

Virtual Private Networks

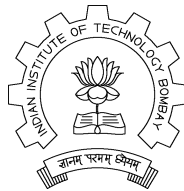
Project Report

Submitted by

Ajit Burad
Sanchit Garg
Prekshu Ajmera

under the guidance of

Prof. Bernard L. Menezes



Department of Computer Science and Engineering,
Indian Institute of Technology, Bombay

Mumbai

November 13, 2006

Abstract

This report describes the major technologies for virtual private networks (VPNs) used today on the Internet. VPN is a cost effective and secure way for different corporations to provide user access to the corporate network and for remote networks to communicate with each other across the Internet. The aim of this report is to present a comprehensive overview of VPNs. We present classification of VPNs such as Secure and Trusted VPNs and describe different VPN protocols such as IPSec, SSL, MPLS. We also present a comparison between these and how it can help us on which VPN methodology to use. In this report we also provide a demonstration on working of IPSec.

Contents

1	Introduction	1
1.1	What is VPN ?	1
1.2	Why VPNs ?	1
1.3	Types of VPN	1
2	Secure VPNs	3
2.1	IPSec	3
2.2	SSL	6
2.3	Comparison between IPSec and SSL	6
2.4	IPSec in Linux-2.6	7
3	Trusted VPNs	10
3.1	L2VPN	11
3.1.1	Types of L2VPN	11
3.1.2	Packet Forwarding	12
3.2	L3VPN	12
3.2.1	Routing Mechanism	13
3.2.2	Packet Forwarding	14
3.3	Multiprotocol Label Switching (MPLS)	14
3.3.1	What is MPLS ?	14
3.3.2	L2TPv3	15
4	Critics / Opinion / Findings	17
5	Conclusion and Future Work	18
	Appendices	18

A	Setting up IPSec in Linux 2.6	19
B	Glossary	25
	Bibliography	27

Chapter 1

Introduction

1.1 What is VPN ?

A **virtual private network (VPN)** is a private communications network often used within a company, or by several companies or organizations, to communicate confidentially over a publicly accessible network. VPN message traffic can be carried over a public networking infrastructure (e.g. the Internet) on top of standard protocols, or over a service provider's private network with a defined Service Level Agreement (SLA) between the VPN customer and the VPN service provider.

1.2 Why VPNs ?

- Extend geographic connectivity.
- Improve security where data lines have not been ciphered.
- Reduce operational costs versus traditional WAN.
- Reduce transit time and transportation costs for remote users.
- Simplify network topology in certain scenarios.
- Provide global networking opportunities.

1.3 Types of VPN

1. **Secure VPN** - It uses cryptographic tunneling protocols to provide the intended confidentiality, sender authentication, and message integrity to achieve privacy. When properly

chosen, implemented, and used, such techniques can provide secure communications over unsecured networks. The key feature is its ability to use public networks like the Internet rather than rely on private leased lines.

Secure VPN protocols include the following:

- IPSec - IP security
- SSL - Secure Socket Layer

2. **Trusted VPNs** - They do not use cryptographic tunneling, instead rely on the security of a single provider's network to protect the traffic.

Trusted VPN protocols include the following:

- L2VPN - Layer 2 VPN
- L3VPN - Layer 3 VPN

Since trusted VPNs work on private lines, we use circuit-switched networks instead of packet-switched network for the QoS reasons. In such a case Multi-protocol label switching (MPLS) becomes important. When customers rely on a provider for secure communication there is a Service Level Agreement (SLA) between the two, which not only decides on security terms but also QoS. Thus Resource Reservation Protocol (RSVP) comes into picture.

Chapter 2

Secure VPNs

Secure VPNs use the tunneling mechanism to carry data on public internet lines. In tunneling data is transmitted through a public network in such a way that routing nodes in the public network are unaware that the transmission is part of a private network. Tunneling is generally done by encapsulating the private network data and protocol information within the public network protocol data so that the tunneled data is not available to anyone examining the transmitted data frames. IPSec and SSL are commonly used in secure VPNs.

2.1 IPSec

IPSec (IP Security) is a standardized framework for securing Internet Protocol (IP) communications by encrypting and/or authenticating each IP packet in a data stream. There are two modes of IPSec operation: **transport mode** and **tunnel mode**.

In **transport mode** only the payload (message) of the IP packet is encrypted. It is fully-routable since the IP header is sent as plain text. Transport mode is used for host-to-host communication.

In **tunnel mode**, the entire IP packet is encrypted. It must then be encapsulated into a new IP packet for routing to work. Tunnel mode is used for network-to-network communications (secure tunnels between routers). Since encryption and encapsulation are done by routers/gateways, end systems need not support this.

IPSec protocols operate at the network layer. This makes IPSec more flexible, as it can be used for protecting both TCP and UDP-based protocols, but increases its complexity and processing overhead, as it cannot rely on TCP (layer 4) to manage reliability and fragmentation. Protocols used for securing traffic in IPSec are AH and ESP.

Authentication Header (AH)

Authentication Header (AH) is intended to guarantee connectionless integrity and data origin authentication of IP datagrams. AH tries to protect all fields of an IP datagram. Only fields changeable during transfer of an IP packet are excluded. AH operates directly on top of IP using IP protocol number 51. An AH packet diagram in transport mode is shown in figure 2.1. The

