IPSEC and the Internet

Manish Karir

December 7, 1999

This comment page is not part of the Thesis.

Typeset by LaTeX using the `umthesis` class by Vijay Bharadwaj, University of Maryland.

ABSTRACT

Title of Thesis:	IPSEC and the Internet
Degree candidate:	Manish Karir
Degree and year:	Master of Science, 1999
Thesis directed by:	Professor John S. Baras Department of Electrical Engineering

Secure and efficient communication between computers is becomming more and more essential as companies attempt to utilize the public network infrastructure for supporting communication between their various sites. The IPSEC protocols have been proposed as a solution to balance the needs of security and networking between computers. The basic IPSEC protocols are based on the end-to-end security model and when used in the most secure mode do not allow any intermediate nodes in the network to access and obtain information from packet headers encrypted by the security end-points. However, with the advent of smart applications in the middle of the network, which attempt to make it more efficient, a tradeoff is created between security and efficiency. This tradeoff is the result of the need for these intelligent applications

to access packet header information which is not possible with secure IPSEC flows.

This thesis analyzes and evaluates several possible solutions to this problem and argues why they all involve an unacceptable loss in the level of security or are not practical in any real system. On the basis of these arguments it then proposes the use of Layered IPSEC to solve the problem. Layered IPSEC adds felxibility to the current IPSEC protocols by providing the ability to use multiple encryption algorithms with separate encryption keys for different parts of a packet. We also describe an experimental implementation of the concept and provide timing measurements from it. On the basis of our experience with the implementation and our experimental measurements we argue for the feasiblity and usefulness of this scheme.

IPSEC and the Internet

by

Manish Karir

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
1999

Advisory Committee:

Professor John S. Baras, Chair
Associate Professor Leandros Tassiulas
Associate Research Scientist M. Scott Corson

© Copyright by

Manish Karir

1999

DEDICATION

To Mom and Dad.

ACKNOWLEDGMENTS

I am grateful to my advisor Dr. John S. Baras for his advice, support and encouragement. I would also like to thank Dr. Leandros Tassiulas and Dr. M. Scott Corson for agreeing to serve on my committee and to review this thesis.

Special thanks are due to Mingyan Liu, Vijay Bharadwaj, Steve Payne, Ravi Vaidyanathan, Brad Barrett, Narin Suphasindhu and numerous other collegues who contributed to a great working environment and influenced both my work and my personal life. I am also grateful to Tina Vigil, who was a source of constant help and support.

The research reported in this thesis was partially supported through contracts from Hughes Network Systems, Lockheed Martin Global Telecommunications and the Center for Satellite & Hybrid Communication Networks, a NASA Commercial Space Center (CSC), under NASA Cooperative Agreement NCC3-528. Their support is gratefully acknowledged. I am also grateful to HRL Laboratories for summer internship opportunities from which I learnt a lot.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Background and Motivation

Security and efficient communication are two opposing concepts. Ideally the only secure computer secure system is one that is disconnected from everything else and sits in an empty room. The need to network and communicate between computer systems is becomming more and more important as networking adds functionality and efficiency to a computer system. Therefore, there is a tradeoff in todays networks between security and the extent to which communication can take place. This must be studied carefully by a network engineer in order to create a secure and efficient system. However, even at the onset of such a study, the network designers or administrators should have well defined limits to which they are willing to sacrifice security in order to add functionality and efficiency to their networks.

Several schemes have been proposed to balance security and communication between computers. Transport Layer Security(TLS), Secure Shell(ssh), Secure

Sockets Layer(SSL), S/MIME Mail Security and Pretty Good Privacy(PGP) are some such solutions. Each solution offers a different level of security. However, one approach which has the potential to replace all preceding schemes is the use of encryption and authentication at the network layer(IP). Security service at the IP layer or IPSEC is currently undergoing standardization by the IETF. What makes this proposal much more attractive than some of the other solutions is the fact that being at a network layer compared to the transport layer or even the application layer, makes it possible to use the same solution for multiple applications and over various transport layer protocols. The fact that IPSEC provides security at the network layer is what makes it so versatile.

IPSEC can be used to provide secure virtual pipes for applications, a pair of computers or even entire LAN's. The IPSEC model of security advocates the use of end-to-end security between any two communicating hosts. The end-to-end paradigm can be applied, in the case of virtual private networks(VPN's), to end-to-end security between security gateways which provide IPSEC services to the entire Local Area Network(LAN). In this scenario, each segment of a VPN has its own IPSEC capable gateway on its boundary with the public network.

IPSEC provides several methods of providing different levels of security. It can be used to provide simple authentication or encryption or both. While authentication provides a simple method to verify the originators of IP packets, encryption provides the ability to securely transmit packet payloads as well. While both can be used by different applications, we will study the use of

encryption more closely as it provides maximum security.

The end-to-end security model by definition does not permit any intelligence to be built into the network which might need to access information from encrypted packet headers. On the other end of the spectrum is the school of thought that advocates building intelligence into the network in order to enhance its efficiency and functionality. Such intelligence, it is argued, can improve network performance, reliability, and reduce network management overheads. This is the primary philosophy guiding the development of proxies, compression agents, active networks and load distribution systems.

Such intelligent systems built into the network, can be rendered useless by the use of end-to-end security, as they often rely on the ability to access information contained in packet headers. Therefore, several proposed and implemented enhancements to the basic Internet protocols loose their functionality and the performance of the network degrades.

This makes us question the viability and usefullness of the end-to-end security promised by IPSEC. Does high level of security have to mean giving up on efficient network performance? Is any mechanism which advocates this an acceptable solution? The situation is further complicated by the fact that mechanims such as firewalls and packet filters themselves rely on information within packet headers to implement security. Therefore, if the packet has been IPSEC encrypted for security, then firewalls can no longer access the information they need to identify packets to which they should apply their security rules. The

use of one tool for providing security has, in this case, hindered the performance of another tool for providing security.

Another scenario in which the end-to-end security model can have a significant impact is in the use of traffic analysis tools. Traffic analysis tools are of fundamental importance to network planners and engineers. These tools can help them make important decisions regarding network design and upgrades. The end-to-end model is specifically designed to prevent traffic analysis and observation by any third party in the middle of the network. However, though network analysis when performed by an unauthorized user can be a serious security breach, its use is often essential for authorized network designers, planners and administrators. Traffic analysis tools, also rely on the ability to access packet header information. Therefore, these tools are not going to be able to access IPSEC encrypted traffic flows and include them in their analysis leading to incorrect or invalid results. Should the solution to keeping out unauthorized use of a service be to disable the service completely?

Limitations and problems introduced by the use of IPSEC flows as defined today will be severely compounded with the introduction of IPv6 which makes the use of IPSEC mandatory, unlike IPv4, where the use of IPSEC is optional.

1.2 Contributions and Organization

There are currently several implementations of IPSEC available for both IPv4 as well as IPv6. While people in the Internet community have acknowledged the problems associated with the use of IPSEC, up till recently there has been no attempt to develop a system to overcome them. This thesis represents one of the first such efforts to analyze the problem and to propose, develop and implement a working prototype of a possible solution. Our goal is to highlight and emphasise the limitations of the IPSEC model as it currently stands and to offer suggestions on how greater flexibility can be added with only slight modifications and without significantly impairing the level of security offered.

In this thesis we examine the issues related to supporting adequate security levels in the Internet and at the same time incorporating the ability to support applications that require information within packet headers. We demonstrate our implementation of such a system and perform experiments to explore its feasiblity and practicality. While there has been some work in analyzing the cryptographic security offered by the IPSEC protocols, in this thesis we do not attempt to make any similar analysis [1]. This thesis does not attempt to analyze IPSEC protocols from a cryptographic point of view but more from a networking point of view. This approach has only recently been used in [2] in an attempt to develop a classification scheme, or model for devices that use layer violations as well as security.

The remainder of this thesis is organized as follows. In chapter 2 we give a brief introduction to IPSEC to provide the necessary background. Chapter 3 provides a discussion of the problems created by the use of traditional IPSEC in different scenarios. Chapter 4 discusses various possible solutions to overcoming the problems described in Chapter 3, and their pros and cons. Chapter 5 provides details of our proposed modifications to the traditional IPSEC protocol followed by a description of its implementation in Chapter 6. Chapter 7 gives some results from experiments on our implementation and finally, Chapter 8 gives conclusions and offers suggestions for future work.

Chapter 2

IPSEC

In this chapter we briefly describe the key features and properties of IPSEC, and explain how the current specification imposes certain constraints and limitations.

2.1 Overview

IPSEC [3] is a framework of open standards that describe a method of providing data confidentiality, data integrity, and data authentication between participating peers. It can be used to protect one or more data flows between a pair of hosts, between a pair of security gateways, or between a security gateway and a host. It can allow data to be sent across a public network without its contents being observed or modified by a third party. This creates the ability to support several applications such as secure Virtual Private Networks, and secure remote user access [4] [5] [6] [7].

IPSEC differs from other methods of providing security as it provides the security service at the IP layer. It uses the Internet Key Exchange Protocol(IKE)

to handle negotiation of protocols and algorithms based on local policy, and to generate the encryption and authentication keys to use.

As IPSEC is a standard it creates the opportunity for secure devices and implementations from various vendors to interoperate. In this context it represents the first large scale effort from the Internet community to consider and impose the use of security. In fact, the IPv6 specification requires that IPSEC use be mandatory [8].

Providing security services consists of providing the following services. Data authentication and integrity, data confidentiality, replay protection and an automated mechanism for the management of cryptographic keys and security associations.

The IETF has defined the following protocols to address each of these security services:

- IP Authentication Header(AH): Provides data origin authentication, data integrity, and replay protection.
- IP Encapsulating Security Protocol(ESP): Provides data confidentiality, data origin authetication, data integrity, and replay protection.
- Internet Security Association and Key Management Protocol(ISAKMP): Provides a method for automatically setting up security associations and managing the cryptographic keys.

In the sections that follow, we will examine each of these in greater detail but

first we need to examine the motivation behind the development of IPSEC.

2.1.1 Why use IPSEC?

Various security protocols have been defined for different applications in the Internet. Secure Sockets Layer(SSL) for web traffic, Secure Shell(SSH) for remote logins, Pretty Good PRivacy(PGP) for email, etc. However, they all have the drawback of behind application specific. If a new application is built, a new security mechanism will have to be built as well. This was the reasoning behind the development of Transport Layer Security(TLS). However, even this proves to be insufficient as it cannot accomodate UDP based applications. Moreover, TLS also exposes important transport layer header information which can be used to mount sophisticated security attacks such as IP spoofing and denial of service attacks.

Therefore, the chief reasons for using IPSEC instead of the various other security protocols mentioned earlier are [9]:

- Transparency: As IPSEC is implemented at the network layer, it provides complete transparency to all applications. IPSEC can work with all TCP and UDP applications alike, including all HTTP, FTP, Telnet and SMTP applications. It can even be used in addition to application layer security protocols such as SSL.

- Security: It generally provides better protection against traffic analysis,

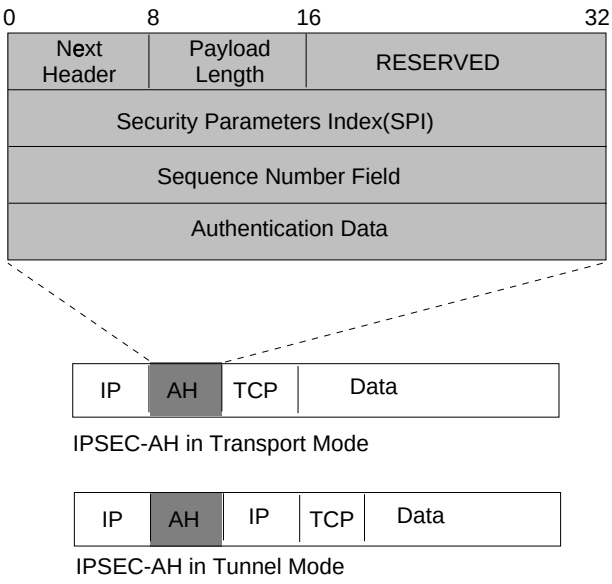

Figure 2.1: IPSEC Authentication Header

denial of service and IP spoofing attacks compared with other application layer or transport layer based mechanisms.

