

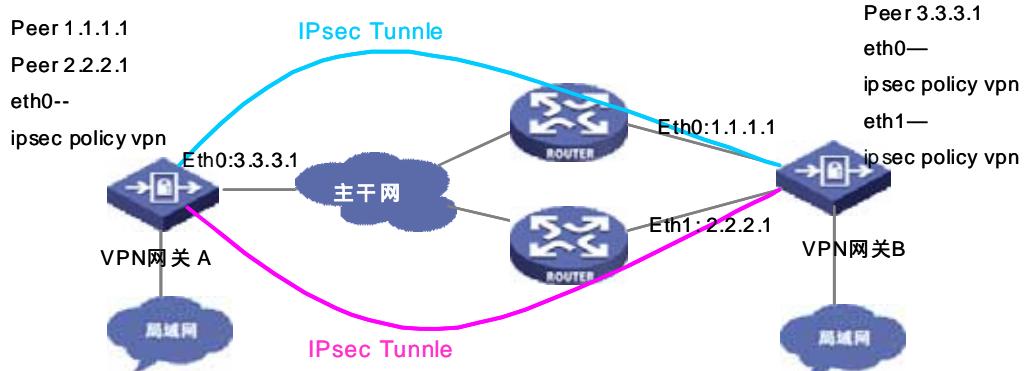


图4 允余的IPsec接入链路

如下图4所示，网关B连接了两个冗余链路，网关A和网关B的每个链路地址分别建立一个IKE邻居，对于网关A来讲是一个IPsec策略下定义了多个IKE Peer，网关A首先使用第一个IKE Peer（即网关B的主链路接口地址）建立IKE SA和IPsec SA，当网关B的主链路失效时，网关A通过IKE Keepalive或者Heartbeat机制发现IKE Peer已经失效，会发起和其配置的第二个IKE Peer的协商（即网关B的备份链路接口地址），将IPsec隧道切换到网关B的备份链路。此时的链路切换是由网关A完成的。

网关B需要在所有冗余链路接口绑定相同的IPsec策略，当网关B的主链路失效时，网关B会将报文发往备份接口即E1接口，触发IKE和IPsec协商。

当网关B的主链路失效时，网关A通过IKE Keepalive或者Heartbeat机制发现IKE Peer已经失效，会发起和其配置的第二个IKE Peer的协商（即网关B的备份链路接口地址），将IPsec隧道切换到网关B的备份链路。此时的链路切换是由网关A完成的。

网关B需要在所有冗余链路接口绑定相同的IPsec策略，当网关B的主链路失效时，网关B会将报文发往备份接口即E1接口，触发IKE和IPsec协商。

多个IKE SA模型的最大缺点就是切换时间很长，远程网关需要很长时间才能够感知IKE Peer失败从而切换到备份链路，而切换时间往往是可靠性设计中需要着重考虑的问题，因此多个IKE SA模型并不是很常用。

IPsec 网关冗余

前面介绍的接入链路冗余模型解决了 VPN 中最常见的故障：接入链路故障。通过接入链路中添加冗余元素来提高系统的适应性，这样也必然会增加系统的复杂度，在配置和维护上都会更加麻烦。鱼和熊掌不能兼得，网络的设计只能在可扩展性、汇聚速度、简单性和性能之间进行折衷。当然，综合各项因素，我们首先关注的是对网络各个组成部分冗余的充分的考虑，虽然 IKE SA 恢复机制提供了链路故障的功能，但这些机制并不能抵御 IPsec 网关故障以及 IPsec 网关后面的子网不可达的故障；这样，IPsec 网关的冗余方案就需要引起重视了。如果 IPsec 节点出现故障，将影响与节点相连的 IPsec 网关的连通性。我们结合传统网关的冗余，在其上添加 IPsec 冗余的特性，形成以下 3 种解决方案。

3.1 简单 IPsec 网关冗余

为抵御 IPsec 节点故障，最简单的方法是再添加一个节点，使用两个 IPsec 网关；通过路由的选择进行主备的切换，当主设备出现故障后，启用备份设备。我们需要在主备设备上配置相同的 IPsec 策略来保证和远端的网关成功协商 IPsec / IKE SA。

如图 5 所示：在 VPN 网关 A 的一个 IPsec 策略中引用两个 IKE Peer，VPN 网关 A 首先和主设

备 VPN 网关 B-1 建立 IPsec / IKE SA，如果尝试失败，它将使用相同的策略尝试和备份设备 VPN 网关 B-2 建立 IPsec / IKE SA。这种方案虽然能够解决当主设备出现故障时，切换到备份设备，但是两个网关之间只是简单的备份关系，没有包括心跳、主从确定等消息，将会不可避免的导致一些问题：

- 1、当主设备故障后，切换到备用设备的过程中，远端网关的 IKE SA 仍然使用和主设备协商出来的，备用设备上没有匹配的 IKE SA / IPsec SA，必须重新协商，将会造成丢包。而从备用设备切换回主设备时同样会出现这个问题。

- 2、因为远端网关和本地网关之间交换 IKE

Keepalive 消息，当本地网关的私网接口（和网络 B 相连的接口）出现故障，这种故障不会和 IPsec 状态同步；在远端网关看来本地网关并没有出现故障。这样，数据报文都将在解密后发送到一个黑洞中。

单纯的设备备份目前已经很少使用，因为这种可靠性的方案的故障恢复时间长，在使用的过过程中需要路由协议作配合才能保证主备的切换，并且不能对本地网关后面的私网的状态进行同步。对 IP 链路的可靠性的维护中，我们通常使用 VRRP 技术确保在通信线路或设备故障时提供备用方案，从而保障数据通信的畅通，有效增强了网络的强壮性和可靠性。下面将介绍如何使用 VRRP 进行 IPsec 网关冗余。

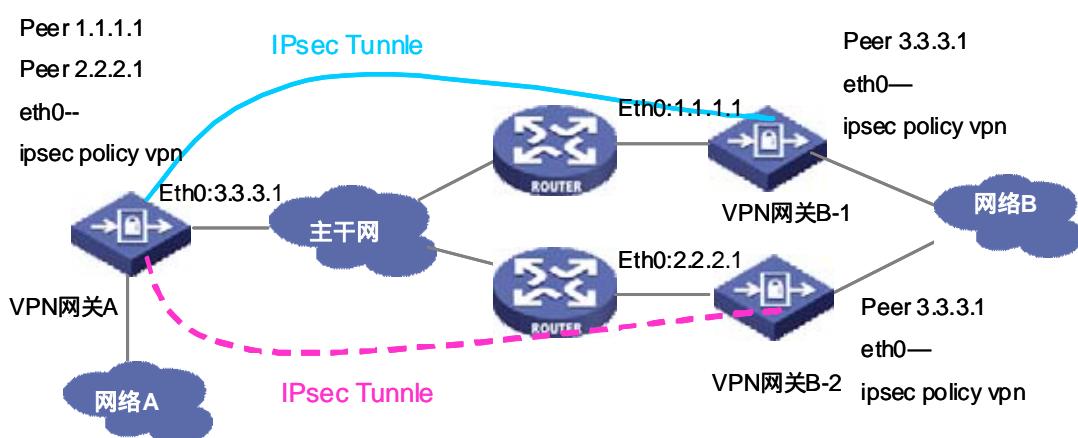


图 5 简单 IPsec 网关冗余

3.2 使用 VRRP 的 IPsec 网关冗余

VRRP 技术通过将多台路由器组成一个备份组作为本地网络统一的出口网关，而备份组内的路由器对远端网络是透明的。在备份组内有一台路由器处于活动状态，承担报文转发任务，一台路由器处于备份状态并随时接替任务，其余路由器处于监听状态。当处于活动状态的路由器出现故障时，将会由处于备份状态的路由器接替其工作，而其余处于监听状态的路由器将再决定出另一台路由器担任备份工作。这样远端主机就可以不作任何修改地继续工作，极大地提高了通信的可靠性。如下图 6 所示

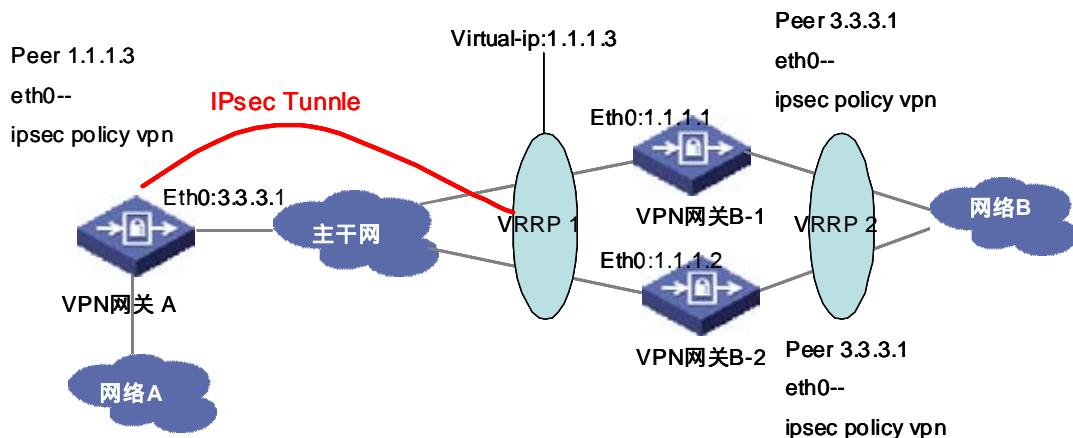


图 6 使用 VRRP 的 IPsec 网关冗余

VRRP 使得本地一个 IPsec 网关处于活动状态，另一个 IPsec 网关处于备用状态。VPN 网关 A 和 VRRP 虚地址建立连接，该地址由 Master 的 IPsec 网关拥有；故障切换时，远端网关并不知道，仍然正常工作。

相对于简单 IPsec 网关冗余，这种使用 VRRP 的 IPsec 网关冗余的优点如下：

- 1、VPN 网关 A 上的 IPsec 策略只用配置一个 IKE Peer，因此不存在 IKE Peer 的配置排序问题。

- 2、VRRP 能够跟踪 VPN 网关 B 上的另一个接口（私网接口）的状态，并决定 IPsec 网关是 Master 或者 Backup。

- 3、不必使用路由协议来确定主从关系，VRRP

通过心跳和抢占自动协商出主备设备。配置和维护更加方便。

使用 VRRP 的 IPsec 网关冗余的一个重要的原因是，路由选择状态和 IPsec 状态是同步的，当网关后面保护的私网出现故障时，VRRP 的外网接口通过监视私网接口的状态来进行主从关系的抢占，最大限度的降低了出现故障时形成丢弃数据的黑洞的可能性。

但是，当主设备出现故障，备份设备接替后，远端网关并不知道这种变化将继续发送加密报文，只有当 VPN 网关 A 等待 IKE SA 的超时后才会进行

IPsec/IKE SA 的重新协商，这样会相当长的时间（3 个 KeepAlive 周期）内出现丢包。IETF 工作组针对这个问题提出了 DPD (Dead Peer Detect) 草案。下面将详细介绍 DPD+VRRP 的模型进行 IPsec 网关冗余。

3.3 VRRP+DPD 的 IPsec 网关冗余

对 IPsec 的增强特性中有一个特性：DPD。DPD 用于 IPsec 邻居状态的检测。启动 DPD 功能后，当接收端长时间收不到对端的 IPsec 加密报文时，能够触发 DPD 查询，主动向对端发送请求报文，对 IKE Peer 是否存在进行检测。

当 IPsec Peer 需要断开连接时，使用 reset IKE SA 可以通过 IKE 发送一个删除通知负载给

给对等体。但是如果由于网络断开等原因远端 peer 非正常下线，本地是无法收到删除通知负载的。这样就会造成本地仍然使用 IPsec SA 加密报文发送给对方。尽管 IPsec 使用 KeepAlive 机制可以保证一段时间以后发现 Peer 断开，但是在 KeepAlive 触发时间内是无法解决这样的问题的。使用 DPD 可以解决 KeepAlive 对 peer 非正常下线处理不善的问题。触发 DPD 的条件是：发送 IPsec 加密报文之前和 DPD 计时器超时后。

IPsec 网关可以根据对每个 Peer 隧道断开时间的容忍度单独定义 DPD 计时器，例如在 VPN 网关 A 上定义 DPD 超时 10 秒，那么当在 10 秒内没有收到 VPN 网关 B 的 IPsec 加密报文，本地触发 DPD。或者当 VPN 网关 A 要发送加密报文前触发 DPD。

关于 DPD 的详细介绍请参看文档《DPD》。

使用 VRRP+DPD，在每次发送 IPsec 加密报文之间发送 DPD 查询，确定对端 IKE SA 的存活情况，如果 IKE SA 处于 Active 的状态，则正常发送报文；如果 3 次 DPD 查询都没有回应，则重新进行 IKE SA 和 IPsec SA 的协商。这样就能有效的避免在 KeepAlive 触发时间内对端 IKE SA 已经删除，但本端仍然给对端发送 IPsec 加密报文。

VRRP+DPD 的组合已经成为目前 IPsec 高可靠性方案中最常见的应用，但是这种方案却不能满足对主备链路切换时延敏感的应用需求：DPD 采足对主备链路切换时延敏感的应用需求：DPD 采用了超时机制来发现 IKE 连接中断，这样就不可

避免的带来较大的切换时延，而且切换到备份链路仍然需要重新协商 IKE / IPsec SA。为了进一步减少切换时延，一种更好的高可用性的解决方案是：让备份组的 IPsec 网关能够共享 IPsec 状态，即使用有状态故障切换模型。有状态的故障切换的模型的主要优点是，因为实时将 IPsec SA 的信息进行了备份，远程分支无需同备用网关之间重新协商 IPsec SA。

3.4 IPsec 状态备份

3.4.1 简介

有状态故障切换模型是指在主备链路的接口上同步 IKE / IPsec 的状态信息，使主备链路接口在任何时候都有相同的 IKE / IPsec SA，这样不管链路如何切换，都不会影响报文的加密和解密。

这种方案的关键是 IKE / IPsec SA 的同步过程，IKE / IPsec SA 的建立和维护过程非常复杂，当切换到备份链路时，备份链路通过同步获得的 IKE / IPsec SA 必须能够和远程安全网关正常通信，这样也就意味着在同步过程中需要将 SA 的关键信息（例如 IKE Cookie、IPsec SA 的生存时间、SPI、序列号）完全的复制。

有状态故障切换一般会和 VRRP 结合使用，和使用路由协议切换链路相比，虽然 VRRP 会带来一定的切换时延，但是采用多组 VRRP 可以同时监控设备的多个链路和接口状态，可以带来更高的可靠性。

有状态故障切换的典型应用如下图 8 所示

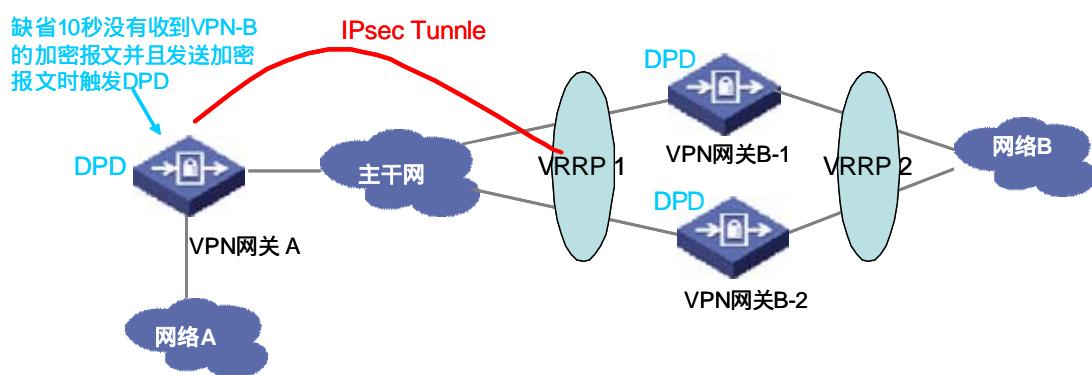


图 7 使用 VRRP+DPD 的 IPsec 网关冗余

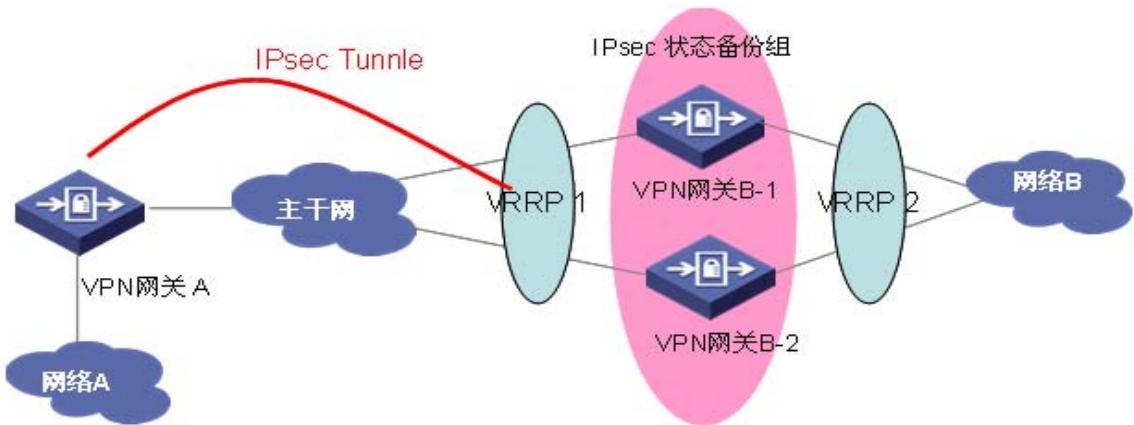


图 8 有状态 IPsec 故障切换

VPN 网关 B-1 和 VPN 网关 B-2 组成了两个 VRRP 组和一个状态备份组，VRRP 组用于监控链路和接口状态，状态备份组用于同步 S A，一般情况下，状态备份组采用专门的可靠链路以提高状态备份的可用性。

3.4.2 IPsec 状态备份的实现

IPsec 相关 RFC 并没有定义状态备份，状态备份完全是由设备厂商自己扩展的 IPsec 特性，我司目前不支持状态备份，下面以思科为例介绍一下状态备份的实现。

思科定义了两个私有协议用于完成 IPsec 的状态备份：SSP 和 SSO，可以采用任意一种。

1. SSP (State Synchronization Protocol)

SSP 是 Cisco 支持 IPsec 状态备份的协议，是在 VRRP 上的扩展，用来同步 VRRP 组中主设备和备份设备的 SADB。

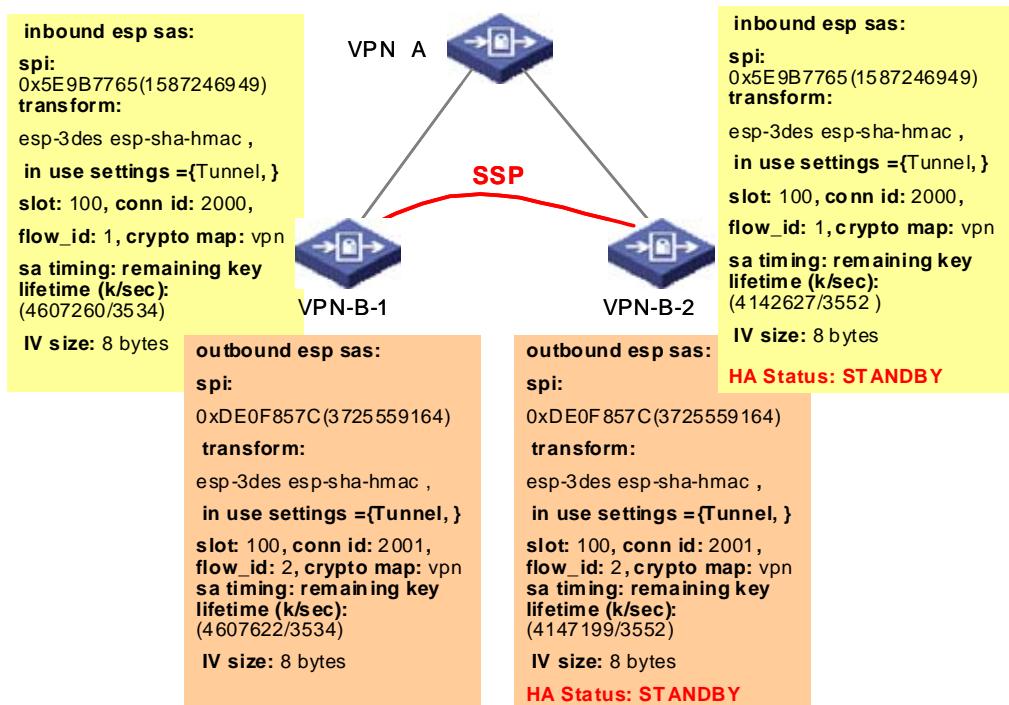


图 9 使用 SSP 进行状态备份

在两个VPN网关上配置VRRP可确保IP路由选择，但如果采用有状态备份，当主IPsec网关出现故障时，大型VPN的收敛时间将相当长。出现这种延迟是因为必须等待原有的SA超时，并需要在所有分支和新的VPN网关之间建立IKE SA和IPsec SA。采用SSP时，使用一个控制信道将SA信息复制到用作备用的VRRP节点的对等体的接口，确保备用网关和备用IPsec对等体有足够的信息，能够立刻承担起主IPsec对等体的角色。在使用有状态故障切换的情况下，无需重新建立IKE SA和IPsec SA。

当然，除了相关的VRRP和SSP的配置，还需要主和备份路由器上有相同的IPsec配置。

2. SSO(Stateful Switch Over)

前面介绍了使用SSP实现IPsec状态备份，另一种实现方法是使用SSO。从最终用户的角度看，这两种机制除配置语法外没有什么不同。SSP机制是专门为实现IPsec状态备份的，而SSO机制使用一种更通用的高可用性基础结构，除IPsec外，它还用于在Cisco IOS中众多其他协议中提供状态备份机制，如OSPF、BGP、IP等。配置IPSec状态备份时，SSO是推荐的方法。

可靠性方案的比较

上面已经介绍了比较常用的IPsec可靠性的解决方案，现在对可靠性比较关心的几个方面做出比较：

	重协商IKE SA	重协商IPsec SA	丢包	切换时延	易用性和可维护性	标准性
单个IKE SA	否	是	较少	较短	很差	否
多个IKE SA	是	是	多	长	较差	否
简单IPsec网关冗余	是	是	多	长	一般	否
使用VRRP的IPsec网关冗余	是	是	多	较短	好	是
VRRP+DPD的IPsec网关冗余	是	是	较少	较短	好	是
IPsec状态备份	否	否	很少	很短	好	否

通过上面的比较我们也可以发现，每种方案都有自己的优缺点，在实际的网络设计中，网络的拓扑、需求千变万化，没有一种万能的方案适合所有网络，网络设计者必须根据实际情况选择最适合的方案，这样就使我们对IPsec的理解提出了更高的要求，只有通过大量的实践积累才能设计更加完善的IPsec网络。

三种 VPN 技术的比较

◆◇□ 刘洋

分别就 IPsec VPN 和 SSL VPN 技术，IPsec VPN 和 MPLS VPN 技术进行一下比较。

比较 IPsec VPN 和 SSL VPN 技术

1.1 IPsec VPN 技术简介

IPsec 是 IETF (因特网工程任务组) 提出的 IP 安全标准。它为数据在通过公用网络 (如因特网) 在网络层进行传输时提供安全保障。IPsec VPN 在 IP 层对数据包进行高强度的加密和验证，使安全服务独立于各种应用程序。通信双方要建立 IPsec 通道，首先要采用一定的方式建立通信连接。因为 IPsec 协议支持几种操作模式，所以通信双方先要确定所要采用的安全策略和使用模式，这包括加密运算法则和身份验证方法类型等。在 IPsec 协议中，一旦 IPsec 通道建立，所有在网络层之上的协议在通信双方都经过加密，如 TCP、UDP、SNMP、HTTP、POP 等，而不管这些通道构建时所采用的安全和加密方法如何。IPsec 协议主要由三部分组成：ESP (Encapsulation Security Payload，封装安全有效负载)、AH (Authentication Header，鉴别头部)、IKE (Internet Key Exchange，Internet 密钥交换)。