IPSec in AH Transport Mode

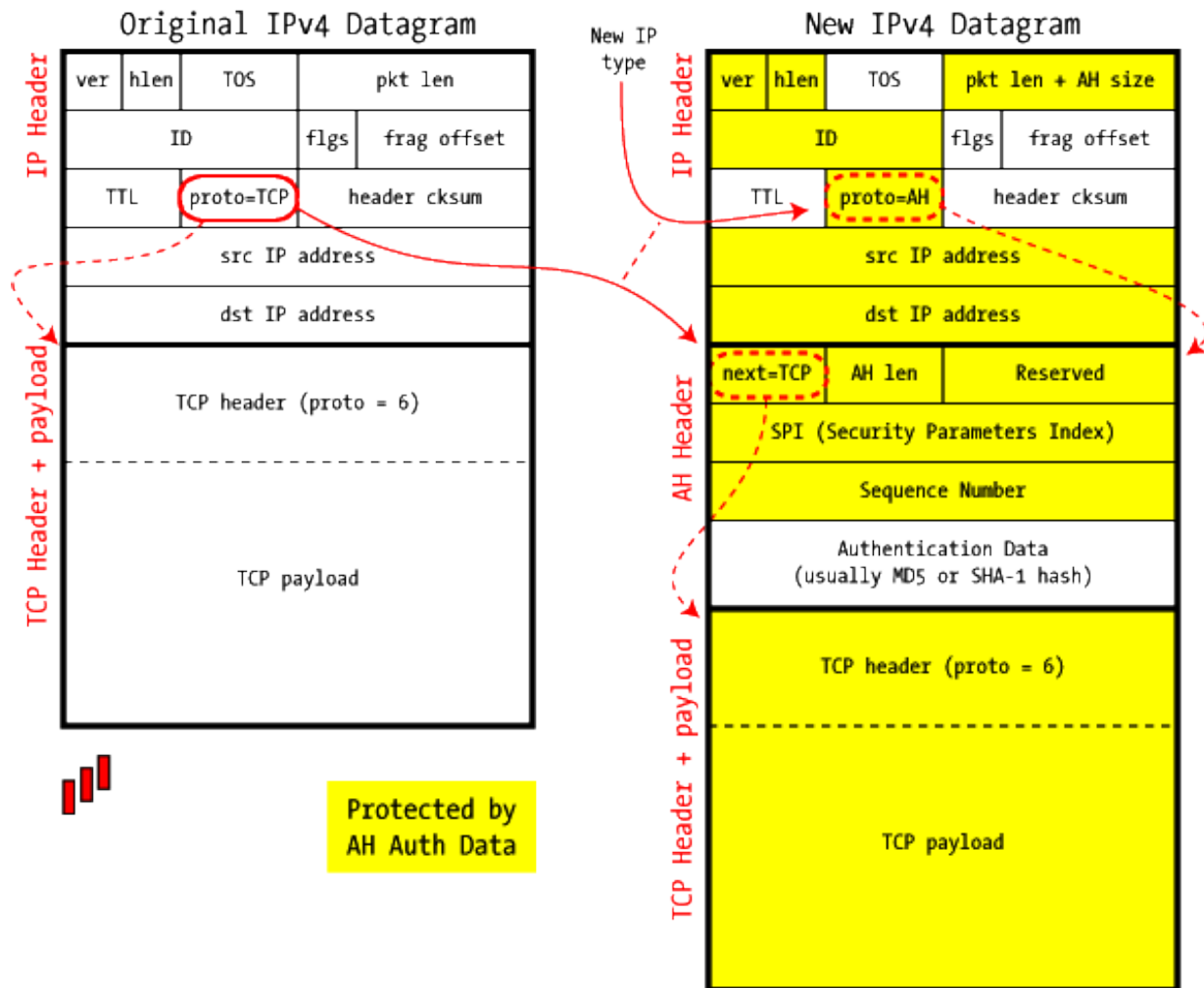


Figure 2.1: AH in Transport Mode [1]

IP packet is modified only slightly to include the new AH header between the IP header and the protocol payload (TCP, UDP, etc.), and there is a shuffling of the protocol code that links the various headers together. This protocol shuffling is required to allow the original IP packet to

be reconstituted at the other end. At receiving end once the IPsec headers have been validated, they're stripped off and the original protocol type (TCP, UDP, etc.) is stored back in the IP header.

Encapsulated Security Payload (ESP)

The Encapsulating Security Payload (ESP) header provides origin authenticity, integrity, and confidentiality of a packet. ESP operates directly on top of IP using IP protocol number 50. An ESP packet diagram in tunnel mode is shown in figure 2.2.