- Independent of Networking Topologies: As IPSEC works on IP packets, it can be used in conjunction with any networking topology such as Ethernet, TokenRing and PPP.

- Standards Based: As IPSEC is standards based, it offers the opportunity for various implementations from different vendors to interoperate.

2.2 IP Authentication Header: Data Authentication and Integrity

Data authentication and integrity deals with ensuring that the data received is the same as the data that was sent and that the claimed sender is in fact the actual sender. This mechanism requires that any tampering with the data in transit be detected and the packet be silently discarded.

The IPSEC specification provides a mechanism to implement data authentication via the use of an Authentication Header(AH) [10]. The security is provided by adding authentication information (to the IP datagram) which is calculated using all of the fields in the IP datagram (including not only the IP header but also the other headers and the user data) which do not change in transit (for instance hop-count field in IPv6, and time-to-live field in IPv4, headers cannot be included as they are altered at each relay the datagram passes through). In IPv4, AH is placed immediately following the IPv4 header and before the information being authenticated. The modified header of a packet using AH is shown in Figure 2.1. The fields of an AH are also shown in the figure. The Security Parameter Index (SPI) field is an arbitrary 32 bit that in addition to the destination IP address and security protocol specifies the Security Association for each datagram. The Sequency Number field is used to provide protection for replay attacks. The Authentication Data field is a variable length field which contains the Integrity Check Value(ICV) for each packet. The

computation of this field depends on which algorithm is being used for authentication MD5, SHA etc. It is essentially a "checksum" type of computation.

AH can be used in either transport mode or in tunnel mode. In transport mode, the original IP header is maintained and an additional authentication header is added in between the IP header and the data. In tunnel mode, a new IP header is generated, the AH header is then placed after this new header. The AH header in this scenario protects the entire new IP packet. Therefore, any change to any fields in the new packet can be detected even though the packet contents are in cleartext form.

Therefore, AH can be used to provide per packet integrity and data origin authentication for IP datagrams as well as protection against replay. Data integrity is assured by "checksum" generated by a message authentication algorithm such as MD5; data origin is assured by including a secret shared key in the data to be authenticated(SPI); and replay protection is provided by use of a sequence number field within the AH header.

Various transform algorithms other than MD5 can also be used in AH. The most common algorithms specified by the IETF are: MD5 [11], Secure Hashing Algorithm(SHA) [12], Hashed Message Authentication Code MD5(HMAC-MD5) [13], and Hashed Messaged Authentication SHA(HMAC-SHA). All systems claiming to be IPSEC compliant must at least implement AH in MD5 mode.

When the destination IPSEC end-point receives an AH packet, it uses the

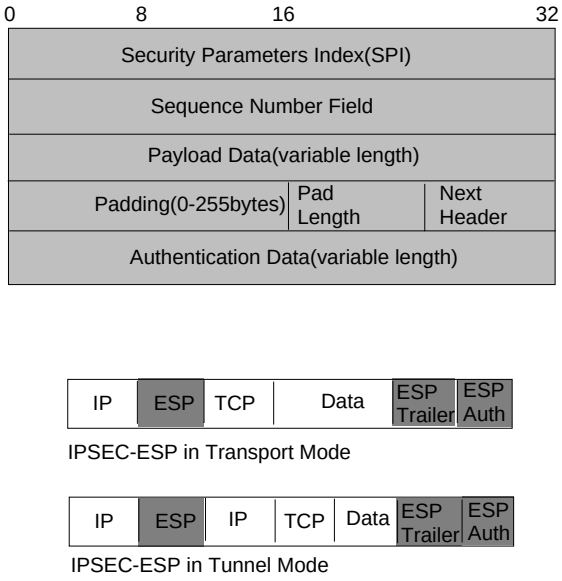

Figure 2.2: IPSEC Encapsulating Security Payload

information in the AH to determine if the packet has been modified during transmission. If it is unable to verify the packet, the packet is discarded.

The security provided by AH might be useful for some services requiring only "weak" security. However, as we are considering primarily applications which require that not only should the data not be modified during transit, but that they also be unreadable to any third party, we focus our attention on the ESP service of IPSEC.

2.3 IP Encapsulating Security Header: Data Confidentiality

Data confidentiality deals with ensuring that only the sender and the receiver are able to read the data being transfered between the two. All intermediate systems cannot determine the contents of the packets that might be passing through them. This functionality is accomplished via the use of encryption and a secure key distribution mechanism.

The IPSEC specification provides a mechanism to implement data confidentiality services via the use of Encapsulating Security Payload(ESP) [14]. There are two ways of providing the ESP service. ESP can be used in tunnel mode or transport mode. In tunnel mode the entire original IP packet is encrypted and placed within another IP packet created by the security gateway. In transport mode the original IP packet headers are maintained while the payload of that packet is encrypted. When ESP is used, the protocol header(IP) immediately preceding the ESP header contains the value 50 in its Protocol (IPv4) or Next Header (IPv6) field.

The format of the ESP header is shown in Figure 2.2. The SPI field is a 32 bit random value that along with the destination IP address, specifies the security association for this packet. The sequence number field is used to prevent replay attacks. The padding fields are used to add padding data to allign it if necessary for encryption. The next header field identifies the type of data contained in the

payload field, and the authentication data field contains an Integrity Check Value(ICV) computed over the the entire ESP except the authentication data.

Various encryption algorithms may be used with ESP. The most common algorithms specified by the IETF are: ESP Cipher Block Chaining(CBC) mode of the US Data Encryption Standard(DES-CBC) [15] and the Triple DES CBC [16]. The RC5-CBC encryption algorithm has been proposed by RSA and is currently an IETF draft. All systems claiming IPSEC compliance must implement ESP with at least the DES-CBC algorithm. MD5 or SHA may be used for computation of ICV.

During transmission, an IPSEC-ESP end-point encrypts the payload of each packet using the chosen encryption method. The receiving IPSEC-ESP end-point, decrypts the payload and forwards the packet to its final destination.

As the use of ESP provides maximum level of security, for the rest of this thesis, we will consider the use of IPSEC primarily in ESP tunnel mode.

2.4 Security Associations and Key Management

In order to use either the AH or ESP mode security, we must agree on how they are going to be used. Security Association (SA) is a set of security information relating to a given network connection or set of connections. The concept of a SA is fundamental to both the IP ESP and the IP AH. A security association identifies the cryptographic algorithm to be used, the keying information, the

identities of the participating parties, etc. A security association is unidirectional, therefore, protecting a bi-directional communication requires two associations. Each SA is uniquely identified by the destination address in an IP packet, a security protocol identified (AH or ESP) and a Security Parameter Index (SPI) which is a 32 bit block in the header of an exchanged packet. The SPI is choosen by the receiver and it is transmitted in the clear.

Security Associations rely on key exchange between communicating parties in order for the authentication and encryption algorithms to be used with AH and ESP. The most common ways for key exchange currently are:

- Manual Key Exchange: This is the simplest and currently the most widely used method of key distribution. In this scheme, a person manually configures each system with its own key as well as the keys of the other communicating systems. This works quite well for small static environments however is not scalable.

- Simple Key Interchange Protocol (SKIP): This key management scheme was proposed by SUN Microsystems, however the IETF chose to use the Internet Security Association and Key Management (ISAKMP) algorithm instead of SKIP for IPv6.

- Internet Security Association and Key Management Protocol (ISAKMP) and Oakley: The Protocol has been picked as the standard for IPv6 and as an option for IPv4. By itself ISAKMP does not establish session keys,

however it can be used with various session key establishment protocols. Oakley is one such key establishment protocol chosen by the IEFT. Oakley Key Determination Protocol uses a Diffie-Hellman technique to establish session keys on Internet hosts and routers.

Chapter 3

Problem Description

The IPSEC specification requires that when end-to-end encapsulated security is used, all protocol headers below the IP layer must be encrypted. While this is the most secure method of providing security at the network layer, it does however introduce several problems for legitimate users or applications who wish to observe protocol headers below the IP header in the middle of the network. In this chapter we provide detailed descriptions of several such applications/scenarios. We only describe some of the applications here, so as to present a general feel for what the problems are in the different cases. Other possible applications and uses, such as [17], are left out because of their similarity to the ones described here.

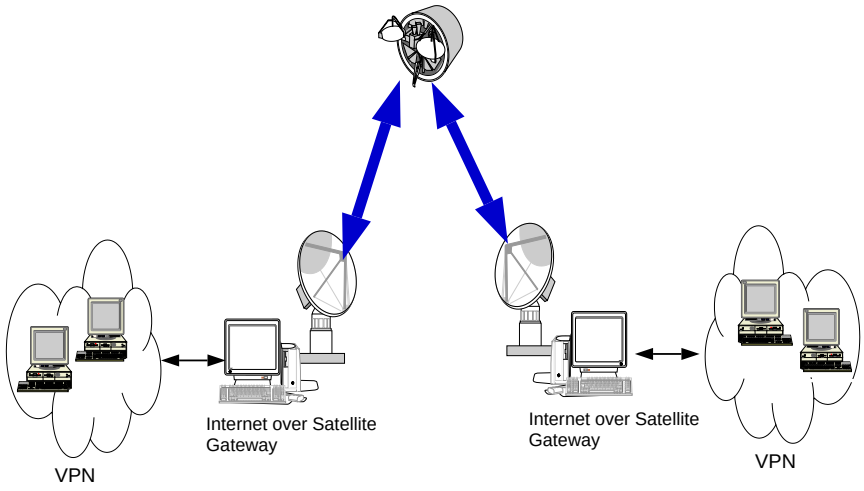

Figure 3.1: A Typical Internet over Satellite Architecture

3.1 Case 1: Internet over Satellite

3.1.1 Background

The first application we consider is the use of TCP connection splitting to enhance the performance of TCP over satellite. But first we need to justify the need for such an application by quantifying the benefits of its use. Therefore we first attempted to obtain simulation results to justify the usefulness of TCP connection splitting in Internet over satellite.

A typical architecture for providing Internet over satellite is shown in Figure 3.1. It consists of two LANs connected across a satellite link via gateways. Various solutions have been proposed to overcome the problem of running conventional Internet protocols over the high delay network topology. These include the use of proxies, protocol enhancements, as well as the use of new protocols. While the most efficient solution is to use a new, specially designed

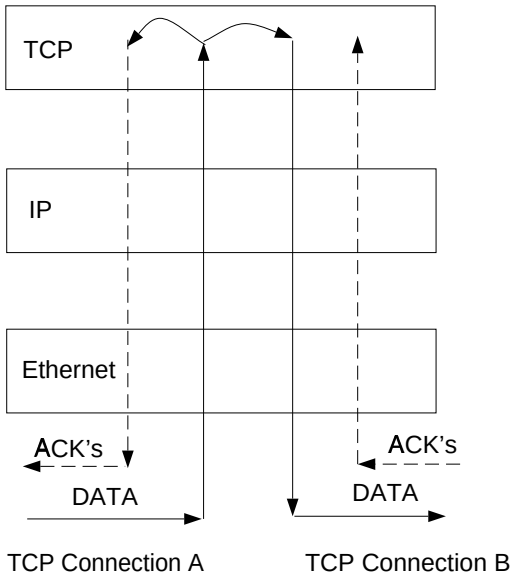

Figure 3.2: TCP Connection Splitting

protocol suite which keeps in mind the heterogeneous nature of today's networks, such a solution is impractical due to the wide installation base of the current protocols. Therefore, the alternative solution, which has been shown to work quite efficiently, is to hide the presence of such links from the protocols via the use of proxies or enhanced gateways [18] [19]. These enhanced gateways implement several protocol enhancements and modifications to improved the end-to-end performance of traffic flows through them. They use TCP enhancements such as window scaling, fast recovery and fast retransmit, Selective Acknowledgements (SACK) and Forward Acknowledgements (FACK), as well as TCP connection splitting.