IPsec 的工作原理类似于包过滤防火墙，也可以看做是对包过滤防火墙的一种扩展。当接收到一个 IP 数据包时，包过滤防火墙是使用其头部在一个规则表中进行匹配。当找到一个相匹配的规则时，包过滤防火墙就按照该规则制定的方法对接收到的 IP 数据包进行处理。一般的处理方法就是两种：转发和丢弃。IPsec 通过查询 SPD (Security Policy Database，安全策略数据库) 决定对接收到的 IP 数据包的处理。但是 IPsec 不同于包过滤防火墙的是，对 IP 数据包的处理方法除了丢弃，直接转发外，还有一种，即进行 IPsec 处理。正是这种新增添的处理方法提供了比包过滤防火墙更进一步的网络安全特性。

进行 IPsec 处理意味着对 IP 数据包进行加密和认证。包过滤防火墙只能控制来自或去往某个站点的 IP 数据包的通过，可以拒绝来自某个外部站点的 IP 数据包访问内部某些站点，也可以拒绝某个内部站点对某些外部网站的访问。但是包过滤防火墙不能保证自内部网络出去的数据包不被截取，也不能保证进入内部网络的数据包未经过篡改。只有在对 IP 数据包实施了加密和认证后，才能保证在外部网络传输的数据包的机密性、真实性、完整性，通过 Internet 进行安全的通信才成为可能。

IPsec 协议应用可以只对 IP 数据包进行加密，也可以只进行认证，还可以同时实施二者。但无论是进行加密还是进行认证，IPsec 都有两种工作模式：传输模式和隧道模式。如果以传输模式操作，源主机和目的主机必须直接执行所有的密码处理操作。数据（密码文件）是由源主机生成并由目的主机检索的。这种操作模式可以实现端对端安全。如果以隧道模式操作，除了源主机和目的主机之外，特殊的网关也将执行密码处理操作。在这种模式里，在网关之间将会产生许多隧道，从而可以实现网关对网关的安全。

IPsec 是网络层的 VPN 技术，这表明它独立于应用程序。IPsec 用自己的封装原始 IP 信息，因此可隐藏所有应用协议的信息。一旦 IPsec 建立加密隧道后，就可以实现各种类型的一对多的连接，如 Web、电子邮件、文件传输、VoIP 等连接，并且每个传输必然对应到 VPN 网关之后的相应的服务器上。

1.2 SSL VPN 技术简介

SSL 的英文全称是“Secure Sockets Layer”，中文名为“安全套接字协议层”。SSL 协议指定了一种在应用程序协议（如 Http、Telnet、SMTP 和 FTP 等）和 TCP/IP 协议之间提供数据安全性分层的机制，它为 TCP/IP 连接提供数据加密、服务器

认证、消息完整性以及可选的客户机认证。而 SSL VPN 指的是以 HTTPS（以 SSL 为基础的 HTTP）为基础的 VPN，而大多数计算机在出货时，都已经安装了支持 HTTP 和 HTTPS 的 Web 浏览器。目前，SSL 已由 TLS 传输层安全协议（RFC 2246）整合取代，它工作在 TCP 之上。如同 IPsec/IKE 一样，它必须首先进行配置，包括使用公钥与对称密钥加密以交换信息。此种交换通过数字签名让客户端使用已验证的服务器，还可选择性地通过签名或其他方法让服务器验证客户端的合法性，接着可安全地产生会话密钥进行信息加密并提供完整性检查。SSL 可利用各种公钥（RSA、DSA）算法、对称密钥算法（DES、3DES、RC4）和完整性（MD5、SHA-1）算法。

SSL 协议的主要用途是在两个通信应用程序之间提供私密性和可靠性，这个过程通过 3 个元素来完成。

（1）握手协议

这个协议负责客户机和服务器之间会话的加密参数的协商。当一个 SSL 客户机和服务器第一次开始通信时，它们在一个协议版本上达成一致，选择加密算法和认证方式，并使用公钥技术来生成共享密钥。

（2）记录协议

这个协议用于交换应用数据。应用程序消息被分割成可管理的数据块，还可以压缩，并产生一个 MAC（消息认证代码），然后结果被加密并传输。接收方接收数据并对它解密，校验 MAC，解压并重新组合，把结果提供给应用程序协议。

（3）警告协议

这个协议用于实时警示在什么时候发生了错误或两个主机之间的会话在什么时候终止。

1.3 二者对比

首先，我们从以上对两种 VPN 技术的分析可知，这两种技术目前应用在不同的领域。SSL VPN

考虑的是应用软件的安全性，其协议工作在传输层，保护的是应用程序与应用程序之间的安全连接，更多应用在 Web 的远程安全接入方面；而 IPsec VPN 应用在两个网络设备之间通过专线或互联网的安全连接，以及两台服务器之间的安全连接，其协议工作在 IP 层，保护的是点对点之间的通信。并且，它不局限于 Web 应用。

IPsec VPN 最大的难点在于客户端需要安装复杂的软件，而且当用户的 VPN 策略稍微有所改变时，VPN 的管理难度将呈几何级数增长。SSL VPN 则正好相反，客户端不需要安装任何软件或硬件，使用标准的浏览器，就可通过简单的 SSL 安全加密协议，安全地访问网络中的信息。SSL VPN 的优势在于 Web 应用方面。SSL 在 Web 的易用性和安全性方面架起了一座桥梁。目前，对 SSL VPN 公认的三大好处是：

(1) SSL 安装简单。它不需要配置，可以立即安装，立即生效。

(2) 客户端不需要麻烦的安装。直接利用浏览器中内嵌的 SSL 协议就行。

(3) 兼容性好。传统的 IPsec VPN 对客户端采用的操作系统版本具有很高的要求，不同的终端操作系统需要不同的客户端软件，而 SSL VPN 则完全没有这样的麻烦。

其次，IPsec 和 SSL 采用 Internet 网络中的不同层来进行安全加密处理，以建立网络安全通道，因此在网络的安全管理等级上，各有所长。以下我们分别从不同的角度来探讨这两种 VPN 应用的安全区别。

(1) 安全通道 (Secure Tunnel)

IPsec 和 SSL 这两种安全协议都有采用对称式 (Symmetric) 和非对称式 (Asymmetric) 的加密算法来执行加密作业。在安全的通道比较上，并没有谁好谁坏之差别，仅在于应用上的不同，满足的需求不同而已。

(2) 认证和权限控管

IPsec 采取 Internet Key Exchange (IKE) 方式，使用数字证书 (Digital Certificate) 或是一组 Secret Key 来做认证，而 SSL 仅能使用数字证书。如果都是采取数字证书来认证，两者在认证的安全等级上就没有太大的差别。SSL 的认证，大多数的厂商都会建置硬件的 token，来提升认证的安全性。对于使用权限的控管，IPsec 可以支持 “Selectors”，让网络封包过滤阻隔某些特定的主机应用系统。但是在实际应用中，大多数人都是开放整个网段，以避免太多的设定所造成的麻烦。SSL 可以设定不同的使用者，执行不同的应用系统，它在管理和设定上比 IPsec 简单方便了许多。

(3) 应用系统的攻击

远程用户以 IPsec VPN 的方式与公司内部网络建立联机之后，内部网络所连接的应用系统，都是可以侦测得到，这就为黑客提供了攻击的机会。若是采取 SSL VPN 来联机，因为是直接开启应用系统，并没有在网络层上建立连接，黑客并不容易侦测出内部网络的实际情况，受到攻击威胁的部分一般只是局限于联机的这台主机系统，从而使得网络不安全的因素大为减少了。

(4) 病毒入侵

一般企业在 Internet 的入口，都会采取适当的防毒侦测措施。不论是 IPsec VPN 或 SSL VPN 联机，对于入口的病毒侦测效果是相同的，但是比较从远程客户端入侵的可能性，就会有所差别。采用 IPsec 联机，若是客户端电脑遭到病毒感染，这个病毒就有机会感染到内部网络所连接的每台电脑。相对于 SSL VPN 的联机，如果某台电脑感染了病毒，一般只会局限于这台主机，不太可能迅速扩散到其他主机。而且所感染的病毒的类型必须是针对某种应用程序的病毒类型，其他类型的病毒是不会通过 SSL VPN 联机感染到这台主机的。

(5) 防火墙上的通讯端口 (port)

在 TCP / IP 的网络架构上，各式各样的应用系统会采取不同的通讯协议，并且通过不同的通讯端口来作为服务器和客户端之间的数据传输通道。以 Internet Email 系统来说，发信和收信一般都是采取 SMTP 和 POP3 通讯协议，而且两种通讯端口是采用 port 25 和 110，若是从远程电脑来联机 Email 服务器，就必须在防火墙上开放这两个端口，否则远程电脑是无法与 SMTP 和 POP3 主机沟通的。IPsec VPN 联机就会有这个困扰和安全顾虑。在防火墙上，每开启一个通讯端口，就多一个黑客攻击机会。反观之，SSL VPN 就没这方面的困扰。因为在远程主机与 SSL VPN Gateway 之间，采用 SSL 通讯端口 (port 443) 来作为这方面的困扰。因为在远程主机与 SSL VPN Gateway 之间，采用 SSL 通讯端口 (port 443) 来作为传输通道，这个通讯端口，一般是作为 WebServer 对外的数据传输通道，因此，不需在

防火墙上做任何修改，也不会因为不同应用系统的需求，而来修改防火墙上的设定，减少 IT 管理者的困扰。

6) 两种 VPN 的选择

表 1 列出了根据不同要求选用不同 VPN 的一些参考标准

(7) 两种 VPN 的层次结构差异比较

比较 IPsec VPN 和 MPLS VPN

2.1 MPLS VPN 技术简介

RFC 2547 提出来的 BGP-MPLS VPN 方案，它属于对等模型。此方案的基本原理是利用扩展的 BGP4 协议分发 VPN 用户站点的路由信息，同时形成第一级的标记交换路径，它用于不同 VPN 的区分；在服务提供商 MPLS 骨干网上利用 IGP 协议进行路由，同时利用 LDP 或 RSVP 或 CR-LDP

需求类型	适合使用 IPsec VPN	适合使用 SSL VPN
用户访问 VPN 方式	固定用户，实时在线用户	移动用户，临时访问用户
用户身份	以公司员工为主	以非公司员工为主
需要访问的资源	所有授权资源	Email，文件共享和基于 web 的应用服务
对秘密信息的安全性要求	较高	较低
客户端的配置	较为复杂，需要操作系统支持	较为简单
可扩展性	注重将来应用程序的可扩展性	注重快速便捷地扩展远程用户
用户主要从事的行业领域	军事，财务，IT，科研等	销售，市场，服务等

表 1 两种 VPN 的应用需求的参考标准的对比

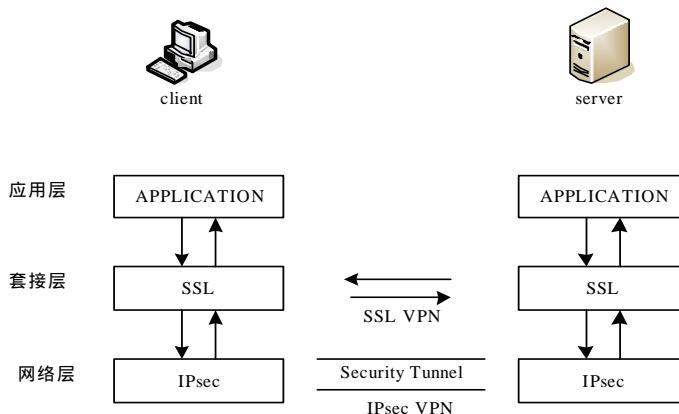


图 1 两种 VPN 的层次结构比较

协议形成BGP 对等路由器对之间的第二级标记交换路径，它只用于VPN 中标记分组的转发。

2.2 二者对比

近年来 IETF 的两个工作组一直在关注于解决 Internet 安全，标签交换标准和 QoS 这三个方面通过 VPN 实现松散联合的机制问题。这两个组织中的 IETF IPsec 工作组的工作主要涉及网络层的保护方面，所以改组设计了加密安全机制，以便于灵活地支持认证、完整性、访问控制和系统加密。另一个 IETF MPLS 工作组则在从另一个方面着手开发了支持高层资源预留、QoS 和主机行为

定义的机制。结果就出现了两种 VPN 架构，这两种架构各自依赖于 IPsec 和 MPLS 技术。下面比较一下 IPsec VPN 和 MPLS VPN。

服务供应商当然可以部署一种或者同时部署多种 VPN 架构来支持其新型增值服务，但是如果它们能够把各类 VPN 融合起来，更可以获得优势互补所带来的巨大利益。提供设计优良，运行正常和综合性的 VPN 服务可以同时提升 IPsec 和 MPLS 的应用层次。服务供应商可以对那些需要较高认证和私密性的数据流实行 IPsec；而对专有数据网络的带宽，流量工程和 QoS 等有要求的情况实行 MPLS。

	IPsec VPN	MPLS VPN
服务模式	高速 Internet 服务，商业质量的 IP 服务，电子商务和应用主机托管服务。	高速 Internet 服务，商业质量的 IP 服务，电子商务和应用主机托管服务。（这点二者基本相同）
可伸缩性	大规模部署需要制定相应计划并且协同解决关键分支机构，关键管理和对等配置各个方面出现的问题。	由于不需要站点对站点的对等性而具有高度的可伸缩性。典型的 MPLS VPN 部署能够支持在同一网络上部署上万个 VPN 组。
网络位置	本地环路，网络边缘或者远离存在加密数据较高曝光性的网络位置最佳，此类地点最适合采用隧道和加密等 IPsec 安全机制。	在服务供应商的核心网络最佳，此处 QoS，流量工程和带宽利用可以得到完全的控制，如果服务供应商的 VPN 服务提供 SLA 或者服务级保证（SLG），那么这一情况下的 VPN 服务更应该配置在网络核心。
透明度	IPsec VPN 位于网络层，对应用透明。	MPLS VPN 运行在 IP+ATM 或者 IP 环境下，对应用完全透明。
网络环境	一般情况下不需要针对可管理的 CPE 服务的网络级环境。在部署了基于网络的 IPsec VPN 服务之后，服务供应商通常还提供了集中的环境和管理支持。	由于 MPLS VPN 站点只同服务供应商网络对等，所以服务激活只需要一次性地在用户侧（CE）和服务供应商侧（PE）设备进行配置准备就可以让站点成为某个 MPLS VPN 组的成员。
服务部署	响应市场变化的速度快捷，可以在现有的任何 IP 网络上部署。	需要在启用了 MPLS 的核心网络共享网络设备，比如在网络升级期间或者必须部署新的 MPLS 网络时都得和核心网络的设备打交道。
会话认证	每个 IPsec 会话都必须通过数字签名或者预先分配的密钥进行认证；不符合安全策略的数据包都被丢弃。	VPN 成员资格由服务供应商决定，这是根据逻辑端口和唯一路由描述符所组成的环境功能实现的；对 VPN 组未经过认证的访问被设备配置所拒绝。
机密性	IPsec VPN 通过网络层上的一整套灵活的加密和隧道机制提供数据私密性。	MPLS 结构用一种类似可信帧中继或者 ATM 网络的方式来区分用户流量，从而实现 VPN 的安全性。
服务质量、服务级协约	IPsec 协议并没有解决网络的可靠性或者 QoS 机制等方面的问题。不过，有的厂商的 IPsec VPN 部署方案可以在 IPsec 隧道内保留数据包分类而实现 QoS。	优秀的 MPLS-VPN 实现方案可以提供可伸缩的，稳固的 QoS 机制和流量工程能力，从而令服务供应商可以提供具有保证 SLA 的 IP 增值服务。
客户支持	需要客户机初始化 IPsec VPN 部署	不适用，MPLS VPN 是基于网络的 VPN 服务。
用户交互	由于客户机需要初始化 IPsec VPN 服务，用户需要同 IPsec 客户端软件交互。	不适用，无需用户交互。

表2 IPsec VPN 和 MPLS VPN 的多方面特点对比

IPsec VPN 应用

◆ ◇ □ 潘冰

IPsec VPN 应用

在 Internet 上的两个安全网关之间建立一个 IPsec VPN 隧道是 IPsec 最简单的应用，在这种应用中可以通过 AH 或者 ESP 来保护报文的机密性和完整性，通过传输模式增加新的 IP 头使私网报文通过 Internet 来传送。那么，这就是 IPsec VPN 的全部吗？

当然不是，只掌握这种用法只是说明你只知道什么是 IPsec，但是你还不知道怎么使用 IPsec。本文的目的就是让你熟练掌握 IPsec 的应用，让最合理、最可靠、最易管理的 IPsec VPN 出现在你的网络中，发挥出 IPsec 最大的价值。

“工欲善其事，必先利其器”，我们接下来要做的就是看看 IPsec 到底能够为我们做什么。

IPsec 可以通过定义不同的感兴趣流来实现各种不同的保护策略，通过 VPN 拓扑的变化和路由的变化来控制报文的流向，这些特性的组合极大的扩展了 IPsec 应用的灵活性和复杂性。

另外，IPsec 本身也有一些局限性，像缺乏对于组播的支持、对于非 IP 报文的支持等等，但是聪明的我们又怎么会被这些问题难住呢？使用 IPsec 和其它 VPN 的组合既发挥了 IPsec 安全的特点，又可以弥补 IPsec 的局限，是目前 IPsec 应用最广泛的一种方式。

下面我们首先来研究一下 IPsec VPN 的应用模型。

IPsec 应用模型

就像刚才所说的那样，IPsec 的应用首先需要面对的一个选择是只使用 IPsec 还是将 IPsec 和其它 VPN 混用？选择合适的应用模型可以使你简化配置、事半功倍。要做到这一点，首先应该对这主要的 IPsec 应用模型各自的特点和优缺点了然于胸。

1.1 IPsec 模型

IPsec 模型是 IPsec VPN 应用的最简单的一种模型，这种模型只采用 IPsec 来完成报文的认证、加密以及 VPN 的构建，虽然说 IPsec 模型简单，但这仅仅是从概念上来讲的，实际上，深入应用 IPsec 模型也完全可以满足大型、复杂网络拓扑的需求。

IPsec 模型主要关心的是路由的设置以及要保护的流。首先，安全网关的路由配置应该保证要保护的数据流必须通过 IPsec 接口离开本地网络，否则，就会有不受保护的报文流出从而导致安全隐患；其次，要保护什么样的流也需要充分考虑，一般情况下，需要为每个要保护的流明确定义安全策略，这要求网络管理人员必须对进出接口的各个报文源、目的地址有足够的了解，同时也要求合理分配网络地址，便于整合安全策略相同的流，简化配置，在一个地址分配不合理的网络中，不仅增加了 IPsec 策略配置的复杂程度，而且可能会产生大量的 IPsec SA，消耗了设备的性能资源，尤其是 Hub 设备。

IPsec 模型最大的缺点就是对组播、广播和非 IP 报文的支持，这就导致了其不支持动态路由协议，只能采用静态路由，在大型的 PN 网络中，这极大的限制了可扩展性，并且同时增加了网络的配置及管理的难度。

由于 GRE 可以支持组播、广播和非 IP 报文，因此首先采用 GRE 对这些报文封装以后，IPsec 就可以认为这些报文就是普通的单播报文，从而可以进行下一步的封装，因此这样就解决了 IPsec 最大的硬伤：组播、广播及非 IP 报文的支持。支持了组播和广播，在 IPsec VPN 上运行动态路由协议就是水到渠成了。有了动态路由，IPsec VPN 就可以完成各个站点和 VPN 与其它网络的路由任务。

采用这种模型的另一个显著优势是配置上的简化和管理的便利。由于所有的报文都经过 GRE 封装以后再进行 IPsec 处理，我们就不需要了解原始的各个报文的原始地址，只需要将要保护的流定义为 GRE 的起点和终点地址即可，这样也同时减少了 IPsec SA 的数量，一般情况下，一个 Hub 和一个 Spoke 只有一组 IPsec SA。

值得注意的是，IPsec+GRE 还有另外一种选择，即先使用 IPsec 对报文进行封装，然后再采用 GRE 封装。但是，显而易见，这种封装方式没有充分利用 IPsec 和 GRE 的优势，同样有着配置复杂、不支持组播、广播和非 IP 报文的缺陷，一般不推荐使用。

由于 IPsec+GRE 的显著优势，这种组合已经成为目前 IPsec VPN 应用最广泛的一种模型。在后面的拓扑结构介绍中，也会重点来介绍这种模型的实际应用例子。

1.3 IPsec+L2TP 模型

IPsec+L2TP 模型和 IPsec+GRE 模型类似，借助 L2TP 对于远程移动用户的强大支持，扩展了 IPsec 的应用。这种模型同样首先采用 L2TP 封装，然后再对其使用 IPsec 封装，大大提高了 IPsec 对于远程移动用户拨号的支持。

IPsec+L2TP 的应用模型比较简单，一般只用于远程拨入用户和企业核心网络的互连，很少考虑到拓扑的变化和路由的设计问题，在 IPsec+L2TP 中 IPsec 需要保护的流只需要以 L2TP 的起点为源、终点为目的即可，因此本文不重点介绍 IPsec+L2TP。

IPsec VPN 拓扑结构

在实际的企业应用中，一般很少有只在两点之间建立 IPsec VPN 的情况，大多数情况都需要核心站点和多个远程节点之间建立 IPsec VPN，根据网 Hub-Spoke VPN 拓扑结构或者 Full-Mesh VPN 拓扑结构。

2.1 Hub-Spoke 拓扑

Hub-Spoke VPN 模型如图所示。

这种拓扑结构是所有的 Spoke 站点都有一条 VPN 隧道连接 Hub 站点，Spoke 之间没有 VPN 隧道，如果 Spoke 之间需要通过 VPN 通信，则必须借助 Hub 转发来完成。

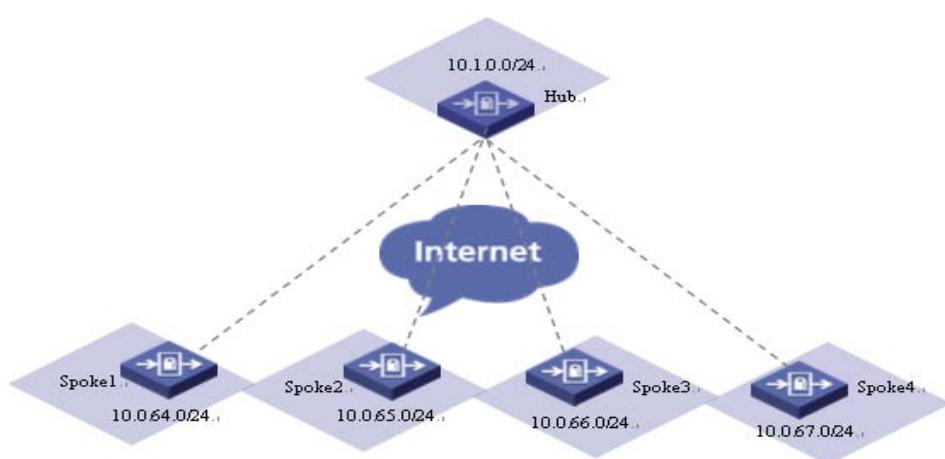


图 1 Hub-Spoke VPN

这种拓扑结构的特点是 V P N 链路较少，拓扑清晰，配置相对简单，尤其适合对于 S p o k e 和 S p o k e 之间没有流量或流量很少的情况。

但是这种模型也有其缺点：过分依赖 H u b 设备，可以说 H u b 设备是整个网络的心脏，H u b D o w n 掉，整个网络就会瘫痪，因此使用 H u b - S p o k e V P N 模型的当务之急就是采用冗余结构保护网络的可靠性。另外，这种模型对 H u b 的性能要求也比较高，尤其是 S p o k e 较多，并且 S p o k e 流量需要 H u b 转发的时候，此时，合理的分配地址以及配置多 H u b 的负载均衡也应该是我们重点需要考虑的。合理的分配地址可以减少 H u b 的 I Psec S A，节约 H u b 的性能资源；多 H u b 负载均衡可以将性能压力分摊到其它设备上。