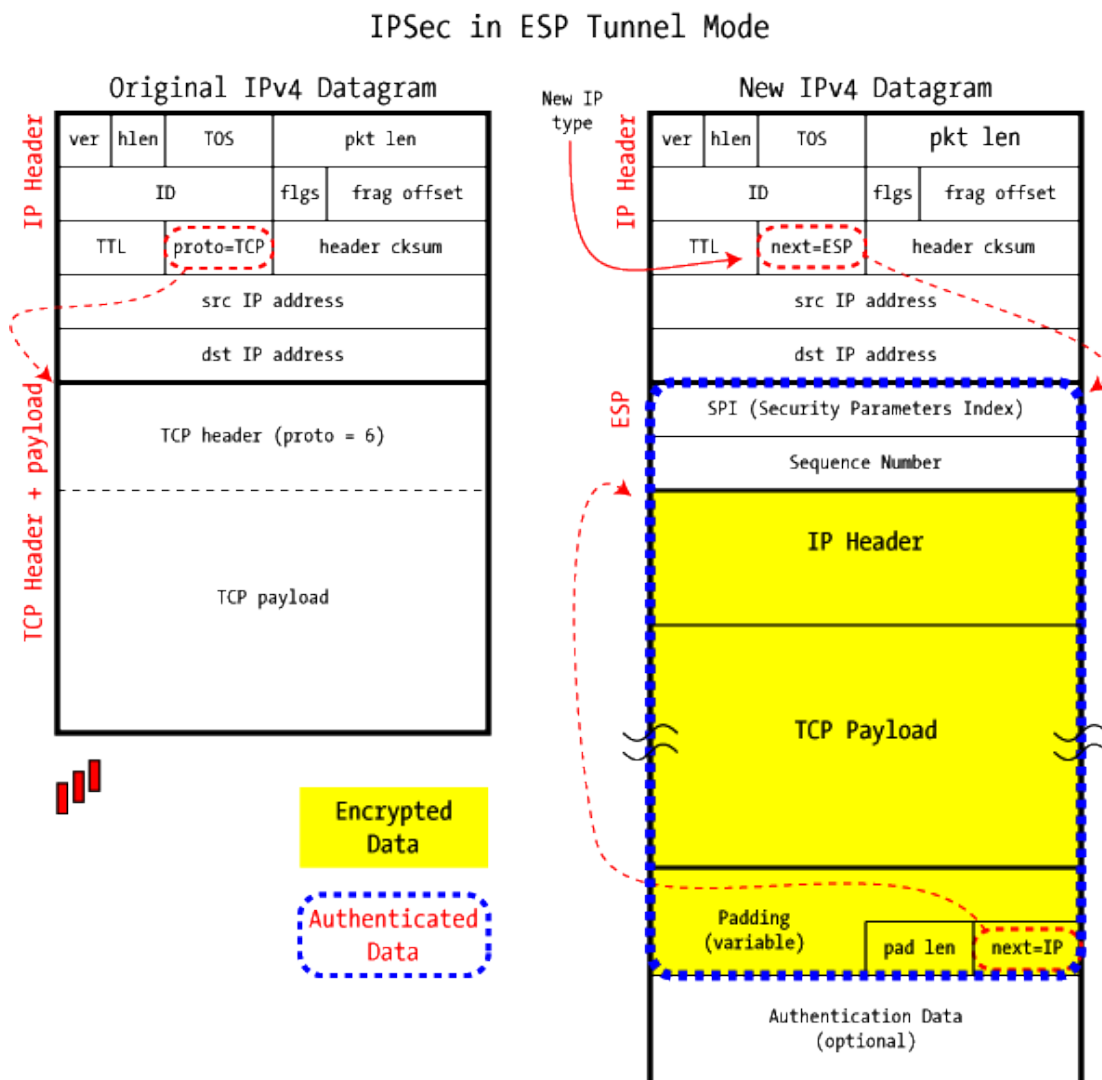


Figure 2.2: ESP in Tunnel Mode [1]

2.2 SSL

SSL uses cryptographic algorithms to encrypt data so that only the two hosts that are supposed to be able to read a message can actually understand it. There are two types of cryptography used within each SSL session, Symmetric and Asymmetric. While symmetric encryption is used for encrypting all the communications within an SSL session, an asymmetric algorithm is used to share the symmetric session key securely between the user and the SSL VPN.

Server Authentication

SSL Certificates are a mechanism by which a web server can prove to users that the public key that it offers to them for use with SSL is in fact, the public key of the organization with which the user intends to communicate. A trusted third-party signs the certificate thereby assuring users that the public key contained within the certificate belongs to the organization whose name appears in the certificate.

Client Authentication

SSL provides for authentication of clients to servers through the use of certificates. Clients present a certificate to the server to prove their identity. Although implementations of such technology are rare, it has special significance in the world of SSL VPNs as it allows SSL VPN servers to identify client machines of different trust levels.

IPSec works at network layer but SSL VPNs work differently. They establish connectivity using SSL, which functions beneath the application layer. It is important to realize that SSL is not strictly a web protocol. It functions at the session and transport layers of the OSI model and can establish encrypted communication tunnels for various application-level protocols that may sit above it. For example, although SSL encrypted web communication (HTTPS) is clearly the most common application of SSL, SSL encrypted **POP3** and **FTP** are utilized in many environments. It is possible to use SSL to encrypt effectively any application-level protocol.

2.3 Comparison between IPSec and SSL

- The advantage of IPSec - it eliminates the overhead caused by each channel. Since IPSec resides at IP layer, it secures the communication irrespective of the application. Thus key exchange and other mechanisms are not repeated for each session unlike SSL which is one connection per one session type. But it may turn out to be unfavourable if the key is compromised, as whole traffic is now vulnerable not just that session.

- IPSec keys are exchanged over UDP (port 500 only) whereas SSL clients are not bound to some specific port.
- IPSec suffers NAT traversal problem as it authenticates even IP header (through AH) and therefore routers cannot change source IP address in the outgoing traffic, as required by NAT.
- IPSec being a framework where hosts decide upon various encryption and authentication algorithms to be used, it doesn't integrate well among vendors. SSL is trouble free.
- IPSec has a high overhead in terms of header size (64 bytes, esp,ah tunnel mode) compared to SSL (21 bytes)
- SSL doesn't work with UDP, whereas IPSec avoids UDP problem by adding an IPSec header to the original packet's field. IPSec has a Sequence number field in its header, it uses this and other TCP mechanisms such as sliding window to facilitate connection-oriented communication.

2.4 IPSec in Linux-2.6

Security Associations & Security Association database

For the peers to be able to encapsulate and decapsulate the IPSec packets they need a way to store the secret keys, algorithms and IP addresses involved in the communication. All these parameters needed for the protection of the IP datagrams are stored in a security association (SA). The security associations are in turn stored in a security association database (SAD).

Security Policy & Security Policy database

The security associations only specify how IPSec is supposed to protect the traffic. Additional information is needed to define which traffic to protect when. This information is stored in the security policy (SP) which in turn is stored in the security policy database (SPD).

IPSec uses internet key exchange protocol (IKE) to exchange secret symmetric keys. This protocol authenticates the peers in the first phase. In the second phase the security associations are negotiated and the secret symmetric keys are chosen using a Diffie Hellmann key exchange. The IKE protocol then even takes care of periodically rekeying the secret keys to ensure their confidentiality.

Manual keyed connections using ‘setkey’

A manual keyed connection means that all parameters needed for the setup of the connection are provided by the administrator. The IKE protocol is not used to automatically authenticate the peers and negotiate these parameters. The administrator decides which protocol, algorithm and key to use for the creation of the security associations and populates the security association database (SAD) accordingly.

Transport Mode

Here we assume that two machines with the IP addresses 10.2.1.90 and 10.2.0.85 communicate using IPSec. The setup file on 10.2.1.90 looks like :

```
# AH
add 10.2.0.85 10.2.1.90 ah 15700 -A hmac-md5 "1234567890123456";
add 10.2.1.90 10.2.0.85 ah 24500 -A hmac-md5 "1234567890123456";

# ESP
add 10.2.0.85 10.2.1.90 esp 15701 -E 3des-cbc "123456789011245679012";
add 10.2.1.90 10.2.0.85 esp 24501 -E 3des-cbc "123456789011245679012";

spdadd 10.2.1.90 10.2.0.85 any -P out ipsec
    esp/transport//require
    ah/transport//require;

spdadd 10.2.0.85 10.2.1.90 any -P in ipsec
    esp/transport//require
    ah/transport//require;
```

The script first creates AH SAs and ESP SAs. The command **add** adds a security association to the SAD and requires the source and destination IP address, the IPSec protocol (ah), the SPI (15700) and the algorithm. The authentication algorithm is specified with -A (encryption using -E). Following the algorithm the key must be specified.

spdadd adds the security policies to the SPD. These policies define which packets are to be protected by IPSec and which protocols and keys to use. The command requires the source and destination IP addresses of the packets to be protected, the protocol (and port) to protect (any) and the policy to use (-P). The policy specifies the direction (in/out), the action to apply (ipsec/discard/none), the protocol (ah/esp/ipcomp), the mode (transport/tunnel) and the level (use/require). A similar file is created on the other peer also. Just the direction of the security policy is reversed (replace **-P in** with **-P out** and vice versa).