TCP connection splitting is conceptually illustrated in Figure 3.2. The end-to-end connection between end-hosts is split into seperate TCP connections,

from end-hosts to gateways, and in between the gateways.

In an effort to gauge the usefulness of this techique, we performed OPNET simulations on different scenarios. We studied four simulation scenarios: terrestrial network, satellite network without enhanced gateways, satellite network without enhanced gateways but with large windows used end-to-end, and finally satellite network with enhanced gateways. The point-to-point link between the two gateways was set at a data rate of a T1 link. A unidirectional delay of 250ms is added to the link when simulating a satellite link. In the cases where the large windows option has been used, the receive TCP buffers are configured to 195K (default is 64K). This value is chosen as it equals twice the bandwidth-delay product (2*(1.5M(bw)*0.5(rtt)). The window scale factor chosen is 3. In all cases a ftp file transfer is studied. A 50M file is transferred from the server to the client.

One of our primary performance criteria was the throughput on the satellite link. Figure 3.3 shows relative performance of different simulation scenarios. The throughtput was measured on the link between two gateways. In the terrestrial case there was no delay on that link, therefore throughtput was high(close to the capacity of the T1 link). When a 250ms delay was introduced on this link the observed throughtput dropped to 500Kbps (about a third of the capacity of the T1 link). When the large windows option was introduced on the client and server, the observed throughtput improved to 800-900Kbps. Finally connection splitting was enabled on two gateways, and the large windows option was used

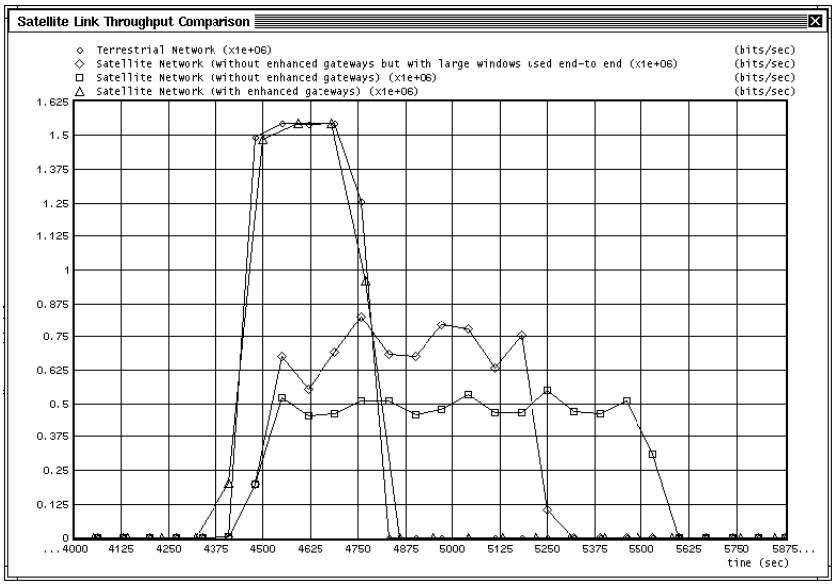

Figure 3.3: Satellite Link Throughput Comparison

just between them. In this case the throughput on the satellite link was again close to the capacity of the T1 link. This clearly shows the benefit of connection splitting and the use of TCP enhancements on the satellite gateways.

The performance of TCP is highly sensitive to the measured round trip time. This is the primary reason why TCP performance suffers when a satellite link is part of the network. Figure 3.4 shows the round trip time measured by the server for different simulation scenarios. In the simple terrestrial case it is close to 150ms. In both cases when the gateways are not used, the presense of the satellite link causes the round trip time to increase to be slightly greater than 500ms. However, when the enhanced gateways are used, the round trip time measured by the server is only 75ms. This implies that the use of the enhanced gateways has removed the negative impact of the satellite link in TCP

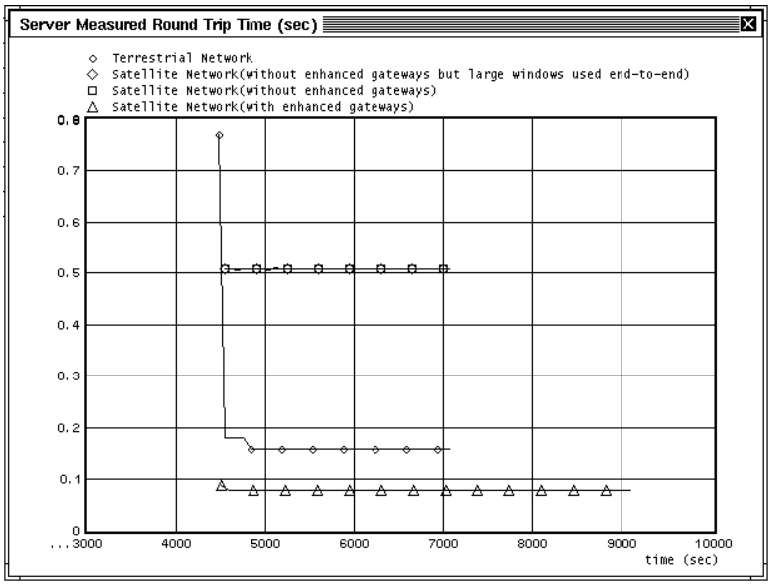

Figure 3.4: Observed RTT Comparison

performance. As the observed round trip time measured by both the server and the client is short they also recover faster from packet loss.

3.1.2 The Problem

Therefore, we see from the results in the previous section that the performance impact of the use of the enhanced gateways and TCP connection splitting in Internet over satellite is significant. However, the implementation of TCP connection splitting relies on the ability to read the TCP header of all incomming packets. When a security mechanism such as IPSEC is used to provide end-to-end security, the TCP header is encrypted. Therefore, the enhanced gateways are unable to perform connection splitting on these secure flows, as they do not have access to the information in the TCP headers of these packets.

This implies that IPSEC flows cannot benefit from the performance gain provided by TCP spoofing gateways. This problem is expected to become an important factor in both the rate at which IPSEC is adopted in the Internet, as well as the acceptability of TCP spoofing as a performance enhancing solution for providing Internet over satellite.

VPN's involving a satellite link can choose to use IPSEC security, in which case their throughput over the satellite link suffers, or the VPN's can choose not to use IPSEC secure flows in order to benefit from the higher network performance provided by TCP connection splitting. The trade-off is extreme, as both high performance and security are desired essentials of a network.

3.2 Case 2: Firewalls

3.2.1 Background

The second application we consider which requires the ability to correctly access packet headers is the use of firewalls. When a private network is connected to the public network, any and all users from the public network can access the private network. This creates a dangerous situation from the point of view of security of the private network. A firewall is a device which can be used to control access from the public network into the private network. Firewalls are used by almost all major companies who have Internet access.

The basic assumptions that must be made for a firewall to work are:

- All traffic from private network to public network and vice versa must pass through the firewall;
- The firewall itself is secure.

The primary function of the firewall is to allow only selected traffic to pass through it and the rest is blocked. This can be a very effective solution in providing security to the private network, as all incomming traffic from the public network is monitored and suspicious or malicious content can be filtered out. The firewall administrator has to define certain rules or tags which describe suspicious content. Often, defining the security policy for a firewall is the network administrators most difficult task.

There are primarily two different kinds of firewalls in use today [20] [21]:

- Packet-Filtering Firewall: A packet-filtering firewall applies a set of rules to each incomming packet and decides whether that packet should be forwarded or discarded. The filtering rules are based on fields in the IP and transport layer headers including IP source and destination addresses, IP protocol field, and TCP and UDP port numbers.
- Application-Level Gateway: An application-level gateway or proxy server, acts as a relay of application level traffic. The user contacts the gateway using a TCP/IP application, the gateway then in turn contacts the remote

host on behalf of the user, and then transfers application data from the remote host to the user. In effect there are two spliced connections between the remote and the gateway and between the user and the gateway. This kind of firewall is more secure than a packet filtering firewall, as this type of firewall allows traffic for specific applications, which it proxies, only. However, there is a large processing overhead in examining traffic on both connections.

3.2.2 The Problem

While firewalls can be configured to provide various degrees of security, the more secure packet-filtering firewall configuration requires that packets containing data to or from certain ports not be allowed to pass though. This requires that the firewall have access to the transport layer headers. However, with the use of IPSEC, firewalls can become ineffective. When IPSEC is used, information required by packet-filtering firewalls is no longer visible to the firewall, therefore the firewall can make incorrect decisions.

There is however an underlying assumption in the above argument. The assumption is that the IPSEC encryption is performed prior to the firewall. Most commercial firewalls solutions available today which claim to be IPSEC compliant perform the firewall functionality before the encryption. One must note that this is not a desired solution as it is open to a very serious security attack. In these

systems, all incomming packets are decrypted, then passed through the firewall. An attacker can simply capture a legitimate packet and then use it to mount a denial-of-service attack on the firewall by transmitting fake packets which will needlessly go through IPSEC processing before being dropped by the firewall. As IPSEC processing is CPU intensive it would be quite easy for the attacker to load the IPSEC processing system, thereby denying service to legitimate packets.

Moreover, it is also desirable to place the IPSEC processing system inside the firewall for the following reasons:

- Bottleneck Avoidance: IPSEC processing can be distributed among several systems thereby enabling the link from the firewall to the public network to be more fully utilized.

- Security: A firewall is supposed to represent the entry point for all external traffic into the private network. This is the most secure method of controlling access to the private network. The use of IPSEC is only supposed to make the network more secure, however most current solutions place the potentially most important computer system, the one that holds the keys to all encrypted traffic, outside the firewall. This makes the IPSEC system itself much more vulnerable to attacks.

Some might argue that firewalls are no longer required as IPSEC ensures that transmitted and received data are secure and encrypted. However, it must be noted that use of encryption and the use of firewalls are aimed at solving two

different problems [22]. The use of encryption is to prevent unauthorized access to data that are being exchanged between computers while the use of firewall is to prevent unauthorized access to a network. The use of an IPSEC system makes the use of a firewall even more essential as the IPSEC system itself must be protected from security attacks.

Therefore we see that in order for a firewall to function correctly there is a need for it to be able to access packet headers. Under current IPSEC standards, a firewall cannot be used correctly and to its full potential.

3.3 Case 3: NAT and Internet Service Providers

3.3.1 Background

Network Address Translation (NAT) is a technique often used by Internet Service Providers to maximize their use of Internet addresses. NAT allows them to use private addresses internal to their network, and then use a fixed address pool when accessing data from the Internet. In this form NAT is also reffered to as Masquerading. In masquerading an almost arbitrary number of connections is multiplexed using TCP port information [23]. For each outgoing packet the source IP is replaced by the NAT gateway's (external) IP address, and the source port is exchanged against an unused port from the range reserved exclusively for masquerading on the gateway. If the destination IP of an incoming packet is the

NAT gateway's IP address and the destination port is inside the range of ports used for masquerading on the router, the NAT router checks its masquerading table if the packet belongs to a masqueraded session; if this is the case, the destination IP and port of the internal host is inserted into the packet and the packet is sent to the internal host. Therefore, the number of simultaneous connections is limited only by the number of TCP-ports available for masquerading. NAT is also used by network administrators who wish to hide the IP addresses internal to their network from the outside Internet. This provides security by means of obscurity. Providing minimal information to potential hackers, can act as an effective detterent against security attacks.

NAT can be used in several applications other than IP address space conservation and security by obscurity. Some of these applications are:

- Load Balancing Servers: NAT can be used to create virtual servers which provide a single reference point to clients. Internally the virtual server routes the requests to one among several servers.

