2005

A framework for IPSec functional architecture.

Mehdi. Fahandezh
University of Windsor

A Framework for IPSec Functional Architecture

by

Mehdi Fahandezh

A Thesis
Submitted to the Faculty of Graduate Studies and Research
through the Department of Electrical and Computer Engineering
in Partial Fulfillment of the Requirements for the
Degree of Master of Science
at the University of Windsor
Windsor, Ontario, Canada
2005

Your file Votre référence
ISBN: 0-494-09865-1
Our file Notre référence
ISBN: 0-494-09865-1

ABSTRACT

In today's network, various stand-alone security services and/or proxies are used to provide different security services. These individual security systems implementing one single security function cannot address security needs of evolving networks that require secure protocol such as IPSec.

In this paper, we provide a framework for implementing IPSec security functions in a well structured functional architecture. The proposed architecture is modular and allows for composing software applications from products commercially available and developed by different suppliers to implement the entire security requirements of IPSec protocol.

In addition the proposed architecture is robust in the sense that it supports open standards and interfaces, and implements security functions of IPSec as an integrated solution under a unified security management system.

DEDICATION

I dedicate this work to my wife, Rose (Zohreh Heshmatzadeh), who supported me all the way and stands beside me. Her enthusiasms for my future have always kept me going and her prayers are always with me. Her patient is most recognized and appreciated.

I also dedicate my work to my family, who always give me boundless supports, especially to my sons, Arash and Arman, to encourage them to reach for higher education and for better life.

Finally, I would dedicate not only this work but also my entire success, to my beloved father. I would consider myself extremely lucky and successful if I could be half of what he was.

ACKNOWLEDGEMENT

I would like to thank my academic advisor, Prof. Shervin Erfani, for his guidance and support, especially for letting me to use his patent on "5-Layer Security Architecture", and lead me all the way through to achieve the novelty of "A Framework for IPSec Functional Architecture".

A special thanks to Mr. Karim Zarafshan for his active support. His encouragements were always my guiding light during the time of my Master's Degree studies at the University of Windsor.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

3DES	Triple DES
ABR	Area Border Router
ACK	Acknowledgment
ACL	Access Control List
AD	Adminstrative Distance
AH	Authentication Header
ARP	Advance Research Project Agency
ARPA	Address Resolution Protocol
AS	Autonomous System Border Router
ASCII	American Standard Code For Information Interchange
ASN.1	Autonomous System Number
ASBR	Abstract Syntax Notation
ATM	Asynchronous Transfer Mode
ATT	Attached
BGP	Border Gateway Protocol
C	Customer
CA	Certificate Authority
CBC	Cipher Block Chaining
CE	Cisco Express Forwarding
CEF	Cipher Feedback
CFB	Customer Edge
CIDR	Classless Inter-Domain Routing
CLNS	Connectionless Network Service
CONFED	Confederation
CoS	Class of Service
CPE	Customer Premises Equipment
CPU	Central Processing Unit
CRL	Constraint-Based Label Distribution Protocol
CR-LDP	Certificate Revocation List
CUG	Closed User Group
DER	Distinguished Encoding Rules
DES	Data Encryption Standard
DF	Don't Fragment
D-H	Diffie-Hellman
DN	Distinguished Name
DoD	Domain of Interprotation
DOI	Department of Defense
DoS	Denial of Service
DS	Digital Signal
DSL	Digital Subscriber Line
EBGP	External BGP
ECB	Electrnic Code Book
EGP	Exterior Gateway Protocol
EIGRP	Enhanced Interior Gateway Routing

ES	End System
ESP	Encapsulating Security Payload
FEC	Forwarding Equivalence Class
FIB	Forwarding Information Base
FQDN	Fully Qualified Domain Name
GRE	Generic Routing Encapsulation
HDLC	High-Level Data Link Control
HMAC	Hashed Message Authentication Code
HSRP	Hot Standby Router Protocol
HTTP	Hypertext Transfer Protocol
IANA	Internet Assigned Number Authority
IBGP	Internal BGP
ICMP	Internet Control Message Protocol
ID	Identifier
IEEE	Institute of Electrical And Electronics Engineers
IETF	Internet Engineering Task Force
IG	Inside Global
IGP	Interior Gateway Protocol
IKE	Internet Key Exchange
IL	Inside Local
IMP	Interface Message Processors
IOS	Internetwork Operating System
IP	Internet Protocol
IPSec	IP Security
IPv4	Internet Protocol Version 4
IPv6	Internet Protocol Version 6
IPX	Internetwork Packet Exchange
IS	Intermediate System
ISAKMP	Internet Security Association And Key Mangement Protocol
ISDN	Intergrated Services Digital Network
IS-IS	Intermediate System To Intermediate System
ISO	International Organization For Standardization
ISP	Internet Service Provider
IV	Initialization Vector
KE	Key Exchange
L1	Level 1
L2	Level 2
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
LDP	Label Distribution Protocol
LFIB	Label Forwarding Information Base
LLC	Logical Link Control
LSA	Link-State Advertisement
LSDB	Link-State Database
LSP	Label Switched Path
LSR	Label Switch Router

MAC	Message Authentication Code
MD5	Message Digest 5
MED	Multi-Exit Discriminator
MIM	Man-In-Middle
MM	Main Mode
MOD	Modulus
MP-BGP	Multiprotocol BGP
MP-IBGP	Multiprotocol Internal BGP
MPLS	Multiprotocol Label Switching
MSS	Maximum Segment Size
MTU	Maximum Transfer Unit
NAT	Network Address Translation
NBMA	Nonbroadcast Multiaccess
NET	Network Entiry Title
NH	Next Hop
NLRI	Network Layer Reachability Information
NSA	National Security Agency
NSAP	Network Service Access Point
NSSA	Not-So-Stubby Area
NTP	Network Time Protocol
NVRAM	Nonvolatile Random Access Memory
OFB	Output Feedback
OG	Outside Global
OL	Outside Local
OSI	Open System Interconnect
OSPF	Open Shortest Path First
P	Provider
PAT	Port Address Translation
PC	Portable Computer
PE	Provider Edge
PEM	Privacy Enhanced Mail
PFS	Perfect Forward Secrecy
PHP	Penultimate Hop Popping
PKCS	Public-Key Cryptography Standard
PKI	Public Key Infrastructure
PMTU	Path Maximum Transfer Unit Discovery
PoS	Packet over Sonet
PPP	Point-To-Point Protocol
PRC	Partial Route Calculation
PREF	Preference
QM	Quick Mode
QoS	Quality of Service
RA	Registration Authority
RD	Route Distinguisher
RFC	Reequest For Comment
RIP	Routing Information Protocol

RIPv2	Routing Information Protocol version 2
RR	Route Reflector
RSA	Rivest-Shamir-Adleman
RSVP	Resource Reservation Protocol
RT	Route-Target
SA	Security Association
SADB	SA Database
SCEP	Simple Certificate Enrollment Protocol
SDH	Synchronous Digital Hierarchy
SHA	Secure Hash Algorithm
SLA	Service-Level Algorithm
SLA	Service-Level Agreement
SONET	Synchronous Optical Network
SOO	Site-of-Orgin
SP	Service Provider
SPD	Selective Packet Discard
SPF	Shortest Path First
SPI	Security Parameter Index
TCP	Transmission Control Protocol
TDM	Time Division Multiplexing
TE	Traffic Engineering
TED	Tunnel Endpoint Discovery
TLV	Type, Length, And Value
ToS	Type of Service
TTL	Time-to-Live
UDP	User Datagram Protocol
VC	Virtual Circuit
VPDN	Virtual Private Dialup Network
VPN	Virtual Private Network
VPNv4	VPN-Ipv4
VRF	Virtual Routing And Network
WAN	Wide Areas Network
WWW	World Wide Web
XOR	Exclusively OR

CHAPTER I:

1.1 Introduction

In the mid 1960's [1] in the height of the cold war, the Department of Defense (DoD) in the USA wanted a command and control network that could survive a nuclear war. The DoD consequently commissioned its research arm- ARPA (Advance Research Project Agency)- to invent the technology that could get data to its destination reliably even if arbitrary part of the network disappeared without warning as a result of a nuclear attack. Traditional telephone technology called circuit switching would not work because it had serious drawbacks. In circuit switching, a route for data to get from one place to another is set up by using relays to make physical connections among pieces of cable. Therefore, if a part of the circuit fails, a new circuit must be set up, which could be quite difficult and time consuming depending on the severity of the damage [2].

ARPA used a different kind of technology called packet switching. The idea of packet switching network was proposed by Paul Baran [3]. In packet switching, data to be sent over the network is divided up into discrete parts called packets. Each packet is routed independently from one computer to the next over the network until it reaches its final destination.

The first experimental network- called the ARPANET – went into operation in December 1969. It consisted of subnets and host computers. The subnets consisted of minicomputers called IMPs (Interface Message Processors) connected by transmission lines. This network contained four nodes, one each at UCLA (University of California at Los Angeles), UCSB (University of California at Santa Barbara), SRI(Stanford Research Institute) and University of Utah. Each node of the network composed of an IMP and

host in the same room, connected by wires. These four sites were chosen because all had large ARPA contract; additionally, all four sites had different and completely incompatible computers. This experimental network grew rapidly: in July 1970 it grew to eight nodes, by March 1971 it expanded to sixteen nodes, in April 1972 it grew to twenty three nodes and by September 1972 it consisted of thirty four nodes [1].

In the 1983 TCP/IP protocols were adopted as the only official protocol of the ARPANET. During the same year the term *Internet* came into common usage. The old ARPANET was divided into MILNET, the unclassified part of the Defense Data Network (DDN), and a new smaller ARPANET. The "Internet" was used to refer to the entire network: the MILNET plus the ARPANET [1]. Many regional networks in the USA joined up to the Internet. In the mid 1980's trans-Atlantic fiber optic cables were laid and consequently, networks in Europe, the Pacific region and Canada were connected to the Internet. Growth of the Internet continued exponentially, and by 1995 there were multiple backbones, hundreds of regional networks, tens of thousands of Local Area Networks (LAN's), millions of hosts, and tens of millions of users. At the close of the twentieth century almost every nation was connected to the Internet.

The nature of the Internet has raised many security concerns. The principal concern over the years is the fact that Internet Protocol (IP) packets are inherently insecure. It is relative easy to forge IP addresses, modify the contents of IP packets, replay old content and inspect the content of packets in transit. Therefore, there is no guarantee that the IP datagrams originated from the source it claims to originate from, or that they will get to the intended destination, or that the contents have not been modified or examined by a third party while they were in transit from the source to the destination.

These security problems have caused many companies to stop using the Internet to transmit electronic data. These companies turned to leased lines and private networks. The Internet Protocol Security protocol developed by Internet Engineering Task Force (IETF) in the late 1990 has addressed most of the IP datagram. The Internet Architecture Board (IAB) in 1994 issued a report identifying issues concerning security needs of the network infrastructure [4]. In response to these issues IPSec has been developed.

The purpose of this thesis is to implement the IPSec by using Functional Architecture Layers. First, however, it is important to have a clear understanding of the related algorithms, technologies, and operational processes. The following chapters will cover the concept of security systems, and the operations of IP Security Protocol (IPSec) and Internet key exchange (IKE), as well as their associated technologies, in depth. These associated technologies should be thoroughly understood in order to deploy a robust IPSec security policy and to implement fully operational IPSec functions.

IPSec is implemented at the network layer. It protects and authenticates IP packets between participating IPSec devices (peers) such as firewalls, routers, clients, and other IPSec-compliant products. IPSec is a framework of open standards, which is not formally limited to any specific set of algorithms. As such, IPSec allows for newer and better algorithms to be implemented without patching the existing standards, and it provides data confidentiality, data integrity, and data origin authentication between

participating peers at the IP layer. In the following sections, we will look at the definitions of these security services and describe the technologies associated with them.

IPSec provides the capability to secure communications across a LAN, across private and public WANs, and across the Internet. In addition, IPSec can play a vital role in the routing architecture required for internetworking [5]. For example, routing protocols such as OSPF (Open Shortest Path First) should be run on top of any security association (SA) between routers that are defined by IPSec. It is seen that the IPSec specification has become quite complex. The numerous documents defining IPSec do not provide a framework for its implementation. The various vendors of traditional security systems are implementing some of the IPSec functionality and security services relying on their own mostly *nonstructured* functional architecture. This nonstructured implementation can only provide some sort of piecemeal implementation, and cannot address security demanded by efficient implementation of IPSec. The key to successful IPSec functional architecture is to consider the security management as an evolving integrated part of the architecture.

This thesis is organized into five chapters.

First three chapters provide necessary background information for this topic stating the problem and thesis contribution. They also provide basic concepts of network security threats, TCP/IP overview, and cryptography tools and techniques.

Chapter IV, which is the body of the thesis, provides the design of the functional structures of IPSec in an integrated form of modular architecture.

Chapter V provides some security session scenarios which will demonstrate the execution of the designed IPSec syste in an IPSec environment.

Finally, we conclude this work by summarizing the results, and making suggestions for future work in Chapter VI.

1-2 Network Security

1-1.1 Introduction to Network Security:

The Internet is changing the way we work, live, play, and learn. These changes are occurring both in the ways that we currently experience (e-commerce, real-time information access, e-learning, expanded communication options, and so forth), and in ways we have yet to experience. Imagine a day when your enterprise can make all its telephone calls over the Internet for free. Or perhaps on a more personal note, consider logging on to a daycare provider's website to check how your child is doing throughout the day. As a society, we are just beginning to unlock the potential of the Internet. But with the Internet's unparalleled growth comes unprecedented exposure of personal data, critical enterprise resources, government secrets, and so forth. Every day hackers pose an increasing threat to these entities with several different types of attacks. These attacks, outlined in the next section, have become both more prolific and easier to implement.

There are two primary reasons for this problem. First is the ubiquity of the Internet. With millions of devices currently connected to the Internet, and millions more on the way, a hacker's access to vulnerable devices will continue to increase. The ubiquity of the Internet has also allowed hackers to share knowledge on a global scale. A simple Internet search on the words "hack," "crack," or "phreak" yields thousands of sites, many of which contain malicious code, or the means with which to use that code.