对于 I P s e c 的冗余以及负载均衡的讨论我们放在后面章节进行，在接下来的讨论中暂时不考虑这两点，请读者注意。

在 H u b - S p o k e 拓扑之上，我们可以选择前面介绍的几种 I P s e c 应用模型，下面就分别介绍一下：

2. 1. 1 I Psec 模型

采用 I P s e c 模型一定要特别关注路由以及要保护流的设计。我们以一个实际网络为例：假设我们有下面的场景：

企业总部为 H u b ，各个分支站点为 S p o k e ，设计 I Psec V P N 满足如下要求

- H u b 和各个 S p o k e 之间有一条 I Psec V P N 隧道，各个 S p o k e 之间没有 I Psec V P N 隧道。
- H u b 内部的私网和各个 S p o k e 内部的私网通过 I Psec V P N 通信，各个 S p o k e 私网之间不需要通信。
- H u b 和各个 S p o k e 都能使内部网段访问 Internet 资源。

首先考虑路由的设计，显而易见，必须采用静态路由来保证各个站点之间的通信。路由的任务有三个：

- 完成各个站点安全网关的公网地址的互通
- 完成 H u b 和 S p o k e 之间私网地址的互通
- 完成 H u b 和 S p o k e 对于 Internet 地址的互通

完成各个站点安全网关的公网地址的互通是建立 V P N 的前提，完成这个目标最简单的方法是每个安全网关设备都有一条指向其默认 I S P 路由器的缺省路由，这样也同时使各个安全网关有了到达其它目的私网的路由和访问 Internet 的路由。当然，我们也可以在各个安全网关上定义目的安全网关的明细路由，但是，这样我们就必须在各个安全网关上再定义一个或者多个目的私网的静态路由，同时还需要为访问 Internet 额外配置路由。所以，在各个安全网关上定义缺省路由是一个比较合理的选择。

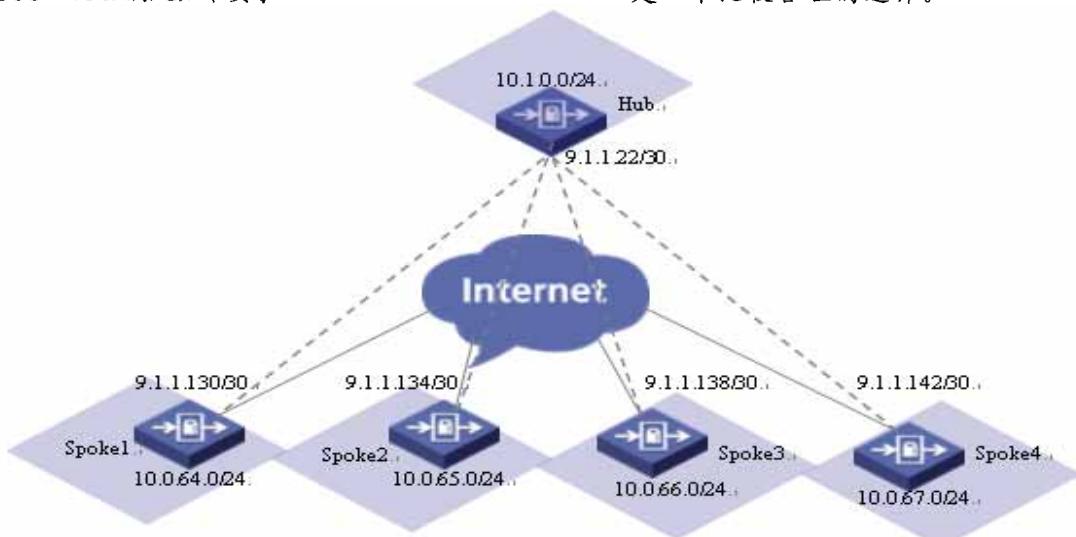


图 2 I Psec V P N H u b - S p o k e 拓扑

其次，需要考虑在 Hub 和 Spoke 上如何定义要保护的流。实际上，这个任务非常简单，各个安全网关要保护的流以自己私网地址为源，目的站点的私网地址为目的即可。以 Hub 和 Spoke1 站点的私网地址为目的即可。以 Hub 和 Spoke1 为例（Hub 和其它 Spoke 的配置类似）：在 Hub 上为 **10.1.0.0/24-10.0.64.0/24**；在 Spoke1 上为 **10.0.64.0/24-10.1.0.0/24**；在 Spoke1 上为 **10.0.64.0/24-10.1.0.0/24**。

综上所述，Hub 的 IPSec 相关配置如下（只列出了关键配置）：

ACL Number 3111 #for Spoke1

```
rule 0 permit ip source 10.1.0.0 0.0.0.255 destination
10.0.64.0 0.0.0.255
```

ACL Number 3112 #for Spoke2

```
rule 0 permit ip source 10.1.0.0 0.0.0.255 destination 10.0.
65.0 0.0.0.255
```

ACL Number 3113 #for Spoke3

```
rule 0 permit ip source 10.1.0.0 0.0.0.255 destination 10.0.
66.0 0.0.0.255
```

ACL Number 3114 #for Spoke4

```
rule 0 permit ip source 10.1.0.0 0.0.0.255 destination 10.0.
67.0 0.0.0.255
```

```
ip route-static 0.0.0.0 0.0.0.0 9.1.1.21 preference 60
```

Spoke 的 IPSec 配置基本相同，这里以 Spoke1 为例：

ACL Number 3111

```
rule 0 permit ip source 10.0.64.0 0.0.0.255 destination
10.1.0.0 0.0.0.255
```

```
ip route-static 0.0.0.0 0.0.0.0 9.1.1.129 preference 60
```

另外，如果我们希望简化配置，可以配置 Hub 的保护流如下：

```
rule 0 permit ip source 10.1.0.0 0.0.0.255 destination
any
```

但是，这样定义 Hub 的保护流只能采用 Spoke 来触发 IPSec VPN，因为这样的保护流无法唯一的标识某个 Spoke。

通过上面的配置可以很好的完成 Hub 和

Spoke 之间通信的需求，但是很多情况下，Spoke-Spoke 之间的通信也是必须的，在不改变现有 VPN 拓扑结构的情况下能否完成这个任务呢？答案是肯定的，我们只需要修改一下路由和保护流的配置即可。置即可。

由于 Spoke 之间没有 VPN，因此 Spoke 之间的通信只能够通过 Hub 的转发完成，我们来分析一下 Spoke1 和 Spoke2 之间的通信过程：

1. Spoke1 到 Spoke2 的报文首先路由到出接口。
2. 报文经过 Spoke1-Hub 的 VPN 到达 Hub，Hub 完成报文的解密。
3. Hub 对报文进行路由，到达出接口。
4. 报文经过 Hub-Spoke2 的 VPN 到达 Spoke2。
5. Spoke2 完成报文的解密。

通过上面的处理过程我们可以看到：问题的关键在于 Spoke1 是否有到达 Spoke2 的路由（步骤 1）；Spoke1-Spoke2 的报文是否匹配 Spoke1-Hub VPN 的要保护的流（步骤 2）；Spoke1 到 Spoke2 的报文是否符合 Hub-Spoke2 VPN 的要保护的流（步骤 4）。

在采用上面配置的前提下，由于各个安全网关配置了缺省路由，因此可以保证 Spoke1 到 Spoke2 的路由可达。而 **10.0.64.0/24-10.0.65.0/24** 的报文（Spoke1 到 Spoke2）是不匹配 Spoke1-Hub VPN 的要保护流的，因此需要修改 Spoke1 的 ACL。

我们注意到，ACL 要修改的只是目的地址部分，需要增加 **10.0.64.0/24-10.0.65.0/24**、**10.0.64.0/24-10.0.66.0/24** 等配置。如果 Spoke 设备较多，配置的繁琐和复杂也是不可想象的，如果采用 **10.0.64.0/24-any** 的配置呢？这样定义虽然可以匹配 Spoke1 访问其它 Spoke 和 Hub 的数据包，但是对于 Spoke1 内部网络访问 Internet 主机也进行了 IPsec 加密，造成 Internet 访问的失败。如何能够简化配置同时满足需求呢？这个时候合理的地址设计就显现出强大的作用，按照图

2 的地址分配原则，我们可以在 Spoke1 上采用如下配置 10.0.64.0/24-10.0.0.0/8，即可轻松解决问题，因为 10.0.0.0/8 这个目的地址可以包含 Hub 和所有其它 Spoke，并且，如果网络以后进行扩展，增加了大量的 Spoke(10.0.70.0/24、10.0.71.0/24 等)，可以不用修改 Spoke1 上的要保护的流。但是如果地址设计混乱，就只能在 Spoke1 上为 Hub 以及其它 Spoke 分别定义要保护的流了。

同样的道理，10.0.64.0/24-10.0.65.0/24 的报文(Spoke1 到 Spoke2)是不匹配 Hub-Spoke2 VPN 要保护流的，也需要修改 Hub 的感兴趣流。相应的，Hub 上针对所有 Spoke 的感兴趣流源地址应该是 10.0.0.0/8 而不是 10.1.0.0/24 了。

综上所述，Hub 的 IPsec 相关配置如下(只列出了关键配置)：

ACL Number 3111 #for Spoke1

```
rule 0 permit ip source 10.0.0.0 0.255.255.255 destination 10.0.  
64.0.0.0.255
```

ACL Number 3112 #for Spoke2

```
rule 0 permit ip source 10.0.0.0 0.255.255.255 destination 10.0.  
65.0.0.0.255
```

ACL Number 3113 #for Spoke3

```
rule 0 permit ip source 10.0.0.0 0.255.255.  
255 destination 10.0.66.0 0.0.0.255
```

ACL Number 3114 #for Spoke4

```
rule 0 permit ip source 10.0.0.0 0.255.255.255 destination 10.0.  
67.0.0.0.255
```

Spoke 的 IPsec 配置基本相同，以 Spoke1 为例：

ACL Number 3111

```
rule 0 permit ip source 10.0.64.0 0.0.0.255 destination 10.0.  
0.0.255.255.255
```

在前面所讲的两个例子中，安全网关都是在本地提供 Internet 访问的功能，也就是说各个 Spoke 和 Hub 都能够为本地私网主机提供 Internet 连接，通过在 IPsec 策略中不保护访问 Internet 主机为目的地址的报文来实现，即所有

加密报文和不加密报文都通过安全网关的外网接口出去，符合 IPsec 策略的报文通过 VPN 离开本地网络，不符合 IPsec 策略的报文直接通过 NAT 离开本地网络，这种实现方式称为“隧道分离”技术。

随着企业对于网络安全认识的不断加深，其对于网络的安全设计要求也越来越严格，典型的，对于安全网关的 Internet 访问功能也提出了更高的要求。一般情况下，为了统一的管理，大部分企业都要求访问 Internet 的需求集中在总部实现，也就是在 Hub 上实现，各个 Spoke 不能提供本地的 Internet 访问功能，这样可以将安全的风险降到最低，并且也便于管理。

我们仍然以图 2 为例，需求如下

- Hub 和各个 Spoke 之间有一条 IPsec VPN 隧道，各个 Spoke 之间没有 IPsec VPN 隧道。
- Hub 内部的私网和各个 Spoke 内部的私网通过 IPsec VPN 通信，各个 Spoke 私网之间不需要通信。
- 各个 Spoke 和 Hub 的私网主机只能通过 Hub 设备访问 Internet 资源。

在上面的例子中 Spoke1 定义要保护的流为 10.0.64.0/24-10.0.0.0/8，而对于其它目的地址的流没有进行 IPsec 保护，这样就造成了隧道的“分离”，不符合 IPsec 策略的报文于是通过分离的隧道离开了本地，解决的方法很简单，只需要将 Spoke1 要保护的流定义为 10.0.64.0/24-any 即可。在 Hub 上和各个 Spoke 之间原来的要保护的流源地址为 10.0.0.0/8，由于从 Internet 上返回的报文需要经过 Hub 到达各个 Spoke，而这些报文的源地址为 Internet 的公网地址，为了使这些地址能够通过 IPsec VPN 从 Hub 返回到 Spoke，需要将 Hub 和各个 Spoke 之间要保护的流源地址改为 any。

综上所述，Hub 的 IPsec 相关配置如下(只列出了关键配置)：

```
acl num 3113 #for spoke1
```

```
rule 0 permit ip source any destination 10.0.64.0 0.0.0.255
```

```
ACL Number 3112 #for Spoke2
```

```
rule 0 permit ip source any destination 10.0.65.0 0.0.0.255
```

```
ACL Number 3113 #for Spoke3
```

```
rule 0 permit ip source any destination 10.0.66.0 0.0.0.255
```

```
ACL Number 3114 #for Spoke4
```

```
rule 0 permit ip source any destination 10.0.67.0 0.0.0.255
```

Spoke 的 IPsec 相关配置如下：

```
ACL Number 3111
```

```
rule 0 permit ip source 10.0.64.0 0.0.0.255 destination any
```

2. 1. 2 IPsec+GRE 模型

IPsec+GRE 模型采用 IPsec 保护的 GRE 隧道，最大的优势就是将我们从 IPsec 复杂的配置中解脱出来，尤其是要保护的流，我们不必关心接口流入和流出的报文地址是什么，要保护的流是什么，只要保护 GRE 隧道的起点和终点即可。

另外，我们也不需要费尽心思的设计如何分配各个站点的网络地址，因为 GRE 已经隐藏了报文的原始地址，简化的配置带来的另一个惊喜就是安全网关性能的节约，更少的 IPsec SA 减少了设备性能资源的占用，提高了 VPN 网络的可靠性。

我们仍然以一个实际的应用场景来学习 IPsec+GRE 的应用：

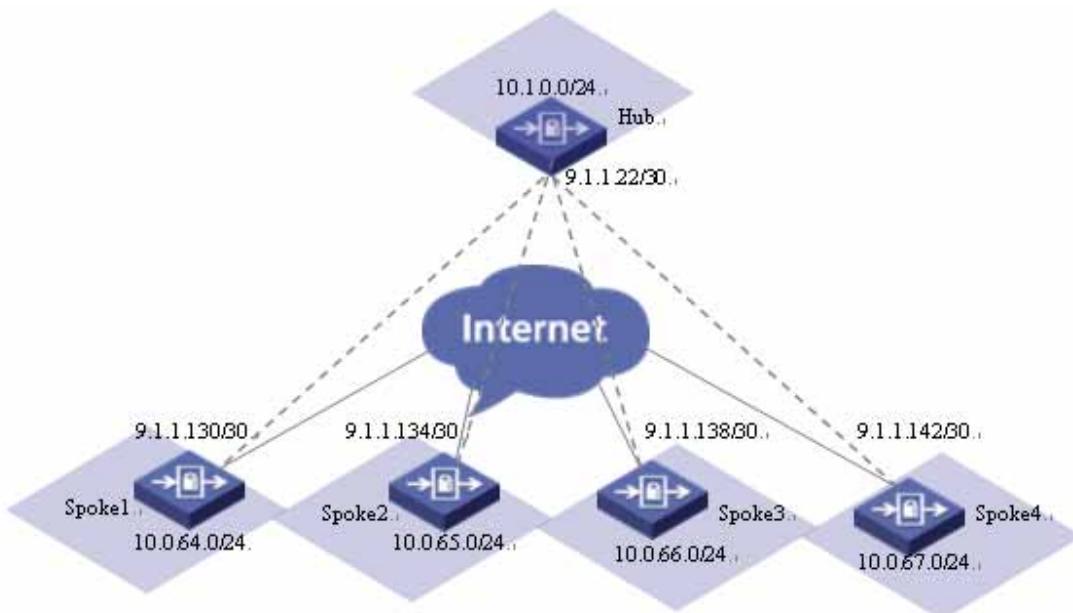


图 3 IPsec+GRE VPN

企业总部为 Hub，各个分支站点为 Spoke，设计 IPsec VPN 满足如下要求

- Hub 和各个 Spoke 之间有一条 GRE 隧道，各个 Spoke 之间没有 GRE 隧道。
- Hub 内部的私网和各个 Spoke 内部的私网通过 GRE VPN 通信，并且通信收到 IPsec 的保护。各个 Spoke 私网之间不需要通信。
- 各个 Spoke 内网主机不能访问 Internet，Hub 内网主机可以访问 Internet。

假定我们首先采用静态路由的方式完成该场景，Hub 的相关配置如下

```
interface Tunnel0
```

```
ip unnumbered Ethernet4/0
```

```
tunnel source 9.1.1.22
```

```
tunnel destination 9.1.1.130
```

```
interface Tunnel1
```

```
ip unnumbered Ethernet4/0
```

```
tunnel source 9.1.1.22
```

```
tunnel destination 9.1.1.134
```

```
interface Tunnel2
```

```
ip unnumbered Ethernet4/0
```

```
tunnel source 9.1.1.22
```

```

tunnel destination 9.1.1.138
interface Tunnel3
ip unnumbered Ethernet4/0
tunnel source 9.1.1.22
tunnel destination 9.1.1.142
interface Serial1/0:0
ip address 9.1.1.22 255.255.255.252
    ipsec policy gre
interface Ethernet4/0
ip address 10.1.0.1 255.255.255.0
# 定义访问各个 Spoke 的出口为 GRE 接口 #
ip route 10.0.64.0 255.255.255.0 Tunnel0
ip route 10.0.65.0 255.255.255.0 Tunnel1
ip route 10.0.66.0 255.255.255.0 Tunnel2
ip route 10.0.67.0 255.255.255.0 Tunnel3
ip route 0.0.0.0 0.0.0.0 9.1.1.21
acl num 3111 #for spoke1
rule 0 permit ip source 9.1.1.22 0.0.0.0 destination 9.1.1.138 0.0.0.0
acl num 3112 #for spoke2
rule 0 permit ip source 9.1.1.22 0.0.0.0 destination 9.1.1.146 0.0.0.0
acl num 3113 #for spoke3
rule 0 permit ip source 9.1.1.22 0.0.0.0 destination 9.1.1.130 0.0.0.0
acl num 3114 #for spoke4
rule 0 permit ip source 9.1.1.22 0.0.0.0 destination 9.1.1.142 0.0.0.0
(IKE Peer 和 IPsec Policy 配置略)
Spoke 的配置基本相同，这里以 Spoke 2 为例：
interface Ethernet0
ip address 10.0.65.1 255.255.255.0
interface Tunnel0
ip unnumbered Loopback1
tunnel source 9.1.1.134
tunnel destination 9.1.1.22
interface Serial0
ip address 9.1.1.134 255.255.255.252
    ipsec policy gre

```

```

ip route 9.1.1.20 255.255.252 9.1.1.133
ip route 10.1.0.0 255.255.255.0 Tunnel0
acl num 3111
rule 0 permit ip source 9.1.1.134 0.0.0.0 destination 9.1.1.22 0.0.0.0

```

上面的配置中，由于 Hub 上定义了缺省路由，因此 Hub 内部的私网可以通过隧道分离的方式访问 Internet (当然前提是必须配置 NAT)，而 Spoke 2 则只定义了到达 Hub 公网和私网地址的静态路由，因此 Spoke 2 内网主机不能访问 Internet，同时，Spoke 2 也没有到达其它 Spoke 私网的路由，因此各个 Spoke 之间也不能够互访。

如果采用动态路由的方式完成上面场景的目标，则动态路由的选择上必须要考虑全面。如果在各个 Spoke 和 Hub 之间启用动态路由协议，那么 Hub 会学习到所有 Spoke 的私网路由，同时，所有的 Spoke 不仅可以学习到 Hub 的私网路由，还会通过 Hub 学习到其它 Spoke 的路由，当 Spoke 1 和 Spoke 2 之间存在了指向 Tunnel1 接口的路由以后，Spoke 1 和 Spoke 2 之间就会通过 IPsec 保护的 GRE VPN 互通，请注意我们是以 Spoke1-Hub 的 GRE 起点和终点的地址为保护的流，而不管 GRE 内部的报文的初始地址，因此无法用要保护的流来控制报文的走向，只能通过路由来控制。

解决这个问题的方式是采用路由过滤的方式，Hub 向 Spoke 发送路由时只发送自己私网的路由，而不发送其它 Spoke 私网的路由。采用路由过滤就意味着这种情况下不能够使用联络状态算法的路由协议 (OSPF / IS-IS)，因为这些路由协议的路由过滤是无法过滤 LSA 的，只能采用距离矢量算法的路由协议，如 RIP 等。

在这个例子里我们选择 RIP v2 作为路由协议，并且在 Hub 设备上定义路由过滤策略。

值得注意的是为了保证 GRE 隧道的建立，各个设备首先必须配置静态路由来完成安全网关公网地址的可达性。

综上所述，Hub 的路由相关配置如下

```

interface Ethernet4/0

```

```

ip address 10.1.0.1 255.255.255.0
interface Tunnel0
ip unnumbered Ethernet4/0
tunnel source 9.1.1.22
tunnel destination 9.1.1.130
rip version 2
interface Tunnel1
ip unnumbered Ethernet4/0
tunnel source 9.1.1.22
tunnel destination 9.1.1.134
rip version 2
interface Tunnel2
ip unnumbered Ethernet4/0
tunnel source 9.1.1.22
tunnel destination 9.1.1.138
rip version 2
interface Tunnel3
ip unnumbered Ethernet4/0
tunnel source 9.1.1.22
tunnel destination 9.1.1.142
rip version 2
ip route 0.0.0.0 0.0.0.0 9.1.1.21
rip
network 10.0.0.0
filter-policy 2000 export
acl num 2000
rule permit source 10.1.0.0 0.0.255.255
Spoke 2 的相关配置如下
interface Ethernet0
ip address 10.0.65.1 255.255.255.0
interface Tunnel0
ip unnumbered Ethernet0
tunnel source 9.1.1.134
tunnel destination 9.1.1.22
rip version 2
ip route 9.1.1.20 255.255.255.252 9.1.1.133
rip
network 10.0.0.0

```

在这个场景中，对于 GRE Tunnel 的目的地址路由一定要小心考虑：如果 GRE Tunnel 的目的地为公网地址，定义一个到该地址的静态路由或者指向 ISP 的缺省路由即可；如果 GRE Tunnel 的目的地为对端的私网地址，则必须为该私网地址额外定义一个静态路由，一定要避免到达 GRE Tunnel 目的地的地址的路由下一跳为该 Tunnel 接口，这样就造成了路由的环路。

如果对于图 3 的场景，将需求改为允许各个 Spoke 之间通过 VPN 通信，那么就不存在动态路由过滤的问题了，每个 Spoke 都可以通过 Hub 学习到所有其它 Spoke 的路由。在这种情况下，也就可以选择 OSPF 或者 ISIS 做为动态路由协议了。

如果对于图 3 的场景，进一步修改其需求，允许 Spoke 和 Hub 内的私网访问 Internet，但是所有的 Spoke 只能通过 Hub 来访问，此时仍然需要考虑的是路由，因为无法通过保护的流来控制报文的走向。为了完成这个任务，需要使 Spoke 有缺省路由指向 Tunnel 接口。如果在各个 Spoke 上采用静态路由来完成这个需求，就丧失了 Internet 访问的统一管理性，即由 Hub 来控制各个 Spoke 能否通过自己访问 Internet。最佳的方式是由 Hub 向各个 Spoke 发布一条指向自己的缺省路由。

以 OSPF 为例，Hub 的配置如下

```

interface Ethernet4/0
ip address 10.1.0.1 255.255.255.0
interface Tunnel0
ip unnumbered Ethernet4/0
tunnel source 9.1.1.22
tunnel destination 9.1.1.130
interface Tunnel1
ip unnumbered Ethernet4/0
tunnel source 9.1.1.22
tunnel destination 9.1.1.134
interface Tunnel2
ip unnumbered Ethernet4/0
tunnel source 9.1.1.22

```

tunnel destination 9.1.1.138

interface Tunnel3

ip unnumbered Ethernet4/0

tunnel source 9.1.1.22

tunnel destination 9.1.1.142

ip route 0.0.0.0 0.0.0.0 9.1.1.21

ospf 1

area 0.0.0

network 10.1.0.0 0.0.255.255

default-route-advertise

Spoke2 的配置如下

interface Ethernet0

ip address 10.0.65.1 255.255.255.0

interface Tunnel0

ip unnumbered Ethernet0

tunnel source 9.1.1.134

tunnel destination 9.1.1.22

ip route 9.1.1.20 255.255.255.252 9.1.1.133

ospf 1

network 10.0.65.0

2.2 Full-Mesh 拓扑

Full-Mesh 拓扑结构在每两个站点之间都建立一条 IPsec VPN 隧道，形成全网状的 VPN 结构，在这种结构下，每个站点都像 Hub 一样提供 IPsec 服务，一般情况下，如果企业网络要求各个站点之间都有较多流量的情况下比较适合使用 Full-Mesh 结构。