- Load Balancing Networks: NAT can be used for transparently using different ISP or routers when accessing a remote host. When an internal host wants to establish a new connection with a destination on the Internet, it just sends its packets to the NAT gateway. The NAT gateway , because it knows all connections, decides which provider will route this connection, replaces the source hosts (internal) address with one of the providers chosen

and sends it out to this provider's router. Since the source address is an address of this provider's network, the answers will also come in that way. The host where the packets originated never gets to know which provider had been chosen by the NAT gateway, so this process is transparent.

3.3.2 The Problem

Based on the functionality of NAT described in the last section, we can easily describe several scenario's where NAT cannot be used in conjunction with IPSEC flows because of its dependence on information within the transport layer header. One such scenario is the use of NAT in a VPN. Usually the NAT and the firewall capabilites are combined into a single device, however if IPSEC flows are present, then it is essential that NAT be performed before IPSEC processing, followed by the firewall processing. Therefore the NAT and firewall functionalities cannnot be placed in the same device.

3.4 Case 4: Network Monitoring and Analysis

3.4.1 Background

While providing someone the ability to monitor and analyze traffic from a network can be a severe security risk, at the same time this ability may be required by legitimate network engineers and administrators in order to monitor

network usage and provide better service.

There are several applications currently in use for network analysis and monitoring. TCPdump, Snoop and Sniffer from Network Associates are some commonly used tools. These tools help network administrators and engineers to monitor and analyze the traffic on their network so that they can identify faults and bottlenecks. In the recent years significant progress has been made in the field of network analysis via the use of such network analysis tools which have forced us to question the very basic assumptions that had been made regarding the nature of network traffic. For example we now know that the simplistic Poisson arrival patterns for traffic which had been assumed for a long time, are no longer valid. Invariably, as new applications develop, traffic patterns in networks also change, sometimes drastically. Without the ability to use these tools we would not be able to identify such changes.

3.4.2 The Problem

Network analysis and monitoring requires the ability to access headers within packets. One of the basic advantages offered by IPSEC is that it prevents such analysis. However, in light of the fact that such analysis is both necessary and essential, we must find a way of accomodating both needs. Such a provision is not present in the IPSEC standard as it currently stands.

Chapter 4

Solutions to Security v/s Networking

In this chapter we explore some schemes for resolving the problems described in the previous chapter and argue why each of them is not suitable for use in the Internet as a general solution. We then describe briefly our proposed solution which is a general solution for adding flexibility to the current IPSEC standard. It aims to achieve this while at the same time not severely reducing the performance of an existing IPSEC system either in terms of throughput or in terms of security.

4.1 Split IPSEC

The first scheme we consider is Split IPSEC. This scheme involves splitting the end-to-end security provided by IPSEC security gateways, into secure flows between the endpoints and the intermediate systems. This is depicted in Figure 4.1 for the case of an Internet over Satellite system using connection splitting. The end-to-end tunnel connection provided by IPSEC is split into

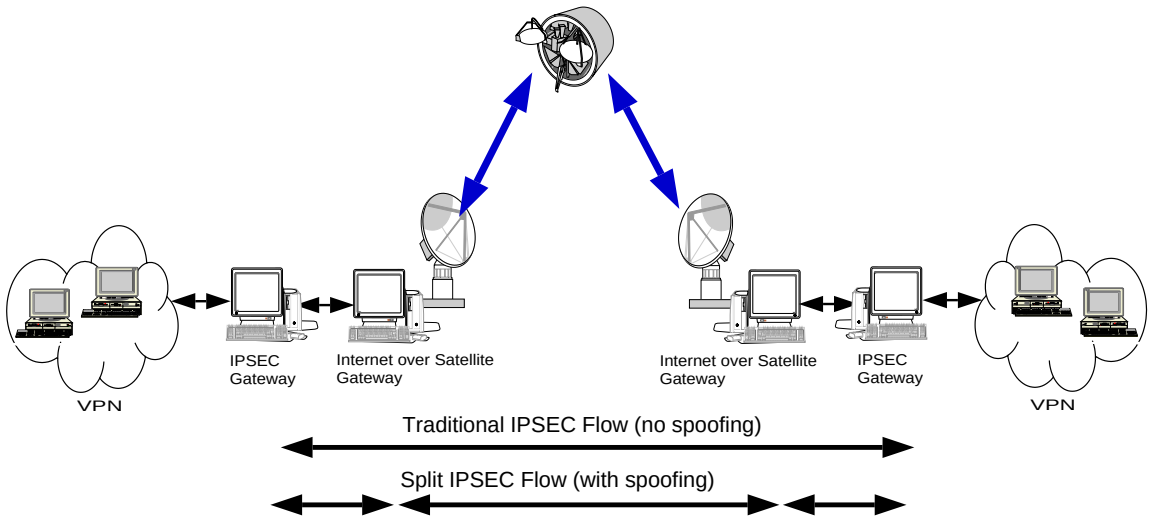

Figure 4.1: Split IPSEC in Internet over Satellite

sections. One for each segment between the IPSEC gateways and one for the segment inbetween the connection splitting gateways.

The packet flow in this scenario is as follows. A packet generated within a VPN and destined for the other side is sent to the IPSEC gateway. This gateway procedes to encrypt this packet using IPSEC tunnel mode and forwards it into the Internet. When this packet reaches the connection splitting gateway (asuming the connection splitting gateway has the right keys) the packet is decrypted, connection splitting is performed, and the resulting outgoing packet is then re-encrypted. At each connection splitting gateway, the same operation is repeated. Finally, when the packet reaches the destination IPSEC gateway, the packet is decrypted and forwarded to the destination within that VPN. In this way, an intermediate system that requires transport header information has to decrypt the entire packet and then reencrypt it.

While this scheme would require no modifications to the existing IPSEC specification, this mechanism would however break the end-to-end security concept of IPSEC tunnels. Each segment of the split IPSEC connection is a valid IPSEC secure connection. However, the intermediate systems now have to be trusted by the end networks, and proper mechanisms are needed to distribute decryption and encryption keys to them. A key distribution protocol for such a scenario can be quite complicated.

Also, in this scenario, the intermediate systems have to implement IPSEC as well as any other functionality they might be providing. These systems must now perform decryption and reencryption of packets as well. The combination of all the functions that such a system now has to run might require a sufficiently large amount of processing to overwhelm the intermediate system or significantly reduce its throughput.

The primary advantages and disadvantages of this scheme are summarized below:

Advantages:

- No changes required to the existing IPSEC specification

Disadvantages:

- All flows through the intermediate gateway are decrypted and then re-encrypted; this requires a lot of processing.

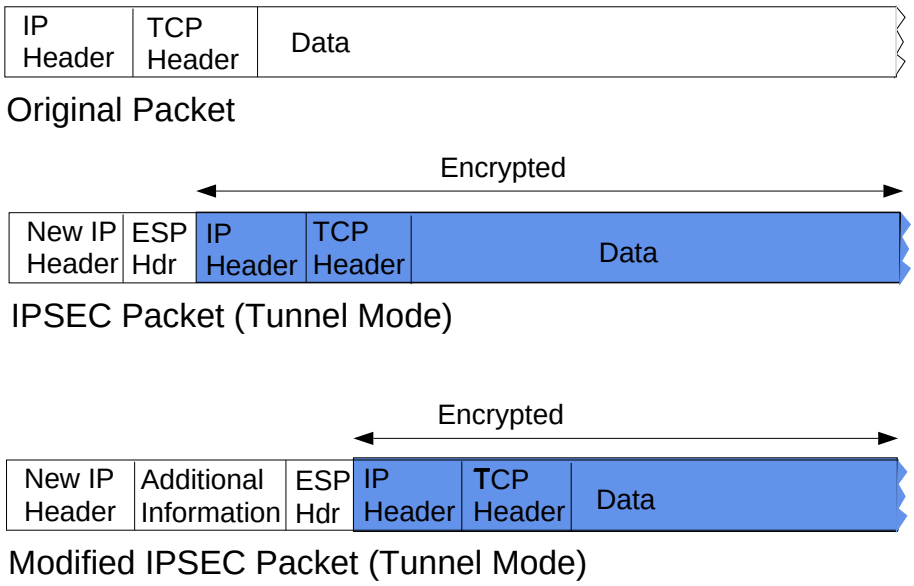

Figure 4.2: Modified IPSEC

- The intermediate systems have to be trusted; this breaks down the end-to-end encryption promised by the tunnel mode IPSEC specification.
- A key distribution protocol must be implemented to ensure that the intermediate gateways get decryption keys.

4.2 Modifications to IPSEC packet format

In this section we consider a proposal to modifiy the IPSEC packet structure to support applications that might need access to transport layer headers. In this method, the IPSEC packet format is modified to add additional information required by intermediate systems. The new packet formats are shown in Figure 4.2. This proposal is similar to the concept of transport friendly ESP outlined in [24].

This scheme aims to overcome the problem created for intermediate systems by IPSEC encryption. Therefore this scheme aims at making selected information from the transport layer headers available to intermediate systems by duplicating it outside the encrypted packet. The IPSEC packet header is expanded to include such information.

The packet flow in one such scenario might be as follows. A packet generated within a VPN and destined for the other side is sent to the IPSEC gateway. This gateway procedes to encrypt this packet using IPSEC tunnel mode. The IPSEC gateway also places information from the TCP header into the new fields in the IPSEC header. This packet is now forwarded towards the second IPSEC gateway. When this packet arrives at the intermediate system, the required information can be read, and the packet processed. Finally when the packet arrives at the destination IPSEC gateway, decryption is performed and the packet is forwarded to its destination within the second VPN.

The information that needs to be added to the IPSEC header in this scheme could open the traffic flows to several attacks that IPSEC was formulated to protect against, such as traffic analysis. Therefore, this scheme would undo to a large extent the security benefits of IPSEC. Moreover, another significant disadvantage of this approach is that the implementation of this scheme would require extensive modification to IPSEC.

The advantages and disadvantages of this scheme are summarized below:

Advantages:

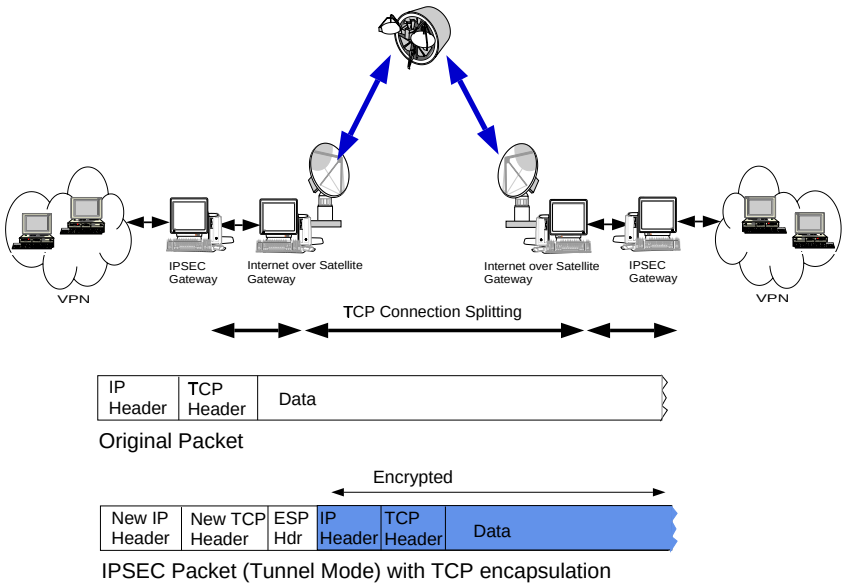

Figure 4.3: TCP encapsulation of IPSEC flows

- Per packet overhead is minimal

Disadvantages:

- Requires changing the IPSEC specification
- Significantly reduces the security of the traffic flow
- Intermediate systems might require keys

4.3 Tunneling IPSEC within TCP flow

In this section we discuss a scheme which might be used to overcome some of the limitations imposed by IPSEC by essentially nullifying some of its benefits. In this method, an IPSEC packet is encapsulated within another TCP packet by the

security gateway. Therefore a new and unencrypted transport layer header is added to the packet, which is visible to all intermediate systems.