Second is the pervasiveness of easy-to-use operating systems and development environments. This factor has reduced the overall ingenuity and knowledge required

by hackers. A truly remarkable hacker can develop easy-to-use applications that can be distributed to the masses. Several hacker tools that are available in the public domain merely require an IP address or host name and a click of a mouse button to execute an attack [6].

1-1.2 Introduction to Security Threats

A *threat* or *security attack* is a potential violation of security or an intrusion for unauthorized, illegitimate, malicious or fraudulent purposes. These attacks are aimed to compromise security. The *points of attack* (or *attacking points*) can occur at various weakness points within a *security perimeter*, and can be at any level or layer of realization, e.g., at the physical system realization level, at the system or network level, at the communication protocol level, and so on.

Table 1.1 lists some known attacks and their brief descriptions. These are examples of specific threats that a security management system may need to counter. The nature of the attacks varies with the circumstances and according to the defined perimeter for the security. Threats may be classified by their *type* (e.g., accidental or intentional, passive or active), by their *consequences*, or by their *sources* (e.g., users or programs) and *objects* of threats. Our purpose is not to categorize various known (or unknown) threats. This is beyond the scope of this study. However, we need to be aware of all potential threats (at all conceivable levels of all natures) in order to come up with *security services* to counter them. Typical intentional known threats and the corresponding security services to counter them are listed in Table 1.1 The goal of this study is to develop an architecture

that counters the intentional threats by implementing appropriate service(s) for the given security domain.

THREAT OR VIOLATION ***(SECURITY ATTACK)***	**DESCRIPTION**	**CONSEQUENSES**	**DEFENSE** ***(REQUIRED SECURITY SERVICE)***
Eavesdropping	A passive attack for disclosure of information	Interception; leak of information; loss of confidentiality; loss of anonymity; traffic analysis deduction	Confidentiality
Masquerade	Spoofing, a bogus party aims to dupe legitimate parties into giving up sensitive information	Misuse of credentials or security parameters; incorrectly executed sign-on procedure; false claim of origin of parameters	Authentication
Repudiation	An authorized involved party subsequently denies the transaction	Denial of origin; denial of delivery; denial of submission, denial of a specific action	Non-repudiation
Forgery or fraud	A party fabricates information and claims it came from another party	Fabrication; insertion of spurious information in a network or to a file	Data Integrity, Access Control, Authentication
Non-authorized access	Attempts to conform to the rules of authentication and access control to gain information in violation to the security policy in force	Modifying the files; performing actions outside of the security profile; illegal associations	Authentication, Access Control
Denial of Service & Interruption	Legitimate access to secured entities is deliberately impeded	A party prevents others from performing their functions; flooding the network with faked error messages; denial of access, denial of communication; deliberate suppression of a particular party	Availability, Authentication, Access Control , Integrity
Trapdoor	A built-in feature that allows security policy to be violated	Intentional or unintentional violation of security services	Firewalling, anti-virus, authentication access control
Modification of Programs (Trojan horse, viruses, and worms)	Software with an invisible malicious part, when executed, compromises the security of the user party	Destruction of the operating systems, communication software, or user application software	Firewalling, anti-virus program, data integrity, authentication
Message modification, data loss & corruption	Integrity compromised by unauthorized deletion, insertion, modification, replay or delay	Modification or deletion of messages, information, and files;	Integrity

Table 1.1: Security Services to Counter Security Threats

Not given in Table 1.1, are famous security products/solutions, which are designed for a particular environment or for a special application. They are considered as *custom-designed* combinations of the above services. Examples of these are:

1. *PGP (Pretty Good Privacy)* - a widely used authentication and confidentiality service.
2. *Kerberos* - an authentication protocol based on conventional encryption to authenticate clients to servers, and vice versa. The Version 5 Kerberos was developed within the Internet community.
3. PEM (Privacy Enhancement Mail)- developed specifically as an Internet Standard for electronic mail.

For illustration purposes, some typical threats associated with various business environment are shown in Table 1.2.

SECURITY REQUIREMENTS	THREATS
Enterprise Network:	
Outside prevention (hackers)	Masquerade, data corruption
Ensure message authenticity	Eavesdropping
Protect corporate/individual privacy	
Banking:	
Fraud protection, prevention of data modification	Forgery or fraud
Identify legitimate parties	unauthorized access
Identify legitimate transactions	repudiation, masquerade
Protect PINs from disclosure	Eavesdropping
Ensure parties' privacy	
Government:	
Protection against unauthorized disclosure or manipulation of information.	Masquerade, unauthorized access, eavesdropping, data loss and corruption
Provide electronic signature on documents	Repudiation
Public Networks:	
Restrict access to management & administration functions	Masquerade, unauthorized access
Protect against service interruptions	Service denial
Protect users' privacy	Eavesdropping
Electronic Commerce:	
Ensure the source and integrity of transaction	Masquerade, data loss and corruption
Protect transaction privacy	Eavesdropping
Provide electronic signatures on transactions	Repudiation

Table 1.2: Typical Threats in Particular Application Environments

1.1.3 Common type of network Attacks

Without security, both public and private networks are susceptible to unauthorized monitoring and access. Internal attacks might be a result of minimal or nonexistent intranet security; while risks from outside the private network stem from connections to the Internet and extranets. Password-based user access controls alone do not protect data transmitted across a network.

Without security measures and controls in place, the data might be subjected to an attack.

- Passive attacks, meaning information is monitored,
- Active attacks, meaning the information is altered with intent to corrupt or destroy the data or the network itself.

The networks and data are vulnerable to any type of attacks listed in Table 1.1, as described in the following bullet items:

- **Eavesdropping**

  In general, the majority of network communications occur in an unsecured or "cleartext" format which allows an attacker who has gained access to data paths in your network to "listen in" or interpret (read) the traffic. When an attacker is eavesdropping on your communications, it is referred to as sniffing or snooping.

  The ability of an eavesdropper to monitor the network is generally the biggest security problem that administrators face in an enterprise. Without strong encryption services that are based on cryptography, the data can be read by others as it traverses the network.

- **Data Modification**

  After an attacker has read the data, the next logical step is to alter it. An attacker can modify the data in the packet without the knowledge of the sender or receiver. Even if the network does not require confidentiality for all communications, you do not want any of messages to be modified in transit. For example, if we are exchanging purchase requisitions, we do not want the items, amounts, or billing information to be modified.

- **Identity Spoofing (IP Address Spoofing)**

  Most networks and operating systems use the IP address of a computer to identify a valid entity. In certain cases, it is possible for an IP address to be falsely assumed— identity spoofing. An attacker might also use special programs to construct IP packets that appear to originate from valid addresses inside the corporate intranet. After gaining access to the network with a valid IP address, the attacker can modify, reroute, or delete the data. The attacker can also conduct other types of attacks, as described in the following bullet item.

- **Password-Based Attacks**

  A common denominator of most operating system and network security plans is password-based access control. This means access rights to a computer and network resources are determined by the user name and their passwords.

Older applications do not always protect identity information as it is passed through the network for validation. This might allow an eavesdropper to gain access to the network by posing as a valid user. When an attacker finds a valid user account, the attacker has the same rights as the real user. Therefore, if the user has administrator-level rights, the attacker also can create accounts for subsequent access at a later time. After gaining access to the network with a valid account, an attacker can do any of the following:

- Obtain lists of valid user and computer names and network information.
- Modify server and network configurations, including access controls and routing tables.
- Modify, reroute, or delete any data.

- **Denial-of-Service Attack**

  Unlike a password-based attack, the denial-of-service attack prevents normal use of a computer or network by its valid users. After gaining access to the network, the attacker can do any of the following:

  - Randomize the attention of the internal IS staff so that they do not see the intrusion immediately, which allows the attacker to make more attacks during the diversion.
  - Send invalid data to applications or network services, which causes abnormal termination or behavior of the applications or services.

- Flood a computer or the entire network with traffic until a shutdown occurs because of the overload.
- Block traffic, which results in a loss of access to network resources by authorized users.

- **Man-in-the-Middle Attack**

  As the name indicates, a man-in-the-middle attack occurs when someone between a legitimate user and the person with whom this user is communicating is actively monitoring, capturing, and controlling the communication transparently.

  For example, the attacker can re-route a data exchange. When computers are communicating at low levels of the network layer, the computers might not be able to determine with whom they are exchanging data.

  Man-in-the-middle attacks are like someone assuming your identity in order to read your message. The person on the other end might believe it is you because the attacker might be actively replying *as you* to keep the exchange going and gain more information. This attack is capable of the same damage as an application-layer attack, described later in this section.

- **Compromised-Key Attack**

  A key is a secret code or number necessary to interpret secured information. Although obtaining a key is a difficult and resource-intensive process for an attacker, it is possible. After an attacker obtains a key, that key is referred to as a compromised key. An attacker uses the *compromised key* to gain access to a

secured communication without the sender or receiver being aware of the attack. With the compromised key, the attacker can decrypt or modify data, and try to use the compromised key to compute additional keys, which might allow the attacker access to other secured communications.

- **Sniffer Attack**

  A *sniffer* is an application or device that can read, monitor, and capture network data exchanges and read network packets. If the packets are not encrypted, a sniffer provides a full view of the data inside the packet. Even encapsulated (tunneled) packets can be broken open and read unless they are encrypted and the attacker does not have access to the key. Using a sniffer, an attacker can do any of the following:

  - Analyze the network and gain information to eventually cause the network to crash or to become corrupted.
  - Read the communications.

- **Application-Layer Attack**

  An application-layer attack targets application servers by deliberately causing a fault in a server's operating system or applications. This results in the attacker gaining the ability to bypass normal access controls. The attacker takes advantage of this situation, gaining control of the application, system, or network, and can do any of the following:

  - Read, add, delete, or modify the data or operating system.

- Introduce a virus program that uses computers and software applications to copy viruses throughout your network.
- Introduce a sniffer program to analyze the network and gain information that can eventually be used to crash or to corrupt systems and the network.
- Abnormally terminate the data applications or operating systems.
- Disable other security controls to enable future attacks.

1.2 TCP / IP Overview

1.2.1 Introduction

The Transport Control Protocol/ Internet Protocol (TCP/IP) suite proved to be quite robust and was very adaptable to the different networks. This along with the fact that it was an open protocol, it was freely available, it was developed independently of any specific computer hardware or operating system, and it was simple and easy to implement made it very popular. By 1983, TCP/IP was integrated into release 4.2 of Berkeley Software Distribution (BSD) UNIX, and the same year, the DoD adopted this protocol suite as Military Standard (MIL STD) and it became the standard of the Internet.
Today, TCP/IP continues to be the standard for the Internet.

The fact that this protocol was designed to be independent of any specific physical network hardware has allowed it to be integrated into many different kinds of networks. TCP/IP is currently integrated into Ethernets, token ring, dialup networks and just about every other type of physical transmission medium, and virtually all modern operating systems.

1.2.2 TCP/ IP Protocol architecture

Networking protocols are normally developed in layers, with each layer responsible for a different facet of the communication [7]. The TCP/IP protocol suite consists of four layers. Each layer has distinct functions and consists of a consists of a combination of different protocols as shown in Figure 1.1. The functionalities of the layers are discussed below.

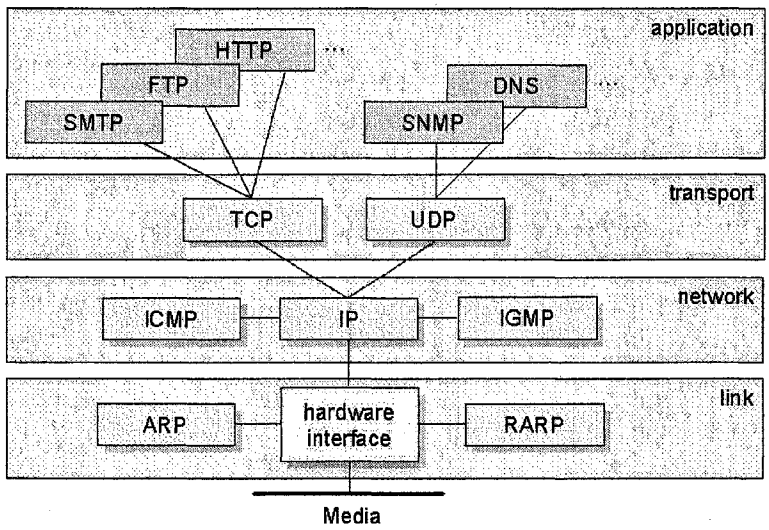


Figure 1.1: Protocols at the four layers of the TCP/IP protocol suite.

- **Data-link Layer**

The data-link layer is also called the data link or network access layer. It is the lowest layer of the TCP/IP protocol stack. Example of data-link layers are Ethernet, Token Ring and ATM (Asynchronous Transfer Mode). For information on Ethernet refer to [8], for information on Token Ring see[9], and for further detail on ATM see [10], [11] and [12]. The protocols in data-link layer provide means for the system to deliver data to other devices on the network. These protocols are concerned with details such as the packet structure and addressing scheme of the underlining network. Two of the protocols in this layer are ARP (Address Resolution Protocol), which maps IP addresses to Ethernet addresses; and RARP (Reverse Address Resolution Protocol), which maps Ethernet address to IP address. See [13] and [14] for further information on these protocols.

When the data-link layer receives a packet from the network layer the layer, above it, encapsulates the packet with the appropriate header then sends it via the physical

network to the specified device. The reverse occurs when a packet arrives from the physical network to the data link layer: the layer removes the data-link header then sends the packet up to the network layer for processing. This is illustrated in Figure 1.2.