Full-Mesh 结构由于 VPN 的数量较多，尤其是站点较多的情况下，拓扑结构看起来相当的复杂，并且每增加一个新的站点，所有站点都需要增加配置，大大增加了配置管理的难度。不过，这种拓扑结构也有其优势，全网状的 VPN 连接提供了最大的可靠性，任何一个站点故障都不会影响其它站点之间的通信，而且这种拓扑将流量分担到各个站点中去，不会将流量集中到一、二个设备上。

2.2.1 IPsec 模型

首先讨论 IPsec 模型在这种拓扑的应用。假定目前的场景如下

- 建立 Full-Mesh 的 VPN 结构，各个站点之间的私网流量通过 IPsec 保护
- 每个站点都可以提供 Internet 访问功能

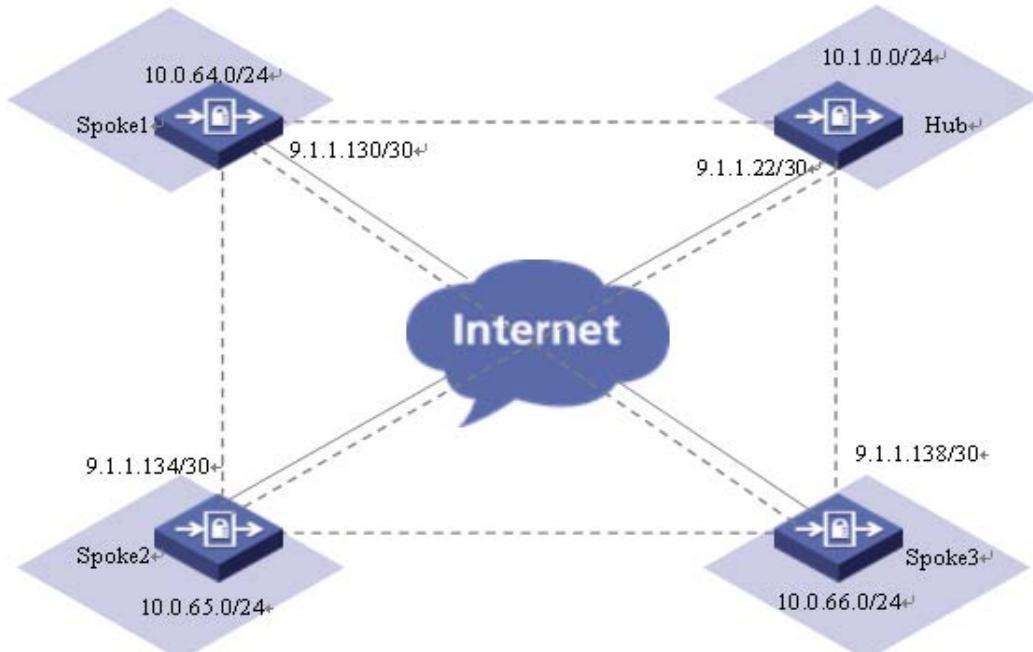


图 4 Full-Mesh 结构

在配置上只需要将各个 Spoke 当作 Hub 来考虑即可，在每个设备上分别定义所有其它设备作为自己的 IKE Peer，同时为每条 VPN 定义要保护的流源为自己私网的地址，目的为目的站点私网地址即可。

在每个站点都配置一条指向 ISP 的缺省路由即可。这样，访问其它私网的流量符合 IPsec 保护的流，根据策略的配置找到 IKE Peer，从而到达目的地；而访问 Internet 的流不符合 IPsec 保护的流，直接离开本地。

这种应用方法的配置思路比较简单，但是非常繁琐，很容易出错。而且其对于地址的设计仍然有较高的要求，如果地址设计不当，不仅仅增加配置的复杂程度，也影响了设备的性能，例如将刚才的例子 10.1.0.0/24 网段对应站点增加一个 10.1.2.0/24 网段，如图所示

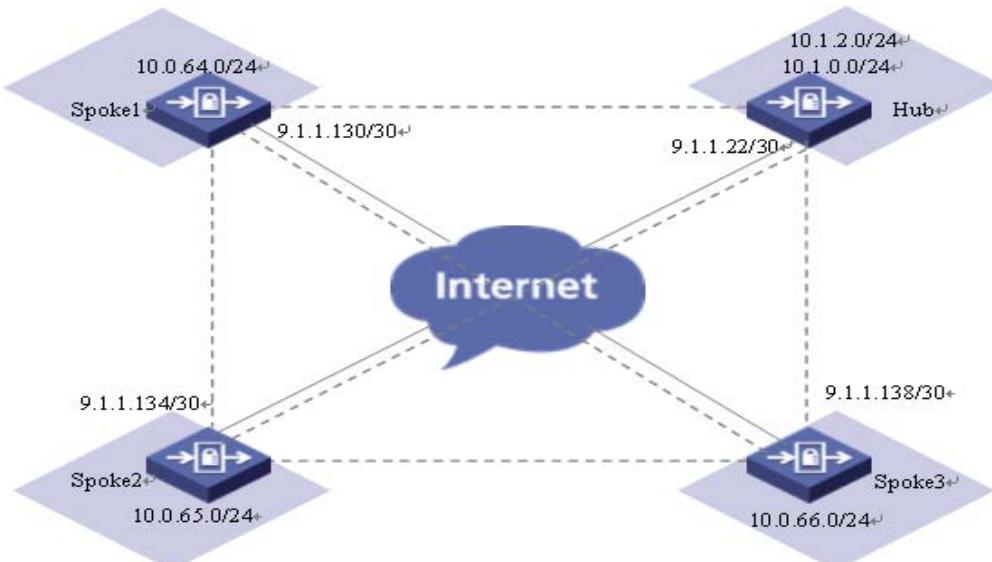


图 5 Full-Mesh 拓扑

这样就增加了所有其它站点的配置，并且也影响了设备的性能，如果我们在其它站点上定义保护的流的目的地址为 10.1.0.0/16，就可以解决这个问题。但如果地址设计不当，可能就不能通过聚合的方式来定义多条流了。

同样，如果在图 4 的场景中我们同时要求 Internet 的访问集中在 10.1.0.0/24 网段对应的站点。那么各个 Spoke 只需要将对应该站点的保

护流目的地址定义为 any，同时在核心站点将对应各个 Spoke 的保护流源地址定义为 any 即可，道理和前面 Hub-Spoke 拓扑一样，这里就不累述了。

通常，Full-Mesh 结构用于 Spoke 之间流量较大或者对于网络速度和延迟比较敏感的应用，例如 VoIP、视频等。Full-Mesh 结构避免了 Spoke 之间通信还需要 Hub 进行转发，在 Hub 上需要二次加解密的情况。

2.2.2 IPsec+GRE 模型

IPsec+GRE 模型在 Full-Mesh 拓扑下仍然保留了其优势：配置简单、支持多播等应用等等。在这种拓扑下，和 IPsec 模型相比，IPsec+GRE 模型将配置的复杂性转移到了路由设计上。很显然，在 Full-Mesh 拓扑下不能够采用静态路由，否则大量的静态路由就会造成网络管理人员的恶梦。。

动态路由只能是唯一的选择。

只需要在 Tunnel 接口启用动态路由协议，动态路由协议会自动学习到达所有其它站点的路由，一个站点到达另外一个站点的最优路径应该是其直连的 VPN 链路，如果该 VPN 链路 Down 掉，动态路由协议也会自动的选取其它链路来达到冗余的目的，当然，如果某个站点的安全网关出现故障，路由协议的冗余也不能够解决通信的问题了，需

借助其它方法。

在 IPsec+GRE 模型中的缺点是动态路由协议会占用安全网关的一部分处理器和内存资源，对于低端的安全设备，可能会造成一些性能的影响。

以 Spoke1 的配置为例，下面列出了相关的配置

```

interface Ethernet0
  ip address 10.0.64.1 255.255.255.0
interface Tunnel0
  ip unnumbered Ethernet0
  tunnel source 9.1.1.130
  tunnel destination 9.1.1.22
interface Tunnel1
  ip unnumbered Ethernet0
  tunnel source 9.1.1.130
  tunnel destination 9.1.1.134
interface Tunnel2
  ip unnumbered Ethernet0
  tunnel source 9.1.1.130
  tunnel destination 9.1.1.138
interface Serial0
  ip address 9.1.1.130 255.255.255.252
  ipsec policy gre
  ospf 1
    area 0.0.0
    network 10.0.64.0 0.0.0.255
  ip route 9.1.1.22 255.255.255.252 9.1.1.129(9.1.1.129 为
    接入当地 ISP 的地址)
  ip route 9.1.1.132 255.255.255.252 9.1.1.129
  ip route 9.1.1.136 255.255.255.252 9.1.1.129
  acl number 3111 #for hub
  rule 0 permit ip source 9.1.1.130 0.0.0.0 destination 9.1.
    1.10 0.0.0.0
  acl number 3112 #for spoke2
  rule 0 permit ip source 9.1.1.130 0.0.0.0 destination 9.1.
    1.134 0.0.0.0
  acl number 3113 #for spoke3

```

rule 0 permit ip source 9.1.1.130 0.0.0.0 destination 9.1..138 0.0.0.0

如果希望各个分支站点能够将 Internet 访问集中到 Hub 上实现，可以在 Hub 上宣告一条 OSPF 的缺省路由即可。

IPsec 可靠性设计

3.1 Hub 备份

在前面的内容中重点介绍了 IPsec 的拓扑设计和路由设计，我们发现，不管是 Hub-Spoke 结构还是 Full-Mesh 结构，企业总部站点的安全网关冗余都是必须要重点考虑的问题。虽然 Full-Mesh 结构可以提高网络的可靠性和冗余能力，但是如果各个站点的 Internet 访问都集中到企业总部站点，那么对于企业总部安全网关的可靠性同样有着很高的要求。

一般从成本和需求的考虑下，IPSec VPN 只考虑核心站点安全网关的冗余，而不考虑其它分支站点。常见的 IPsec 冗余有如下两种情况：

3.1.1 通过 VRRP 冗余

为了提高核心站点的可靠性，采用两台安全网关设备配置 VRRP 提供双机热备是一种很常用的方法。分别在两台安全网关的内网接口和外网接口定义两个 VRRP 组，使用虚地址和内网及外网通信。

为了减少 VPN 的配置以及简化拓扑，不需要每个分支站点都分别和核心站点的两个安全网关建立 IPsec VPN，而是使用核心站点安全网关 VRRP 组的虚地址和其它分支站点分别建立一个 VPN 链路，这样冗余的设备没有改变原有拓扑结构，要保护的流和路由的设备也不需要进行修改。只需要在各个分支站点配置 IKE 邻居时定义 VRRP 虚地址即可。

VRRP 建立 IPsec 的拓扑如下所示

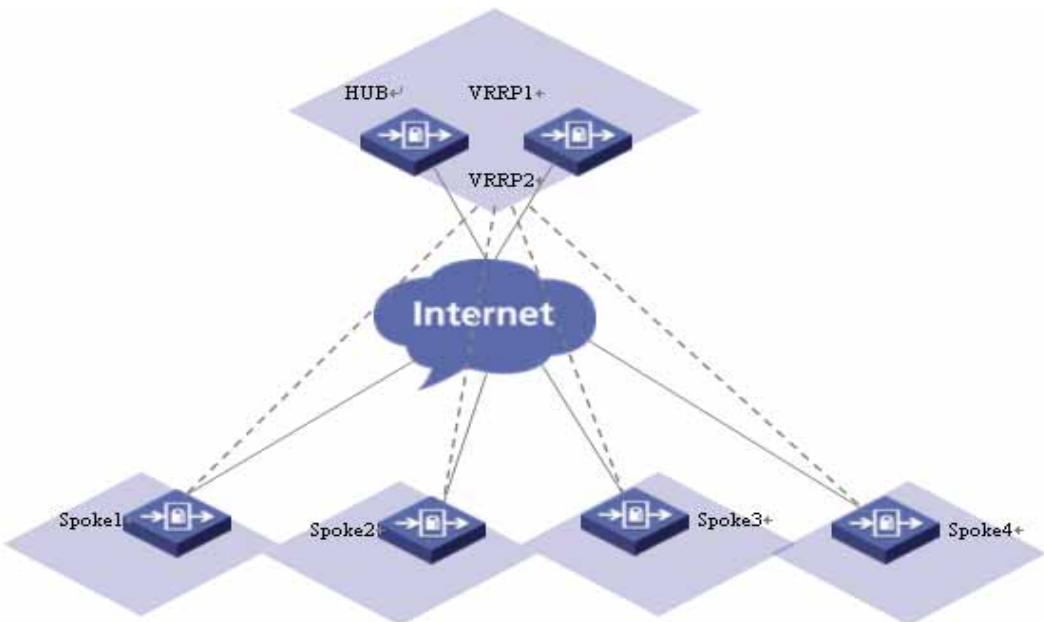


图 6 VRRP+IPsec

使用 VRRP 虚地址建立 IPsec VPN 带来最大的问题是 VRRP 状态切换时 IPsec 状态能否正常切换。当远端站点和本地的 VRRP 组 Master 建立 IKE 连接以后，当 VRRP 出现主备切换时，远端站点无法检测到 IKE Peer 已经切换，因此仍然使用原有的 IKE/IPsec SA 和 VRRP 组建立 IPsec，导致 IPsec 失败。

DPD 的出现解决了这个问题，IPsec DPD (IPSec Dead Peer Detection on-demand) 为按需型 IPsec / IKE 安全隧道对端状态探测功能。启动 DPD 功能后，当接收端长时间收不到对端的报文时，能够触发 DPD 查询，主动向对端发送请求报文，对 IKE Peer 是否存在进行检测。

如果在上面的方案中同时采用 DPD 特性，当 VRRP 出现主备切换时，远端站点会在进行 IPsec 通信之前发送 DPD 探测对方的 IKE 通道是否存在，经过 3 次失败以后远端站点会重新和 VRRP 组建立新的 IKE 通道。

值得注意的是，在使用 IPsec+DPD 方案时，如果 IPsec 连接由核心站点安全网关的 VRRP Master 设备发起，由于该 IKE 报文的源地址为 Master 出

接口地址而不是 VRRP 的虚地址，因此无法和远端设备建立 IKE 连接，从而无法完成 IPsec VPN，只能由远端设备来发起 IPsec 连接。解决这个问题的最好方法是在 VRRP 的所有设备上定义 IKE 的 local-address 地址为 VRRP 虚接口地址，这样 VRRP 设备发起 IPsec 连接时的源地址就是虚地址，可以完成 IKE 和 IPsec 的协商。

通过 VRRP+IPsec 的方式来实现 IPsec VPN 的冗余，优点是配置简单，不改变原有拓扑结构、路由和感兴趣流的配置。但是由于 VRRP 的切换存在延迟，而且 VRRP 切换以后 IPsec 还要等待 DPD 的超时才能建立新的 VPN，因此整个切换的时间在 1 分钟左右，对于延迟较敏感的应用是无法接受的。

3.1.2 通过路由协议冗余

为了满足一些对于状态切换时延较敏感的应用，可以使用动态路由协议 + IPsec + GRE 的方案。这种方案是在核心站点使用两台安全网关设备，每个安全网关设备都和其它的分支站点建立一条 IPsec VPN 隧道，如下图所示

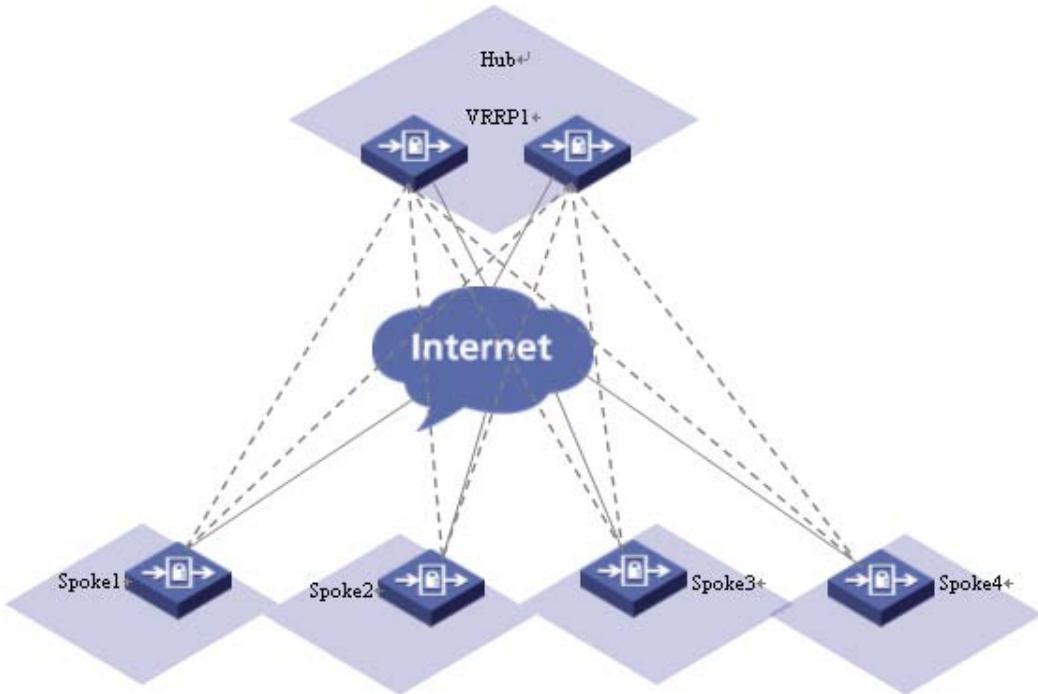


图 7 GRE+IPsec+ 动态路由

对于核心站点的内网接口，可以选择使用 V R R P ，也可以通过一些路由策略来控制用户的出口。配置了动态路由协议以后，各个分支站点都有两条到达核心站点的路由，分别指向两个 Tunnel 接口，如果希望核心站点的两个设备能够负载均衡，可以不修改路由协议的 Cost 值，如果希望另外一个设备只做备份，则可以调高备份链路的 Cost 值。

这种方式的冗余优点是切换时间非常短，但是拓扑结构较复杂， V P N 链路较多，相应的 IPsec S A 较多，对各个分支站点的性能要求较高。

介绍了这么多 I P s e c 的典型应用，相信读者已经对 I P s e c 有了深层次的理解，但是仅仅掌握了 I P s e c 各种模型和拓扑结构是不够的，实际的网络千变万化，充分利用 I P s e c 这个工具扬其所长、避其所短，设计出最适合需求的网络仍然需要我们不断的学习和积累才能够完成。

IPsec 典型组网

◆ ◇ □ 刘胜伟

概述

传统专线网基于现有的物理网络，例如数字电路、ATM/FR、DDN等，这种方式安全性和可靠性非常高，但对于客户来说，相应的建设成本和使用费用昂贵，并且只能实现有限的专网访问，缺乏联网的灵活性。因此伴随互联网的发展，通过互联网搭建企业VPN的模式越来越引起客户的兴趣。

通过互联网组建VPN有两种方式，一种是企业自建，一种是租用运营商的VPN业务。如果企业采用自建方式，通过在其公司总部架设VPN服务器（防火墙、NAT设备等），以允许可信赖的伙伴或分支机构访问公司内网。另外放置在每个节点的安全设备还用于对每一个在广域网上传输的数据包进行物理加密和解密。这种方式对于客户来说，专业技术要求较高，并且建设和维护成本大，客户对象一般是中、高端客户。还有一种就是采取运营商转售的方式，由运营商提供给企业这种VPN的接入平台，并进行代维。

IPsec VPN基于一组开放的网络安全协议，业务数据流通过IPsec加密，IPsec能够提供服务器及客户端的双向身份认证，并且为IP及其上层业务数据提供安全加密保护，能够支持数据加密（包括常见的DES, 3DES, AES加密算法），数据完整性验证，数据身份验证，以及防重放等功能，充分保证业务数据的安全性。IPsec VPN利用Internet构建三层隧道VPN的方式，可以允许用户以任意方式接入VPN，并且不受地理因素的限制，无论用户在外地或海外，只需要从当地接入Internet即可。IPsec VPN不仅适用于SOHO用户或移动办公用户接入，而且适用于企业分支机构之间的互连互通。正因为IPsec VPN具有其它VPN不可替代的业务优势，在VPN业务较为普及的海外一些国家得到了比较普遍的应用。

典型应用

IPsec试图解决的VPN两个主要设计问题：

- 为把两个专用网络组合成一个虚拟网络的无缝连接。
- 将虚拟网络扩展成允许远程访问用户成为可信网络的一部分。

基于两个设计的基础上，IPsec VPN可以被分为两类：

- LAN - to - LAN IPsec 实现 (也被称为 site - to - site VPN)。
- 远程访问客户端 IPsec 实现。

2.1 LAN-to-LAN IPsec

LAN - to - LAN IPsec 描述的是在两个局域网之间建立的 IPsec 隧道的概念，也被称作 site - to - site IPsec VPN。建立 LAN-to-LAN VPN 时，两个专用网络之间跨越一个公用网络，这样在任意一个专用网络中的用户都可以访问另一个专用网络中资源，就像他们在各自的专用网络上一样。

IPsec 提供了在两个站点间协商和建立加密隧道的途径。一般地，隧道技术是重要地，因为两个专用网络通常使用 RFC1918 地址机制，这需要使用隧道跨越公共网络。IPsec 允许定义流量使用什么加密以及怎样加密。

2.2 远程访问客户端 IPsec

当一个远程用户连接到一个 IPsec 路由器或使用安装在其上的 IPsec 客户端访问服务器时，就会创建远程访问客户端 IPsec VPN。一般情况下，这些远程访问机器使用拨号或是类似的连接方式连接到公共网络或者 Internet。一旦到 Internet 的连接建立起来后，IPsec 客户端就可以建立一条跨越公共网络或者 Internet 而连接到一个位于专用网络边缘的 IPsec 终端设备的封装隧道。远程访问客户端正在试图与这个专用网络建立连接并成为其中的一部分。这些 IPsec 终端设备也被称为一个 IPsec 远程访问集线器。

IPsec 的远程访问实现也有一些自己独有的挑战。在 site-to-site 情形中，IPsec 对等体的数量，即 IPsec 隧道的终端设备，是有限制的。不过，在远程访问 IPsec VPN 的例子中，对等体的数目是很多的，在工程量巨大的实现中甚至达到了成百上千个。这些情形需要特殊的可扩展性好的认证密钥管理机制，因为所有的用户保存所

有密钥是不可能的任务。

L2TP 也是远程用户接入专用网络的一种常用方式。但默认情况下，L2TP 不提供加密服务。尽管 L2TP 具有让 LAC 和 LNS 互相认证的机制，但是仍然有为 L2TP 通信加密提供进一步安全性的需要。PPP 也可以提供一种加密机制。不过，它只是初级的并且不提供数据完整性、重放攻击保护和密钥管理，其中密钥管理是一个适当的加密、数据完整性和认证系列所必需的。除此之外，它提供的认证是针对每一个会话的，而不是针对每一个分组的。因此，L2TP 常常依赖 IPsec 为它提供安全服务。

在《IPsec 应用》这篇文档中已介绍了 Hub-Spoke、Full-Mesh 拓扑结构下 IPsec、GRE 模型的设计。本文将重点针对两个典型组网分别介绍 LAN-to-LAN、远程访问客户端 IPsec 两类典型应用的实现。

LAN-to-LAN IPsec 实现

3.1 典型组网

图中两台 SECPATH1000F 产品作为企业网总部的接入设备，实现了冗余备份，SECPATH100、SECPATH10 作为分支机构的接入设备。

企业总部中的 BIMS 服务器、VPN Manager 服务器、Radius 服务器，分别提供对 VPN 配置的下发，VPN 的管理和用户的认证等功能。

CA 服务器放在公网上，为 IPsec 的实现提供证书认证支持。

3.2 IPsec 模型

3.2.1 实现介绍

最简单的连接性就是使用 IPsec 模型在两个场点之间创建 IPsec VPN 来实现的。场点可以使用传输模式通过诸如运行 IP 的帧中继或 ATM 网络等专用网络连接起来，也可以使用隧道模式通过 Internet 连接起来。