In this scenario, when an encrypted flow reaches an intermediate gateway that requires access to transport layer headers, it sends back an explicit notification to the originating security gateway. The security gateway then opens a new TCP connection with the destination set to the end-point security gateway, and places the entire original encrypted packet into the data payload of this new packet. The resulting packet format from this scheme is shown in Figure 4.3. The intermediate system can now access the transport header, and possibly use the information, even though it is different from the infromation within the transport header of the original packet. At the second security gateway, the original packet is retrieved from the payload of the new TCP packets and decryption and forwarding is performed as usual.

This method requires several siginificant changes to the IPSEC gateways. However, the security of the original packet is not compromised, and even though the TCP headers of the packets flowing between the VPN's are visible, this does not reveal any information about the real encrypted packet which is the payload of the outer packet. The major drawback of this scheme is the significant per packet overhead due to the extra encapsulation.

The advantages and disadvantages of this scheme are summarized below:

Advantages:

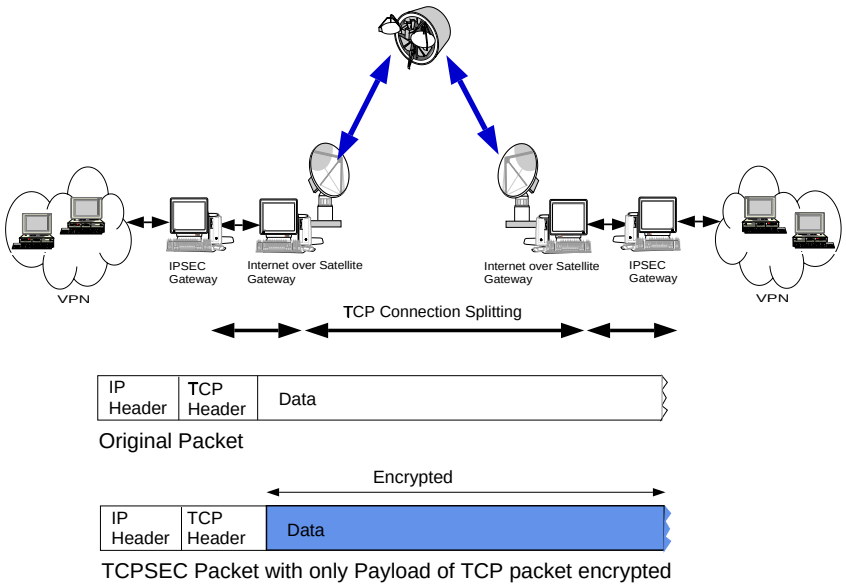

Figure 4.4: TCPSEC in Internet over Satellite

- Security of the original packet is not compromised

Disadvantages:

- Requires changing the IPSEC specification
- Requires some changes to the security gateways to add the TCP encapsulation functionality
- There is a large per packet overhead
- Significant loss of security

4.4 Transport Layer Security(TLS) or TCPSEC

In this section we discuss an alternative to IPSEC which has been proposed in the Internet community as an alternate solution for providing security. This

scheme proposes the use of security at the transport layer. TCPSEC refers to the TCP analog of IPSEC. In this only the payload of the TCP packet is encrypted leaving the TCP headers unencrypted. This is shown in Figure 4.4

In this scenario a packet generated within a VPN and destined for the other side is encrypted using TCPSEC. This packet is now forwarded towards the second TCPSEC gateway. When this packet arrives at an intermediate system, the information from the TCP headers is available. Finally the packet is decrypted at the second TCPSEC gateway and the packet is forwarded to its destination in the second VPN.

The major drawback of this scheme is that it reduces the effectiveness of security as the TCP headers of the packets are now unencrypted and open to several well known attacks.

A more general approach to this scheme is the use of Transport Layer Security (TLS). TLS provides security between two communicating applications. It is layered on top of some reliable transport protocol such as TCP. Using this protocol for security in place of IPSEC significantly reduces the security of the communication as it leaves the transport layer header unencrypted. This can potentially leave the network vulnerable to several security risks. Moreover, because TLS depends on a reliable transport protocol it cannot be used for applications that use UDP. Voice over IP is one such major application that will not be able to use encryption. Therefore, video conferencing sessions can easily be snooped on. Moreover, TLS cannot be used by any multicast applications

either. Therefore, TLS is not a viable and acceptable solution.

The advantages and disadvantages of this scheme are summarized below:

Advantages:

- Requires no changes to the IPSEC specification

Disadvantages:

- Requires changes to the security gateway to add the TCPSEC functionality
- Security is significantly reduced as now the TCP header is unencrypted
- For TCP only

4.5 Layered Encryption in IPSEC

Keeping in mind the failure of any of the other approaches to provide a satisfactory answer, we propose a new scheme which seeks to modify IPSEC slightly in order to make it more flexible. This scheme enables different parts of a packet to be encrypted with different encryption schemes with different keys or even the same encryption scheme but with different keys for different parts of the packet. This gives us the flexibility to selectively distribute one key to provide access to different parts of the packet without compromising the security of the other parts of the same packet. This scheme is described and analyzed further in the following chapters.

4.6 Discussion

The two main system parameters of interest to us when examining the feasibility of any system are, first of all, the ability to meet the functional requirements, and secondly, the ease of implementation. In this scenario, an acceptable solution to the problem of selective access to packet headers should have the ability to support applications that require access to headers within a packet as well as the ability to do so without significant compromise on security. Therefore, keeping these parameters in mind we now reexamine the different schemes described in the previous sections.

All schemes described in the previous section do infact hinder the level of security provided by IPSEC. While the split IPSEC, TCP encapsulation of IPSEC packets and the separate encrypted packet scheme both maintain original packet integrity, they do introduce weakness at different points in the system. The split IPSEC weakens the system as a whole due to its reliance on the ability to locate and trust intermediate systems in a network. The TCP encapusulation scheme does not depend on this. However, its point of weakness is the introduction of a TCP header which is visible to a potential attacker. The remaining two schemes of using TLS or TCPSEC and proposals for modification of the IPSEC packet format both severly diminish security and are therefore not appropriate for consideration in any system where security is of primary importance.

From the point of view of ease and feasibility of implementation the split IPSEC scheme places a significant additional load on intermediate systems by adding the need to decrypt and re-encrypt entire packets. The scheme of modifying the IPSEC packet format though not difficult to implement would not be acceptable as it significantly impairs security. The TCP encapsulation of IPSEC packets is fairly complex and impractical for implementation in any real system.

Chapter 5

Layered IPSEC

This chapter describes our proposed scheme for supporting secure IPSEC flows in a network in which intermediate nodes might require the ability to read packet headers. The core concept of the scheme was developed in conjunction with HRL Laboratories [25]. This scheme aims at adding flexibility to the existing IPSEC framework. By using multiple keys, it provides the end-points of a flow with the ability to selectively distribute keys to different nodes, depending on the level of acces they require. Using separate encryption keys and algorithms for different parts of the same packet also enhances security as now for the data to be completely compromised we require that multiple keys be obtained by the attacker. This makes even brute force key cracking attacks harder.

In traditional IPSEC a single encryption scheme is used to encrypt the entire packet. The encryption scheme chosen can be RC5, DES or 3DES or any other algorithm. However, IPSEC does not make any provision for the use of multiple keys or multiple encryption algorithms. This makes the IPSEC framework quite inflexible.

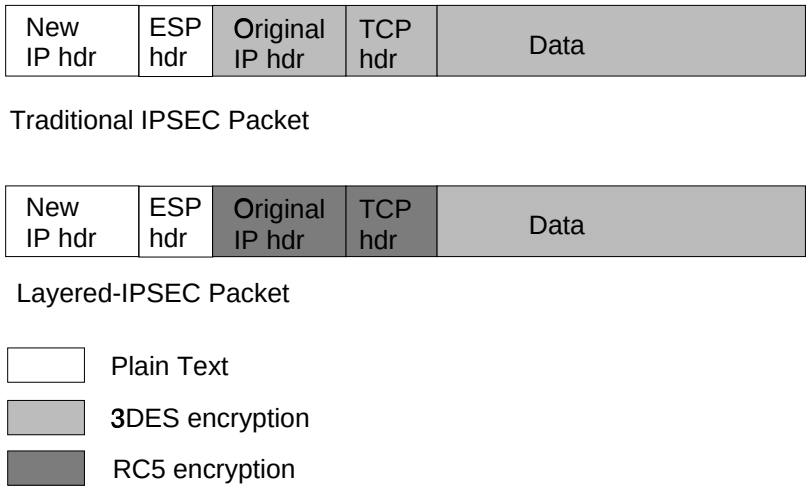

Figure 5.1: Layered IPSEC Packet Encryption

We propose a scheme which allows different encryption algorithms with different keys to encrypt different parts of the same packet. This allows us the ability to distribute one key without completely compromizing the security of the entire packet. This key can therefore be distributed to all intermediate systems that are interested in accessing parts of a packet. The primary features of this scheme are listed below:

- Multiple key generation: Each end-point of the secure connection generates separate keys for each of the encryption algorithms it will use for that particular connection. Multiple keys can be used even if the same encryption algorithm is used to perform the encryption. In the original IPSEC specification, each end-point has two keys, one for sending and one for receiving data. In our scheme, if for example two encryption schemes or keys are being used, then there are a total of four keys at each end-point. One set is used for sending data and the other for receiving data.

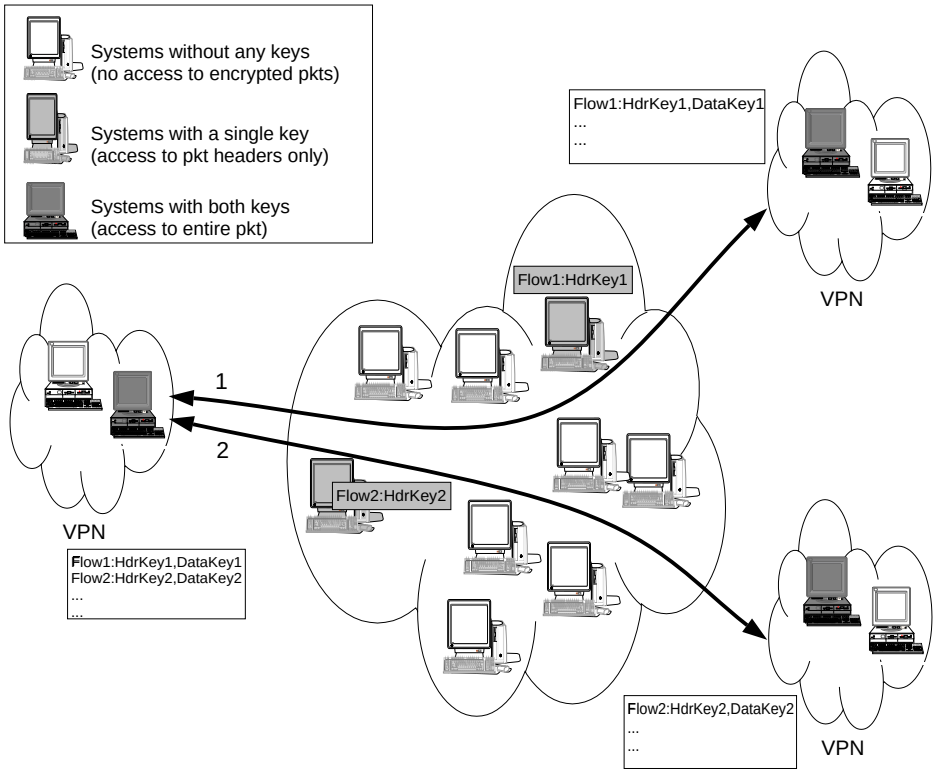

Figure 5.2: Layered IPSEC Packet Encryption

- Multiple key distribution: Similar to the original IPSEC single key scenario, the keys are symmetric and therefore need to be exchanged between the connetion end-points. However, in layered IPSEC, key distribution is much more complicated. Not only is the number of keys that need to be exchanged two or more times the number of keys in original IPSEC, but in addition, if other intermediate systems require a key the appropriate key needs to be distributed to them as well.