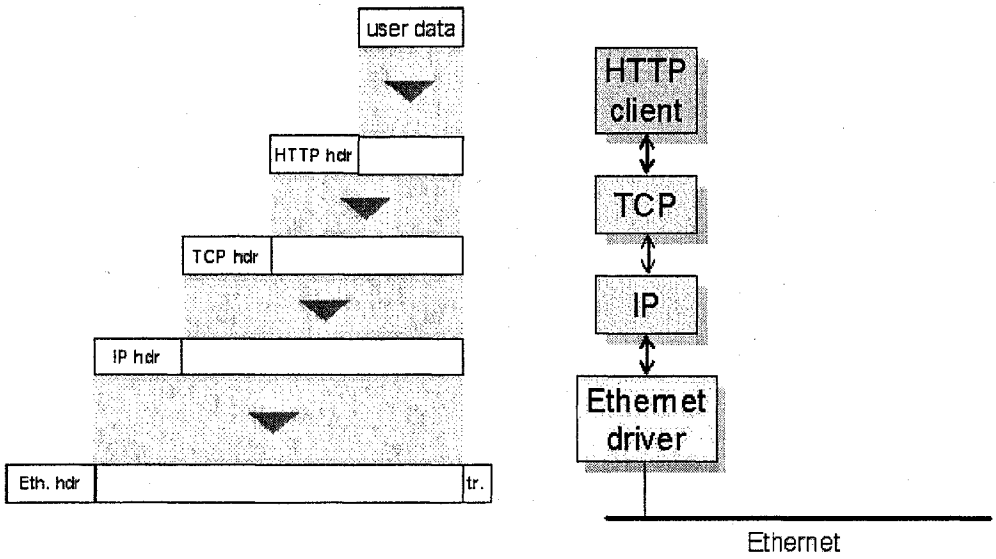


Figure 1.2: Encapsulation and de-capsulation of data as it goes down and up the protocol stack, respectively.

- **Network Layer**

The network layer is also called the internet layer. This layer is responsible for the routing of packets from the source host to the destination host. The network layer consists of three protocols: Internet Control Protocol (ICMP), Internet Group Management Protocol (IGMP) and Internet Protocol (IP).

The official specification of ICMP is outlined in [15]. ICMP is the protocol that is used to communicate error conditions that occurred during the transmission of IP packets. The ICMP message is usually encapsulated in the IP datagram. Another important role of

ICMP is the debug of network connectivity problems. The popular debugging tools are ping and traceroute using this protocol.

IGMP is the protocol that is used by hosts and routers to ascertain information about the hosts in multicast groups; i.e., groups of hosts to which IP datagrams are to be sent simultaneously. As is the case with ICMP, IGMP messages are encapsulated in IP datagrams. For further detail on IGMP refer to [16] which contains the original specification of IGMP.

IP is the protocol that holds the whole TCP/IP protocol suite architecture together. The data of the protocols in all of the layers of the TCP/IP protocol stack, except the data-link layer, are transmitted as IP datagrams. IP allows hosts to inject packets into the data-link layer, which eventually puts them on the physical network and has them travel on potentially different networks to their final destination. IP offers a connectionless service. In connectionless services, each message or datagram carries the full destination address, and is routed independently of the other datagrams. With connectionless services, it is possible for the datagrams to arrive at their destination in different order that they were sent. This is not possible with connection-oriented services, since the latter is like a telephone system which establishes a connection or a path from the host to the destination uses the path to send data, and then releases the connection after the data transmission is completed.

There are two versions of IP that are available for use with the TCP/IP protocol, IPv4 and IPv6, as discussed in Appendix A1.

- **Transport Layer**

The layer above the network layer in the TCP/IP protocol stack is the transport layer. This layer provides a flow of data between two hosts, for the application layer above. Two protocols are implemented at the transport layer, 1) TCP (Transport Control Protocol) and 2) UDP (User Datagram Protocol). TCP is a reliable connection-oriented protocol that allows byte streams from a host to be delivered without error to any other hosts on networks that are reachable. The TCP protocol also breaks up received packets, re-assembles packet fragments it received from the network layer, and perform flow control to ensure that a fast hosts does not overwhelm a slower host with more packets that it can handle. See [17] for further detail on this protocol.

The UDP, unlike TCP, offers an unreliable and connectionless service. It does not perform any error checking or flow control; it simply sends datagrams from one host to another and does not provide any guarantee that a datagram will get to its destination. There is much less overhead involved in processing UDP datagrams compared to TCP packets. Consequently, the throughput of UDP datagrams is usually greater than that of TCP. UDP protocol is usually use by applications for which prompt delivery is more important than accurate delivery; such as in the transmission of speech or video. See [18] for outline of the original specification of UDP.

- **Application Layer**

The top of the TCP/IP protocol stack is the application layer. This layer includes all processes that use the transport layer protocols to deliver data. Application layer protocols usually provide services to the users of the system. There are many application protocols. The ones that just about all TCP/IP implementation provides include:

-Telnet: The Network Terminal Protocol, which provides remote login.

-FTP: File Transfer Protocol, which is used for file transfer to or from remote hosts.

- SMTP: Simple Mail Transfer Protocol, which delivers electronic mail.

- SNMP: Simple Network Management Protocol, which is used for monitoring network segments.

- DNS: Domain Name Service, which is used for mapping host names onto their IP addresses.

- HTTP: Hypertext Transfer Protocol, which is used for fetching web pages on the World Wide Web.

1.3 Cryptographic tools and Techniques

1.3.1 Introduction

There is no single cryptographic tool. There are various techniques for encrypting messages, for securely exchanging keys, for maintaining the integrity of messages, and for guaranteeing authenticity of a message. All of these techniques combined to provide the services necessary to keep secrets in an increasingly public world.

There is really no such thing as absolute security, but for every secret there is a threat, and analysis must be made to determine the viability of that threat and the damage that can be done if they are acted upon. Generally, the strength of a cryptosystem is measured by its complexity. If the complexity of a particular crypto-system is 2^{32} then it is assumed that 2^{32} separate operations are required to break it. That is a considerable number of operations, but if each operation can be performed by a modern computer in hundredths or thousandths of a second, the system might not be secure enough for the secret it is protecting. Because of this the term computationally secure is used to express the security of a modern cryptosystem.

1.3.2 Cryptographic Building Blocks

A good portion of public key cryptography relies upon a foundation of one-way functions and trapdoors. A one-way function is something that is easy to compute in one direction but difficult, bordering on impossible, to compute in the other direction. A trapdoor is a way to sneak back, in effect a way to cheat and return using a secret passage.

For a one-way function to be useful in cryptography it must exhibit its one way-ness with any input. For example, in a finite field it is easy to compute the product of numbers but difficult to factor that product. Another number example is the Discrete Logarithm Problem: with a larger prime, p, and a base generator, g, for a particular value y, find x where

$$g^{x} = y \bmod p$$

Modular exponentiation is easy, but doing a discrete logarithm to recover the exponent is hard. For any class of numbers: odd numbers, palindromic numbers, numbers divisible by 47 - the problem of solving the discrete logarithm is still very hard; see [19].

1.3.3 Encryption Concepts and Algorithms

1) Encryption Concepts

Unencrypted message is referred to as "plaintext". The process of camouflaging a message in such a way as to conceal its content is known as encryption. An encrypted message is referred to as "ciphertext". The process of converting ciphertext back into plaintext is known as decryption.

2) Cryptographic algorithm

A cryptographic algorithm, also called a cipher, is the mathematical function used for encryption and decryption. For better security, quality control and standardization, modern cryptography incorporates the concept of a key. This key can take many values (usually denoted in binary bits), and is analogous to the combination of a lock. Although the concept of a combination lock is well known, it is often difficult to open a combination lock without knowing the combination. In addition, the more numbers in a

given combination, the more time consuming it will be to guess the combination. The same is true for cryptographic keys; hence the more bits in a key the less susceptible a key is to being compromised. Therefore, the security robustness in modern day ciphers is based on the key rather than the details of the cipher known as encryption algorithm. This doesn't matter if an eavesdropper knows the algorithm; if the particular key is not revealed, the message cannot be read by anyone.

3) **Confidentiality**

Confidentiality is the security service that protects data and the external characteristics of communication from unauthorized disclosure by concealing source and destination addresses, message length or frequency of communication. With confidentiality, the intended recipients know what was being sent but unintended parties cannot determine what was sent. Encryption algorithms can be used to provide confidentiality.

There are two general types of key-based algorithms: symmetric and asymmetric.

- **Symmetric Algorithms**

  In symmetric algorithms, the encryption key can be calculated from the decryption key and vice versa. In this case, encryption and decryption use the same mathematical functions, and the encryption key and the decryption key are identical. Symmetric algorithms, also known as secret-key algorithms, require that the sender and receiver agree on a key before they can transmit and receive information securely, as it demonstrated in Figure 1.3.

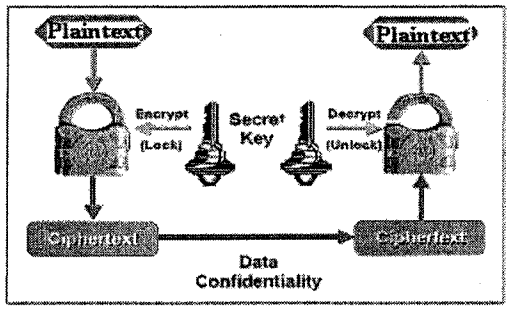


Figure 1.3: Schematic Example for Symmetric Cryptography

The security of a symmetric algorithm lies in the key; disclosing the key implies that anyone could encrypt and decrypt the messages. Therefore, it is mandatory that the key remain secret.

Symmetric algorithms are further categorized into stream and block ciphers. Block ciphers operate on one block of data (usually in 64-bit block size) at a time. Stream ciphers conversely operate on data one bit (or one byte) at a time. In either case, they are ideal for bulk encryption. As block ciphers are used only in IPSec, stream ciphers are beyond the scope of this study.

Cipher mode refers to a set of techniques used to apply a block cipher to a data stream. Four basic modes are used:

- Electronic code book (ECB)
- Cipher block chaining (CBC)
- Cipher feedback (CFB)
- Output feedback (OFB)

An example of a symmetric block in CBC mode is the Data Encryption Standard (DES). DES encrypts data in 64-bit blocks and the same algorithm and key are used for both encryption and decryption. The key length is 56 bits. The key can be

any 56-bit number and can be altered any time. DES breaks a massage into 64-bit blocks, XORs the first block with the Initialization Vector (IV), and encrypts it with the 56-bit key. The result is a 64-bit block of cipher text that is further XORed with the next 64-bit block. This process is repeated until the entire message has been encrypted.

There is a more secure variant of DES. called "3DES". In 3DES, pronounced "Triple-DES" the DES algorithm is applied 3 times to each 64-bit plaintext block, using a different 56-bit key each time. A typical approach uses two different 56-bit keys, K1 and K2, yielding a length of 112 bits, with K1 being reused for the final (third) encrypt. Other applications use three different 56-bit keys, K1, K2, and K3, and K3, yielding a total key length of 168 bits. Using the same key in all three steps will yield the same result as encrypting with conventional DES. Due to the second step being a decryption operation [20].

Over the years, a number of symmetric cryptosystems have been proposed as possible replacement for DES. Some of these cryptosystems will be introduced below.

- **Blowfish**: Blowfish was developed by Bruce Schneier [21]. It uses a key of variable length: from 32 to 448 bits to encrypt blocks of 64-bit plaintext into blocks of 64-bit ciphertext. Blowfish has a simple structure which makes it easy to implement. Currently, Blowfish is one of the fastest encryption systems. The variable length key allows for varied degree of

security. It is believed that Blowfish is very secure, particularly when longer keys are used.

- **CAST-128**: CAST was developed by Carlisle Adams and Stafford Tavares [22]. It uses a key that varies between 40 to 128 bits in length, in 8-bits increments; and produces 64-bit blocks ciphertext from blocks of 64-bit plaintext. The CAST algorithm involves 16 rounds of computations which are quite complex. This algorithm has received widespread review and it is believed to be quite safe.
- **International Data Encryption Algorithm (IDEA)**: IDEA was developed by Xuejia Lai and James Massey at the Swiss Institute of Technology (ETH in Zurich). The original specification and later modifications are outlined in [23] and [24] respectively. IDEA uses a 128-bit key and encrypts plaintext in blocks of 64 bits and output 64-bits blocks ciphertext. IDEA enciphering algorithm has received a fair bit of attention and it is believed that to be quite secure for a number of years.
- **RC2:** RC2 was developed by Ronald Rivest [25]. It is designed to be easily implemented on 16-bit microprocessors. The algorithm uses a key of variable length: from 8 bytes to 128 bytes and enciphers block of 64-bit plaintext to output blocks of ciphertext of 64 bits, inlength.
- **RC5:** RC5 was developed by Ronald Rivest [26]; a later description is given in [27]. RC5, like DES, is suitable for both hardware and software implementation. It is adaptable to processor of different word length. The number of bits in the processor's word is one of the inputs to the RC5

algorithm. RC5 algorithm also has variable number of rounds, which allows tradeoff between higher speed and higher security. The algorithm uses a variable length key to encrypt plaintext blocks of 32, 64 or 128 bits and output blocks of ciphertext of corresponding length. RC5 has a low memory requirement which therefore, makes it suitable to be used in applications such as smart cards that have limited memory.

- **Asymmetric Algorithms**

Asymmetric algorithms are designed so that the key used for encryption is different from the key used for decryption as it is shown schematically in Figure1.4.

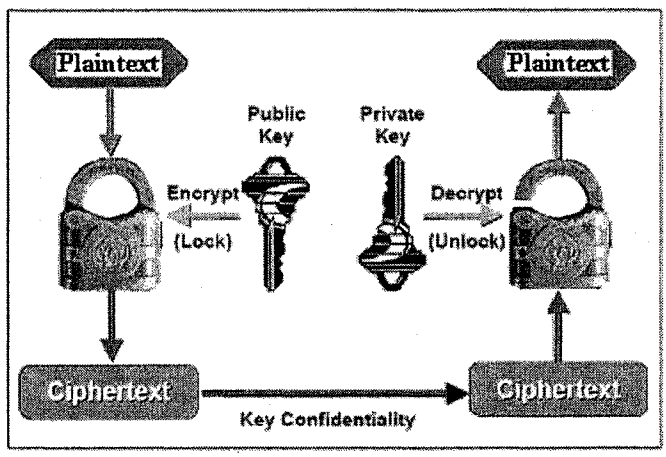


Figure 1.4: Schematic Example for Asymmetric Cryptography

In addition, the decryption key cannot be derived from the encryption key. These algorithms are referred to as public-key algorithms, because the encryption key is made public. In other words, anyone can use the encryption key (public key) to encrypt a message, but only a specific recipient with the corresponding decryption key (private key) can decrypt the message. In other instances the message can be

encrypted with private key and decrypted with the public key. This is typically used in digital signatures.