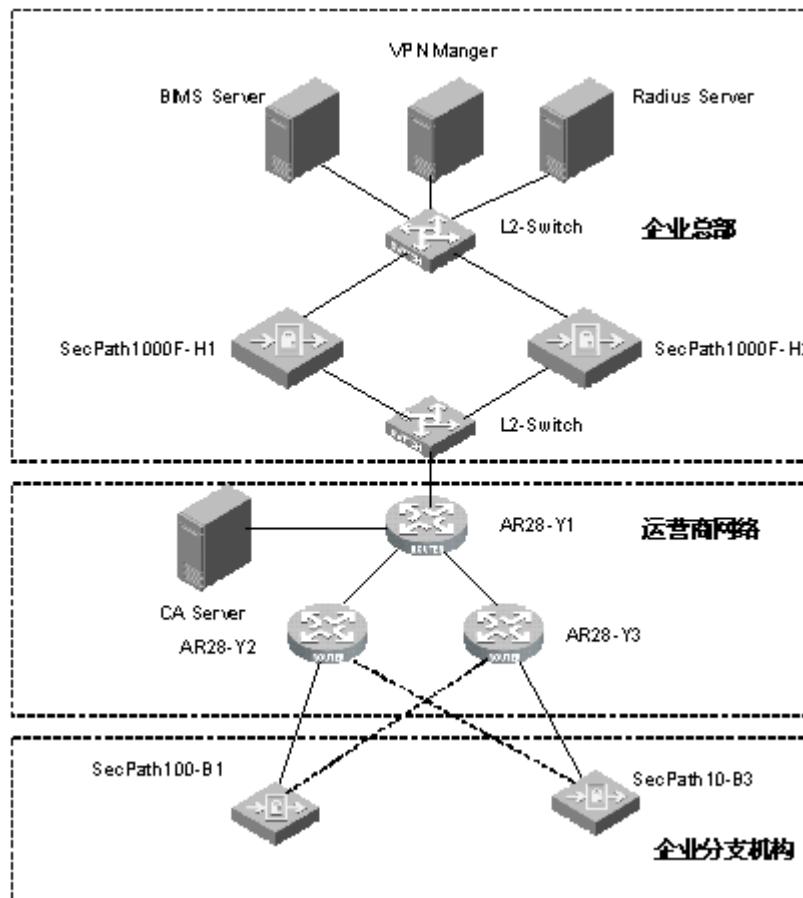


图 1 LAN-to-LAN IPsec 实现典型组网图

3.2.2 组网设计说明

如典型组网图中的 SecPath100-B1 和 SecPath1000F-H1 之间建立 IPsec 隧道。该组网在分支和总部之间配置了一条 IPsec 隧道，由 IPsec 保证了私网通信的私密性。

1、路由方面我们需要考虑中心场点 (SecPath1000F-H1) 有到每个远程分支场点 (SecPath100F-B1 和 SecPath10-B3) 的显示路由，同样，各个分支场点也必须有一条前往中心场点的路由。另外，企业中心和企业分支进行私网通信路由一般采用是在网关上配置静态路由，需要将这些静态路由引入到路由协议中，使私网中的设备可以通过网关访问对端私网。

2、IKE 的认证方法除了可以使用预共享密钥方法外还可以通过公网上的 CA 服务器进行认证。

其它关于采用何种认证、加密算法等可以根据具体的要求在设备上进行相应的配置。若公网上配置了 NAT，企业分支设备对企业中心的访问需要穿越 NAT，此时 IKE 应使用野蛮模式。

3、由于 IPsec 要保护的是总部、分支私网数据流量，所以 IPsec 封装模式应配置为隧道模式。至于 IPsec 采用的加密、认证协议应根据隧道流量是否要求加密，认证强度等决定使用 AH、ESP 还是 AH-ESP。

4、为了实现总部中心网关的冗余备份，可以在企业总部再部署一台网关设备 (SecPath1000F-H2)。在 SecPath1000F-H1 和 SecPath1000F-H2 上启用 VRRP 协议，使用 VRRP 地址作为私网网关设备指定的总部 IKE Peer 地址。当总部主设备 down 掉后，链路会自动切换到从设备当主设备 up

后，链路又会自动切回主链路。

5、为了在VRRP切换后，分支设备的IKE Peer能够迅速切换到master上，可以在安全网关上启用DPD协议。

6、既然企业总部中有两台网关设备，就可以考虑实现负载均衡。在每一台中心网关上都配置两个VRRP组，每个VRRP的master对应不同的网关设备。一部分企业分支设备（如SecPath100-B1）的IKE Peer地址指向第一个VRRP组的虚地址，另一部分的分支设备（如SecPath10-B3）指向另一个VRRP虚地址。这样假设SecPath1000F-H1 down掉后，所有的分支都将SecPath1000F-H2建立连接；当SecPath1000F-H1恢复后，部分网关会重新与之建立连接。

3.3 GRE+IPsec模型

3.3.1 实现介绍

设计使用IPsec提供对等体间连接性的VPN有其固有的局限性，这包括：

- IPsec只能加密/解密IP数据流；
- IPsec无法处理多播或广播IP数据流，这意味着IP多播数据流无法通过IPsec隧道。另外，很多路由选择协议（如EIGRP、OSPF和RIPv2）使用一个多播或广播地址；因此不能在IPsec对等体上使用这些路由选择协议来配置动态路由选择。

要克服这些局限性，可在对等体之间配置一条封装IP的GRE隧道，并用IPsec来保护GRE/IP隧道。对于前往GRE端点的IP多播分组用GRE封装其中的有效负载。

通过结合使用GRE隧道和IPsec，切断了动态路由选择需求和子网间的数据流同IPsec保护策略的紧密关系。只需要在两个VPN网关之间传输所有数据流的GRE隧道定义一个IPsec代理描述，而不用关心这些数据流的类型、信源和目的地，这极大地简化了IPsec保护。

3.3.2 组网设计说明

如典型组网图中的SecPath100-B1和SecPath1000F-H1之间首先建立一个GRE隧道，然后IPsec保护GRE隧道中流量，解决单独使用IPsec时的上述局限性。

1、GRE模型和IPsec模型之间的主要区别在于IPsec保护的地址范围。在GRE模型中，总部网关(SecPath1000F-H1)的加密映射指定：对从它的GRE隧道源地址出发，前往分支(SecPath100-B1)的GRE隧道接口的数据进行封装。所有用户数据流都被封装到用一个源地址和目标地址表示的隧道中。同样，在分支路由器上只需配置一个保护隧道源地址和目标地址的加密映射。这样实现，SecPath1000F-H1上前往分支SecPath100-B1的LAN IP地址的数据流先被路由到GRE隧道中，然后使用指定的GRE隧道参数封装分组，然后使用IPsec对封装的分组进行加密，再将其从出接口路由出去。

2、为了实现链路备份可以在分支设备(SecPath100-B1)上分别与中心网关(SecPath1000F-H1和SecPath1000F-H2)建立两条GRE隧道，隧道的源和目的地址分别采用分支和中心网关的公网接口地址。通过配置两条隧道路由的cost值不同，使隧道流量经过主网关进行转发。当主网关down掉后，流量经由备网关转发。

3、和IPsec模型一样，GRE模型下也可以实现负载分担。如通过路由的cost值可以设置一部分分支设备以SecPath1000F-H1为主网关，其它分支设备以SecPath1000F-H2为主网关。这样正常情况下，所有流量的转发可以由两台中心网关执行。

4、为了提高可靠性，可以在中心和分支网关设备上创建回环接口，并把它们作为GRE隧道的源和目的接口。在私网和回环接口上启用动态路由协议。

远程访问 IPsec 实现

移动远程客户（如PC、PDA等）和VPN网关之间的数据流进行加密的需求带来了新的挑战。远程接入客户模型要求一组完全不同的功能，以适应动态主机地址分配、缺乏配置控制和暂时连接。IPsec模型和GRE模型非常适合用于提供场点到场点的连接性，在这种情形下，这两种模型都不可行，因为接收远程办公人员的IPsec连接的公司场点预先不知道其IP地址。需要采用一种高效和可扩展的方式来让远程接入客户访问公司VPN。使用IKE模式配置特性能够让客户使用动态分配的IP地址，还让网络供应人员能够定义提供给客户的策略，以简化网络的运营管理。关于这种特性在《模式配置》中已有详细介绍，本文不多赘述。

此处介绍另一种远程接入实现方式：L2TP + IPsec模型。

4.1 典型组网

此典型组网和LAN-to-LAN典型组网主要区别在远程接入端点区域。

图中两台SECPATH1000F产品作为企业网总部的接入设备，实现了冗余备份。远程接入端点中SECPATH10F作为LAC设备；装有iNode客户端软件的PC可以通过LAN接入或ADSL接入两种方式和中心网关设备建立隧道。

4.2 L2TP + IPsec 模型

4.2.1 实现介绍

L2TP是在公共网络上使用IP来隧道传送PPP的协议。使用L2TP实现远程访问可以对用户进行

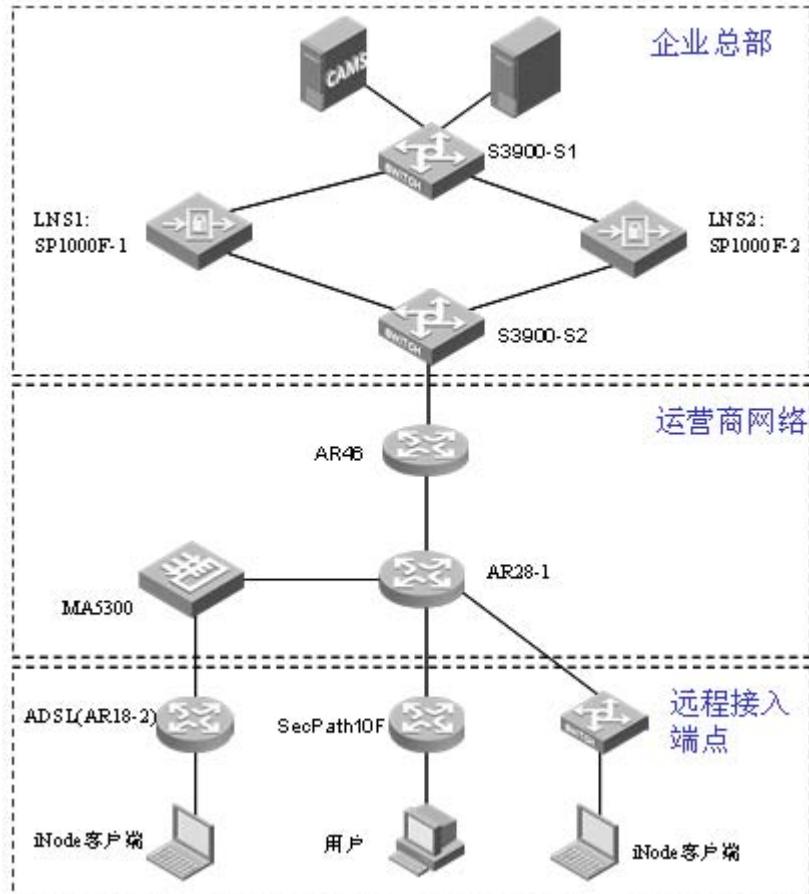


图2 远程访问IPsec实现典型组网图

身份认证和动态分配地址。但 L2TP 本身不为它隧道传送的通信提供加密机制，可以依赖 IPsec 来为它提供安全性。

IPsec 可以在作为 LAC 和 LNS 的网关设备上创建。这样做是由于 L2TP 封装是在 IPsec 过程之前运行的。所以，分组进入 IPsec 处理之前，它已经具有了 L2TP 头和 UDP 头并可以用 IPsec 加密。

4.2.2 组网设计说明

1、当用户采用 LAN 或 ADSL 接入方式接入运营商网络时，用户 PC 先和运营商设备 AR28-1 建立 PPP 连接。然后使用 iNode 客户端与企业总部的 SecPath1000F 建立 L2TP 隧道（此时的用户 PC 的角色是 LAC），并启用 IPsec 协议对 L2TP 隧道流量进行数据加密保护。

2、在企业总部的 LNS 设备上可以配置接入用户的认证方式为本地数据库认证或远端 AAA 服务器认证，用户通过认证后才能够与总部设备成功建立隧道连接。

3、可以根据情况选择使用 IKE 的主模式或者野蛮模式进行 IKE 协商；为了在网络中存在 NAT 设备的情况下成功建立隧道，可以在 iNode 客户端上启用 NAT 穿越功能。

4、也可以在典型组网中的 SecPath10F 和总部 LNS 之间建立 L2TP 隧道，实现小型分支机构的用户接入。通过在 LAC (SecPath10F) 设备上启用 l2tp-auto-client 和隧道连接保持等，可以实现一个用户认证通过后整个分支都能上网并保持连接的功能。

5、通过在客户端的 LNS 地址处填入总部中心网关设备的 VRRP 地址，可以实现在主网关 down 掉的情况下使用另一台网关设备和客户端建立隧道连接。

— 本文完 —

IPsec 专题之 测试方法

使用 Avalanche 进行 IPsec 性能测试

◆◇□ 李红霞 潘冰

VPN 连接的方式

VPN 的两种基本的类型是 LAN-to-LAN (site-to-site) 和 Client-to-LAN (remote access)。VPN 的基本原理是对报文的再次封装从而在 Internet 上创建 tunnel 传递数据，数据的安全则由安全协议（如：IPsec）来保证。VPN 连接使得通过 Internet 通讯的主机就像在自己的私网中通讯一样。

1.1 LAN-to-LAN

下图是 LAN-to-LAN VPN 连接模型。在这种 VPN 连接中，两个安全网关通过 Internet 连接，

使用 VPN 来保护安全网关后面的私网，通过在 Internet 上创建使用安全协议的 tunnel，两个安全网关为两个私网中的主机间的通讯提供了安全的路径。

1.2 Client-to-LAN

下图是 Client-to-LAN (dialup) 的 VPN 连接模型。在这种 VPN 连接中，在 Client 端没有安全网关，被保护的 LAN 在安全网关后面。通过建立 Client 和 Server 端安全网关之间的 tunnel，Client 可以和受保护的 LAN 中的主机安全的通讯。

LAN-to-LAN VPN Connection

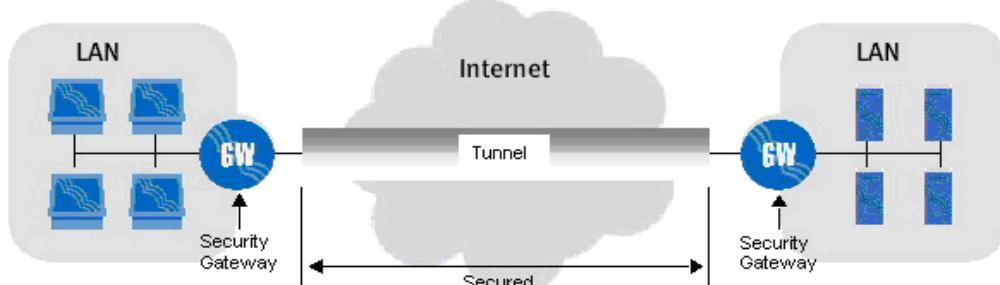


图 1 LAN-to-LAN VPN 连接模型

Client-to-LAN VPN Connection

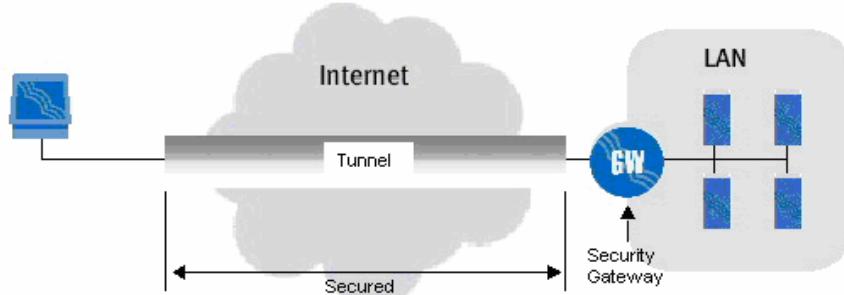
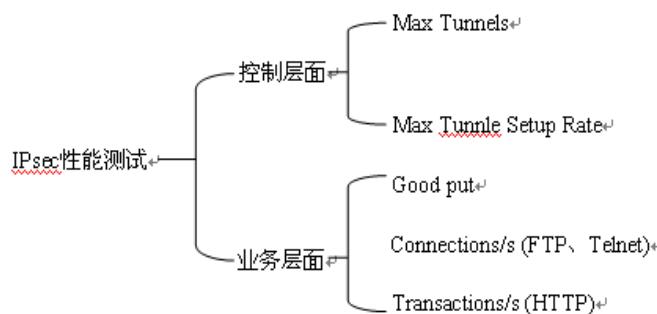


图 2 Client-to-LAN 连接模型

不管是 LAN-to-LAN 还是 Client-to-LAN，IPsec VPN 必须采用隧道模式才能支持，这是由于 IPsec 传输模式只能保护安全网关之间的通信，而不能保护安全网关后面的网络或者主机。

IPsec 测试方法

IPsec 的测试一般可以从控制层面和业务层面来进行，控制层面一般关心隧道数量、隧道建立速率等性能参数，而业务层面则关系数据的转发和加解密性能。



控制层面的测试一般能够测试 IPsec 的仪表都可以完成，而业务层面的测试，很多仪表只能最高测试到第三层的转发加解密性能，这样得出的数据往往只是理想情况下的，而这个性能指标对于用户的实际使用几乎没有太大的意义，用户关心的是实际使用的感觉而不仅仅是某些毫无意义的理论值。因此对于 IPsec 业务性能测试必须能够测试 4-7 层的应用，以用户的实际使用体验为标准进行测试。Avalanche 最大的特色就在于此。

Site to Site Scenario

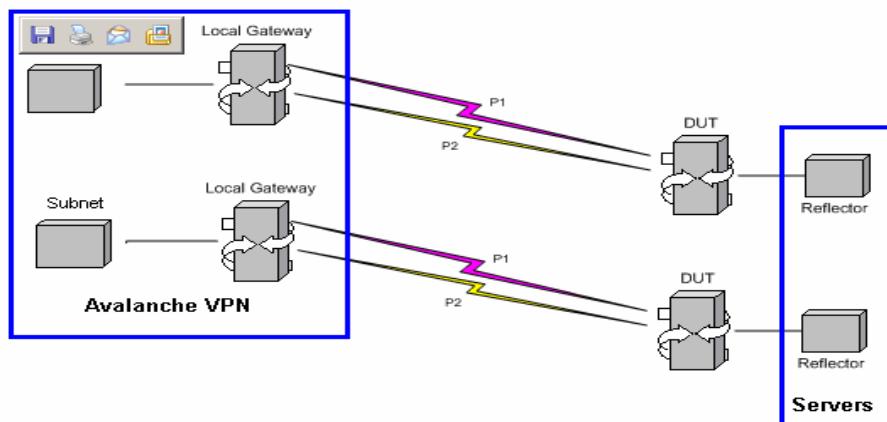


图 3 LAN-to-LAN 测试组网图

Avalanche 简介

Avalanche 是思博伦推出的用于进行 4-7 层测试的工具，它可以分为在 Smartbits 上使用的 Avalanche Smartbits 和专用硬件组成的 Avalanche 和 Reflector 设备。它可以用模拟大型的应用网络环境来对测试设备进行测试，支持对防火墙、VPN、路由等的测试。

Avalanche 的基本特性：

1. Avalanche 可以模拟客户端每秒建立成千上万的连接。
2. Reflector 可以模拟大量的应用服务器，与 Avalanche 配合对于 DUT 进行相应的测试。
3. Avalanche 可以模拟一个用户连接所作出的反应，如在 HTTP 测试中的 User Behavior 选项。

同时 Avalanche 可以模拟 LAN-to-LAN 和 Client-to-LAN 的方式对设备进行测试。同时能够模拟承载在 IPsec Tunnel 上的流量，对设备对具体的应用进行测试。下面介绍如何使用 Avalanche 进行 IPsec 的测试。

IPsec 性能测试

分为 LAN-to-LAN 和 Client-to-LAN 两种方式进行介绍。以 Avalanche 7.5 为例。

4.1 测试组网图

LAN-to-LAN:

Client-to-LAN:

Remote Access Scenario

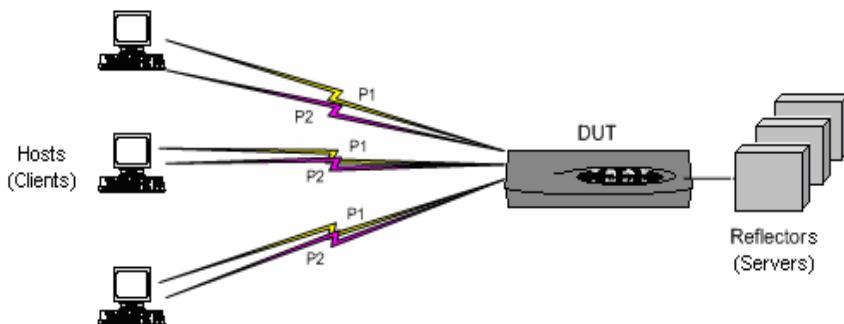


图4 Client-to-LAN 测试组网图

测试中使用Avalanche 模拟Client 端的网络，包括Client 端的安全网关和Client 端后面的网络。Avalanche 和被测设备（DUT）建立IPsec 隧道。Reflector 模拟DUT 保护的网络中的服务器。数据流由Avalanche 产生，访问Reflector，Avalanche 和DUT 建立IPsec 隧道，数据加密传输，在DUT 上解密后转发给Reflector，模拟真实的业务访问过程。

IPsec 隧道建立成功后，流量从Client 端流入Server 端时，需要DUT 对报文进行解封装操作，那么测试的就是DUT 解封装下的转发性能；流量从Server 端流入Client 端时，需要DUT 对报文进行封装处理，那么测试的就是DUT 封装下的转发性能。

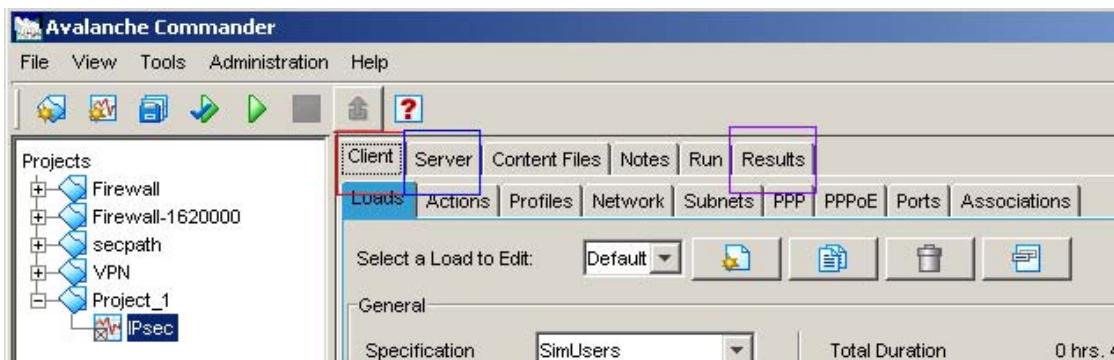
4.2 基本配置

连接上测试仪器后，通过控制台，可以看到以下的界面。

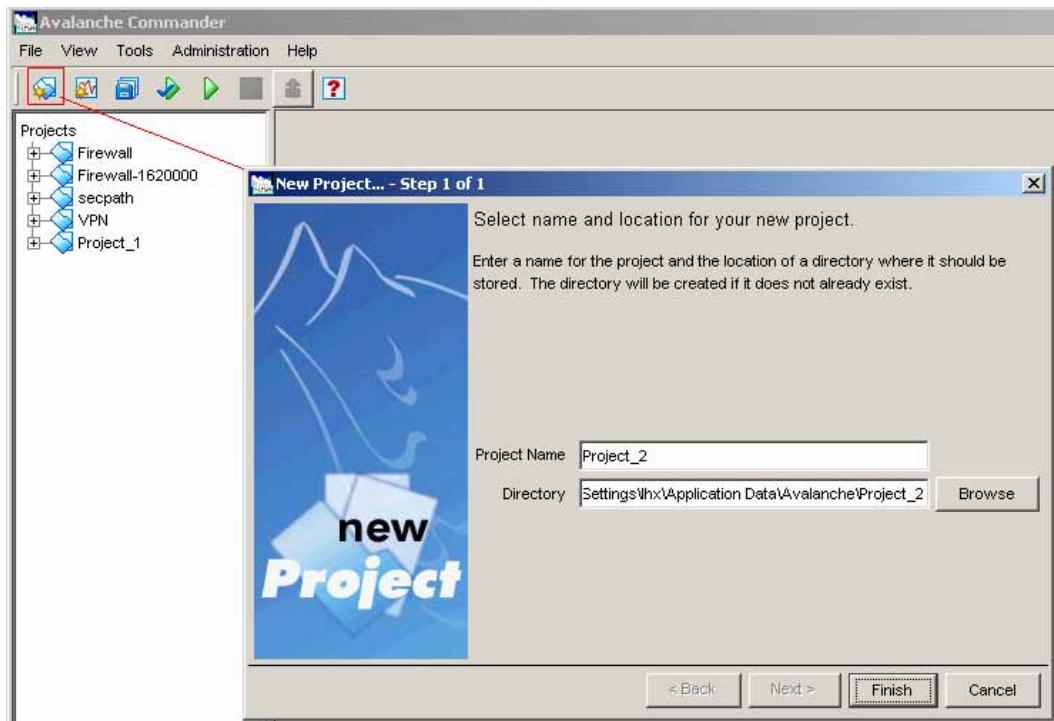
在Avalanche 的主界面上可以分为以下几个部分：

- ◆ 左边的测试项目导航栏，显示了已经存在的Project 和Test。
- ◆ 右边的测试配置栏，对具体的Test 进行配置，使用较多的标签为Client、Server 以及Run。下面将会有详细的介绍。
- ◆ 最上方的菜单栏。
- ◆ 工具栏。