- Selective Decryption and Re-encryption: While the end-points of the secure connection decrypt the entire packet, any intermediate systems that have been given keys to certain parts of a packet perform partial decryption on those portions. In some cases if the original packet is modified, re-encryption must be performed so that the end systems will still get what appear to be end-to-end encrypted packets. As the decryption and encryption keys are the same, this does not add much complexity to the system.

In the problems described in Chapter 3, the part of the packet that is of interest to the applications is primarily the packet headers. Therefore, one direct application of this scheme is to encrypt the header of a packet with one encryption scheme and to encrypt the remaining portion of the packet with another encryption scheme. Therefore, the end systems have two keys for encryption and decryption, but all interested intermediate systems only have the

key for the packet headers. By allowing two seperate keys, this scheme aims at providing the much needed packet header visibility to intermediate systems, but at the same time does not completely reveal the contents of the entire packet to those systems as the data portion of the packet is still encrypted by a key that is not known to them.

We now describe how the use of layered IPSEC can solve the problems described in Chapter 2 without either increasing the complexity of the system unacceptably or significantly diminishing the overall security of the system.

- Internet over Satellite: The key used to encrypt the header is distributed to the connection splitting gateways. When a packet from an encrypted flow reaches the connection splitting gateway, the header of the packet is decrypted, acknowledgements are spoofed and the decrypted packet headers are reencrypted before transmission.

- Firewalls: The firewall obtains keys for the packet headers of the IPSEC flows passing through it. The packet header of an incomming packet is decrypted, and examined to see if the firewall rules are satisfied. The outgoing packet is then re-encrypted before retransmission.

- Network Address Translation: This functionality it similar to the functionality of the firewall described previously.

- Network Monitoring and Analysis: The network monitoring system obtains keys for decrypting headers of all IPSEC flows passing through it. For each

IPSEC encrypted packet, the header is decrypted and all relevant information is extracted. There is no need to re-encrypt as a captured packet is never retransmitted, network monitoring applications only receive a copy of the original packet.

Therefore, the primary benefits of using a layered IPSEC scheme is not only that it enables us to design and build efficient networks, but also that it enables the existence of secure flows without sacrificing on efficiency. The advantages of using a layered IPSEC approach are summarized below.

- No compromise on security: The end-to-end security for data payloads is preserved. Even the packet headers are encrypted and are safe from attackers who do no have the appropriate keys.

- Privileged applications: Layered IPSEC gives us the ability to supprort priviledged applications in the Internet, by allowing us to control up to a finer level who can read specific parts of a packet.

- Feasiblity and ease of implementation: The intermediate systems in this scheme only have to decrypt a small portion of the entire packet. This takes much less computation than having to decrypt the entire packet making this scheme much more practical and feasible than using split IPSEC.

- Security: Probably plaintext cryptoanalysis can be used to aid in cracking an encryption key. A probably plaintext attack works by looking at certain

bit positions for which a likely value can be predicted [26]. A comparison engine can count the number of such matches, and pick up certain packets for further analysis by a second stage cracking engine. Given that the IPSEC protocols when used in tunnel mode encrypt the IP header, a large number of bit fields have values that can be predicted. Therefore, the IPSEC protocols are vulnerable to probably plaintext attacks. Using the encrypted IP and TCP headers, attackers can obtain the encryption key, and then proceed to decrypt the data itself. However, use of Layered IPSEC can protect against this type of an attack, as different keys are being used for the headers and the data. Therefore, there is very little probably plaintext information available to help in cracking the key used to encrypt the data.

- Not a new scheme: This scheme is still essentially IPSEC, the added modifications only make IPSEC more flexible.

Chapter 6

Implementation of Layered IPSEC

In this chapter we describe some details of the IPSEC testbed and the various modules that were implemented in order to demonstrate the feasibility of the layered IPSEC concept.

6.1 Background

6.1.1 FreeS/WAN

The IPSEC testbed we developed was based on the Free Secure WAN(FreeS/WAN) implementation of IPSEC and IKE for Linux. We used version 1.0 of the FreeS/WAN written by developers at the Electronic Frontier Foundation (EFF) in Canada [27]. This is a stable version of the software and runs on Linux version 2.0.36. FreeS/WAN provides us with a base IPSEC implementation on top of which we can implement our modifications.

Version 1.0 of the FreeS/WAN software supports both manual key distribution as well as automatic key distribution. Manual keys can be specified in a

configuration file. The automatic key distribution method uses a shared secret string stored in a different configuration file to create session keys as required. All connections are re-keyed periodically. Automatic key distribution using the Internet Key Exchange (IKE) protocol is implemented via the Pluto daemon.

The FreeS/WAN software supports both the required encryption schemes recommended by IPSEC for use in ESP in addition of some others. Both the null encryption transform and the DES encryption transform are implemented, for conformance with other IPSEC requirements, however, they are generally disabled by default as they are both regarded as insecure [28]. At a recent contest, a 56 bit DES key was cracked in only 22 hours using brute force exhaustive searching of the key space. However Triple DES (3DES) performs three DES encryptions on a single data block, and therefore provides a much higher level of security than is available from a single DES pass. The three-key version of 3DES is the default encryption algorithm for Linux FreeS/WAN.

For IPSEC in AH mode, the FreeS/WAN software implements both keyed MD5 and keyed SHA transforms.

6.1.2 The RC5 Algorithm

RC5 is a fast symmetric block cipher suitable for hardware or software implementations. This algorithm has been developed by Ronald L. Rivest at MIT Laboratory for Computer Science [29]. RC5 has a variable-length secret key,

providing flexibility in its security level. RC5 provides several tunable parameters such as the word size, the number of rounds, and the number of bytes in the secret key. Therefore, this algorithm can offer a variable level of security depending on the parameters which are chosen. The DES algorithm does not have this flexibility. As the security needs of an application increase, the parameters of RC5 can be changed to provide increased protection against brute force attacks. Detailed security analysis of the algorithm has shown that RC5 with 12 rounds and 64 bit block size gives roughly the same security as DES against analytical attacks [30].

In general, by choosing greater key sizes and more rounds in RC5, leads to a higher level of security. The performance of RC5 is directly proportional to the number of rounds, and is not affected by the key size. This flexibility of RC5 makes it ideal for our implementation of layered IPSEC, as the tradeoff between speed(and hence throughput) and security can be balanced via appropriate parameter settings. A word size of 32 bits, a round number of 12, and a key length of 16 bytes are recommended as the nominal choice of parameters for RC5.

The input to the encryption and decryption algorithm of RC5 are 2 "word" sized registers. Therefore RC5 can be used to selectively encode and decode parts of a data block in 2 word sized chunks independent of each other. This once again is perfectly suited for our implementation of layered IPSEC as we can selectively decrypt as little a portion of the data packet as we have to and thereby save considerably on processing. The RC5 algorithm is considered

superior to DES and other encryption and decryption algorithms with respect to software performance. This makes it an excellent candidate for use in IPSEC where it can be used even on workstations and laptops without the need for special hardware and without any severe performance penalties.

We implemented an RC5 encryption and decryption library for use in our layered IPSEC testbed. The library was derrived heavily from published reference code for the algorithm. Various modifications were added to make it more suitable for our purposes, however the core algorithm functionality was not modified in any way.

Though the ESP RC5-CBC transform for using RC5 with IPSEC in ESP mode has been described in [31] this was not used, as there was no readily available implementation of this for IPSEC. The RC5 is a patented algorithm, therefore support for this algorithm from various IPSEC vendors and implementors is scarce.

6.1.3 Linux IPCHAINS

Linux IPCHAINS is the core functionality in Linux that enables applications such as firewalls, transparent proxies, and masquerading to be used. Linux IPCHAINS enable users to specify a set of rules and actions for all incomming packets [32]. A chain is a checklist of rules. Each rule specifies what action should be taken on each packet. If a rule doesn't match the packet, then the next

rule in the chain is consulted. Each packet that enters the IP layer in Linux is tested against the various specified rules and if the packet matches a specific rule, then that packet has the associated action performed on it. For example, a firewall application registers DENY and ALLOW actions within the Linux kernel along with the address and port information. All incomming packets have theirs ports and addresses compared against this rule set and are forwarded or dropped according to the policy specified for those packets.

In Linux, a firewall rule can be specifed at three stages, input, forwarding and output. An incomming packet is first checked against input firewall rules, and if it is accepted there then it is checked against the forwarding rules (after a routing decision), and finally just before a packet is sent out, it is checked against the output rules. Each stage can have muliple sets of rules; this is why these are known as chains.

Recognizing this structure in the Linux kernel is important for our implementation approach. All packets that enter a system will pass through the IPCHAINS. Since what we want to do is modify IP packets, IPCHAINS is the right tool for us to use. We can use Linux IPCHAINS, to modify a packet by inserting a rule early in the IP input chain. All applications that receive the packet after this rule will get the modified packet. Finally we can insert a rule in the IP output chain, which is called just before the packet is sent out. This rule can undo the changes made in the input chain.

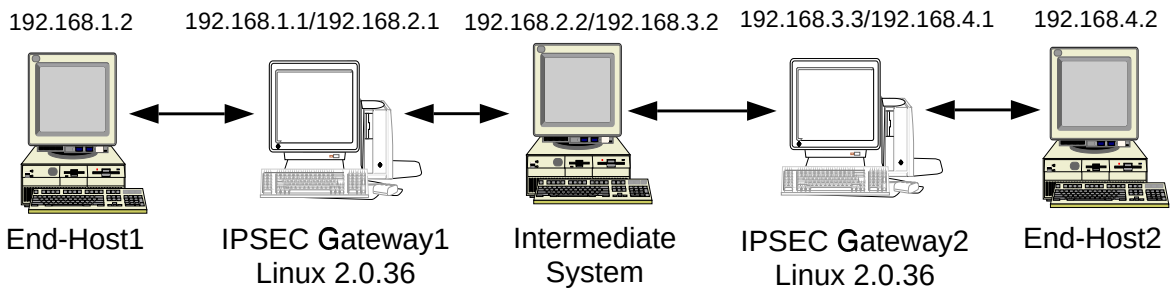

Figure 6.1: IPSEC Testbed Architecture

6.2 Testbed Network Architecture

The testbed we setup for implementation of layered IPSEC is shown in Figure 6.1. In our testbed, the IPSEC Gateways run the FreeS/WAN software. The secure tunnel is setup between these two gateways. End-to-end communication is tested by running applications between the two end hosts (ping, telnet, and ftp). The intermediate system between the two security gateways, functions as a general router in the base case when normal IPSEC is used, but is used to test and verify the correct operation of the layered IPSEC concept in other scenarios. It basically represents a specific trusted and authorized host in the Internet that wants to access information contained in the packet headers of the encrypted flow between the two security gateways.

6.3 Implemented Changes and Modifications

6.3.1 Modifications to FreeS/WAN

For our testbed, all changes to the FreeS/WAN software distribution were kept to a minimal. No modifications were made to paket formats, key distribution, etc. The only changes that were made were to add support for layered IPSEC. This essentially involves two steps. In the first step, the starting point for the default IPSEC encryption, i.e. 3DES is moved forward, such that only the portion of the packet following the transport layer header is encrypted with it. In the second step the portion of the packet that has now been exposed is then encrypted using RC5 encryption.

Corresponding changes are also made in the decryption routines of the FreeS/WAN software, so that the right portions of the packet are decrypted with the corresponding algorithm.

This forms our base test system, which demonstrates IPSEC end-to-end security with layered encryption. In order to demonstrate the usefullness of the layered IPSEC approach we also implemented some example applications.