An example of a public key cipher is the RSA algorithm developed by Ron Rivest, Adi Shamir, and Leonard Adleman. The public and private keys are functions of a pair of large prime numbers. The security of this approach is based on the fact that it can be relatively easy to multiply these large prime numbers together but extremely difficult to factor the resulting product. In other words, anyone attempting to recover the text plain from the cipher text and the public key will have to go through the knotty task of guessing the decryption (private) key by factoring the resulting product of two large prime numbers. RSA uses a key length of 512 bits, 768 bits, 1024 bits, or longer. A longer key length offers stronger security, but also uses up more CPU processing cycles. This is due to the fact that the mathematical computations in the RSA algorithm are much more complicated than the DES algorithm [28]. As a result, in hardware, RSA is about 1000 times slower than DES, and in software, DES is up to 100 times faster than RSA. These numbers may change slightly as technology advances, however, RSA will never approach the speed of symmetric algorithms, which makes it an unlikely candidate for bulk data encryption [29].

Public-key algorithms such as RSA are rarely used for data confidentiality (encryption) because of the performance constraints mentioned earlier. Instead, they are most often used in applications involving authentication via digital signatures and key management. In real life, public key algorithms are not

surrogates for symmetric algorithms. They are typically used to encrypt keys instead of messages.

1.3.4 Hashing Concepts and Algorithms

1) Hashing Concepts

A one-way hash function is a mathematical function that is easy to compute in the forward direction, but computationally infeasible to reverse. An input message runs through the mathematical function (the hash function) and results in a string of output bits called a hash (or a message digest). The process is irreversible, that is, the input cannot be derived from the output. It is analogous to a coffee grinder. Imagine the message is the coffee beans and the output hash is the resulting ground coffee. It is quite impossible to take the ground coffee and recreate the original coffee beans. In addition, a hash function takes a message of arbitrary length and produces a fixed-length output [30].

It is important to implement keyed hashing because only performing a one way hash on some data does not provide any authentication. A hash function alone is like a checksum. Anybody can modify the data and just run the hash algorithm over the modified data. A keyed hash function is a message authentication code (MAC), which, in other words, is a one-way hash function encrypted using private key. A MAC is generated through hashing a shared privet key along with message.

The concept is the same as hash functions, except now only the holder of the secret key can verify the hash value. Figure 1.5 illustrates how keyed hashing works [31].

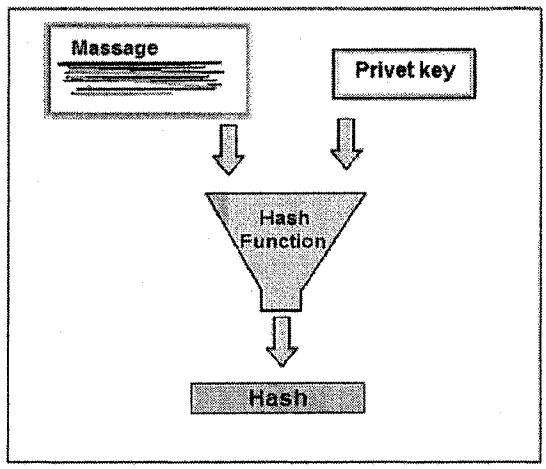


Figure 1.5: A keyed hashing function (MAC)

Keyed hashing can be used to generate a hash (MAC) on each IP packet in an IP data stream. These hashes then become part of the IP packet and are used to verify the integrity of the IP packet when it is received. The recipient runs the same hash (plus shared secret key) on each received IP packet and compares it with the inserted hash value. If nothing in the packet has been modified in transit the results are the same and the integrity of the IP packet remains intact. Common keyed hash functions include:

- **Message Digest 5 (MD5)** is a one-way hash that combines a shared secret and the message, to produce a hash. MD5 processes the input text in 512-bit blocks, divided into 16 × 32-bit subblocks. The output is a set of 4 × 32-bit blocks, concatenated to form a single 128-bit hash value.
- **Secure Hash Algorithm (SHA)** is similar to MD5 but the algorithm output is a set of 5× 32-bit blocks, concatenated to form a single 160-bit hash value.

Note that all message authentications in IPSec use hashed MAC (HMAC) described in [32] so MD5 becomes HMAC-MD5 and SHA becomes HMAC-SHA. HMAC is a

variant that provides an additional level of hashing making it cryptographically stronger than principal hashing function.

2) **Integrity Service**

Integrity is a security service that ensures that any modifications to data are detectable. There are two types of integrity in a TCP/IP environment: connectionless and partial sequence integrity. Connectionless integrity is a service that detects the modification of individual IP packets without regard to the ordering of the packet in a traffic stream. Partial sequence integrity is referred to as antireplay integrity, and it detects arrival of duplicate IP packets. With integrity, data is transmitted from source to destination without undetected change. Hashing algorithms can be used to provide integrity. Although authentication and integrity services are often mentioned separately, in practice they are closely related, and almost always offered together.

3) Key Exchanges

Symmetric algorithms and keyed hashing both require a shared key. The security of the encryption and authentication techniques can be completely undermined by an insecure key exchange. With symmetric encryption, the data encryption key is always present and remains fixed. Any intruder who compromises the encryption key can decrypt messages encrypted with it. A common cryptographic technique is to encrypt each individual conversation with a separate key. This is referred to as a *session key* since it is used for only one particular communication and destroyed when it is no longer needed. As such, the risk of compromising the session (encryption) key is reduced significantly. In the following section, we look at how the Diffie- Hellman key exchange algorithm can be used to ensure the secure exchange of keys.

- Diffie- Hellman key exchange

Diffie- Hellman was the first public key algorithm developed [33]. It is secure because of the complexity of calculating discrete logarithms in a finite field, as compared with the ease of calculating modular exponentiation in the same field. Diffie- Hellman can be used to distribute a shared secret key between parties in a key exchange across an insecure communication channel such as the Internet.

All participants in a Diffie- Hellman exchange must first choose a group. This group defines the prime p and the generator (primitive) g to be used. The Diffie-Hellman exchange comprises two parts. In part 1 each side, Alice and Bob, choose a random number (which becomes their private key) and does exponentiation to produce a public value.

- Alice: public-value-A = g^a MOD p (where a is the random number chosen by Alice).
- Bob: public-value-B = g^b MOD p (where b is the random number chosen by Bob).

In part 2, Alice and Bob exchange their public values. In other words, Alice provides public-value-A to Bob, and Bob provides public-value B to Alice. Each side then does an exponentiation again, this time using the other party's public value as the generator, which creates the shared secret key, Z;

- Alice: $(\text{public-value-B})^a \text{ MOD } p = g^{ab} \text{ MOD } p = Z$
- Bob: $(\text{public-value-A})^b \text{ MOD } p = g^{ab} \text{ MOD } p = Z$

Once Alice and Bob know the shared secret key Z, they can use it to protect their communications with any conventional secret-key algorithms. Anyone

who knows p or g will not be able to calculate the shared secret value Z easily because it is very complicated to compute a *discrete logarithm* to recover the secret exponents (private keys) Alice and Bob used.

One of the inadequacies with key exchange systems is that the same session key is distributed to numerous sessions. If this key is leaked, it can be used to decrypt the earlier encrypted traffic. Diffie-Hellman can be used to achieve Perfect Forward Secrecy (PFS), which eliminates the need for using long-life session keys. With PFS, a separate Diffie-Hellman session key (secret key) is generated for each session. If any of these keys are recovered, they will only provide access to that specific session's information. The remaining keys that are used to encrypt other sessions are not compromised.

Nevertheless, the Diffie-Hellman exchange still has some limitations. Alice and Bob can use this algorithm to generate and distribute a shared secret key, however, it cannot be used to encrypt and decrypt messages. Another drawback of the Diffie-Hellman exchange is that Alice and Bob have no way to verify that they are talking to each other, so the exchange can be subject to a man-in-the middle attacks. In other words, Diffie-Hellman provides for confidentiality but not for authentication. In this case, authentication can be achieved via the use of digital signatures in the Diffie-Hellman message exchanges.

1.3.5 Authentication Mechanisms

1) Introduction

Data origin authentication is a security service that verifies the identity of the claimed source of data. This service is usually bundled with connection-less integrity service. With data-origin authentication, we have to ensure that the data received is the same as the data that was sent and that the claimed sender is, in fact, the actual sender. Mechanisms such as the exchanges of digital certificates can be use to provide data origin authentication.

Confidentiality is mandatory to protect a secret. Without authentication, there is no way of validating and identifying the person with whom you share the secret as who he or she claims to be. In the following sections, we will examine additional authentication mechanisms that can help us identify and validate a sender [36].

2) Digital signatures

A digital signature is an encrypted message digest that is appended to a document. It is used to validate the identity of the sender and the integrity of the document. In practical implementations, digital signatures are created by combining hash functions with public-key algorithms. Because the entire document must be known before signature generation, HMAC rather than digital signatures are used to provide the message integrity of an ongoing data stream.

Instead of signing (encrypting) a document, Alice signs (encrypts) the hash of the document to produce the digital signature and appends it to the original document. Figure 1.6 shows an example of how a digital signature is created.

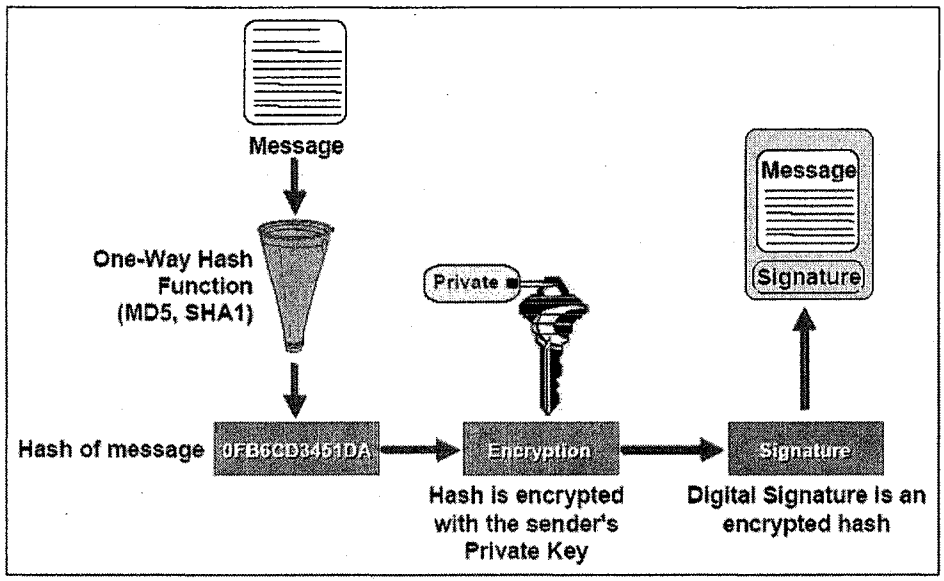


Figure 1.6: digital signature generation.

Digital signatures require the exchange of public keys beforehand. This exchange of public keys is again susceptible to man-in the middle-attack because during the exchange process, there is no way to guarantee that the public keys, which Alice and Bob have received from each other, are genuine. Some imposter (man-in-the middle) can substitute Alice's public key with his own fabricated public key, therefore, tricking Alice into believing that the public key she has received is actually from Bob. In the next section, we will look at how digital certificates can be used to thwart this kind of attack [34].

3) Digital certificates

A digital certificate (also commonly referred as public-key certificate) is someone's public key, signed and verified by another trustworthy party. Certificates are used to foil

attempts to substitute one's key for another. Typically, a digital certificate uses the X.509 standard format [35].

4) Digital envelopes

Digital signature and digital certificate provide proof of identity but they do not provide for privacy. To incorporate confidentiality, we put the signed document into an envelope and encrypt the envelope. Typically, the envelope contains encrypted content and encrypted content-encryption keys for the recipient. The combination of encrypted content and encrypted content encryption keys for a recipient is referred as a digital envelope (encrypted message) for that recipient. The process by which enveloped data is constructed by the sender involves the following steps:

1) A content-encryption key for a particular content-encryption (symmetric) algorithm is generated at random.
2) The content is encrypted with the content-encryption key.
3) The content-encryption key is encrypted with the recipient's public key.

The encrypted message and encrypted key are represented together according to the syntax of the public key cryptography standard#7 (PKCS #7) [35]. PKCS #7 is a general syntax for data that may be encrypted or signed, such as digital signatures or digital envelopes. The syntax is recursive, so that envelopes can be nested. Digital envelopes are used during CA (Certificate Authority) transactions to ensure the secure distribution of the certificates (PKCS#7 wrapped) to the users [35]. At the receiving end, the recipient opens the envelope by decrypting the encrypted content-encryption keys with the recipient's private key and decrypting the encrypted content with the recovered content-encryption key.

CHAPTER II: INTERNET PROTOCOL SECURITY (IPSEC) ARCHITECTURE:

2.1 Introduction:

IPSec is a framework of open standards developed by IETF [36]. It identifies a set of methods that intend to protect network traffic at the IP or network layer. It is a mandatory component of IP Version 6, (IPv6) and an optional one in IP Version 4, (IPv4).

IP packets have no inherent security. It is relatively easy to forge the addresses of IP packets, modify the contents of IP packets, replay old packets, and inspect the contents of IP packets in transit. Therefore, there is no guarantee that IP datagrams received are:

(1) from the claimed sender (the source address in the IP header);

(2) that they contain the original data that the sender placed in them; or

(3) that the original data was not inspected by a third party while the packet was being sent from source to destination. IPSec is a method of protecting IP datagrams. This protection takes the form of data content confidentiality, data origin authentication, connectionless data integrity authentication, anti-replay protection, and limited traffic flow confidentiality.

IPSec can be used in three types of scenarios. The first one is protection of one or more data communications between two hosts. The second one is protection of data communication between two security gateways which connect two networks. The third one is protection of data between a security gateway and a host such as remote login. These three scenarios can take place across a public network such as the Internet, or within a LAN. In either case, IPSec content can not be observed, in ESP protocol, or modified, in AH protocol, by a third party. This provides a transparent service for building applications that are unaware of the existence of IPSec in the Network Layer.