Avalanche 测试仪由Avalanche 和Reflector 两个设备组成，这里Client 标签是对Avalanche 进行配置，而Server 标签是对Reflector 进行配置。

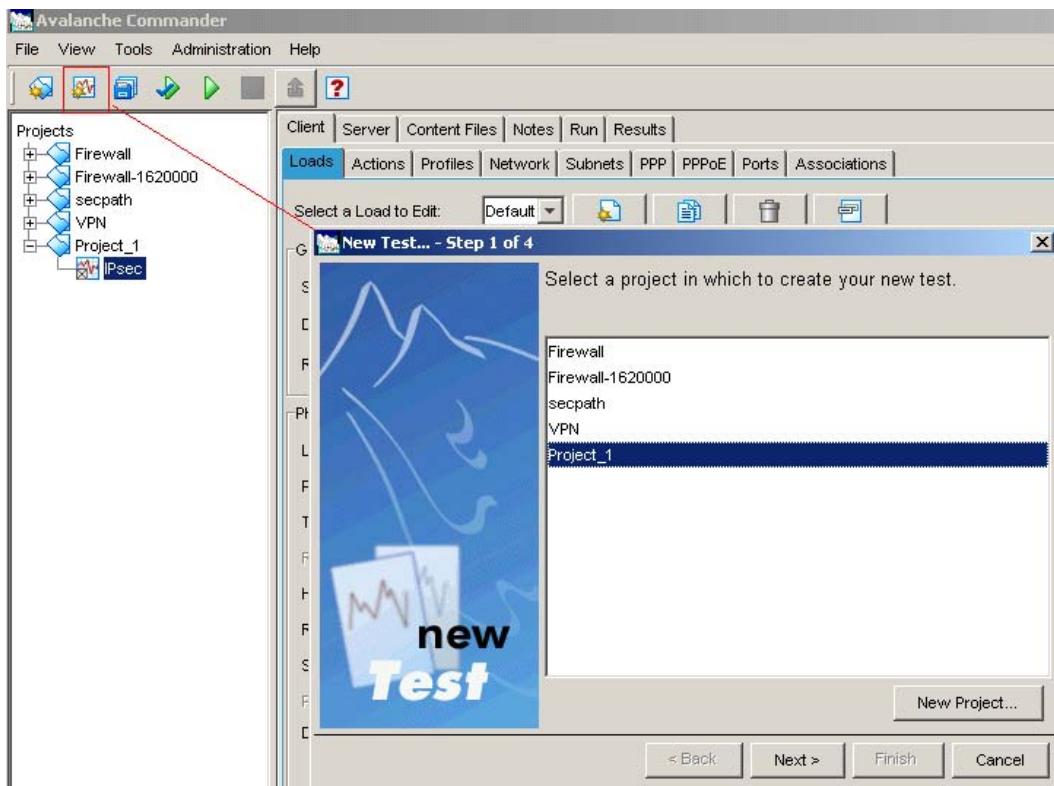


1. 首先创建一个 Project。



2. 创建 Test:

新建 Test，并选择对应的 Project。

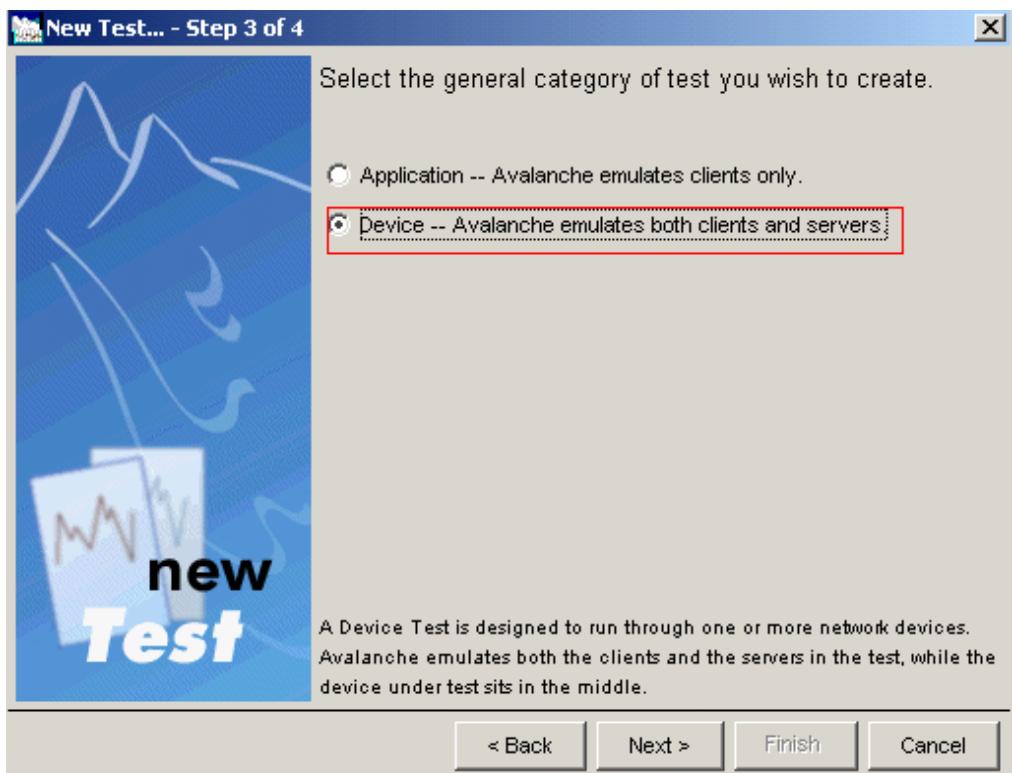


点击 Next, 填写 Test Name:

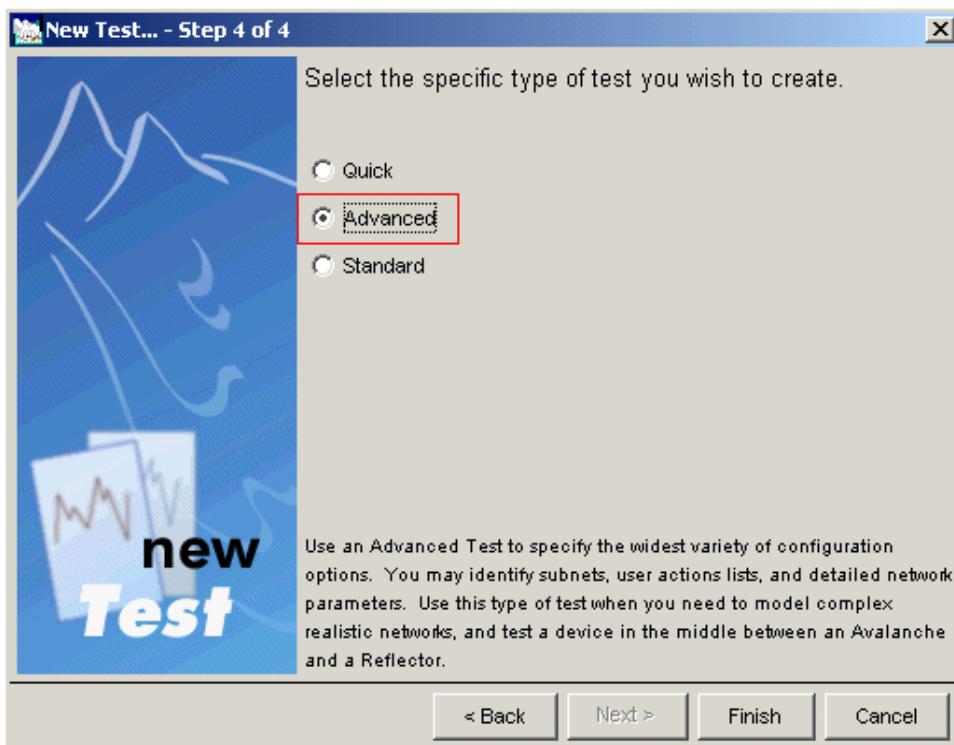


点击 Next, 选择 Device。

因为需要 Avalanche 同时模拟 Client 和 Server。



点击 Next，选择 Advanced，点击 Finish。



4.3 LAN-to-LAN 方式

4.3.1 Client 端的配置

首先熟悉一下，Client 端的配置标签：



基础配置参数：Load、Actions、Network、Subnets、Ports、Associations 参数在这里不做累述。主要说明一下和 IPsec 相关的配置。

1. 选择 Client-->Subnets：

Client | Server | Content Files | Notes | Run | Results |
 Loads | Actions | Profiles | Network | **Subnets** | PPP | PPPoE | Ports | Associations |

Show Subnet Profiles using: IPv4 IPv6 VLAN MAC IPSec Policy Generator...

Subnet Name	IP Address (Range)	Netmask	Network	Default Gateway	Gateway Ad...	Randomize IP	Enable IPSec	Remote
3_0190	172.16.191.1	/24	172.16.191.0	<input type="checkbox"/>		<input type="checkbox"/>	<input checked="" type="checkbox"/>	
1_0011	172.16.12.1	/24	172.16.12.0	<input type="checkbox"/>		<input type="checkbox"/>	<input checked="" type="checkbox"/>	
1_0012	172.16.13.1	/24	172.16.13.0	<input type="checkbox"/>		<input type="checkbox"/>	<input checked="" type="checkbox"/>	
1_0013	172.16.14.1	/24	172.16.14.0	<input type="checkbox"/>		<input type="checkbox"/>	<input checked="" type="checkbox"/>	
1_0014	172.16.15.1	/24	172.16.15.0	<input type="checkbox"/>		<input type="checkbox"/>	<input checked="" type="checkbox"/>	
1_0015	172.16.16.1	/24	172.16.16.0	<input type="checkbox"/>		<input type="checkbox"/>	<input checked="" type="checkbox"/>	
1_0016	172.16.17.1	/24	172.16.17.0	<input type="checkbox"/>		<input type="checkbox"/>	<input checked="" type="checkbox"/>	
1_0017	172.16.18.1	/24	172.16.18.0	<input type="checkbox"/>		<input type="checkbox"/>	<input checked="" type="checkbox"/>	
1	172.16.1.1	/24	172.16.1.0	<input type="checkbox"/>		<input type="checkbox"/>	<input checked="" type="checkbox"/>	
1_0018	172.16.19.1	/24	172.16.19.0	<input type="checkbox"/>		<input type="checkbox"/>	<input checked="" type="checkbox"/>	

Selected Range: 3_0190 (1)

3_0190

Static Routing | IP Fragmentation | Realism | PPP / PPPoE | DHCP | **IPSec**

IPSec Policies

General Message Parameters Forms Database

Persistent Tunnel Send Timeout (sec): 5 Max Re-transmit: 5 File:

Phase 1 Phase 2 Digital Certificates

Encrypt Packet 3 Transform: ESP Certificate File:

Re-connect Time (sec): 20 Re-key Threshold (sec):

IKE Lifetime (sec): 28800 SaLD Lifetime (sec):

Use Commit Bit CA Certs

Phase 2 Policies

界面上方配置 Client 端安全网关保护的子网。可以是多个子网。每个子网创建一个 Tunnel。Local Gateway 设置 Avalanche 模拟的子网的网关，Remote Gateway 填写 DUT 的接口地址。ISAKMP ID 填写 IKE Peer 中 remote name。

需要说明的是，当选择主模式的时候，id-type 为 ip-address，此时和我司设备的互通存在问题，建议选择野蛮模式。

勾选 IPSec，才能看到界面下方的 IPSec 标签。这里配置 IPSec 策略，主要配置第二阶段的策略，目前只支持 ESP。

Phase 2 Policies						
Enabled	Tunnel Option	PFS	Hash	Encryption	Dest Subnet	Dest Mask
<input checked="" type="checkbox"/>	One per Subnet	NONE	MD-5	ESP-DES	2.2.2.2	/0

注意填写 Dest Subnet，默认为 0.0.0.0。

2. Client—>Associations:

Client | Server | Content Files | Notes | Run | Results |
 Loads | Actions | Profiles | Network | **Subnets** | PPP | PPPoE | Ports | Associations |

Load Profile Type: User Based Global Global Profile Name: Default

Enabled	Action	Profile	Weight	Port	Subnet
<input checked="" type="checkbox"/>	Default	Default	100	10.154.0.34:1	3_0190 (IPv4)

选择在 Subnets 中配置的子网，这里选择的就是测试仪器将要发送的流量。

4.3.2 Server 端的配置

Server 端没有特别需要说明的，只要注意 Server 的地址和 Client 端配置的一致。

Associations				
Profile	Port	Subnet	IPv4 Address Range	
Default	10.154.0.35:1	Default (IPV4)	2.2.2.2	

4.4 Client-to-LAN

Avalanche 7.5 版本在该模式中启用 XAUTH，而我司的设备不支持这种方式，IKE 第一阶段协商失败。因此目前无法使用测试仪在该模式下对我司设备进行测试。

在 Client-->Subnets 中勾选 Remote Access，同时选择 XAUTH 的方式。

Subnets								
Gateway Ad...	MAC	Byte 1	Byte 2	Randomize IP	Enable IPSec	Remote Access	Gateway Versi...	Local Gateway
	<input type="checkbox"/>			<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
	<input type="checkbox"/>			<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	IPV4	1.1.1.13
	<input type="checkbox"/>			<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	IPV4	1.1.1.14

Server 端的配置和前面相同。

测试时可以结合 Load 中流量测试设备在真实应用下的性能。

4.5 开始测试

4.5.1 控制层面指标的测试

1. 最大隧道数

LAN-to-LAN 方式：

- ◆ Client-->Subnets 标签中创建的子网数 = 最大隧道数；
- ◆ Client-->Association 标签中创建子网数个联系；
- ◆ Client-->Load 标签中创建的连接数 >= 最大隧道数；
- ◆ 查看 RUN 标签中是否连接全部建立，是，则可以增加子网；否，则减少子网重新测试。

Client-to-LAN 方式：

- ◆ Client-->Subnets 标签，不需要创建多个子网，只需创建多个 IP，在 IP Address (Range) 中配置多个 Client 地址。

- ◆ 其他同上。

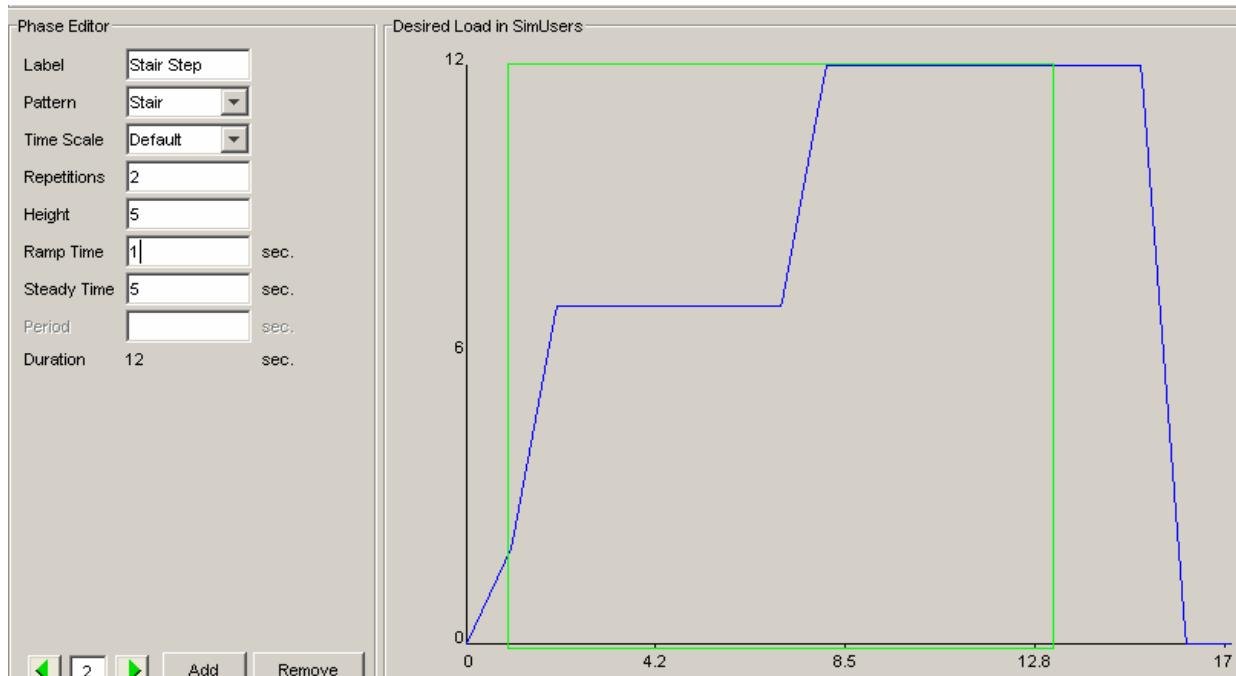
2. 最大隧道建立速率

IPsec 标签的配置和最大隧道数相同。其他配置项举例说明。

例如：测试设备的最大隧道建立速率是否达到 5 tunnels/s。

LAN-to-LAN 方式：

- ◆ 创建 > 5 个子网，建议至少创建 20 个子网；
- ◆ Client-->Load 标签中连接建立的速率为 5，根据斜率来确定。如图所示：



Load 第二阶段中，Ramp Time=1, Height=5，这样斜率为 5，连接建立的速率为 5，连接建立之初为每个连接创建一个 tunnel，因此连接建立速率=tunnel 建立速率，也可以在 Client Statistic 中查看到 tunnel 建立的速率；

- ◆ 查看 RUN 标签中是否连接全部建立，是，则可以增加速率；否，则减少速率重新测试。

Client-to-LAN 方式：

- ◆ 创建 > 5 个 IP 地址，建议至少创建 20 个 IP 地址。
- ◆ 其他同上。

4.5.2 业务层面指标测试

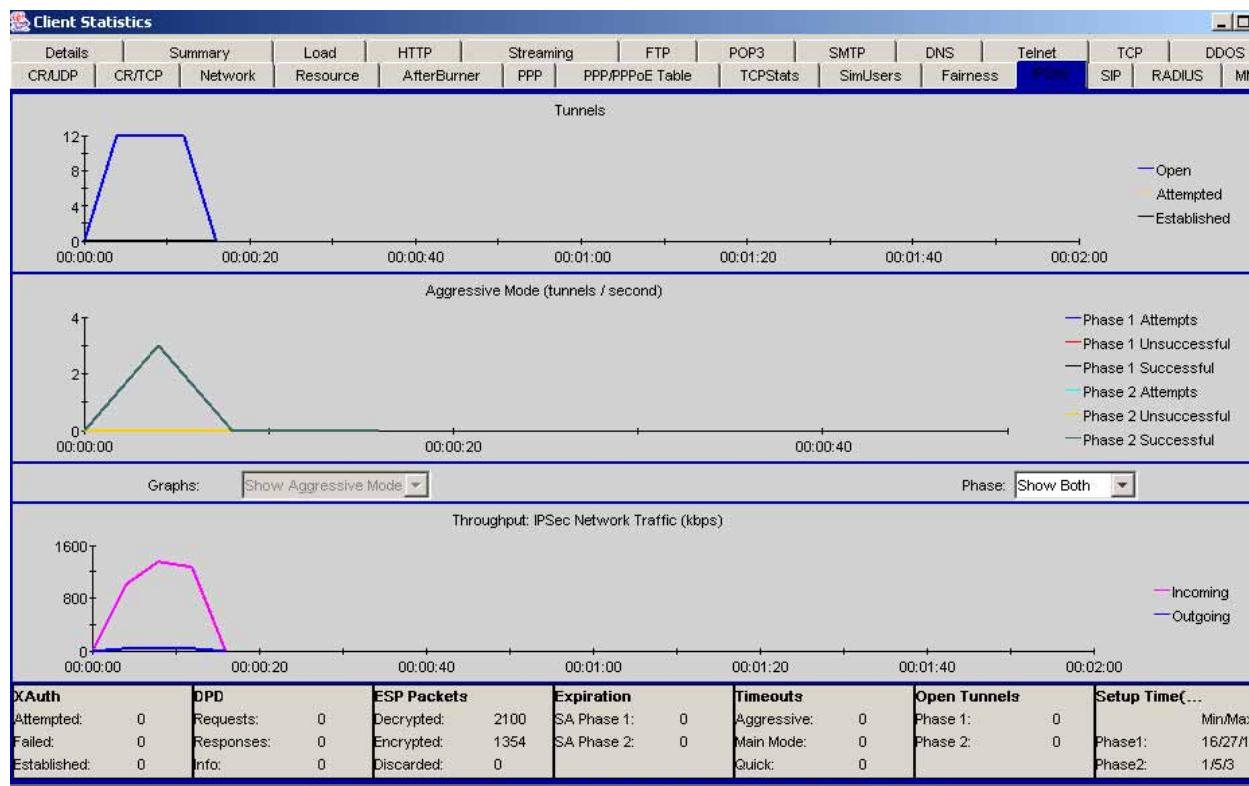
本测试项测试设备在 IPsec 隧道上承载的各种业务的处理能力，配置同 Avalanche 的基础配置，在本文中不再累述，可以参看 Avalanche 的帮助手册。

4.6 测试结果分析

4.6.1 控制层面指标

可以在 RUN-->Client Statistic-->IPsec 标签中的看到隧道建立的情况：

1. 图中最上端说明了已经建立的隧道数，目前是 12 个。这个指标可以说明设备支持的最大隧道数。
2. 图中中间部分说明了当前的隧道建立速率，最大的速率是 3 tunnels / s。这个指标可以衡量设备隧道建立速率。



3. 图中最下端给出了 IPsec 隧道的吞吐量，同时最下端的表格中描述了设备当前的各项指标。目前为测试完成状态，其他指标为 0。

该指标只有在所有连接都成功建立的情况下才是有效的。即 RUN 中没有 Unsuccessful 的连接。



4.6.2 业务层面指标

该指标可以查看 RUN—> Client Statistic 中 Detail 标签、HTTP、TCP 等相关业务的标签。

以上的所有的指标可以在统计报表中获得。在 Results 标签中选择当前的测试，查看统计数据。

一般察看 Client-->realtime 的报表，包括 Goodput 等指标。

Client | Server | Content Files | Notes | Run | Results

	Test Name	Run Name	Date & Time	
<input type="checkbox"/>	IPsec		2006/09/03 02:50:08	
<input type="checkbox"/>	IPsec		2006/09/03 02:51:22	
<input type="checkbox"/>	IPsec		2006/09/03 02:52:32	
<input type="checkbox"/>	Test_1		2006/09/01 03:11:20	
<input type="checkbox"/>	Test_1		2006/09/01 03:12:40	
<input type="checkbox"/>	Test_1		2006/09/01 03:14:20	
<input checked="" type="checkbox"/>	Test_1		2006/09/01 03:16:04	