After executing the files on corresponding machines, all the traffic between the two peers will be encrypted. Complete details for setting up IPsec in Linux are given in Appendix 1.

Chapter 3

Trusted VPNs

Trusted VPNs are provisioned and managed by Internet service providers by defining paths through their networks to ensure that customers' traffic is routed over a trusted path. A customer might choose a trusted VPN because there is no equipment to buy, it's completely managed by the service provider and thus requires no maintenance, and they often include service-level agreements. Typically, trusted VPNs are less expensive upfront but more expensive over time. The general scenario of trusted VPN network is shown in figure 3.1

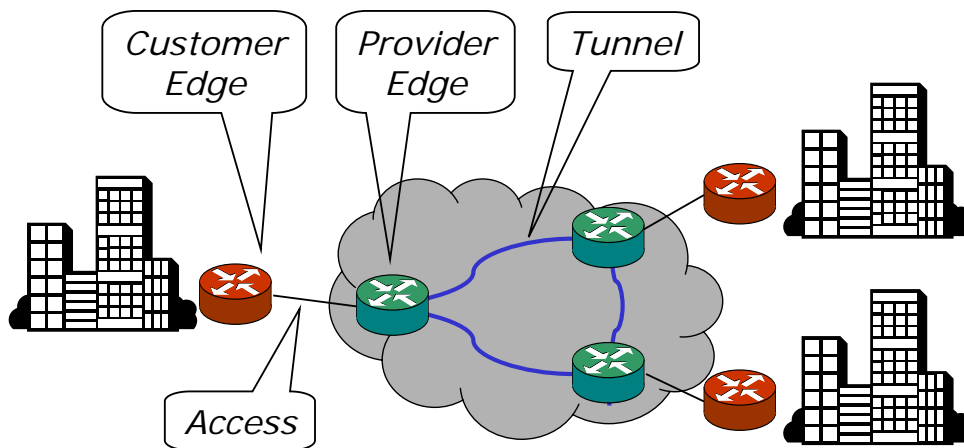


Figure 3.1: *Basic idea behind trusted VPNs*

A provider edge (PE) is a device that connects customers to the provider's backbone network. A PE can be an IP router or an MPLS label switch router (LSR). The backbone devices in the provider's network are called provider (P) devices, and they too can be routers or LSRs.

Provider Edge and Customer Edge based IP VPNs are site-to-site networks. In a PE-based IP VPN (or network-based IP VPN), all VPN configuration and management including VPN route

exchange (with CEs and PEs) and tunnel setup (to other PEs) is performed on the PE. The customer need only attach a particular site CE to the nearest provider PE via a LAN or WAN data link. The provider must then forward the packet to the correct VPN destination, be it another locally attached CE site or a separate PE or CE located elsewhere in the network.

We can subdivide PE-based IP VPNs into L3VPNs and L2VPNs. An L3VPN forwards packets based on the VPN customer's internal routing information that is, the packet header IP address. An L2VPN forwards packets based on layer-2 (MAC address, VC connection identifiers, and so on) or port information; the customer and provider exchange no customer routing information.

3.1 L2VPN

In a L2VPNs, the provider network emulates a learning bridge, and forwarding decisions are taken based on MAC addresses or MAC addresses.

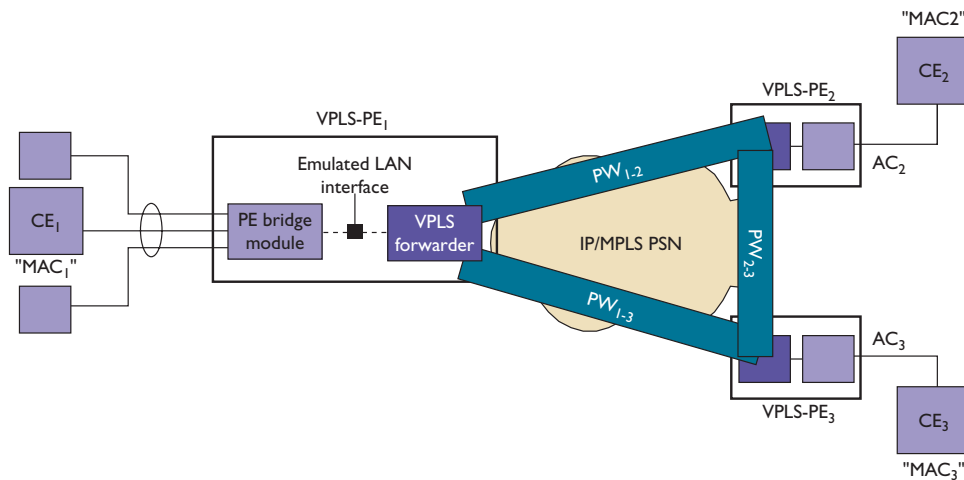


Figure 3.2: *Virtual private LAN service (VPLS) provider-edge (PE) nodes. These VPLS PE nodes contain a PE bridge module and a VPLS forwarder [6]*

3.1.1 Types of L2VPN

There are two types of L2VPN:

- **Virtual Private Wire Service (VPWS)**
A Virtual Private Wire Service (VPWS) is a point-to-point circuit (link) connecting two Customer Edge devices. The CE in the customer network is connected to a PE in the

provider network via an Attachment Circuit . The Attachment Circuit is either a physical or a logical circuit. The PEs in the core network are connected via a Pseudo Wire (PW).