Consequently, the application development process is simplified by not integrating any security mechanism for transport service purposes.

IPSec provides specific security services at the IP layer. It enables a host or a security gateway to select one or more specific security protocol(s), such as AH and ESP, encryption and authentication algorithm(s), such as DES, RSA, and cryptographic keys. The set of security services that IPSec can provide includes are listed below for quick reference:

- **Confidentiality(Encryption)**

Confidentiality is "the property of communicating such that the intended recipients know exactly what was being sent but unintended parties can not determine what was sent".

- **Data Origin Authentication**

Data Origin Authentication is "knowing that the data received is the same as the data that was sent and that the claimed sender is in fact the actual sender".

- **Connectionless Integrity**

Connectionless integrity (for protocols such as UDP) is "the property of ensuring that the data is transmitted from source to destination without undetected alteration". The integrity check is done on a per-packet basis. This is in contrast to connection integrity, where the integrity check is done on a stream of packets.

- **Access Control:**

Access control is the ability to limit and control the access to a host system and application via communications links. In order to achieve this control, each entity (a user or a program) trying to gain access must first be identified or authenticated, and then authorized, based on access rights assigned to it.

- Rejection of Replayed Packets:

Rejection of replayed Packets is the ability to detect and reject any replayed packets that can be saved by an attacker.

All of these services are implemented in the IP layer as shown in Figure 2.1.

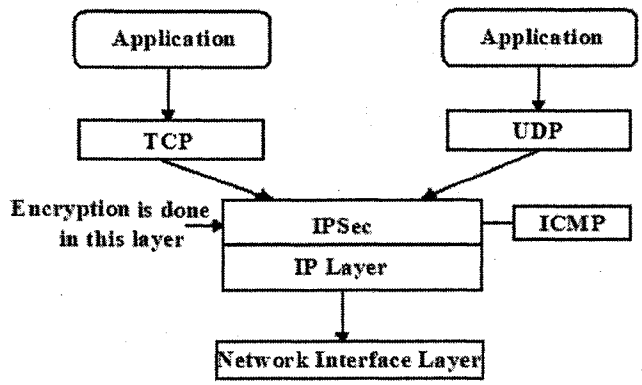


Figure 2.1 IPSec Encryption not evident in applications

Therefore, all other higher layer protocols such as TCP, UDP, and ICMP can utilize IPSec without knowing it even exists. This provides an outstanding feature for software developers because they do not need to implement any security mechanisms in their software for transport purposes.

2.2 IPSec Implementation in TCP/IP Stack

There are three ways to implement IPSec within the TCP/IP protocol stack [5]:

A) Above the IP Layer - In this kind of implementation, source code of the IP layer must be available in order to modify it. In addition, the logical interface between IP and TCP layers must be implemented.

B) Within the IP Layer - In this kind of implementation, source code of the IP layer must be available so that it can be modified to integrate the functionality of IPSec protocol.

C) Below the IP layer - In this method, the implementation is started from scratch and it does not need the TCP/IP source code. However, certain TCP/IP functions must be re-implemented such as dealing with fragmentation and reassembly of fragmented packets.

2.3 The IPSec Roadmap

The IPSec protocols include AH, ESP, IKE, ISAKMP/Oakley, and transforms. In order to understand, implement, and use IPSec, it is necessary to understand the relationship among these components. The IPSec roadmap and related documents define how various components of IPSec interact with each other. This is shown in Figure 2.2, and explained as follows.

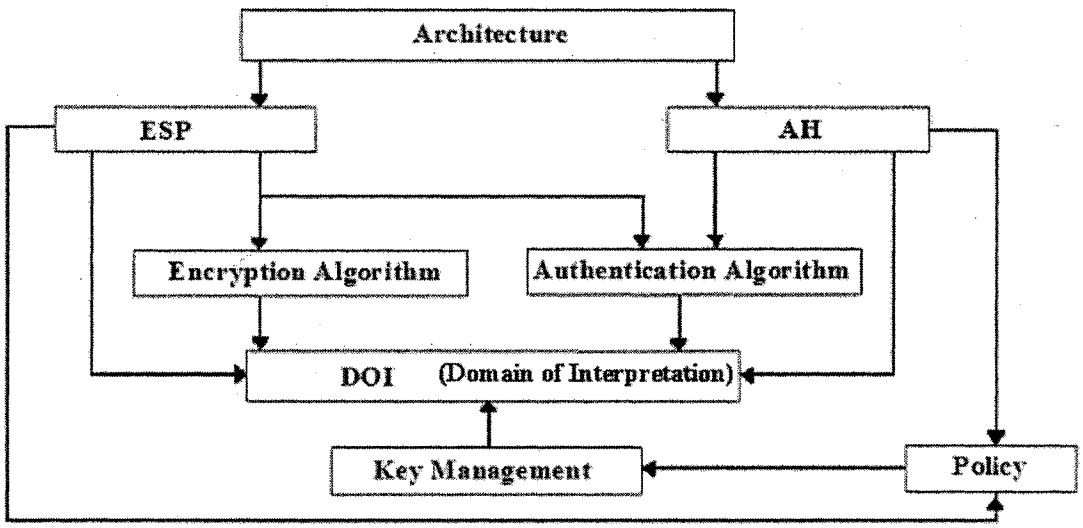


Figure 2.2 IPSec Roadmap (this figure has been reproduced from the draft with permission of the authors)

- **Architecture**: Covers the general concepts, security requirements, definitions, and mechanisms defining IPSec technology.
- **Encapsulating Security Payload (ESP):** Covers the packet format and general issues related to the use of the ESP for packet encryption and, optionally, authentication.
- **Authentication Header (AH):** Covers the packet format and general issues related to the use of AH for packet authentication.
- **Encryption Algorithm:** A set of documents that describe how various encryption algorithms are used for ESP.
- **Authentication Algorithm:** A set of documents that describe how various authentication algorithms are used for AH and for the authentication option of ESP.
- **Key Management:** Documents that describe key management schemes.
- **Domain of Interpretation (DOI):** Contains values needed for the other documents to relate to each other. These include identifiers for approved encryption and authentication algorithms, as well as operational parameters such as key lifetime [8].

2.4 How IPSec Works

IPSec is designed to provide high quality, interoperable cryptographic - based security for IPv4 and IPv6 datagrams. IPSec achieves these objectives by use of two traffic security protocols Authentication Header (AH) and Encapsulation Security Payload (ESP) and through the use of cryptographic key management procedures and protocols such as Internet Key Exchange protocol (IKE).

The IP AH protocol provides data origin authentication, connectionless integrity and an optional anti-replay service.

The IP ESP protocol provides data confidentiality, limited traffic-flow confidentiality, connectionless integrity, data-origin authentication and anti-replay service. There are two modes of operation of both AH and ESP: transport mode and tunnel mode. AH and ESP can be applied, in the desired mode alone or in combination with each other. The Internet Key Exchange (IKE) protocol is used to negotiate the choices of the cryptographic algorithms to be utilized by AH and ESP, and put in place the necessary cryptographic keys that the algorithms require. AH, ESP and IKE protocols will be discussed in more detail in later sections.

The protocols that IPSec uses are designed to be algorithm independent. The choice of algorithms is specified in the Security Policy Database (SPDB)*. The possible choices of algorithms that are available are dependent on the IPSec implementation; however, a standard set of default algorithms are specified by IPSec to ensure interoperability in the global Internet.

IPSec allows the user or administrator of a system or a network to control the granularity at which the security service is offered. For example an organization's policy might specified that data traffic that originated from certain subnet(s) should be protected with both AH and ESP and that the encryption should be done with triple-DES with three different keys; whereas, the policy might specified that data traffic from another site should be protected with only ESP and that these traffic should be afforded encryption with DES†. IPSec is able to differentiate between the security service it offers to different data traffic by the us of Security Association (SA).

---

* In the proposed architecture used in this thesis, we refer to the required database as Security Management Information Base (SMIB), see chapter IV.

† See "IPSec Policy and Business Management Layer," Section 4.2

2.5 IPSec Modes

IPSec sessions can be implemented in two different modes: Transport and tunnel mode IPSec in the transport mode is used to establish an end-to-end secure communication channel between the source and destination hosts. All packets are secure once they leave the source host, until they reach the destination host as shown in Figure 2.3. Generally, this mode is used within LANs.

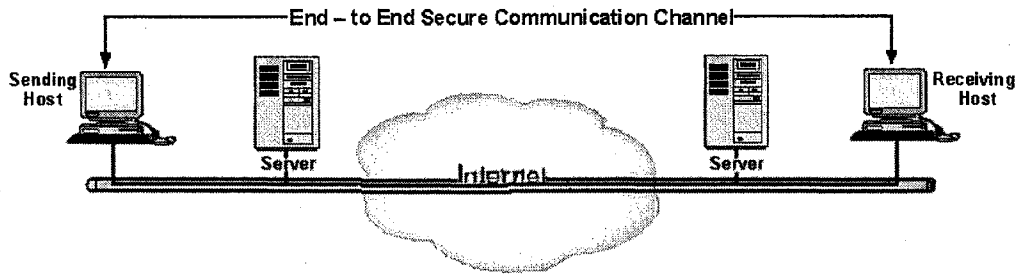


Figure 2.3: IPSec in Transport Mode

In the tunnel mode shown in Figure 2.4, IP packets are protected only between this two defined tunnel points or security gateways.

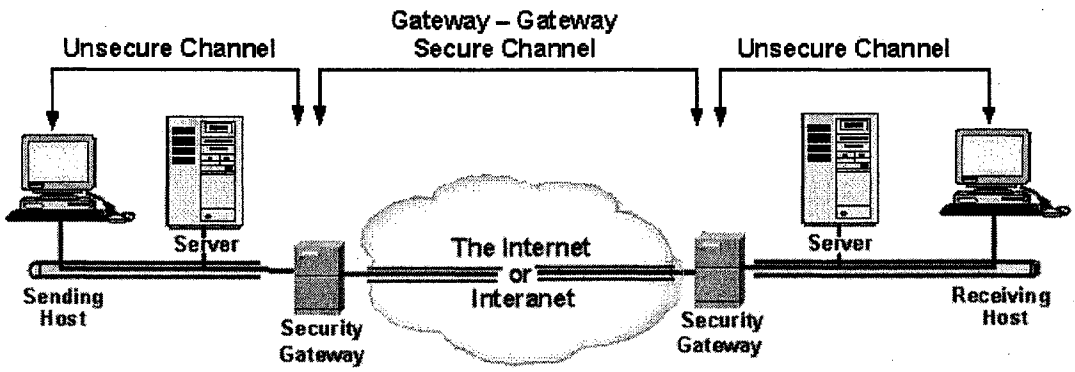


Figure 2.4: IPSec Tunnel Mode

When packets are transmitted between the client and the security gateway, they are sent in an unprotected format until they reach the initial gateway. Then protection is applied and the packets are sent to the receiving network security gateway, where they are decrypted and forwarded in clear text to the destination host.

2.6 Security Association

2.6.1 Security Association (SA) Concept

Before any IPSec traffic is exchanged between a source and destination host, both parties must establish their security parameters through a handshake that determines several security parameters such as encryption algorithm, authentication algorithm, encryption keys, and the chosen protocol (AH or ESP). The IPSec protocol uses the Security Association (SA) mechanism to provide groundwork to manage these security parameters. Both AH and ESP IPSec protocols need an established SA before any communication can take place. Thus, SA is a unidirectional communication that identifies the security parameters. To protect a distinctive bi-directional communication using AH or ESP protocol, two SAs are required, one in each direction [37]. If both AH and ESP protocols are used simultaneously, two or more SAs are needed. This is called an SA bundle. Each SA is uniquely identified by the following three parameters [37]:

a) The packet destination address

b) The IPSec protocols, either AH or ESP

c) A Security Parameter Index (SPI) which is chosen by the receiver and is transmitted in a clear text.

The IPSec protocol stack uses a Security Association Database (SADB) to store all active SAs. This database contains all the parameters pertaining to each SA. Sender and receiver hosts consult the SADB for every received or sent packet to determine how to deal with them. The SA, or the contract between the sender and receiver hosts, contains fields that are needed by both parties to process every incoming and outgoing IP packet. Some fields are used for outbound processing, some for inbound processing, and some for both, depending on usage of the field [37] The fields for the SA is given in Appendix [A2].

2.6.2 Security Databases

There are at least two databases that are necessary for the processing of IPSec traffic: Security Policy Database (SPDB) and Security Association Databases (SADB). SPDB specifies the policies that are to be applied to the traffic that is destined to, or originated from a given host or network. On the other hand, SADB contains the active SA parameters. For both SPDB and SADB, separate inbound and outbound databases are required. The objects stored in these database are summarized in Appendix A3.

Each entry in the SPDB consists of one or more selectors, an indication of whether the packets that match the selector(s) in the entry should be discarded, be subject to IPSec processing or not be subject to IPSec processing. If the packets are to be subjected to IPSec processing, the entry also contains a pointer to a Security

Association (SA) specification which details the IPSec protocols, modes and algorithms to be applied to the packets matching this policy entry.

The first entry with selector(s) matching that/those corresponding to the traffic under consideration will be applied. If no matching entry is found, the packets for the traffic under consideration will be discarded. The entries in the SPDB should therefore be ordered according to the desired preference of application.

The entries in the SPDB determine the granularity with which the traffic is processed. For example, the policies could specify that IPSec service corresponding to a given SA or SA bundle should be applied to all traffic to or from any source or destination; or, the policy could specify the application of different SAs or SA bundles based on specifed selectors. The SPDB therefore plays a very important role in the control of the flow of all traffic through an IPSec system.

Separate SADB are kept for inbound and outbound IPSec processing for inbound or outbound traffic, the respective SADB is searched for selectors matching the SPI, the source or destination address and IPSec protocol, extracted from the packet header. If a matching entry is found, the parameter for this SA are compared with the appropriate fields in the AH or ESP headers. If the header fields correspond with the SA parameters in the database, the packet is processed. However, if there are any discrepancies, the packet is discarded. If no SA entry matches the selectors, the packet is discarded if it is an inbound packet. If the packet however is outbound, a new SA or SA bundle will be created and entered in the outbound SADB.