6.3.2 IPSEC module for Intermediate Systems

We implemented a dynamically loadable module for use in intermediate systems which might want to access the packet header information from a flow encrypted using layered IPSEC. This module uses the IPCHAINS feature in Linux. It adds

two rules to IPCHAINS. The input rule removes the ESP tunnel headers and decrypts the IP and TCP headers from the original packet (before ESP). All applications that sit in between this input stage and the output stage see the actual packet header and can perform any required operations on them. Applications like IP Masquerade, NAT and Firewalls work in this manner. The output stage is the inverse operation of the input stage, the IP and TCP headers are reencrypted using RC5 encryption, the IPSEC IP and ESP headers are replaced and the packet is sent out.

As far the ESP end-points are concerned, they obtain a packet that appears to be essentially unmodified to them. The ESP header is the same as was initially transmitted from the source IPSEC gateway.

6.3.3 RC5 Encryption/Decryption Library

In order for the intermediate systems to be able to handle a large volume of traffic, its decryption and reencryption has to be extremely efficient and fast. Therefore, rather than simply using 3DES with different keys on different parts of the packet, we use RC5 on the packet headers instead. RC5 is known for its speed in encryption/decryption. In order to do this we implemented an RC5 library for use both by the IPSEC gateways as well as intermediate systems.

6.3.4 Enhanced TCPdump

We also implemented an enhanced version of TCPdump which enables us to examine ESP flows. Normally TCPdump simply examines copies of packets it picks up from the network and displays the information contained in the headers. In this way, TCPdump is a good tool for use by network engineers and designers. However, when IPSEC in ESP mode is used, the only information that TCPdump can provide is limited to the the outer IP header of the tunnel between the IPSEC gateways. We modified this behaviour of TCPdump such that it can now display the decrypted contents of the packets encapsulated within the tunnel as well.

Chapter 7

Results and Analysis

In this chapter, we present some results and analysis related to our implementation of layered IPSEC. We provide justification for various design choices, in the form of timing measurements and provide feasibility arguments on the basis of both implemented applications as well as overhead measurements.

7.1 Applications

In order to demonstrate the feasibility of our approach, we implemented sample applications. We implemented modification for both the firewall implementation in Linux 2.2.12 as well as TCPdump 3.4. In both cases only minimal changes were required in order to implement correct functionality. In both cases the approach is identical. The ESP encrypted packet is decrypted, the tunneled packet is extracted and passed on for further default processing. In the case of TCPdump the application picks up a copy of the packet from the network, therefore, there is no need to re-encrypt the packet. However, for the case of the

firewall, the packet may need to be forwarded back out into the network, and therefore, it needs to be reencrypted; and the ESP tunnel header needs to be replaced on the packet.

7.1.1 Firewalls

We were successfully able to demonstrate correct firewall functionality on an encrypted flow using layered IPSEC and our module for intermediate systems. With an encrypted flow, a firewall cannot examine the complete headers of the packets, and therefore cannot function correctly. However, with our modifications, the firewall application is able to see the relevant information from the packet headers, and make correct decisions about dropping or forwarding these packets.

We performed the following experiment. The intermediate system is running as a firewall. When IPSEC is not used the firewall works correctly. As an example we add a rule to block all telnet traffic (as passwords are sent clear text). The firewall correctly blocks the attempt to telnet from one end-host to the other. However, now if IPSEC is enabled at the security gateways, the telnet session is permitted. This is a significant security risk as even though the traffic up to the destination security gateway is encrypted, there might be a malicious intruder on the other network who will be easily able to capture the password. Therefore, we have a situation where the use of IPSEC is infact reducing the

effectiveness of other security measures.

In the second part of the experiment we run Layered IPSEC on the security gateways, and our layered IPSEC module for the intermediate system on the firewall. The Layered IPSEC module decapsulates the IPSEC ESP packet, decrypts the header information and passes the internal packet to the firewall. Therefore, in this case, the firewall rule is correctly able to determine that a telnet session has been requested and blocks the connection attempt.

7.1.2 Network Monitoring and Analysis

We were successfully able to demonstrate correct operation of a network monitoring and analysis tool such as TCPdump. Layered IPSEC was used at the IPSEC gateways, and our modified TCPdump was used on an intermediate system. All incomming IPSEC ESP packets from the network are identified and the encapsulated packet is extracted, the header is decrypted and then passed onwards so that the various fields in the header can be identified. We were correctly able to examine and record information regarding not only the IP tunnel between the IPSEC gateway, but information from the IP and TCP headers of the tunneled packets as well. In our prototype implementation, we only tested our setup for one flow, therefore only one key was needed. In a more robust implementation there might be some additional overhead involved in locating appropriate header keys for each traffic flow.

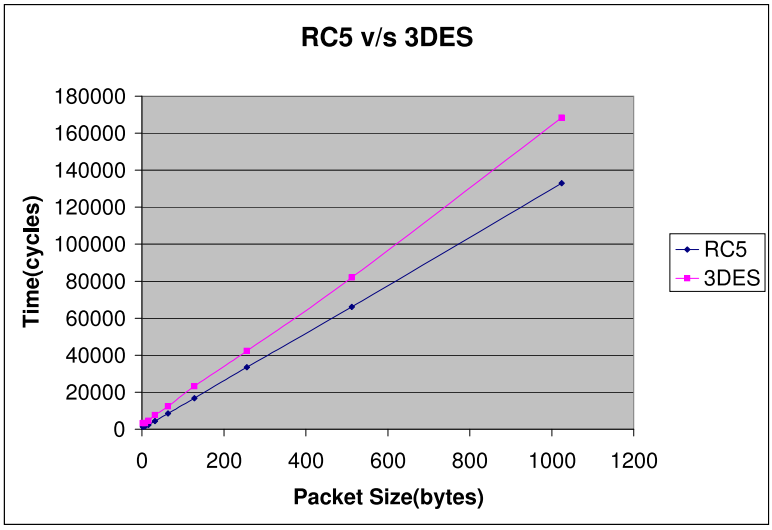

Figure 7.1: Comparison of encryption times for RC5 versus 3DES

7.2 Timing Analysis

An important factor to consider when analyzing a new scheme is its practicality. In this section we present some measurement results from our implementation, and explain some design choices made during our implementation.

For the purposes of timing measurements we use the time stamp counter on the Pentium chip. The RDTSC (Read Time Stamp Counter) instruction is a two byte instruction, that returns the number of clock cycles since the CPU was powered up. Therefore, by reading this counter twice we can compute the number of cycles that have elapsed between the first and the second call. This provides us with a much more accurate and meaningful measurement of time, which are independent of the CPU clock speed.

7.2.1 3DES v/s RC5

When first considering a system that would do layered IPSEC, we were faced with a design choice of what encryption algorithm to use for the different parts of the packet. The default FreeS/WAN implementation uses 3DES as its basic encryption mechanism. However, in studying the literature, we found that there are other choices available as well. One notable alternative, was RC5. The RC5 algorithm has been specially designed for speed and fast implementation. Though different figures have been quoted providing the speeds of these algorithms, we wanted to compare these two on the same platform, with the same measuring setup. So we could choose to use either 3DES or RC5 to encrypt the packet header and data portion with different keys, or perhaps even both on the different parts of the packet.

Figure 7.1 shows a comparison of the time taken to encrypt a block of data using the 3DES algorithm and the RC5 algorithm. Measurements were taken for various block sizes, ranging from 2 to 1024 bytes. The measurements are in numbers of cycles and can be easily converted to seconds by dividing by the clock speed of the CPU, which was 166MHz in our case. As these encryption algorithms operate on small protions of data at a time, the time taken to encrypt increasingly larger blocks grows linearly. However, it is clear that the RC5 encryption(RC5 32/12/16 with 32 bit word size, 12 rounds, and 16 bytes of key length) takes much less time than 3DES(16 bytes of key length). The data from

Packet Size(bytes)	RC5(cycles)	3DES(cycles)
2	1285.992	3217.761
4	1587.342	3347.378
8	1587.568	3207.518
16	2324.53	4516.135
32	4388.633	7625.232
64	8497.234	12351.9
128	16706.51	23253.11
256	33512.03	42314.46
512	66183.67	820002.76
1024	132944.6	168445.6

Table 7.1: Average Number of Cycles to Encrypt a Packet using RC5 and 3DES

our measurements is sumarized in Table 7.1.

However, the relative security offered by RC5 versus 3DES for same size key lengths have not been evaluated in the literature; though RC5 claims to be as secure as 3DES. Therefore, keeping in mind the faster speed of RC5, and weighing it against its unproven strength, for our implementation of layered IPSEC, we decided to use RC5 for encryption of the packet headers and maintain the default use of 3DES for encryption of the data portion of the packet.

This is a favorable design choice, as the faster speed of RC5 means that it is possible to decrypt and re-encrypt the packet header in the middle of the

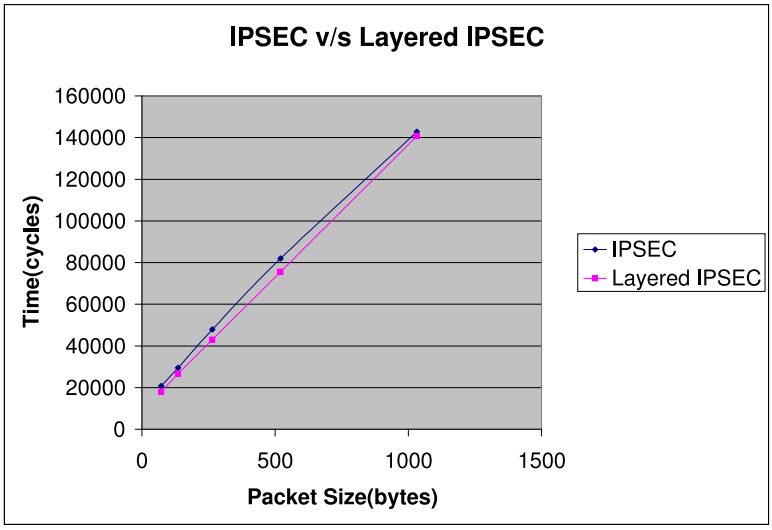

Figure 7.2: Comparison of time required for normal IPSEC encryption versus Layered IPSEC encryption

network, much more efficiently. This makes the layered IPSEC concept much more feasible and practical.

7.2.2 Overhead at IPSEC Gateways

In order to justify our approach as a viable modification, we also have to show that it does not add too much overhead at an IPSEC Gateway compared with an unmodified IPSEC Gateway. For this purpose we constructed an experiment where we first measured execution time for encrypting a packet entirely with 3DES as normal IPSEC does and then measure the time required to use Layered IPSEC on a Packet. We repeated this experiment for packet sizes of 64, 128, 256, 512 and 1024 bytes. An 8 byte header is added to our payload size specified. Our results are shown in Figure 7.2. As can be seen the time measured in cycles to perform Layered IPSEC on a packet(with RC5 and 3DES) is slightly less than

Packet Size(bytes)	Normal IPSEC(cycles)	Layered IPSEC(cycles)
72	20713.26	17963.3
136	29555.53	26677.53
264	47864.7	42948.4
520	81971.09	75522.94
1032	142753.5	140684.2

Table 7.2: Average Number of Cycles to Encrypt a Packet IPSEC v/s L-IPSEC

the time required to perform 3DES on the entire packet. RC5 is a much faster algorith than 3DES as shown by our previous experiment. However, since it is used on such a small portion of the packet, the gains from its faster speed are not very large. Nevertheless, the difference is sufficient to offset the overhead of 2 function calls that Layered IPSEC must make for encryption, versus 1 for normal IPSEC.

Therefore, we see that implementing Layered IPSEC has not added significant overhead to an IPSEC gateway. It should be noted that in this case the overhead has been offset by the use of much faster RC5 algorithm. If 3DES were to be used on both parts of a packet in Layered IPSEC, this would not be the case.