- **Virtual Private LAN Service (VPLS)**

A VPLS is a provider service that emulates the full functionality of a traditional Local Area Network (LAN). A VPLS makes it possible to interconnect several LAN segments over a packet switched network (PSN) and makes the remote LAN segments behave as one single LAN. An example of how packet forwarding is done in L2VPN is discussed below.

3.1.2 Packet Forwarding

VPLS PE nodes support a LAN-bridging function between attached CE Ethernet sites and the emulated LAN. Figure 3.2 decomposes a VPLS PE (PE_1) to show how this bridging function is accomplished. A PE bridge module takes MAC frames from many different CE-facing Ethernet attachments and bridges them onto a PSN-facing emulated LAN interface (which is internal to the PE). The emulated LAN interface then presents itself as an AC to the VPLS forwarder. Each VPLS forwarder has a PW connection with every other remote VPLS forwarder that belongs to the same VPLS. A PE can support multiple VPLS instances. The VPLS forwarder (also called a virtual switching interface, or VSI) performs several important functions to emulate LAN functionality across the PSN. One such function is replicating and forwarding broadcast, multicast, and unknown unicast MAC frames. The VPLS forwarder inspects the MAC destination address (DA) of the unicast frame received from the emulated LAN interface. If there's no entry for this address in its forwarding table, the VPLS forwarder will replicate the frame and send a copy over a point-to-point PW to each of the remote forwarders defined in the VPLS. Broadcast and multicast MAC frames are automatically sent to each remote site.

For eg. if CE_2 wants to send a unicast MAC frame to CE_1 . It first transmits the MAC frame with a DA of "MAC₁". The VPLS forwarder in PE_2 doesn't have a "DA=MAC₁" entry in its forwarding table, so it replicates the frame and sends a copy over PW_{1-2} to PE_1 and PW_{2-3} to PE_3 . The MAC frame is not forwarded over PW_{1-3} because the PW full mesh invokes a **split horizon** to prevent loops. (A **split horizon** means that a packet received over one PW will not be forwarded over another one.)

3.2 L3VPN

An L3VPN interconnects sets of hosts and routers based on Layer 3 addresses. We explain routing and packet forwarding mechanism in L3VPN by an example.

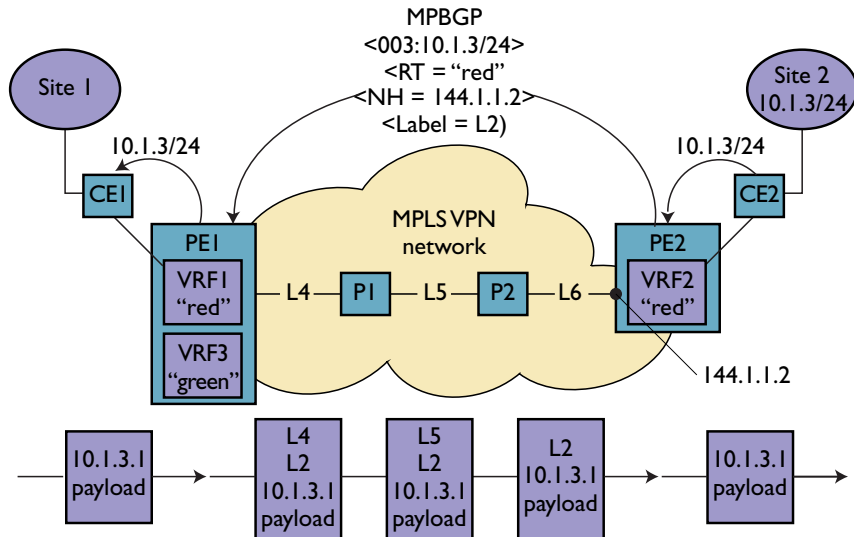


Figure 3.3: An MPLS virtual private network. MPBGP propagates VPN routes and labels between PEs. MPLS label switching is used to forward VPN data packets across the provider network. [5]

3.2.1 Routing Mechanism

Routing mechanism is quite complex in L3VPN for the reason that each PE should know the location of every customer based on its IP. If CE1 has to communicate with CE2, PE1 should know that CE2 lies in domain of PE2. For this CE2 first advertises address 10.1.3/24 at CE site 2 to PE2 via a dynamic routing protocol. The PE then converts this IPv4 address into a VPN IPv4 address and stores it in VRF2(Virtual Routing and Forwarding Instance) with an RD(Route Distinguisher) of 003. Next, MPBGP advertises the address (VPN route) with an RT(Route Target) equal to "RED", a label equal to L2, and the BGP next-hop address of 144.1.1.2, which is a loop-back address on PE2. When the MPBGP advertisement arrives at PE1, PE1 compares the RT value of the VPN route with RT values in VRF1 and VRF3. Because VRF1 is configured to import routes with RT = "red", it stores the VPN route, the label, and the associated BGP next hop. PE1 will then advertise this route to CE1 with its own address serving as the next hop.

Separate from this inter-PE message exchange, the PE and P devices automatically build a separate MPLS label-switched path(LSP) leading from PE1 to the PE2 loop-back address. This LSP is denoted by the tunnel labels on the PE and P data links. Now the MPLS VPN is ready to forward packets from CE1 to CE2.

3.2.2 Packet Forwarding

When a host at CE site 1 needs to send a packet to a host connected to 10.1.3/24 (say it's 10.1.3.1) at CE site 2, CE1 directs the packet to PE1. A route lookup in VRF1 yields the destination prefix (10.1.3/24), a label, L2, and a BGP next hop of 144.1.1.2. PE1 then checks with the default routing table (not part of any VRF) to figure out how to reach the next hop address. This lookup yields another label, L4. So, PE1 appends two labels, L2 and L4, to the packet, which it injects into the provider network toward PE2.

Each P device swaps the top label as the packet traverses the provider network along the LSP leading to PE2, leaving the VPN label alone. When the packet reaches PE2, the VPN label, L2, informs it that the packet should be forwarded to CE2. PE2 pops (removes) the VPN label and restores the packet to its original IPv4 format as it finds its way to CE2 and on to its destination.