A SA lifetime has two kind of limit: a soft limit and a hard limit. When the soft limit is reached, the communicating peers must re-negotiate a new SA to replace the existing one. However, the existing SA is not delected form the database until the hard limit expires. Unlike the SPDB, the entries in the SAD are not ordered. Nonetheless, as is the case with SPDB lookups, the first matching SA entry that is found in the SAD is used for the IPSec processing of the packets that are associated with the given SA.

2.7 IPSec Principle of Operation

The IPSec processes ensure secure communication between two communicating hosts based on a policy that is defined in a Security Policy Database (SPDB). The system administrator is responsible for establishing and maintaining the SPDB. Source and destination hosts check with the SPDB to make a decision, for each IP packet, whether the IP packet has to be protected by IPSec, or discarded, or allow it to bypass the IPSec stack.

SPDB is organized as an ordered list of rules. Each rule is identified by one or more selectors that define the type of IP traffic that will be affected by IPSec. Selectors might be the source and destination IP addresses the SPI, source and destination port numbers, and the protocol ID. Once the SPDB is consulted by the source or destination hosts with a specific selector, the established SA parameters are retrieved. Table 2.2 shows an example of SPDB.

Security Policy Parameters	Example Values
SPI	3219011
IP Destination Address	172.16.16.10
IP Destination Port	23
IP Source Address	172.16.16.15
IP Source Port	3290
Protocol	TCP

Table 2.2: Example of SPDB

The processing of IPSec packets are done as follows [5], [6], [7], [8]:

- **Outbound Traffic**

  Once the IPSec processes receive an IP packet to be sent from an application, it consults the SPDB to decide how to process this IP packet. If the packet must be IPSec enabled, an IPSec process retrieves the required SA from the SADB. If the necessary SA already exists, it is used to process the IP packet. In the contrary case, an IPSec process calls Internet Key Exchange protocol (IKE) to establish a new SA with specific security parameters.

  For example, let us consider the network shown in the Figure 2.5. In this case, the host is tunneling a packet to the gateway using ESP but is authenticating to the end host B.

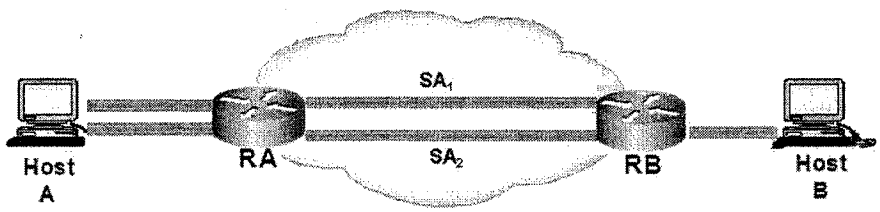


Figure 2.5: Outbound IPSec Processing

The correct header is shown in Figure 2.6.

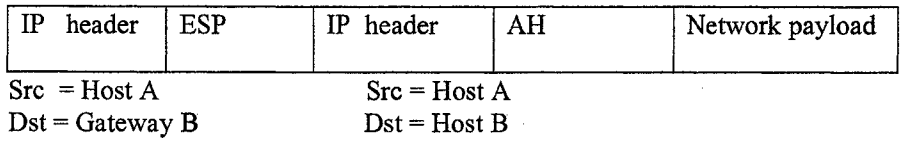


Figure 2.6 packet format

In this case, IKE establishes four SAs two for sending and two for receiving. As we are discussing the outbound processing, we will ignore the SAs established for processing inbound packets. The two outbound SAs are SA_1 and SA_2 where SA_1 is the SA between A and the gateway and SA_2 is the SA between the host and the destination. The ordering of IPSec processing is very important. If SA_2 is applied after SA_1, the packet is formed incorrectly. It is very important to maintain ordering in the SA bundle so that IPSec processing is applied in the correct order for outbound packets.

- **Inbound Traffic**

Once the ISPec process receives an IP packet from the network, it inspects the header to decide if this packet was IPSec-enabled. If so , the IPSec process uses information in the incoming packet to determine the selectors that will be used to retrieve the defined policy from the SPDB. It also retrieves a pointer to the corresponding SADB where the security parameters are stored. These security parameters will be used for checking and/or deciphering of the packet in question.

- **Fragmentation**

IPSec does not fragment or reassemble packets. On outbound processing, the transport payload is processed and then passed on to the IP layer for further processing. On inbound processing; the IPSec layer gets a reassembled packet from the IP layer. However, as IPSec does add IPSec header, it impacts the Path Maximum Transfer Unit (PMTU) length. If IPSec does not participate in PMTU discovery, the IP layer ends up fragmenting a packet as the addition of the IPSec header increases the length of the IP datagram beyond the PMTU. It is important for IPSec to participate in the PMTU discovery process. This is discussed in greater detail in the Chapter IV on IPSec implementation.

- **Internet Control Massage Protocol (ICMP)**

ICMP (Internet Control Massage Protocol) processing is critical to the operation and debugging for a network. This protocol is runs on top of a network protocol such as IPv4 and IPv6. When IPSec is used end-to-end, it does not impact ICMP. However, when IPSec is used in tunnel mode, it impacts ICMP and the operation of the network. The problem arises in the tunnel mode, particularly when the tunnel header is added by an intermediate gateway. This is because ICMP messages are required to send only 64 bits of the original header. ICMP is used to determine if the host is reachable or not. ICMP is used in PMTU (Path Maximum Transfer Uunit) discovery If a router needs to fragment a packet but the "do not fragment" bit is set, then the router sends back an ICMP massage to host indicating the MTU of its link so that the host can generate packets whose size does not exceed this MTU.

In order to handle ICMP error messages correctly, IPSec needs to maintain some state and perform extra processing.

- **Path MTU**

By default, the Maximum Segment Size (MSS) is 536 bytes. It is the maximum number of bytes that can be transported by TCP in a single packet. TCP fragments packets in a transmit queue into 536 byte chunks before passing them down to the IP layer. With a 536-byte MSS, packets are not likely to be fragmented at an IP device along the path to the destination because most links use an MTU of at least 1500 bytes. However, smaller packets also increase the amount of bandwidth used to transport overheads.

Most routers of Border Gateway Protocol (BGP) are equipped with a commode that can dynamically determine the largest MSS value without generating packets that need to be fragmented. It allows TCP to determine the smallest MTU size among all links in a TCP session and then this MTU value, less the number of bytes taken up by the IP and TCP headers, as the MSS for the session.

CHAPTER III: IPSec PROTOCOLS

3.1 Introduction

In this Chapter we briefly review various IPSec protocols and their features. IETF has defined three protocols in IPSec:

1) IP Authentication Header (AH) Protocol "AH provides connectionless integrity, data origin authentication, and an optional anti-replay service" for each IP datagram packet [38].

2) Encapsulating Security Payload(ESP) Protocol "ESP provides data confidentiality (encryption), limited traffic flow confidentiality, connectionless integrity, data origin authentication, and an optional anti-replay service" for each IP datagram packet [43].

1) Internet Security Association and Key Management Protocol (ISAKMP) "ISAKMP provides a method for automatically setting up security associations and managing the cryptographic keys" [44]. The details of these protocols are explained in the following sections.

3.2 The IP Authentication Header (AH) Protocol

3.2.1 Authentication Header (AH) Concept

The IP protocol is inherently insecure. The authentication mechanism that is used to provide integrity for the IP datagrams is rather primitive. The Authentication Header (AH) protocol was designed to improve the security of IP datagrams. AH provides connectionless integrity and data origin authentication and an anti-replay protection service [23]. AH uses cryptographic Message Authentication Codes

(MAC) to authenticate IP datagrams. A MAC requires a secret key to generate the message digest of a bit-string. IPSec uses Internet Key Exchange (IKE) to negotiate, generate and put in place necessary cryptographic keys for AH or ESP. Since only the source and the destination nodes know the secret key, it should be computationally infeasible for an intruder to modify the datagram and recompute an authentication data that will be validated by the hosts. This is illustrated in Figure 3.1.

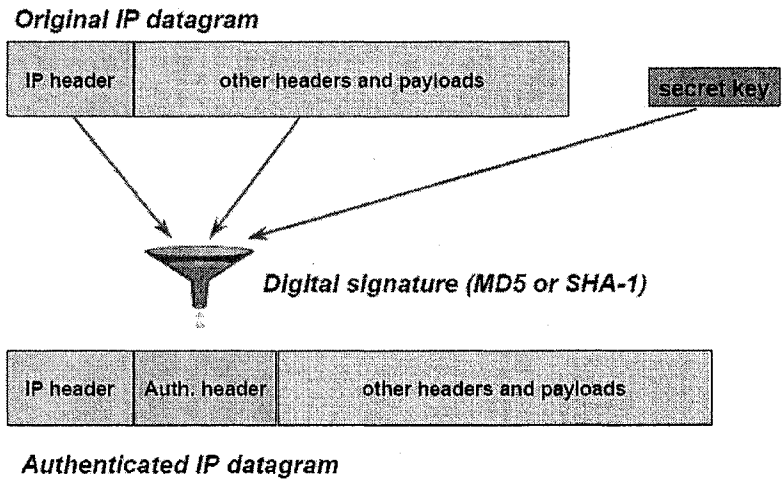


Figure 3.1: Procedure for Authenticated IP Datagram

3.2.2 AH Protocol Format

According to IPSec specifications, data authentication is implemented by using an Authentication Header (AH) [23]. The AH is additional information, i.e.,a field, that is added to each IP datagram. It is calculated using all the fields in the IP datagram (IP header, TCP header, and user data). IP fields that change, such as Time-to-Live (TTL), are not included in the calculation. Figure 3.2 shows the AH protocol format in IPv4 for both the Transport Mode and the Tunnel Mode. As

can be seen in Figure 3.2 for IPv4, the AH header is placed immediately after the IP header and before any transport protocol and user data.

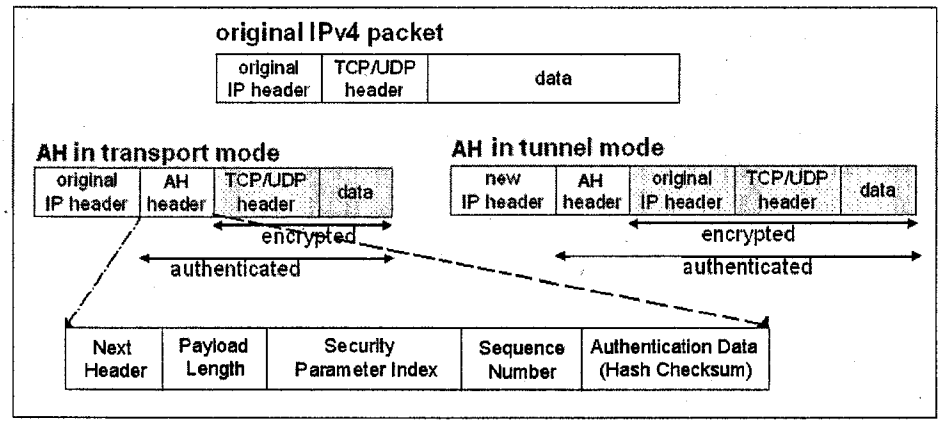


Figure 3.2: IPSec AH Protocol Format

AH consists of five fixed-length fields and a variable length Authentication Data field. Figure 3.3 shows the relative position of these fields in AH.

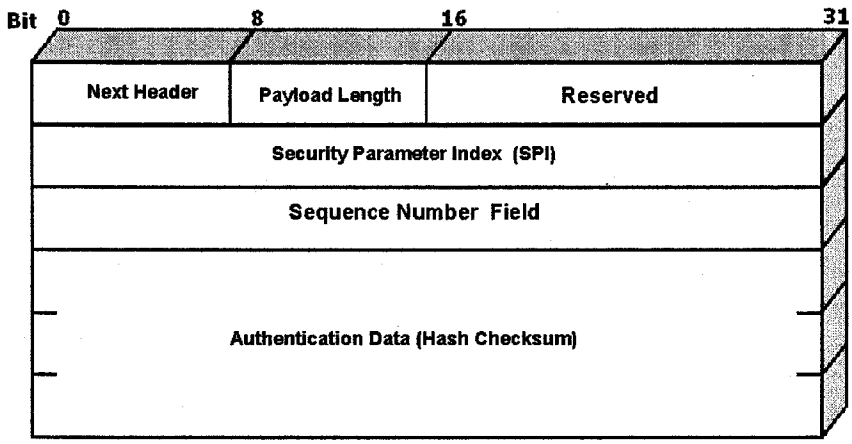


Figure 3.3: Authentication Header Format

2. **Next Header:** This is an 8-bit field that identifies protocol ID of the transport protocol that is included in the payload after the Authentication Header. The protocol ID is a number that is chosen from a standard set of protocols ID defined by IETF.

3. **Payload Length:** This is an 8-bit field that specifies the length of AH in 32-bit words.
4. **Reserved:** This 16-bit field is reserved for future use.
5. **Security Parameters Index (SPI):** SPI is a random 32-bit value that is used with the destination IP address and security protocol (AH) to exclusively identify the Security Association for each IP datagram.
6. **Sequence Number (SN):** This unsigned 32-bit field contains a monotonically increasing counter value that provides anti-replay security service against replay attacks. For each IP packet issued for a specific SA, the sequence number is incremented by one to guarantee that each IP packet is assigned a unique sequence number. The receiver host verifies each IP packet to guarantee the SN has not been reused. In this way, an attacker can not capture an IP packet, modify its content, and play it back.
7. **Authentication Data:** This is a variable-length field that contains a hash or Integrity Check Value (ICV) for each IP packet. It is produced by the sender host to protect a packet from modification during transit. A recipient host performs the same integrity check and compares the result of the hash algorithm with the result stored in this field. There are several types of authentication algorithms, such as MD5 and SHA-1. Each one produces different ICV values.

The AH protocol can be used when there is a need to protect the entire IP packet from being modified. The AH protocol can also be used when communications must be restricted to specific computers in a network. AH protocol ensures mutual

authentication between communicating computers, such as trusted computers in a domain. If a host can not negotiate an SA, communications do not take place.