All/None [Delete Selected Results](#) [Archive Selected Results](#) Filter: [Show All Tests](#)

▼

[Details](#) | [Highlights](#) | [Notes](#)

Test_1--2006/09/01 03:16:04

- Merged Statistics

Client Summary:	View ...	Save As ...	Client Real-time	View ...	Save As ...
Server Summary:	View ...	Save As ...	Server Real-time	View ...	Save As ...
Event Log:	View ...	Save As ...			
SNMP Statistics:	View ...	Save As ...			
- Individual Load Generator Statistics
 - client-subtest_0**
 - realtime.csv
 - summary.csv
 - server-subtest_0**
 - Http_summary.csv
 - realtime.csv
 - summary.csv

使用 IXIA VPN 进行 IPsec 性能测试

◆◇□ 田浩博

IPsec/IKE 简介

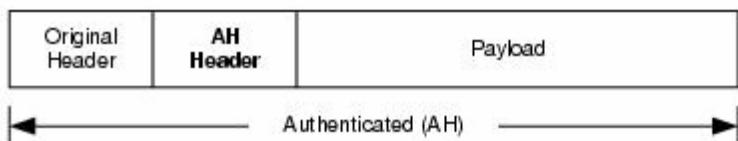
IPsec (IP Security) 是 IETF 制定的三层隧道加密协议，它为 Internet 上传输的数据提供了安全保证。IPsec 为 IP 数据报提供了高质量的、可互操作的、基于密码学的安全性。特定的通信方之间在 IP 层通过加密与数据源认证等方式，来保证数据报在网络上传输时的机密性、完整性、真实性。

IPsec 包括 AH (Authentication Header, 认证头协议, 协议号 51) 和 ESP (Encapsulating Security Payload, 报文安全封装协议, 协议号 50) 两个协议。AH 和 ESP 可以单独使用，也可以同时使用。另外，IPsec 又分隧道 (tunnel) 和传输 (transport) 两种工作模式。需要注意的是，AH 头会对整个报文进行认证，而 ESP 头则对加密后报文的部分进行认证。在同时使用这两种协议的时候，先使用 ESP 协议对初始的报文进行加密和认证，再使用 AH 协议对整个报文进行认证，这样可以为要传输的报文提供最大强度的保护。图 1 展示了不同工作模式下 AH 和 ESP 协议分别对报文的保护情况，以及隧道模式下联合使用 AH 和 ESP 协议对报文的保护情况。

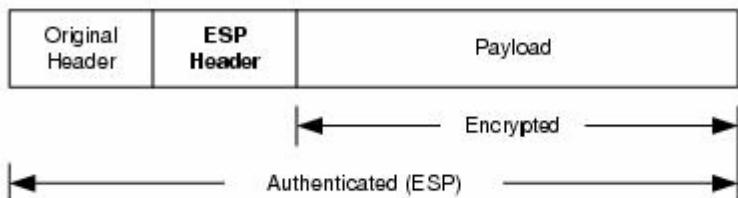
IPsec 通过 SA (Security Association, 安全联盟)，对不同的数据流提供不同级别的安全保护。安全联盟是单向的，在两个对等体之间的双向通信，最少需要两个安全联盟来分别对两个方向的数据流进行安全保护。同时，如果希望同时使用 AH 和 ESP 来保护对等体间的数据流，则分别需要两个 SA。安全联盟可以通过手工配置的方式建立，但是当网络中结点增多时，手工配置将非常困难，工作量非常巨大，而且难以保证其安全性。这时就要使用因特网密钥交换协议。这时就要使用因特网密钥交换协议 (Internet Key Exchange) 来自动地进行安全联盟的建立与密钥交换的过程。

IKE 协议建立在由 Internet 安全联盟和密钥管理协议 ISAKMP (Internet Security Association and Key Management Protocol) 定义的框架上。IKE 为 IPsec 提供了自动协商交换密钥、建立安全联盟的服务，能简化 IPsec 的使用和管理，大大减少 IPsec 的配置和维护工作。IKE 不是在网络上直接传输密钥，而是通过一系列数据的交换，最终计算出双方共享的密钥，并且即使第三者截获了双方用于计算密钥的所有交换数据，也不足以计算出真正的密钥。(Internet Key Exchange) 来自动地进行安全联盟的建立与密钥交换的过程。

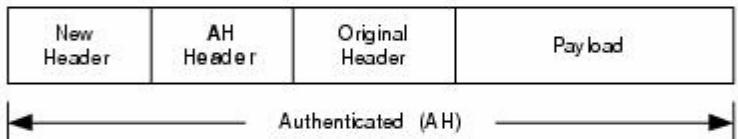
Transport Mode (AH)



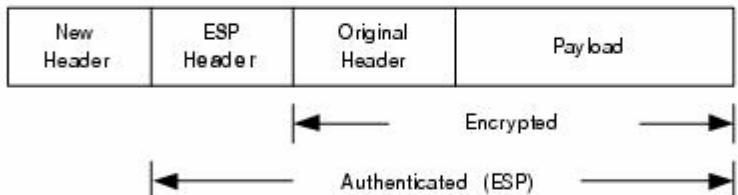
Transport Mode (ESP)



Tunnel Mode (AH)



Tunnel Mode (ESP)



Tunnel Mode (AH + ESP)

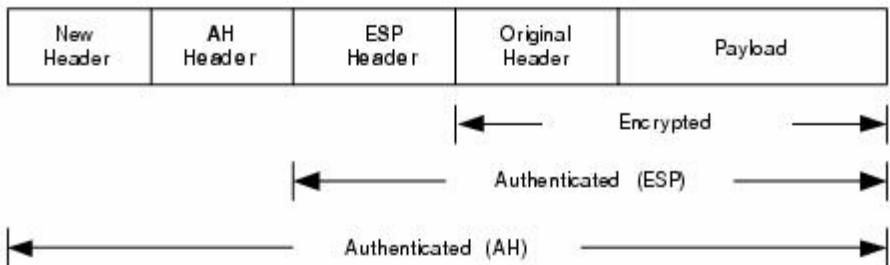


图 1 不同模式下 AH 和 ESP 协议对报文的保护情况

在 IPsec 的实际使用中，有两种典型的组网：

- Site to Site：如图 2 所示。两个站点之间通过一对 IPsec 安全网关相连，这样，对于任意一个站点的任意一个主机来说，当它和另外一个站点的主机通信时，都会认为连接是安全的。安全网关间的 IPsec 隧道保证了在不安全的网络中，能够进行安全的数据传输。

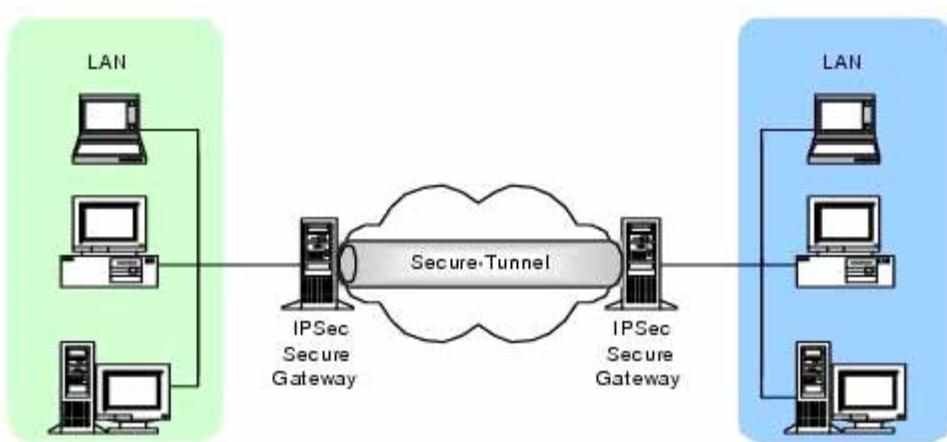


图2 site to site

- Client to LAN: 在图3所示的拓扑中，主机将自己作为 IPsec 安全网关，与远端的 IPsec 安全网关建立隧道，实现了和远端站点内主机进行安全通信。

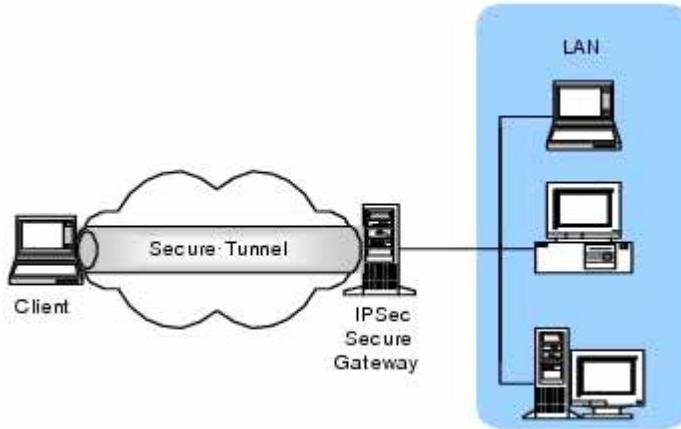


图3 Client to LAN

IxVPN 软件就是根据这两种拓扑，通过模拟安全网关和网关后的主机，对待测设备进行测试。

使用 IXIA VPN 进行 IPsec 的测试

2.1 IXIA VPN 简介

IxVPN 是 IXIA 公司推出的专门用于测试 IPsec VPN 安全网关性能的工具。每个端口都有一个全面的 IPsec 与 IKE 协议栈，可模拟很多个安全网关（Secure Gateway）与客户端，以建立成千上万个 IPsec 隧道。下图是实际环境和 IXIA 模拟的环境对比：

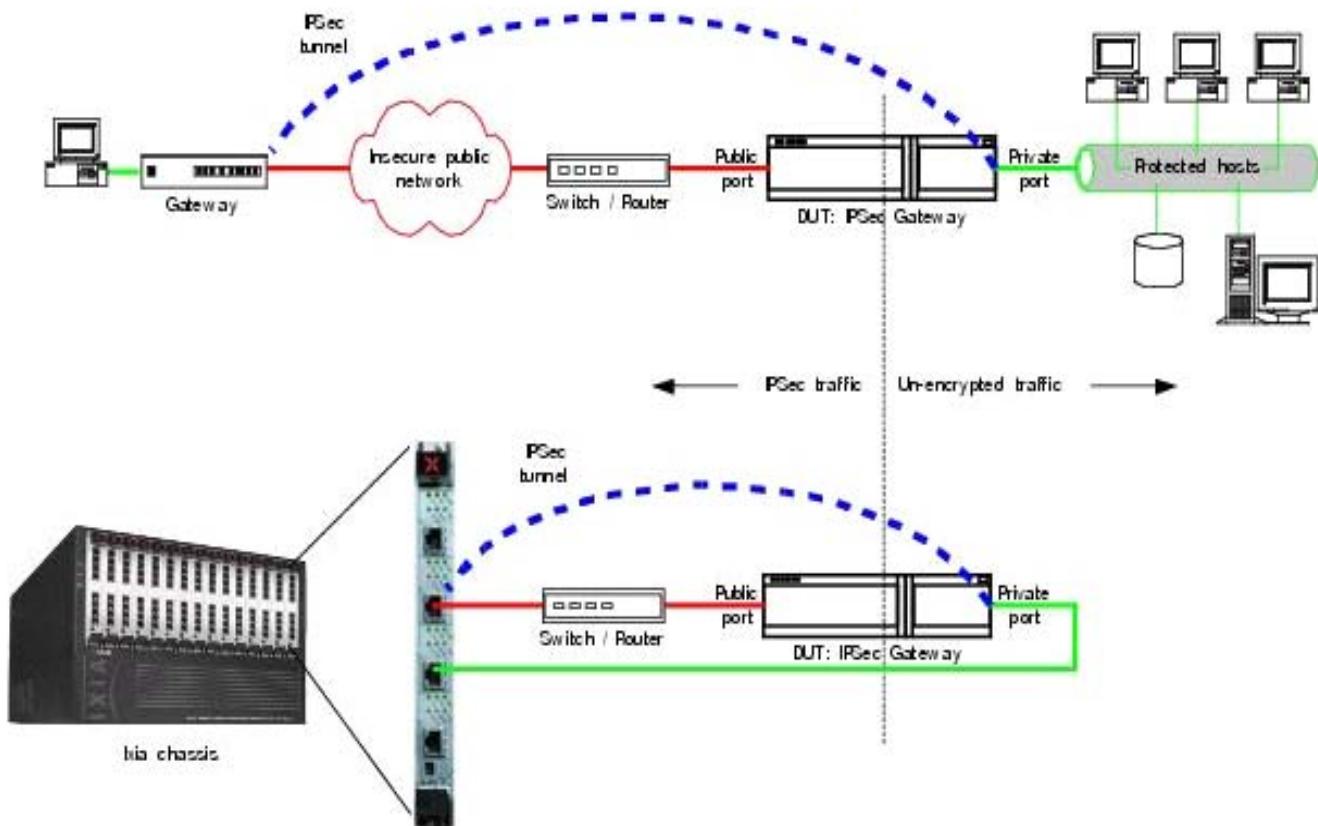


图 4 实际环境和 IXIA 模拟环境

对于 IXIA 测试仪的一个端口来说，不仅可以模拟一个安全网关和这个安全网关后的一台主机，还可以模拟很多个安全网关以及这些网关后的多个主机。当然，如果 DUT 的性能超强的话，可以利用测试仪的多个端口和 DUT 建立 IPsec 隧道。

隧道一旦建立起来，IxVPN 便可采用 RFC 2544 方法测试 IPsec 网关的加密与解密性能。配合 IxChariot 可以模拟 IPsec 隧道中的各种业务流量负载（此部分内容不在本次介绍中）。IxVPN 所能提供的主要测试项是：

- 隧道数量
- 隧道建立率
- 通过 VPN 隧道的 IxChariot 业务负载
- 通过 IPsec 隧道的 RFC 2544 测试

2.2 IXIA VPN 配置 step by step

运行 IXIA VPN 后，首先看到的是这个界面

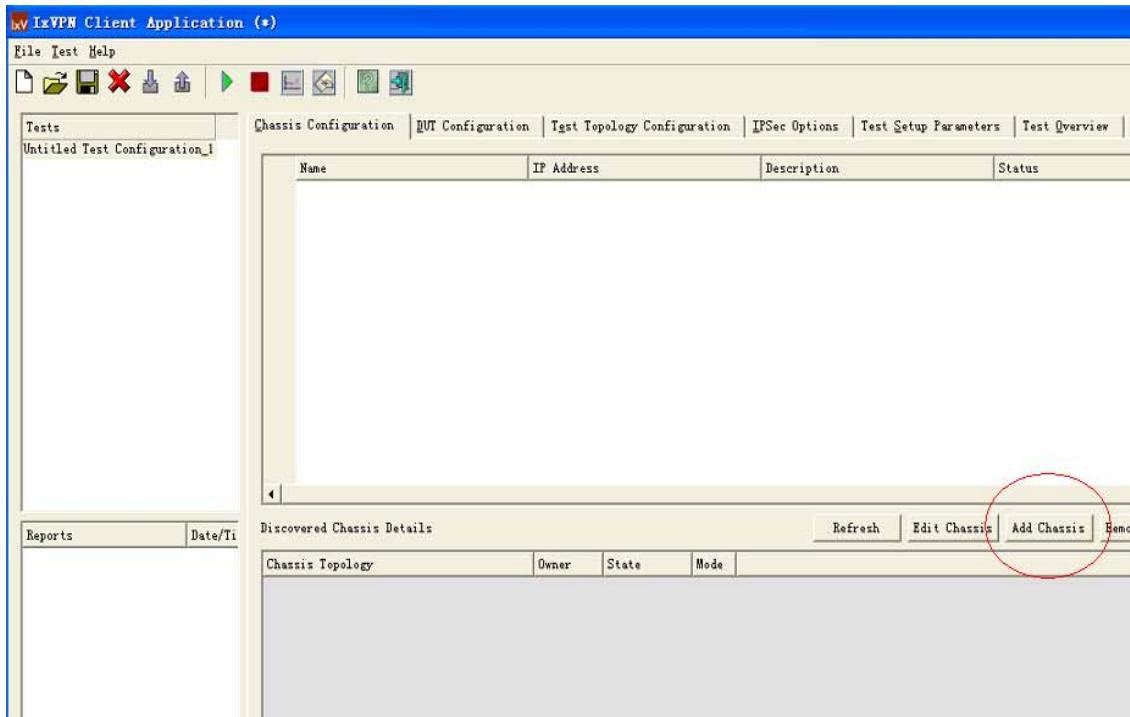


图5 Chassis configuration界面

因为是第一次运行，所以没有任何可用的板卡，需要手工添加，点击图5中红色的部分，出现下面对话框：

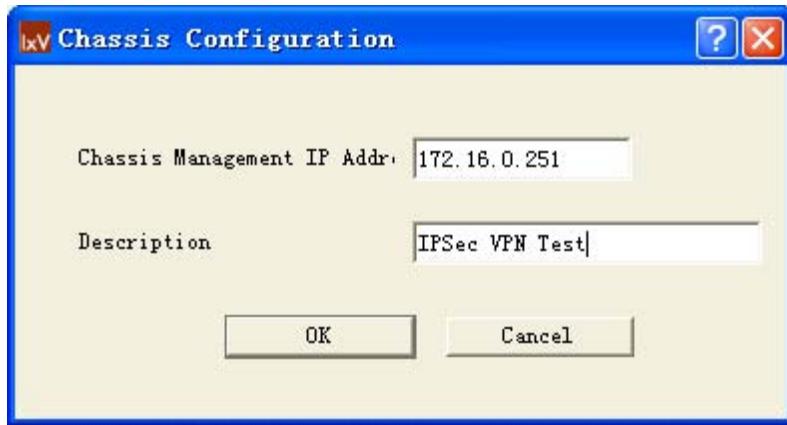


图7 DUT Configuration界面

在“Chassis Management IP Address”后的方框中填入测试仪的IP地址，“Description”中的内容可填也可以不填。确认后，在图5中的“Discovered Chassis Details”下面的灰色框中就可以看到测试仪上的所有板卡了。

然后选择“DUT Configuration”页面，添加被测设备。如下



图 7 DUT Configuration 界面

单击图 7 中红色圆圈的按钮，出现下面的界面

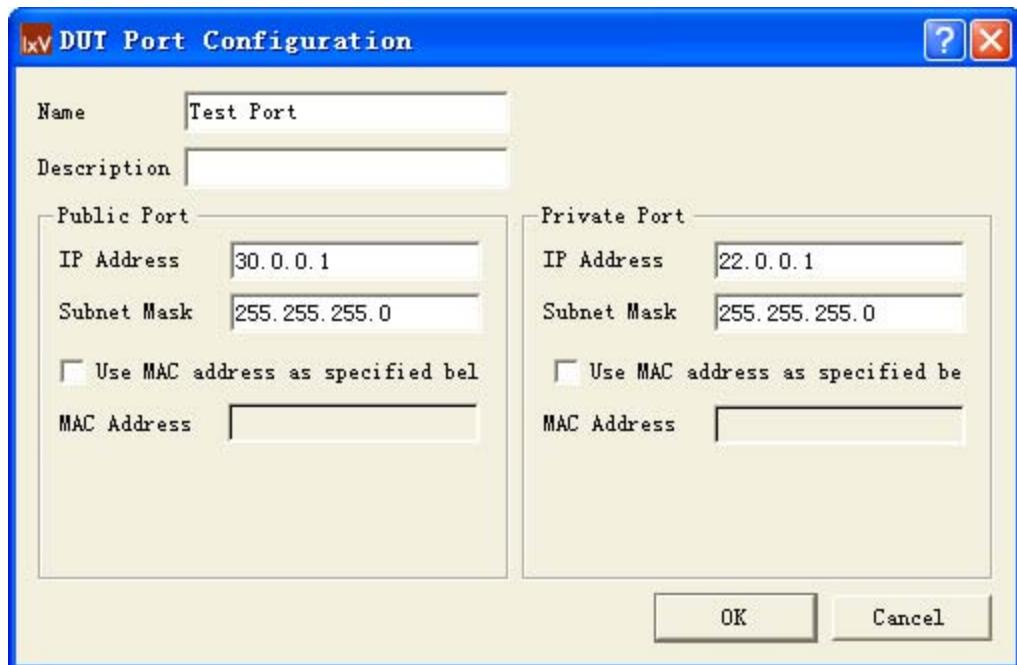


图 8 DUT Port Configuration 界面

在图 8 中，“Name”和“Description”是对 DUT 端口的描述。需要注意的是“public port”和“private port”两个配置项。回头看一下图 4，我们可以知道，public port 就是和公网相连的端口。测试仪端口模拟的大量安全网关，就是和这个端口建立的 IPsec 隧道。而 private port 是和私网相连的端口，不需要再建 IPsec 隧道了。IPsec 隧道建立成功后，流量从公网流入私网时，需要 DUT 对报文进行解封装操作，那么测试的就是 DUT 解封装下的转发性能；流量从私网流入公网时，需要 DUT 对报文进行封装处理，那么测试的就是 DUT 封装下的转发性能。这两个地址需要和 DUT 上的接口地址一致。

完成上面的操作后，进入“Test Topology Configuration”配置界面。在上一步中，我们配置了 D U T 的端口，在这一步中，就要完成和 D U T 端口相连的测试仪端口的配置了。我们已经知道了 D U T 的哪个端口连着什么样类型的网络，但是测试仪端口并不知道它要模拟什么，这就需要人工去指定了。

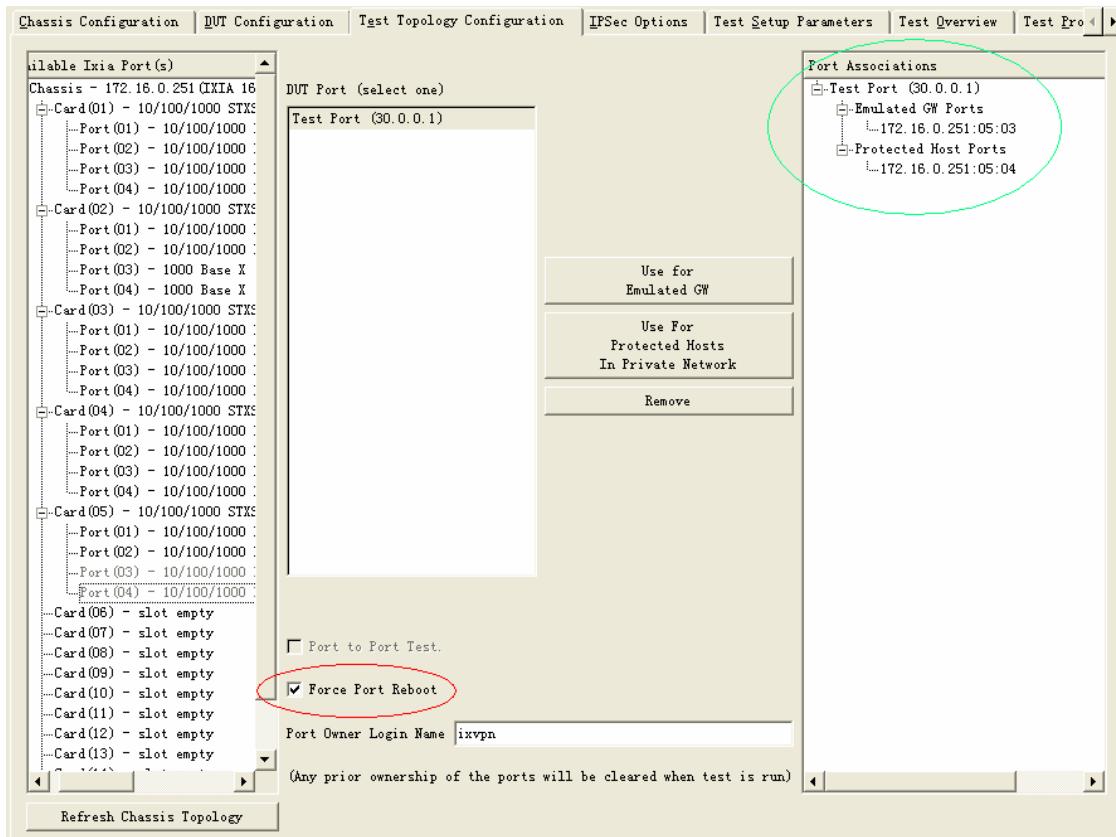


图 9 Test Topology Configuration 界面

首先选中图中左侧部分的“card 5 port 3”端口，然后单击图中中部“Use For Emulated GW”按钮；再选中图中左侧部分的“card 5 port 4”端口，然后单击图中中部“Use For Protected Hosts In Private Network”按钮。完成后，会出现图中绿色圆圈部分。这就意味着“card 5 port 3”端口模拟安全网关，和DUT的public port相连，“card 5 port 4”端口模拟私网内主机，和DUT的private port相连。