We have talked a lot about MPLS when it comes to packet forwarding over leased lines. Now we discuss shortly about what exactly MPLS is.

3.3 Multiprotocol Label Switching (MPLS)

3.3.1 What is MPLS ?

It is a data-carrying mechanism which emulates some properties of a circuit-switched network over a packet-switched network. MPLS operates at an OSI Model layer that is generally considered to lie between traditional definitions of Layer 2 (data link layer) and Layer 3 (network layer), and thus is often referred to as a “Layer 2.5” protocol. It was designed to provide a unified data-carrying service for both circuit-based clients and packet-switching clients which provide a datagram service model. MPLS works by prepending packets with an MPLS header, containing one or more ‘labels’. Each MPLS label contains four fields (Figure 3.4)

- a 20-bit label value.
- a 3-bit field for QoS priority.
- a 1-bit bottom of stack flag. If this is set, it signifies the current label is the last in the stack.
- an 8-bit TTL (time to live) field.

These MPLS labeled packets are forwarded (switched is the correct term) after a Label Lookup/Switch instead of a lookup into the IP table. When a labeled packet is received by an MPLS router, the topmost label is examined. Based on the contents of the label a swap operation is performed on the packet's label stack. Routers can have prebuilt lookup tables that tell them which kind of operation to do based on the topmost label of the incoming packet so they can process the

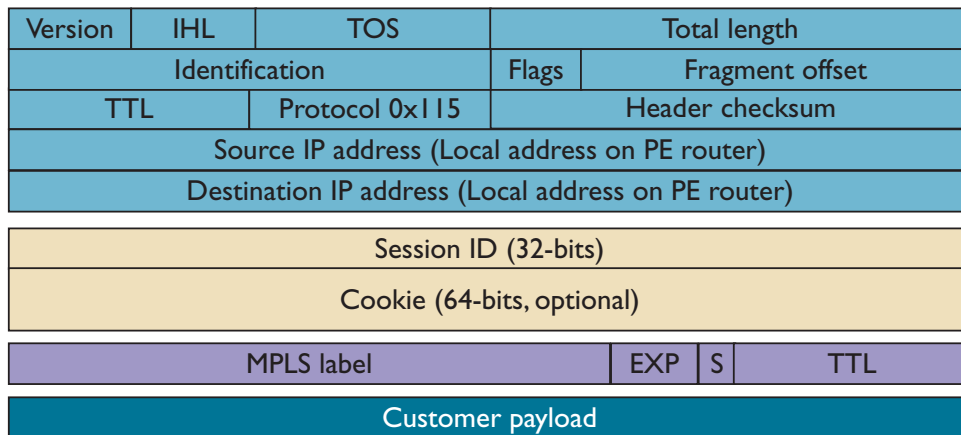


Figure 3.4: *MPLS-over-L2TPv3 encapsulation. The MPLS-labeled packet is encapsulated in an L2TPv3 header comprising an IP delivery header, session ID, and optional cookie [4]*

packet very quickly. In a swap operation the label is swapped with a new label, and the packet is forwarded along the path associated with the new label. Since labels that are swapped at routers are same for a flow, we can provide specific QoS for the whole flow based on Service Level Agreement (SLA). Protocols like Resource Reservation Protocol (RSVP) are used to arrange such arrangements for a flow.

But there are problems with MPLS especially when there are long distance private lines. In such case operating MPLS in the backbone requires extra work in configuring and managing the protocols that distribute labels. A provider with thousands of PEs would incur a substantial operational burden in carrying and managing all the host routes required by MPLS VPNs. Therefore long distance secure communication can be made by establishing and employing IP tunnels (rather than MPLS LSPs) to forward packets across native IP networks in support of MPLS VPN services. Thus solutions like GRE, L2TPv3 come into picture.

3.3.2 L2TPv3

One solution to the problem discussed in 3.3.1 that has recently emerged is to encapsulate MPLS packets in L2TPv3 (Layer-2 Transport Protocol, version 3) headers. Figure 3.4 illustrates the encapsulation of an MPLS packet in L2TPv3, which comprises several elements:

- The 20-byte **IP delivery header** contains the sending and receiving PE routers' IP addresses.
- The **session ID** is an automatically generated 32-bit number used to define individual services or service contexts on the egress PE router.

- The **cookie** is an automatically generated (optional) 64-bit random number that is associated with each session ID. It allows remote or receiving PE routers to quickly verify that each arriving packet was originated by a valid sending or source PE.

Chapter 4

Critics / Opinion / Findings

- Connecting IITs via VPN - We can connect IITs through VPNs, that will make communication between institutes secure. Since IITs work on a LAN methodology, we can implement IPsec at router/proxy server layer.
- Secure firewall + VPN
- SSL is simple and light weight but its under utilized, systems should support SSL by default so that basic application like Mail, Website access can be done securely.
- IPsec is complex framework which is difficult to scale due to various key exchange algorithms and headers. More work on standardizing the protocol should be done in order to make it easily compatible between different vendors, which is not the case now.
- In later stage we found that Linux kernel (>2.6) has an inbuilt support for IPsec. We can configure SADB and SPDB easily using shell commands (see Appendix). So small companies can save money/time by using this opensource instead of wasting on purchasing softwares.
- Browsers like firefox maintain database for Certificate Authorities, which are used to verify the certificates of web servers. Security of this database should not be compromised, as it will allow malicious certificates to pass through.

Chapter 5

Conclusion and Future Work

We analyzed the strengths and the weaknesses of the approaches like SSL and IPsec. SSL VPNs are now ready for prime time. In actual implementation projects, between 90% and 95% of an organization's remote users only require web access and e-mail. The percentage of users whose needs are met is even higher if some "standard" applications (as defined by the enterprise) are supported through plug-ins. There will be a continued need for IPSEC-based remote access VPNs in order to tunnel communication to unsupported applications, to serve clients on Macintosh and Linux, and for site-to-site VPNs. However, SSL VPNs offer near-equivalent functionality for most enterprise remote access users. By offering SSL as the default option to all remote users, and providing IPSEC VPN clients only to the few users who need non-standard application support, the enterprise can reduce the complexity of the overall remote access infrastructure, while enabling access from more places, including airport Internet kiosks and web.