In summary, the AH protocol provides data integrity for the entire IP packet by using authentication algorithms such as MD5. It also provides data origin authentication by using a shared secret key or the SPI. Finally, AH protocol provides a mechanism against replay attack by using sequence numbers.

3.2.3 AH Modes

AH can be used in either transport or tunnel mode. The difference is the data being protected, either an upper-layer protocol or an entire IP datagram. In either case, AH also authenticates immutable portions of the outer IP header.

3.2.3.1 AH Transport Mode

When used in transport mode, AH protects end-to-end communication. The communications endpoint must be the IPSec endpoint. The AH header is inserted into the datagram to protect it by placing it immediately following the IP header (and any options) and before the upper layer protocol to protect. Figure 3.4 shows the position of AH-in transport mode relative to other header fields.

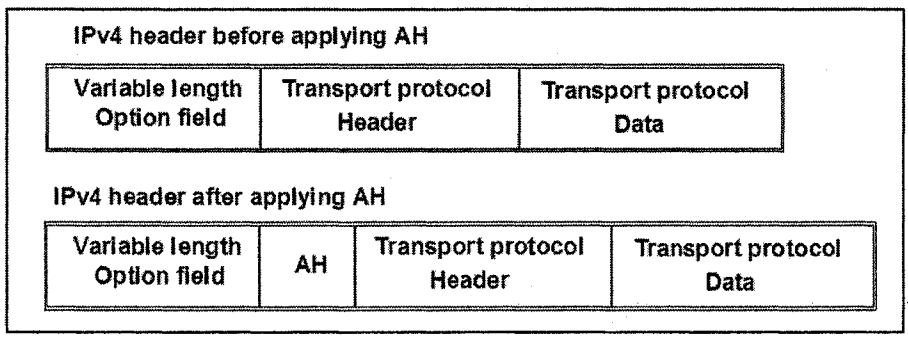


Figure 3.4: AH relative to other IPv4 header fields, in transport mode.

The option field was eliminated in IPv6. The options, in this version of IP. are treated as separate headers called extension headers that are inserted after the IP header. This feature speeds up packet processing time in that, except for the hop-by-hop extension header which contains routing information that routers need to examine, the extension headers are not examined or processed by any intermediate node on the path from the source to the destination. They are only processed by the destination host. In transport mode for IPv6, the AH is inserted after the hop- by- hop, Routing and Fragmentation extension headers. Figure 3.5 illustrates this.

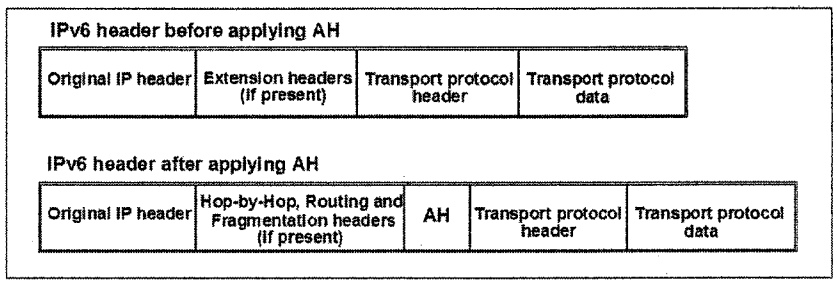


Figure 3.5: AH relative to other IPv6 extension headers, in transport mode.

3.2.3.2 AH Tunnel Mode

When used in tunnel mode, AH encapsulates the protected datagram and an additional IP header is added before the AH header. The "internal" IP datagram contains the original addressing of the communication and the "outer" IP datagram contains the addresses of the IPSec end-points. Tunnel mode can be used as a replacement for transport mode for end-to-end security, but since there is no confidentiality and therefore no protection against traffic analysis, there is really no point in AH tunnel mode. AH is simply used to guarantee that the received packet

was not modified in transit, that it was sent by the party claiming to have sent it, and optionally, that it is a fresh, nonreplayed packet.

In tunnel mode, AH is inserted before the original IP header and a new IP header, is consequently inserted in front of the AH. This is illustrated diagrammatically for IPv4 in Figure 3.6 and for IPv6 in Figure 3.7, respectively.

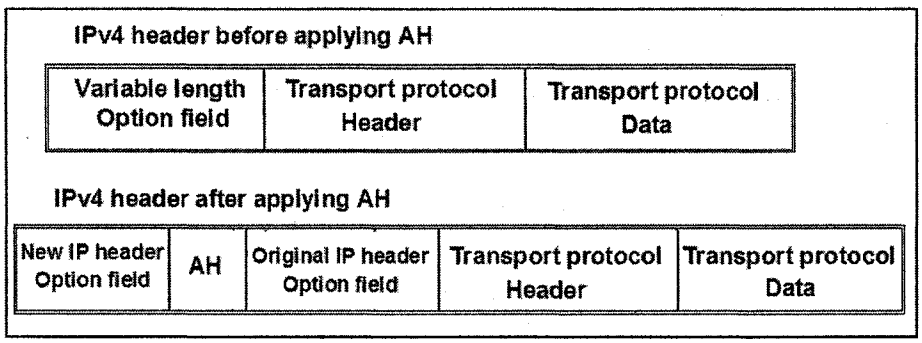


Figure 3.6: AH relative to other IPv4 header fields, in tunnel mode

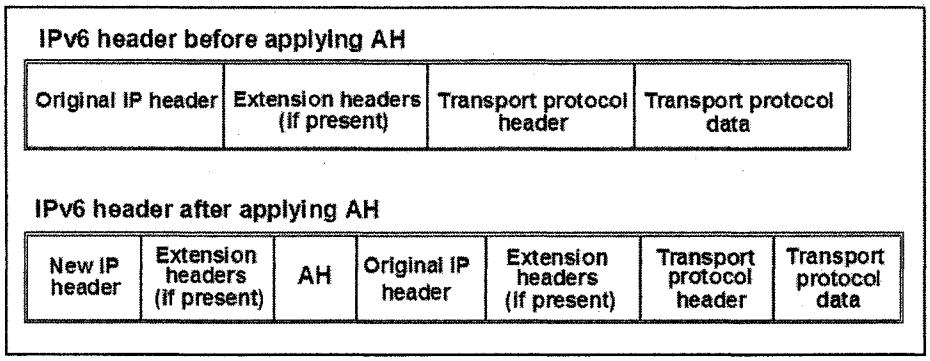


Figure 3.7: AH relative to other IPv6 extension headers, in tunnel mode

The inner IP header carries the true source (the node that generated the packet and the final destination address. Whereas, the outer IP header typically carries the sender's security gateway address as the source address, and the recipient's security gateway

address as the destination address. Consequently, the source address in the inner and outer IP headers may be different. The same holds for the destination address.

3.2.4 AH Processing

When an outbound packet matches an SPDB entry denoting protection with AH, the SADB is queried to see whether a suitable SA exists. If there is no SA, IKE can be used to dynamically create one. If there is an SA, AH is applied to the matched packet in the mode dictated by the SPDB entry. If there is an SPDB bundle, the order of application depends on the protocols involved. AH always protects ESP, not the other way around.

3.2.4.1 Outbound Processing

When an IPSec implementation receives an outbound packet it uses the relevant selectors (destination IP address and port, transport protocol, etc) to search the Security Policy Database (SPDB) to ascertain the policy that is applicable to the traffic. If IPSec processing is required and a SA or SA bundle has already been established, the SPDB entry that matches the selectors in the packet will point to the appropriate SA bundle in the SADB. If a SA has not been established, the IPSec implementation will create a SA and link it to the SPDB entry. The SA is then used to process the packet as follows:

1. Generate or increment Sequence Number as the sequence number is used to prevent replay of previously transmitted packets. When a new SA is established the sender initializes its sequence number counter to zero. For each packet that the sender transmits, it increases the sequence number by one and insert the resulting value of the sequence number counter into the sequence number field of the AH header.

2. Calculate Integrity Check Value. The ICV is calculated using the values in the immutable fields of the IP header and the values of the mutable fields that are predictable mutable fields for the purpose of the ICV computation. The SA for the datagram stipulates the cryptographic authentication algorithm that should be used to generate the ICV. The available choices of authentication algorithms vary with different IPSec implementations. The ICV is generated into the variable length Authentication Data field of the AH, then if required, the datagram is fragmented into the appropriate fragment size and sent to the destination node.

3.2.4.2 **Inbound Processing**

When a datagram arrives at an IPSec host or security gateway, if the more fragment bit is set, this is an indication that there are other fragments that are yet to arrive. The IPSec application therefore waits until a fragment arrives with a sequence number that is similar to the previous ones, and has the more fragment bit not set. It then reassembles the IP fragments, then performs the following steps:

1. It uses the SPI, destination IP address and IPSec protocol in the IP header (outer IP header if tunnel mode) to look up the SA for this datagram in the inbound SADB. If the look up fails, it drops the packet and logs the error.
2. It uses the SA found in (1) to do the IPSec processing. This involves first checking to determine whether the selectors in the IP headers (inner header if tunnel mode) match those in the SA. If the selectors do not match, the application drops the packet and audits the error. If the selectors match, the IPSec application keeps track of the SA and the order in which it is applied

relative to the others and continues to do (1) and (2) until it encounters a non-IPSec extension header or a transport layer protocol.

3. It uses the selectors in the packet to find a policy in the inbound SPD whose selectors match those of the packet.
4. It checks whether the SA's found in (1) and (2) match the policy specified in (3). If the check fails, repeat steps (4) and (5) until all policy entries have been check or until the check succeeds.
5. If anti-replay is enabled use the anti-replay window of the SA to determine if the packet is a replay. If the packet is a replay, drop it and log the error.
6. It uses the authentication algorithm specified by the SA bundle to calculate the ICV for the packet and compares it with the value stored in the Authentication Data field. If the two values differ, then it discards the packet and audit the error.

At the end of these steps, if the packet has not been discarded, it is passed to the transport layer protocol, or is forwarded.

3.3 The IP Encapsulating Security Payload (ESP) Protocol

3.3.1 Encapsulating Security Payload (ESP) Protocol Concept

The ESP protocol can be used to guarantee one or any combination of the following security services [4]:

1. **Data confidentiality** - It guarantees that only the sender and the receiver are able to read user data. All intermediate nodes that are on the path can not read user data while in transit. This is accomplished by using a specific data encryption algorithm in the source and destination hosts with a secure key distribution mechanism.
2. **Partial protection against traffic analysis** - This service is only available in the tunnel mode of ESP protocol where the actual source and destination IP addresses are hidden. Only the source and destination IP addresses of the security gateways are exposed on the IP packet. An attacker in the middle can only see an IP packet that is going from one security gateway to another one without knowing the actual sender and receiver hosts.
3. **AH Security Services**

   All the AH services, connectionless data integrity, data origin authentication, and protection against replay, can also be enabled in ESP protocol.

3.3.2 ESP Protocol Format

Figure 3.8 shows IPSec ESP protocol format in both transport and tunnel modes. The confidentiality security service can be activated alone or with other services. Allowing confidentiality without other services makes traffic vulnerable to certain types of active attacks [25].

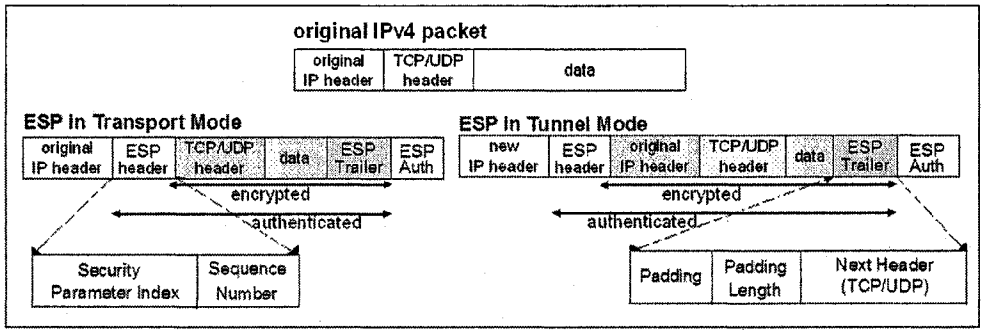


Figure 3.8: IPSec ESP Protocol Format

Similar to AH protocol, authentication and integrity are enabled together and provided by the use of an ICV or hash value. Protection against replay attacks is achieved by the use of a sequence number which the receiver uses to check every incoming packet. ESP protocol provides integrity service by signing the ESP header, TCP/UDP header user data, and the ESP trailer. It also prevents IP packets from inspection by encrypting the TCP/UDP header, user data, and the ESP trailer.

AH and ESP protocols can be combined together in the same SA. While both AH and ESP provide integrity protection for user data, AH protects the entire IP packet from modification while ESP protocol protects only the TCP/UDP header and user data from inspection through encryption.

ESP packets consist of four fixed-length fields and 3 variable-length fields. Figure 3.9 shows the packet format of this protocol. A brief description of each field is as follows.

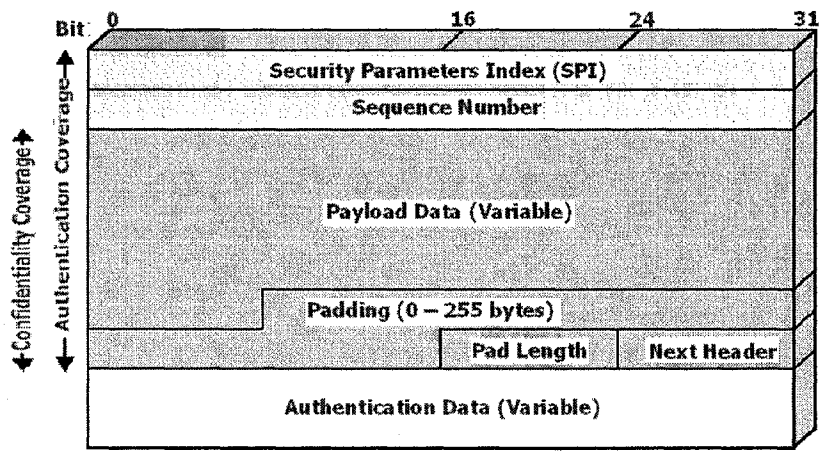


Figure 3.9: ESP Packet Format.