请注意，图中红色部分如果没有勾上的话，那么最好把它勾上。因为测试仪端口每做完一次测试，端口都会有大量的隧道信息需要保存。如果不重启，可能会对下一次的测试有影响。

然后进行的是 I P s e c 的配置，这一步比较简单，无非是使用什么什么样的加密算法，什么样的认证算法等等。如图 1 0：

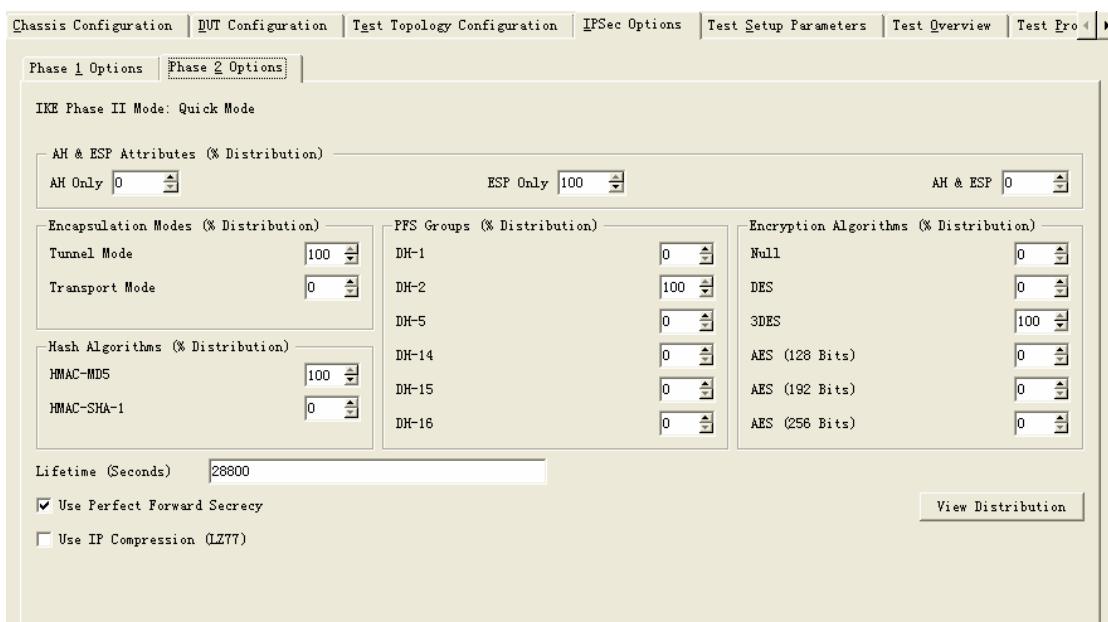


图 10 IPsec Options 界面

这里有一点需要注意的地方，在IPsec 协商中，不能使用“AH + ESP”这种方式，否则的话测试仪和我们的设备之间无法建立IPsec 隧道。和IXIA 的工程师交流过，他们也不知道为什么。目前只能单独使用“AH”或“ESP”方式。

在前面，我们已经选择了“card 5 port 3”端口模拟安全网关，不过模拟什么样的安全网关？安全网关后模拟多少主机？没有明确说明。下面让我们进行这步比较重要的配置。选择“Test Setup Parameters”界面，如图 11

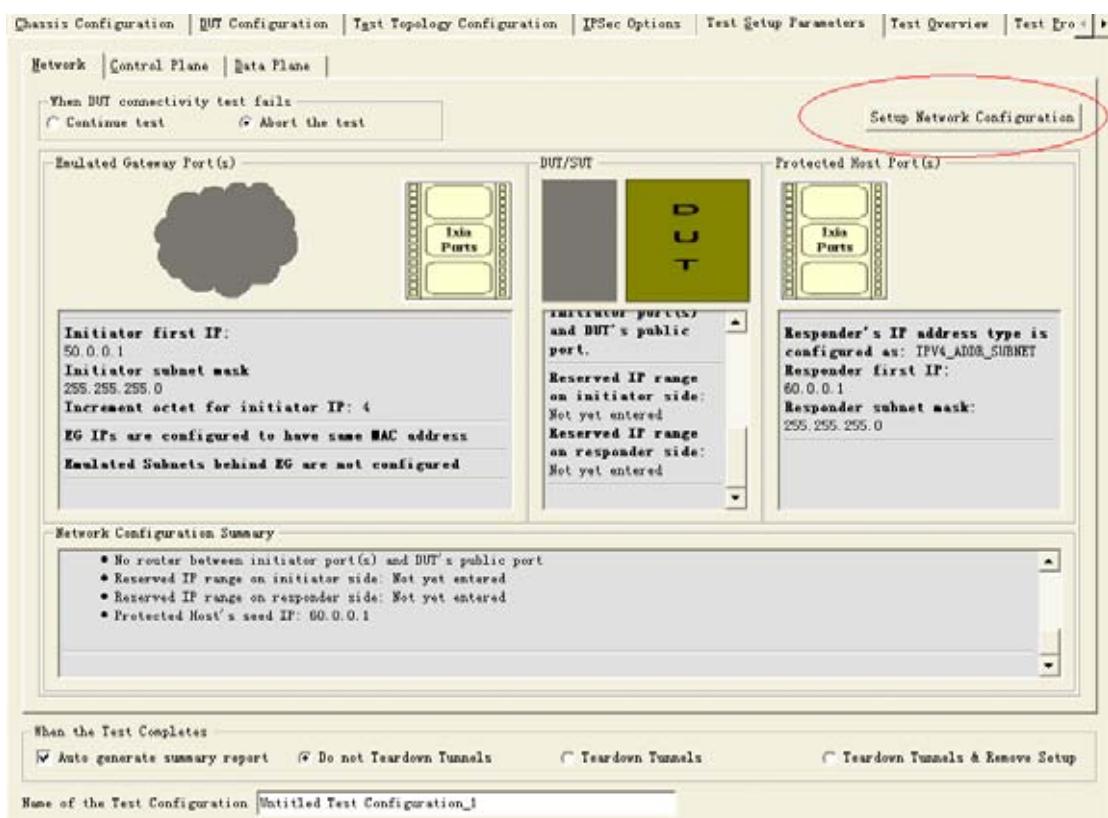


图 11 Test Setup Parameters 界面

这个界面下，又分三个页面，分别是“Network”、“Control Plane”、“Data Plane”。“Network”页面配置安全网关的IP和网关后的主机，“Control Plane”页面配置建立隧道的数量及重复建隧道的次数，“Data Plane”页面配置进行哪方面的性能测试。

配置比较复杂的是“Network”页面，见图11。先看一下页面下部的“When the Test Completes”单选框，请根据需要设定。在“Name of the Test Configuration”提示框中可以给本次测试起一个好记的名字，也可以就使用默认的。然后单击右上部红色圆圈中的“Setup Network Configuration”按钮，又会出现一个配置界面，如下：

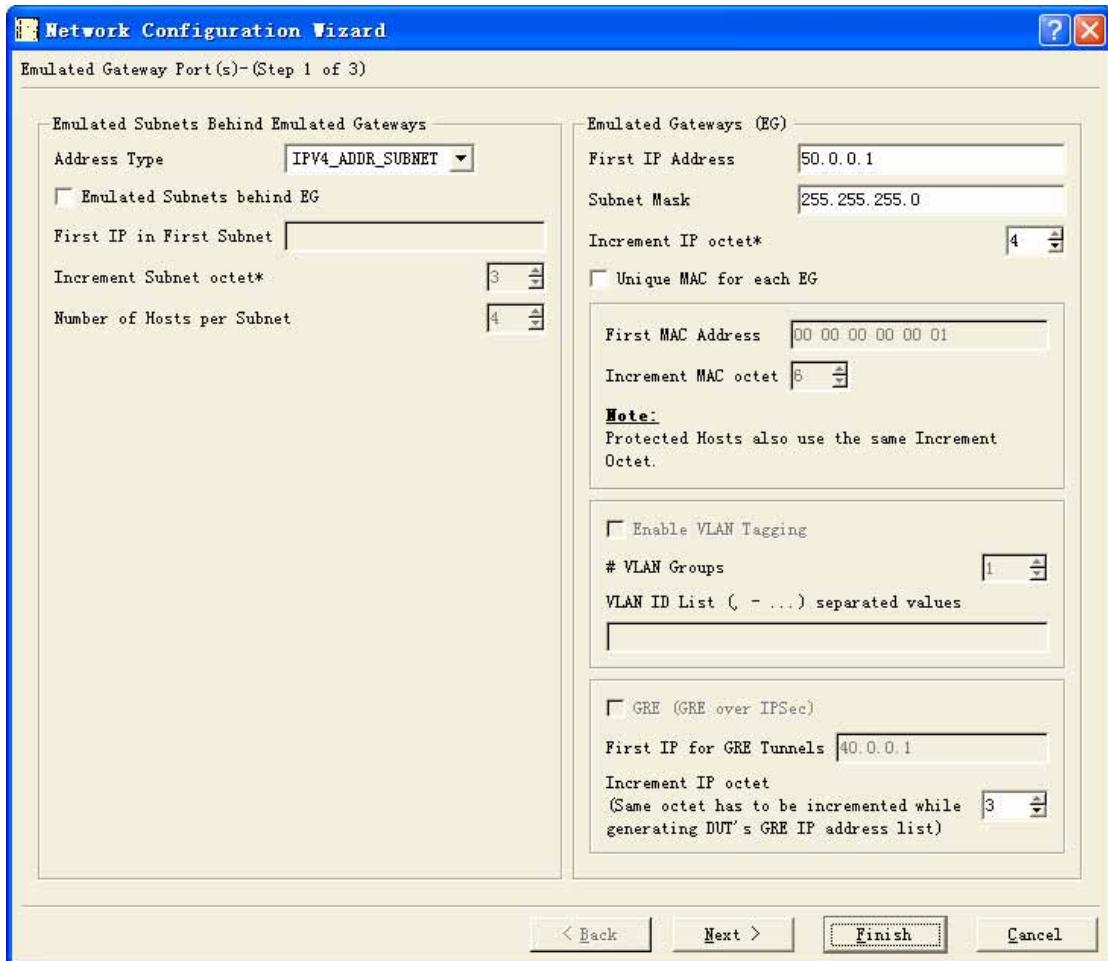


图12 Network Configuration Wizard界面1

如果在安全网关后需要模拟多个子网和主机，那么请把“Emulated Subnets behind EG”勾上，然后在下面的“First IP Address in First Subnet”方框输入起始IP地址。“Increment Subnet Octet”指的是从第几个字节开始，网段开始变化。默认值为3。比如，如果“First IP Address in First Subnet”中的IP是1.0.0.1，那么第一个子网就是1.0.0.1，第二个是1.0.1.1，第三个是1.0.2.1，以此类推。这里需要注意，如果修改这个默认值，测试仪模拟的网段有可能会产生重叠。而“Number of Hosts per Subnet”想必大

家也猜到了，指的就是每个子网后有多少主机。在“Emulated Gateways”方框中，添入的是网关的地址。同样，测试仪可以模拟多个网关，“Increment IP Octet”指的是开始变化的字节。这样的配置对应的就是 site to site 的拓扑结构；如果“Emulated Subnets behind EG”没有被勾上，那么对应的就是 client to site 的拓扑结构。

完成这一步的配置后，点击“Next”按钮，出现下一个界面

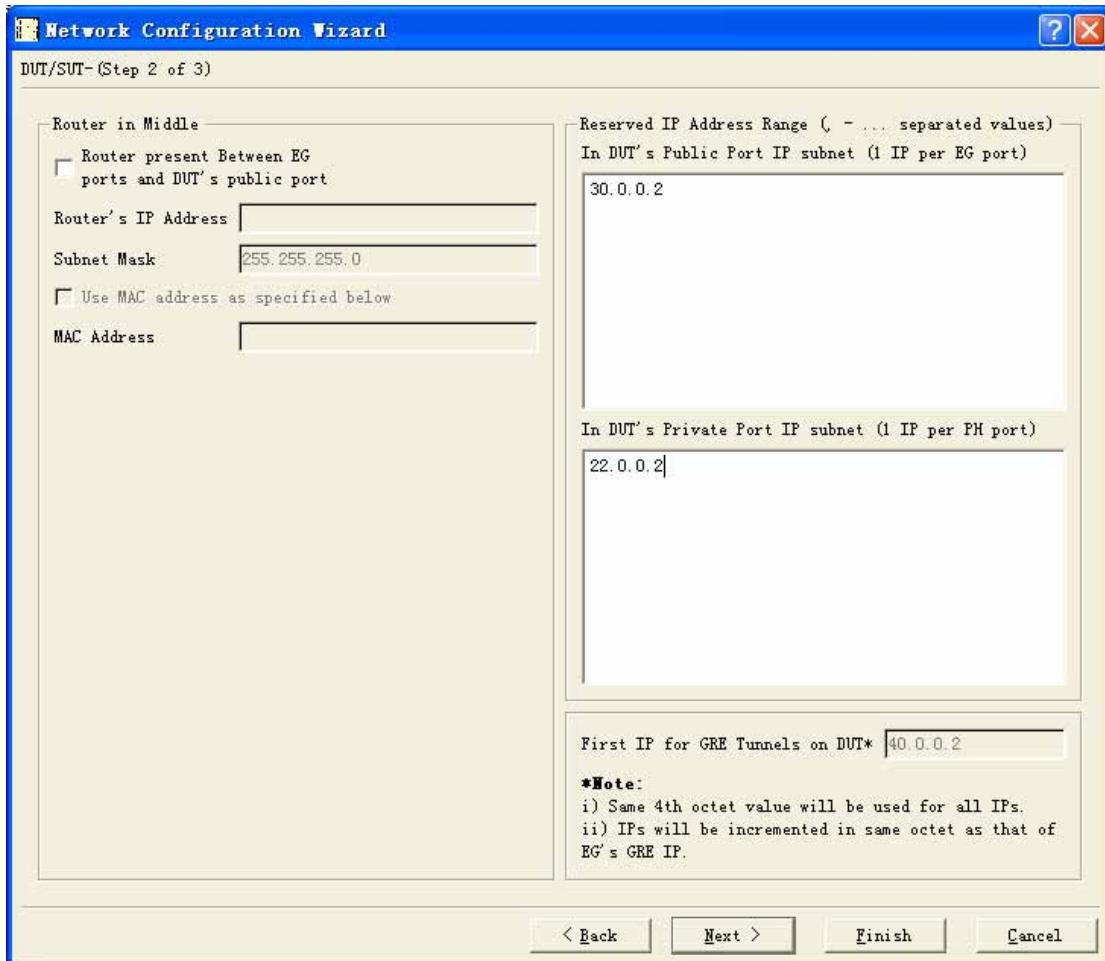


图 13 Network Configuration Wizard 界面 2

如果在测试仪和 DUT 间还有其它的设备，那么请把“Router present between EG ports and DUT 把 public port”选项勾上，并且将中间设备的地址信息填写完整。再看图 13 的右侧，“In DUT 把 Public Port IP Subnet”需要至少配置一个和 DUT 上 public port 在同一网段的 IP。这样做的目的有两个：(1) 测试仪也需要学习 DUT 的 MAC，通过配置同一网段的地址可以完成一次完整的 ARP 过程；(2) 这个地址也是路由的下一跳。在图 12 中，我们配置了 EG，IPsec Tunnel 需要建立在 EG 和 DUT 端口之间（本例中为“50.0.0.1”和“30.0.0.1”间）。一般来说，EG 和 DUT 都不在同一网段。这时 DUT 就把这个地方配置的地址作为到达 EG 的下一跳，即在 DUT 上配置这样的一条路由“ip route-static 50.0.0.1 24 30.0.0.2”。同理，“In DUT 把 Private Port Subnet”也需要至少配置一个和 DUT 上 private port 在同一网段的 IP。

继续 next，那么会看到下一个界面：

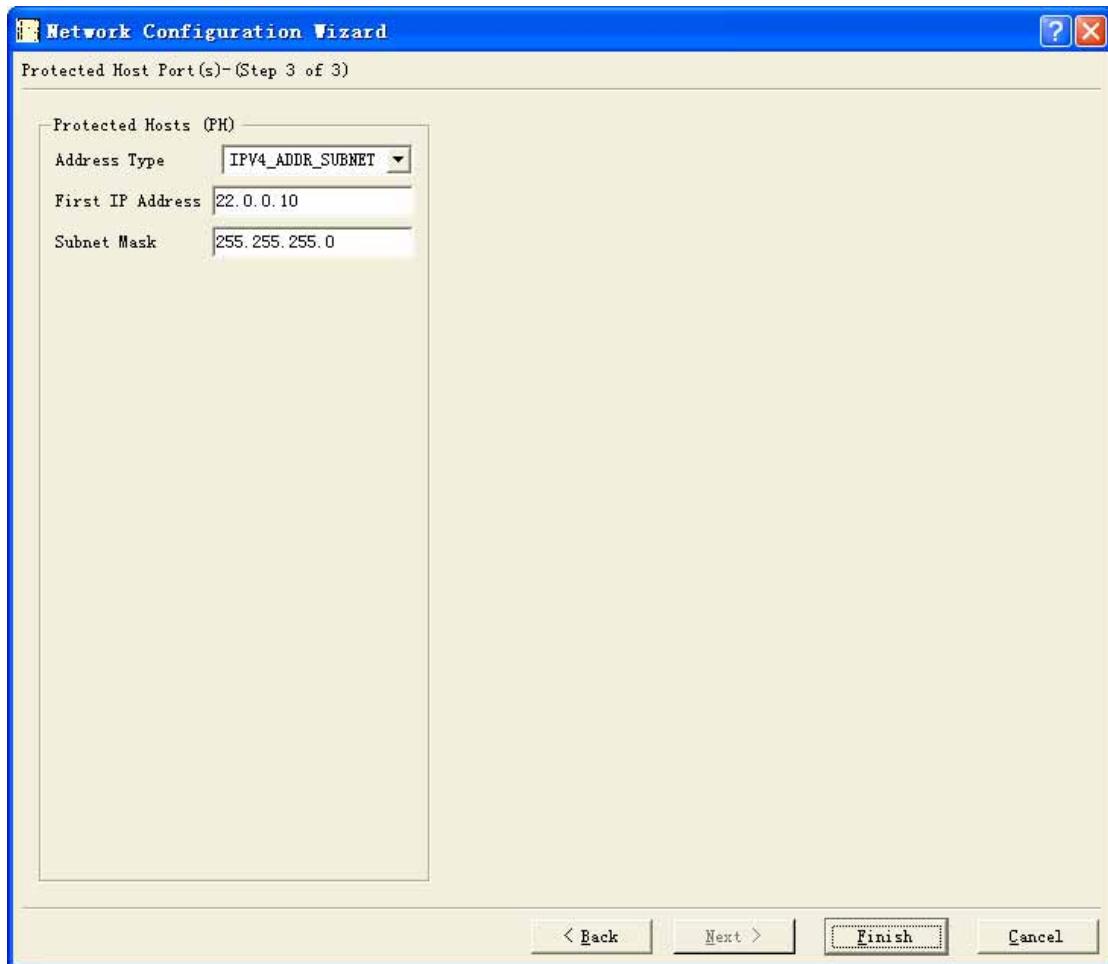


图 14 Network Configuration Wizard 界面 2

这个界面完成功能和图 12 中右侧部分差不多，只不过这里配置的是 DUT 上 private port 端口后的网络。完成了这一步后，“Network”的配置才算真正完成。

现在我们来配置“Test Setup Parameters”界面下的“Control Plane”页面，如下：

这个页面主要选择测试哪些内容。相信大家一看就能明白，现在对“Tunnel Rate Test”的几个参数做简单说明：

- Iterations：测试的次数。比如说要建立 100 条 IPsec 隧道，那么就可以建立 100 条隧道几次，然后结果（比如建立隧道的时间）取这几次的平均值。
- Number of Tunnels Requested：要建立的 IPsec 隧道的数量
- Initial Count 和 Increment：这两个参数是共同起作用的。比如要建立 100 条隧道，但是不希望一下都建立，而是想先建立 20 条，然后再以每次 10 条的速率建，那么 Initial Count 就应该为 20，而 Increment 为 10

Chassis Configuration | DLT Configuration | Test Topology Configuration | IPSec Options | Test Setup Parameters | Test Overview | Test Progress

Network | Control Plane | Data Plane

Tunnel Capacity Test

Tunnel Rate Test

Initial count: 10 Increment: 10
Iterations: 1 # Tunnels Requested: 100

(Maximum # tunnels possible to support in current configuration = 3000)

Tunnel setup options:

- Tunnel Setup Timeout (sec): 30
- Number of Retries: 0
- Retry Interval (Sec): 0
- Increment Retry Interval (Sec): 0

Tunnel rekey options:

- Rekey
- Number of Rekey Retries: 0 Stop Rekey After: 0 Seconds
- Rekey Fuzz Percentage: 0 Rekey(s)
- Rekey Margin (Seconds): 0

Multiple Phase 2 SAs

Multiple Phase 2 per Phase 1

Number of Phase 1 SAs: 100
Number of Phase 2 SAs per Phase 1 SA: 2

Cumulative Failure Threshold

of Tunnels Failed: 1
% of Tunnels Failed: 1.000

图 15 Control Plane 界面

完成了这步的配置，现在开始“Test Setup Parameters”界面下的“Data Plane”页面的配置，如下：

Network | Control Plane | Data Plane

RFC 2544

RFC 2544 - Frame Loss Test

RFC 2544 - Throughput & Latency Test

Throughput Test - Frame Loss Threshold %: 0.000

Frame Loss Test - # of Samples: 1

of Tunnels to Use: 1

Protocol: UDP

Port Number: 7

Best possible offered load

Unencapsulated Offered Load (Mbps): 1

Test Duration (MM:SS): 00:10

Frame Size(s) (Bytes):

RFC 2544 recommended List Range

RFC 2544 | List | Range |

64, 128, 256, 512, 1024, 1280, 1424

Soak Test

Soak Test

Simultaneous bi-directional traffic

of Tunnels to Use: 1

Protocol: UDP

Port Number: 7

Test Duration (MM:SS): 00:10:00

Unencapsulated Offered Load (Mbps): 1000

Frame Size(s) (Bytes):

RFC 2544 recommended List Range

RFC 2544 | List | Range |

64, 128, 256, 512, 1024, 1280, 1424

这个界面的配置也很简单了，和 Ix Script Mate 差不多。在结果当中，会分别测试出加密时的性能和解密时的各种性能。

剩下的两个界面“Test Overview”和“Test Progress”就没有什么可配置的了。通过“Test Overview”可以预览网络拓扑和 IPsec 属性；测试进行时，可以通过“Test Progress”适时查看测试进程。

测试仪是测试人员手中的利器，希望大家能够充分发掘测试仪的功能，发现更多的问题！

— 本文完 —

Route to Network

网络之路

—— IPsec专题

策划：刘宇 陈旭盛

主编：陆宇翔

编委：杜祥宇 文旭 徐庆伟

王慧升 周迪 陈国华

刘胜伟 潘冰 李丹

李红霞 刘倩 刘洋

美编：皮妮娜 梁坤

纷吾既有此内美兮 又重之以修能

扈江蓠与薜芷兮 纫秋兰以为佩

—— 《楚辞·离骚》



你身边的好网络