As a necessary future work, we have to study secure firewalls so that if a company need to deploy both VPNs and firewalls, then they both could be supported. (MPLS PART 2 KA THOK DO)

Appendix A

Setting up IPsec in Linux 2.6

Here we show how to manually setup secure communication between two hosts. A large part of this process can also be automated, but here do it by hand so as to acquaint ourselves with what is going on ‘under the hood’.

Manual Keying

IPsec supports ‘Encapsulated Security Payload’ (ESP) for encryption and ‘Authentication Header’ (AH) for authenticating the remote partner. We can configure both of them, or decide to do only either.

Both ESP and AH rely on security associations. A security association (SA) consists of a source, a destination and an instruction. A sample authentication SA may look like this:

```
add 10.2.0.85 10.2.1.90 ah 15700 -A hmac-md5 "1234567890123456";
```

This says ‘traffic going from 10.2.0.85 to 10.2.1.90 that needs an AH can be signed using HMAC-MD5 using secret 1234567890123456’. This instruction is labelled with SPI (‘Security Parameter Index’) id ‘15700’. The interesting bit about SAs is that they are symmetrical. Both sides of a conversation share exactly the same SA, it is not mirrored on the other side. Do note however that there is no ‘autoreverse’ rule - this SA only describes a possible authentication from 10.2.0.85 to 10.2.1.90. For two-way traffic, two SAs are needed.

A sample ESP SA:

```
add 10.2.0.85 10.2.1.90 esp 15701  
-E 3des-cbc "123456789012123456789012";
```

This says ‘traffic going from 10.2.0.85 to 10.2.1.90 that needs encryption can be encrypted

using 3des-cbc with key '123456789012123456789012'. The SPI id is '15701'.

So far, we've seen that SAs describe possible instructions, but do not in fact describe policy as to when these need to be used. To do actual crypto, we need to describe a policy. This policy can include things as 'use ipsec if available' or 'drop traffic unless we have ipsec'.

A typical simple Security Policy (SP) looks like this:

```
spdadd 10.2.1.90 10.2.0.85 any -P out ipsec
esp/transport//require
ah/transport//require;
```

If entered on host 10.2.1.90, this means that all traffic going out to 10.2.0.85 must be encrypted and be wrapped in an AH authenticating header. Note that this does not describe which SA is to be used, that is left as an exercise for the kernel to determine. In other words, a Security Policy specifies **WHAT** we want; a Security Association describes **HOW** we want it.

Outgoing packets are labelled with the SA SPI ('the how') which the kernel used for encryption and authentication so the remote can lookup the corresponding verification and decryption instruction.

What follows is a very simple configuration for talking from host 10.2.1.90 to 10.2.0.85 using encryption and authentication. Note that the reverse path is plaintext in this first version and that this configuration should not be deployed. On host 10.2.1.90:

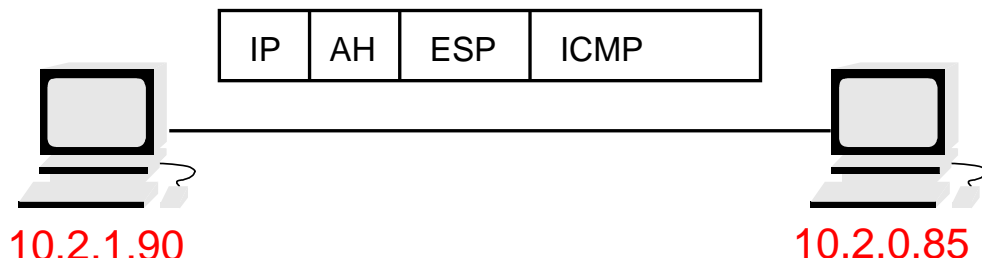


Figure A.1: *IPsec Transport Mode*

```
#!/sbin/setkey -f
add 10.2.1.90 10.2.0.85 ah 24500 -A hmac-md5 "1234567890123456";
add 10.2.1.90 10.2.0.85 esp 24501 -E 3des-cbc "123456789012123456789012";

spdadd 10.2.1.90 10.2.0.85 any -P out ipsec
esp/transport//require
```

```
ah/transport//require;
```

On host 10.2.0.85, the same Security Associations, no Security Policy:

```
#!/sbin/setkey -f
add 10.2.1.90 10.2.0.85 ah 24500 -A hmac-md5 "1234567890123456";
add 10.2.1.90 10.2.0.85 esp 24501 -E 3des-cbc "123456789012123456789012";
```

With the above configuration in place 'ping 10.2.0.85' from 10.2.1.90 looks like this using tcpdump:

```
22:37:52 10.2.1.90 > 10.2.0.85: AH(spi=0x00005fb4,seq=0xa): ESP(spi=0x00005fb5,seq=0xa)
(DF)
22:37:52 10.2.0.85 > 10.2.1.90: icmp: echo reply
```

Note how the ping back from 10.2.0.85 is indeed plainly visible. The forward ping cannot be read by tcpdump of course, but it does show the Security Parameter Index of AH and ESP, which tells 10.2.0.85 how to verify the authenticity of our packet and how to decrypt it.

The problem with the above configuration is that it contains policy on how 10.2.1.90 should treat packets going to 10.2.0.85, and that it explains how 10.2.0.85 should treat those packets but it does NOT instruct 10.2.0.85 to discard unauthenticated or unencrypted traffic! Anybody can now insert spoofed and completely unencrypted data and 10.2.0.85 will accept it. To remedy the above, we need an incoming Security Policy on 10.2.0.85, as follows:

```
#!/sbin/setkey -f
spdadd 10.2.1.90 10.2.0.85 any -P IN ipsec
esp/transport//require
ah/transport//require;
```