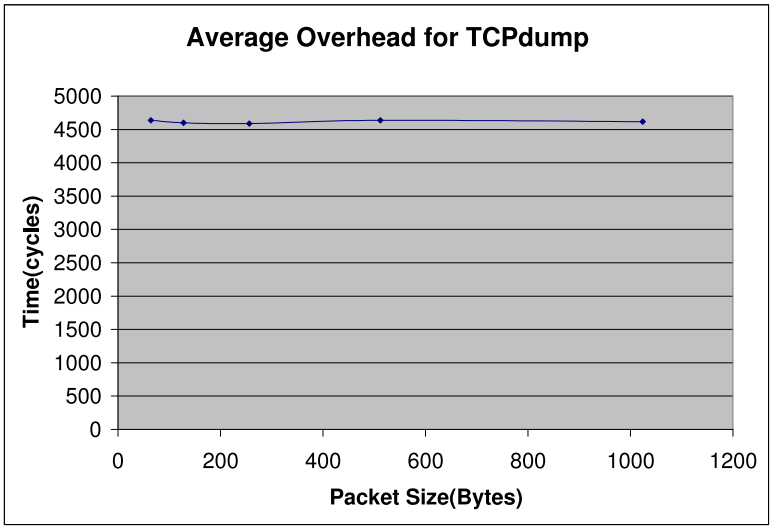

Figure 7.3: Average Number of Cycles added to TCPdump by header decryption

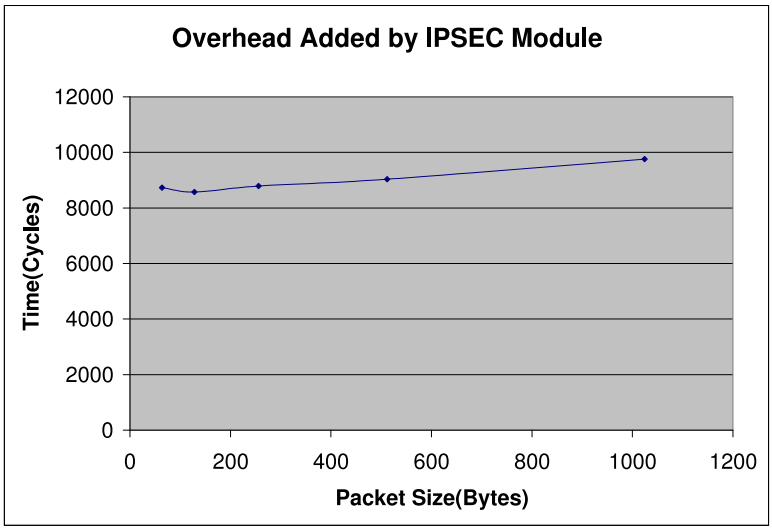

Figure 7.4: Average Number of Cycles added by IPSEC module for intermediate nodes

7.2.3 Overhead at Intermediate Gateways

There can be two types of intermediate nodes. The first kind are passive and do not need to allow packets to pass through them. The second kind are those that do allow packets to pass throught them, with or without modification. For the first kind, the overhead comes simply from the need to decrypt the packet headers. There is no need to reencrypt as these nodes are only passive observers of traffic. TCPdump is a good example of this kind of an application. In the second case, packet headers need to be reencypted before retransmission back into the network. A firewall is an example of this type of a node.

The data from our measurements for the two different cases are summarized in Table 7.3. Figure 7.3 shows the overhead incurred by a traffic monitoring node that is running our enhanced TCPdump. The additional decryption overhead added is constant as irrespective of packet size; only the same amount of packet header bytes need to be decrypted.

Figure 7.4 shows the overhead of using Layered IPSEC on an intermediate node that needs to modify packet headers. In this case the overhead is dependent on the packet size as the additional processing required for handling Layered IPSEC includes a memory allocation for creating a new packet for retransmission (packet headers need to be added). This operation is dependent on the size of the packet. However, even for packet sizes ranging from 64 bytes to 1K, the number of cycles only increases from 8500 to 10,000. This provides a good indication of

Packet Size(bytes)	TCPdump(cycles)	L-IPSEC Module(cycles)
64	4637.133	8732.88
128	4598.633	8537.08
256	4587.333	8793.82
512	4637.133	9038.70
1024	4617.933	9755.92

Table 7.3: Overhead at Intermediate Nodes

how much overhead is added by the Layered IPSEC module.

Chapter 8

Conclusions and Future Work

In this thesis we have described an alternative mechanism for adding flexibility to the current IPSEC standard. The current IPSEC standard has been in development for a long period of time. The work was initiated at a time when there was no forseable need for intermediate nodes to access information contained within encrypted packets. This has changed quite dramatically over the years, and applications which require such ability are gaining popularity. The use of Layered IPSEC can provide a way of accomodating applications in the middle of the network which violate the end-to-end security paradigm. In Layered IPSEC, the end-to-end security concept is not violated for the data being transmitted, only the packet header information is made visible to intermediate nodes with valid packet header keys.

We also provided some results from our prototype implementation, regarding the practicality of this approach. We showed how Layered IPSEC can be implemented with only slight modifications to the base IPSEC implementation. We also demonstrated sample applications such as Firewalls and TCPdump

which use Layered IPSEC to obtain packet header information.

Analysis of the timing measurements from our implementation shows that the overhead in using Layered IPSEC is not unacceptably large. Infact any system that currently implements IPSEC can implement Layered IPSEC without any loss in throughput. Due to the nature of Layered IPSEC, the overhead incurred at intermediate nodes in the network is also small.

There is currently significant opposition to the concept of Layered IPSEC, in the Internet community. Most opposition is based on arguments regarding end-to-end security violation, unclear motivation for its use, as well as performance drawbacks. This thesis aims at answering some of these questions and concerns, by providing a prototype implementation as well as actual measurements from the implementation.

On the basis of the results and arguments presented earlier in this thesis we argue that Layered IPSEC is an important tool for adding flexibility to security in the Internet. We also propose the use of a faster encryption algorithm such as RC5 for data such as packet headers which need relatively less security.

Further work is needed in developing and adding a key distribution mechanism for Layered IPSEC. For the purposes of simplicity we used manually distributed keys for distributing RC5 keys between Layered IPSEC end-hosts and Layered IPSEC intermediate nodes. Significant work also needs to be done in convincing the Internet community of the value behind this approach. Formal Internet drafts and detailed specifications need to prepared and presented at the

IETF, before the Internet community will find it an acceptable option to add to the IPSEC standard. Our prototype implementation only verified correct functionality with firewalls and TCPDump. The implementation of NAT and connection splitting gateways for Internet over satellite is similar in structure to that of a firewall application. Both use the same Linux IPCHAINS feature that is used by the firewall application. Therefore, the implementation of those two applications was left for future work. Currently, a correctly working firewall implementation is sufficient to prove the viablity of our approach. The second implementation of TCPDump was done as it is necessary to have such a tool when debugging a Layered IPSEC implementation and as it demonstrates an example of a passive application which only needs to decrypt packet headers and never has to encrypt them.

While security and efficient networks will always be in conflicts, the use of Layered IPSEC, we hope, can bring us one step closer to finding an acceptable balance.

BIBLIOGRAPHY

[1] S. Bellovin. Problem areas for the IP security protocols. *Proceedings of the Sixth Usenix UNIX Security Symposium*, July 1996.

[2] H. Kruse. Protocol interactions and their effects on Internet based e-commerce. *Second International Conference on Telecommunications and Electronic Commerce (ICTEC)*, October 1999.

[3] S. Kent and R. Atkinson. Security architecture for the Internet protocol. ftp://ftp.isi.edu/in-notes/rfc2401.txt, November 1998.

[4] CISCO Systems. An introduction to IP security(IPSEC) encryption. http://www.cisco.com/warp/public/105/IPSECpart1.html, 1992.

[5] SSH. SSH IPSEC express. http://www.ipsec.com/tech/whitepapers/ipsec-wp.pdf, March 1999.

[6] IBM. Using IPSEC to construct secure virtual private networks. http://www-4.ibm.com/software/network/library/whitepapers/vpn/, 1998.

[7] G. Labouret. IPSEC: A technical overview. http://www.hsc.fr/ressources/veille/ipsec/papier/papier.html.en, April 1999.

[8] M. Dobruki. Security project at the TCM laboratory.
http://www.tcm.hut.fi/Tutkimus/IPSEC/ipsec.html, 1997.

[9] CyLAN Technologies Inc. CyLAN IPSEC white paper.
http://www.cylan.com/files/whpaper.htm, 1997.

[10] S. Kent and R. Atkinson. IP authentication header(AH).
ftp://ftp.isi.edu/in-notes/rfc2402.txt, November 1998.

[11] R. Rivest. The MD5 message-digest algorithm.
http://www.ietf.org/rfc/rfc1828.txt, April 1992.

[12] P. Metzger and W. Simpson. IP authentication using keyed SHA.
http://www.ietf.org/rfc/rfc1852.txt, September 1995.

[13] M. Oehler and R. Glenn. HMAC-MD5 IP authentication with replay prevention.
http://www.ietf.org/rfc/rfc2085.txt, February 1997.

[14] S. Kent and R. Atkinson. IP encapsulating security payload(ESP).
ftp://ftp.isi.edu/in-notes/rfc2406.txt, November 1998.

[15] P. Karn, P. Metzger, and W. Simpson. The ESP DES-CBC transform.
http://www.ietf.org/rfc/rfc1829.txt, August 1995.

[16] P. Karn, P. Metzger, and W. Simpson. The ESP triple DES transform.
http://www.ietf.org/rfc/rfc1851.txt, September 1995.

[17] G. Apostolopoulos, V. Peris, P. Pradhan, and D. Saha. L5: A self learning layer-5 switch. *IBM Research Report*, April 1999.

[18] V. Bharadwaj, J. Baras, and N. Butts. Internet service via broadband satellite networks. *Proceedings of the SPIE, vol. 3528, 169-180.*, pages 169–180, February 1999.

[19] V. Bharadwaj. Improving TCP performance over high-bandwidth geostationary satellite links. Master's thesis, Center for Satellite and Hybrid Communication Networks(CSHCN), University of Maryland, College Park, Maryland, 1999.

[20] W. Stallings. *Cryptography and Network Security.* Prentice Hall, Inc., 2 edition, 1998.

[21] R. Oppliger. *Internet and Intranet Security.* Artech House, Inc., 1 edition, 1998.

[22] U. Ellerman. IPv6 and firewalls. http://www.cert.dfn.de/eng/team/ue/fw/ipv6fw, 1996.

[23] L. Vepstas. Linux network address translation. http://linas.org/linux/load.html, November 1998.

[24] S. Bellovin. Transport friendly ESP or layer violations for fun and profit. 44th IETF, Minneapolis, http://www.research.att.com/ smb/talks/tf-esp.pdf, March 1999.

[25] Y Zhang. Multi-layer protection scheme for IPSEC.
http://www.wins.hrl.com/people/ygz/ml-ipsec/draft-zhang-ipsec-mlipsec-00.txt, October
1999.

[26] S. Bellovin. Proable plaintext cryptanalysis of the ip security protocols. *Proceedings of the 1997 Symposium on Network and Distributed Systems Security*, 1997.

[27] Linux frees/wan overview.
http://www.xs4all.nl/ freeswan/freeswan_trees/freeswan-1.00/doc/overview.html.

[28] P. Kocher. Breaking DES. *CryptoBytes, Vol 4. No. 2*, pages 1–5, Winter 1999.

[29] R. Rivest. The RC5 encryption algorithm. *Proceedings of the 1994 Leuven Workshop on Fast Software Encryption*, pages 86–96, 1994.

[30] B. Kaliski and Y. Yin. On the security of the RC5 encryption algorithm. *RSA Laboratories Technical Report TR-602*, September 1998.

[31] B Howard and R. Baldwin. The ESP RC5-CBC transform.
http://rsa.com/pub/rc5/draft/rsadsi-esp-rc5-00.txt, March 1996.

[32] P. Russel. Linux IPCHAINS-HOWTO.
http://www.rustcorp.com/linux/ipchains/HOWTO.html, March 1999.