This instructs 10.2.0.85 that any traffic coming to it from 10.2.1.90 is required to have valid ESP and AH. Now, to complete this configuration, we need return traffic to be encrypted and authenticated as well of course. The full configuration on 10.2.1.90:

```
#!/sbin/setkey -f
flush;
spdf flush;

# AH
add 10.2.0.85 10.2.1.90 ah 15700 -A hmac-md5 "1234567890123456";
```

```
add 10.2.1.90 10.2.0.85 ah 24500 -A hmac-md5 "1234567890123456";

# ESP
add 10.2.0.85 10.2.1.90 esp 15701 -E 3des-cbc "123456789012123456789012";
add 10.2.1.90 10.2.0.85 esp 24501 -E 3des-cbc "123456789012123456789012";

spdadd 10.2.1.90 10.2.0.85 any -P out ipsec
esp/transport//require
ah/transport//require;

spdadd 10.2.0.85 10.2.1.90 any -P in ipsec
esp/transport//require
ah/transport//require;
```

And on 10.2.0.85:

```
#!/sbin/setkey -f
flush;
spdf flush;

# AH
add 10.2.0.85 10.2.1.90 ah 15700 -A hmac-md5 "1234567890123456";
add 10.2.1.90 10.2.0.85 ah 24500 -A hmac-md5 "1234567890123456";

# ESP
add 10.2.0.85 10.2.1.90 esp 15701 -E 3des-cbc "123456789012123456789012";
add 10.2.1.90 10.2.0.85 esp 24501 -E 3des-cbc "123456789012123456789012";

spdadd 10.2.0.85 10.2.1.90 any -P out ipsec
esp/transport//require
ah/transport//require;

spdadd 10.2.1.90 10.2.0.85 any -P in ipsec
esp/transport//require
ah/transport//require;
```

IPSec Tunnels

In the previous section we had seen IPSec in so called ‘transport’ mode where both endpoints understand IPSec directly. As this is often not the case, it may be necessary to have only routers understand IPSec, and have them do the work for the hosts behind them. This is called ‘tunnel mode’.

Setting this up is a breeze. To tunnel all traffic to 172.168.0.0/16 from 10.2.1.90 via 10.2.0.85, we issue the following on 10.2.1.90:

```
#!/sbin/setkey -f
flush;
spdflush;

add 10.2.1.90 10.2.0.85 esp 34501
-m tunnel
-E 3des-cbc "123456789012123456789012";

spdadd 192.168.0.0/16 172.168.0.0/16 any -P out ipsec
esp/tunnel/10.2.1.90-10.2.0.85/require;
```

Note the ‘-m tunnel’, it is vitally important! This first configures an ESP encryption SA between our tunnel endpoints, 10.2.1.90 and 10.2.0.85.

Next the actual tunnel is configured. It instructs the kernel to encrypt all traffic it has to route from 192.168.0.0/16 to 172.168.0.0/16. Furthermore, this traffic then has to be shipped to 10.2.0.85. 10.2.0.85 also needs some configuration:

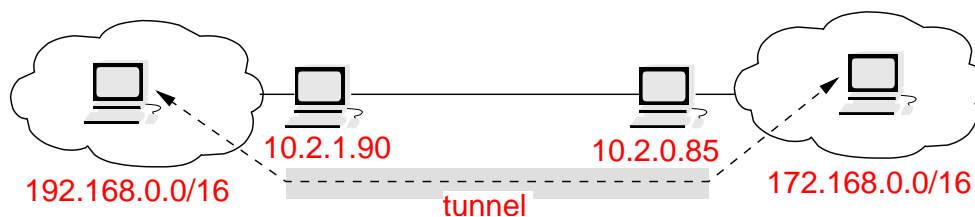


Figure A.2: *IPSec Tunnel*

```
#!/sbin/setkey -f
flush;
spdflush;
```

```
add 10.2.1.90 10.2.0.85 esp 34501
-m tunnel
-E 3des-cbc "123456789012123456789012";
```

```
spdadd 192.168.0.0/16 172.168.0.0/16 any -P in ipsec
esp/tunnel/10.2.1.90-10.2.0.85/require;
```

Another name for this setup is 'proxy ESP', which is somewhat clearer. Note that the IPsec tunnel needs to have IP Forwarding enabled in the kernel !

Appendix B

Glossary

1. **AC** - Attachment circuit (In a Layer 2 VPN the CE is attached to PE via an Attachment Circuit)
2. **BGP** - Border Gateway Protocol
3. **CE** - Customer edge (router in the customer network interfacing the provider network)
4. **DA** - Destination address
5. **L2VPN** - Layer-2 virtual private network (forwarding decisions are taken based on MAC addresses)
6. **L3VPN** - Layer-3 virtual private network (interconnects sets of hosts and routers based on Layer 3 addresses)
7. **LSR** - Label-switch router (routers supporting MPLS)
8. **MAC** - Media access control
9. **MPLS** - Multiprotocol label switching (used in circuit switched networks)
10. **PE** - Provider edge
11. **PSN** - Packet switch network (public IP network)
12. **PW** - Pseudo-wire (emulated point-to-point connection over a PSN)
13. **SA** - Source address
14. **VPN** - Virtual private network (secure communication over internet)
15. **VPWS** - Virtual private wire service (point to point Layer 2 VPN)

16. **VSI** - Virtual switching interface (performs standard LAN bridging functions)
17. **RD** - Route Distinguisher (8-byte value that, together with a 4 byte IPv4 address, identifies a VPN-IPv4 address family)
18. **RT** - Route Target (identify a set of VRFs)
19. **VRF**- VPN Routing and Forwarding(VRF is a per-site forwarding table)

Bibliography

- [1] An Illustrated Guide to IPsec, <http://www.unixwiz.net/techtips/iguide-ipsec.html>.
- [2] The official IPsec Howto for Linux, <http://www.ipsec-howto.org/>.
- [3] AbdelNasir Alshamsi and Takamichi Saito. A Technical Comparison of IPsec and SSL.
- [4] Brian Daugherty and Chris Metz. Multiprotocol Label Switching and IP, Part I: MPLS VPNs over IP Tunnels. *IEEE Internet Computing*, 9(3):68–72, 2005.
- [5] Chris Metz. The Latest in Virtual Private Networks: Part I. *IEEE Internet Computing*, 7(1):87–91, 2003.
- [6] Chris Metz. The Latest in Virtual Private Networks: Part II. *IEEE Internet Computing*, 8(3):60–65, 2004.
- [7] Chris Metz. Multiprotocol Label Switching and IP, Part II: Multicast virtual private networks. *IEEE Internet Computing*, 10(1):76–81, 2006.