1. **Security Parameter Index (SPI)** - The SPI is a 32-bit integer that is used in combination with the source or destination address and the IPSec protocol to uniquely identify the SA for the datagram. The range of numbers from 1 to 255 are reserved by the Internet Assigned Number Authority (IANA) for future use, and 0 is reserved for local and implementation specific use. Therefore the current valid SPI values are between 256 and 2^{32} - 1 inclusively. This is similar to the AH SPI field.
2. **Sequence Number (SN) -** As is the case for the AH, this contains a 32-bit unsigned integer which servers as a monotonically increasing counter. The sender's and receiver's sequence number counters are initialized to 0 when the SA is established. The sender consequently increases its sequence number by one for

every packet it sends using a given SA. The sequence number is used to prevent intruders from capturing and re-sending previously transmitted datagrams.

3. **Payload Data** - This field contains the padding bits, if any, that are utilized by the encryption algorithm, or that are used to align the pad Length field so that it begins at the third byte within the 4-byte word. The length of this field can be between 0 and 255 bytes inclusively.
4. **Pad Length** - The Pad Length field is a 8-bit field which indicates the number of padding bytes in the Padding field. The valid values for this field are integers between 0 and 255 inclusively.
5. **Next Header** - This is a 8-bit field that identifies the type of data encapsulated in the payload. It may indicate an IPv6 extension header or a transport layer protocol; refer to [47] for the set of assigned IP protocol numbers.
6. **Authentication Data** - This is a variable-length field that contains the Integrity Check Value (ICV), which as indicated by Figure 3.14, is calculated over the length of the ESP packet minus the Authentication data field. The length of this field is specified by the authentication algorithm employed for the authentication.

3.3.3 ESP Modes

The ESP protocol is used when there is a need to protect user data and the TCP/UDP header from being observed. The ESP protocol is also used to protect the TCP/UDP header and application data from modification during transmission. In summary, the ESP protocol provides confidentiality for user data in each IP packet. It also provides partial protection against traffic analysis when it is used in

the tunnel mode. Finally, ESP protocols provides data integrity where only the user data and TCP/UDP header are authenticated, compared to the authentication of the entire IP packet in AH protocol as shown in Figure 3.8 and 3.14.

As is the case for AH, the location of the ESP data in the packet depends on the mode of operation of ESP. There are two modes of operation: transport mode and tunnel mode. These modes of operation will be discussed in the following two subsections.

3.3.3.1 ESP Transport Mode

In transport mode, the ESP header is inserted after the IP header and any options it contains but before any transport layer protocol, or before any IPSec protocol that has already been applied, as illustrated in Figure 3.10.

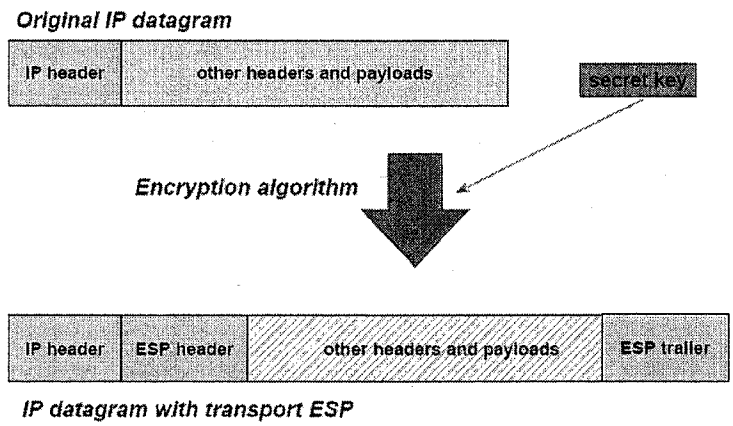


Figure 3.10: IP Datagram with Transport ESP Mode

So, for IPv4 in transport mode, ESP is inserted after the variable length "options "filed show in field, Figure 3.11.

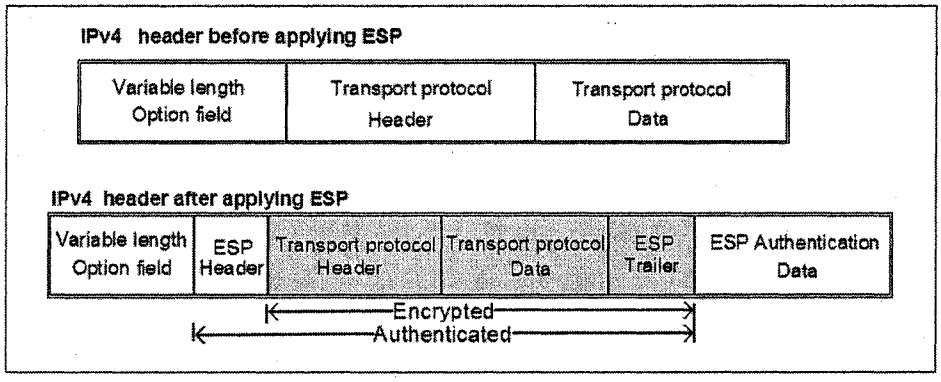


Figure 3.11: ESP relative to other IPv4 header fields, in transport mode.

Figure 3.11 shows the position of ESP in transport mode relative to other header fields. In this diagram, the ESP Header field consists of the SPI and Sequence number fields' whereas the ESP Trailer authentication services are applied as shown in Figure 3.12. In transport mode, for IPv6, the ESP header is inserted after the Hop-by-Hop. Routing and Fragmentation extension headers .Figure 3.16 illustrates this.

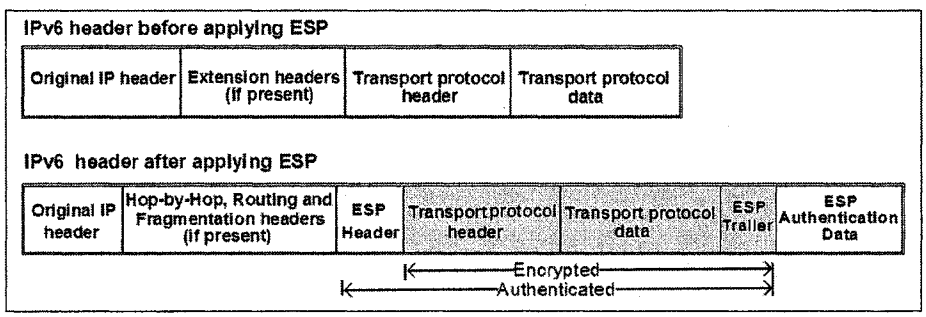


Figure 3.12: ESP relative to other IPv6 extension headers, in transport mode.

- **3.3.3.2 ESP Tunnel Mode**

In tunnel mode, ESP is inserted before the original IP header and a new IP header is consequently inserted in front of the ESP header. This is

illustrated in Figure 3.13 for IPv4 and for IPv6 in Figure 3.14, respectively.

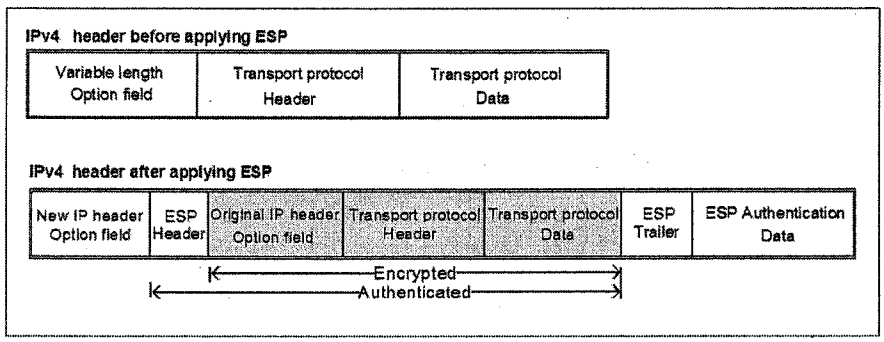


Figure 3.13: ESP relative to other IPv4 header fields, in tunnel mode.

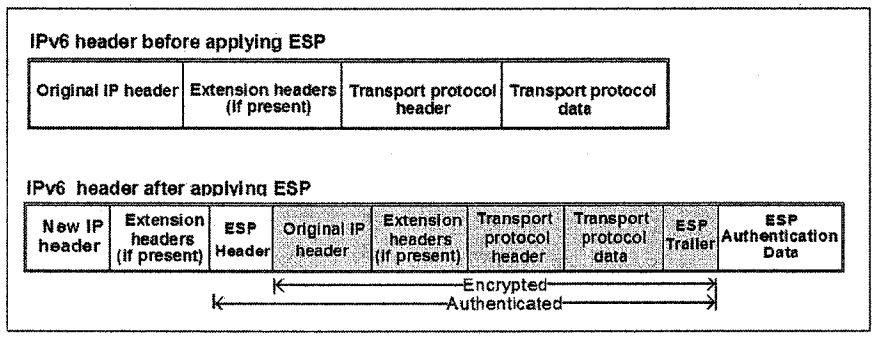


Figure 3.14: ESP relative to other IPv6 extension headers, in tunnel mode

The inner IP header carries the true source (the node that generated the packet) and the final destination address. The outer IP header typically carries the sender's security gateway address as the source address, and the recipient's security gateway address as the destination address.

Consequently, the source address in the inner and outer IP headers may be different; the same holds for the destination address.

3.3.4 ESP Processing

Processing of an IP packet with ESP depends partly on the mode in which ESP is being employed. In either case, though, one thing to remember is that with ESP, the ciphertext is authenticated.

3.3.4.1 Outbound Processing

When an IPSec implementation receives an outbound packet it uses the relevant selectors (destination IP address and port, transport protocol, etc) to search the security Policy Database (SPDB) to ascertain what policy is applicable to the traffic. If IPSec processing is required and a SA or SA bundle has already been established, the SPDB entry that matches the selectors in the packet will point to the appropriate SA bundle in the SADB. If a SA has not been established, the IPSec implementation will create a SA and link it to the SPDB entry. The SA is then used to process the packet as follows:

1. **Generate or Increment Sequence Number** - The sequence number is used to prevent replay of previously transmitted packets. When a new SA is established the sender initializes its sequence number counter to zero. For each packet that the sender transmits, it increases the sequence number by one and insert the resulted value of the sequence number counter into the sequence number field of the ESP packet.
2. **Encrypt the Packet** - If the traffic requires confidentiality services the SA will specify the encryption algorithm to be used. The available choices of encryption algorithms depends on the IPSec implementation. However, only private-key cryptosystems are currently used because of the slow execution speed of public key cryptosystems compared to private-key enciphering

systems, as discussed in Chapter 1. For interoperability, all IPSec implementations must provide DES in CBC mode as an encryption algorithm option. When the encryption algorithm requires an Initial Vector(IV) as is the case with DES in CBC mode, the IV is carried in the first few bytes of the Payload Data field. As illustrated in Figures 3.14, the ESP header, the SPI (Security parameter index) and the Sequence number, are not encrypted. This is so because the destination node needs the SPI to look up the SA of the packet, and it requires the Sequence Number to ascertain whether or not the packet is a re-play of a previously transmitted packet. The authentication data is similarly not encrypted because the receiving node must authenticate the packet before it decrypts it. This will be discussed further in the discussion of ESP Inbound Processing. When encryption service is required the packet must be encrypted before the calculation of the ICV.

3. **Calculate the Integrity Check Value** - If the SA for the packet stipulates that ESP authentication service should be applied, the ICV is calculated using the values in all the fields of the ESP packet except the Authentication Data field, then the ICV is inserted in the Authentication Data field. The SA for the packet specifies the algorithm that should be used to generate ICV. The available choices of authentication algorithms vary with different IPSec implementation. However, for interoperability, all implementations are required to support HMAC with MD5, and HMAC with SHA-1. If fragmentation is required, the packet is broken into the appropriate sizes then sent on route to the destination node.

3.3.4.2 Inbound Processing

When a packet arrives at an IPSec host or security gateway, if the more fragment bit is set, this is an indication that there are other fragments that are yet to arrive. The IPSec application therefore waits until a fragment arrives with a sequence number that is similar to the previous ones, and has the more fragment bit not set. It then reassembles the IP fragments, then performs the following steps:

1. Use the destination IP address and IPSec protocol in the IP header (outer IP header if tunnel mode), and the SPI in the ESP header to look up the SA for the packet in the inbound SADB. If the lookup fails, it drops the packet and logs the error.
2. Use the SA found in (1) to process the ESP packet. This involves first checking to determine whether the selectors in the IP headers (inner header if tunnel mode) match those in the SA. If the selectors do not match, the application drops the packet and audit the error. If the selectors match, the IPSec application keeps track of the SA and the order in which it is applied relative to the others, and continues to do (1) and (2) until it encounters a non-IPSec extension header or a transport layer protocol.
3. Use the selectors in the packet to find a policy in the inbound SPD whose selectors match those of the packet.
4. Check whether the SAs found in (1) and (2) match the policy specified in (3). If the check fails, repeat steps(4) and (5) until all policy entries have been check or until the check succeeds.

5. If anti-replay is enabled, use the anti-replay window of the SA or SA bundle to determine if the packet is a replay. If the packet is a replay, drop it and log the error.
6. If the SA stipulates that authentication service is required, then uses the authentication algorithm and the private-key specified by the SA bundle to calculate the ICV for the packet and compares it with the value stored in the ESP Authentication Data field. If the two values differs, then discard the packet and audit the error.
7. If the SA indicates the application of confidentiality service use the cryptographic algorithm and the private-key that the SA specifies to decrypt the packet. Decryption processes are in general quite CPU and memory intensive. If the IPSec system is allowed to perform unnecessary decryption or encryption of packets, then the system will be vulnerable to denial of service attack. Consequently, when decryption or encryption is required, this service is applied after the packet has been successfully authenticated.

At the end of these steps if the packet has not been discarded, it is then passed to the transport layer protocol, or is forwarded.

3.4 The Internet Key Exchange (IKE) Protocol

3.4.1 IKE Protocol Concept

Prior to an IP packet being secured by IPSec , a security association (SA) must exist. As stated earlier, SAs may be created manually or dynamically. The Internet Key Exchange (IKE) is used to create them dynamically. IKE negotiates SAs on behalf of IPSec and populates the SADB.

IKE, described in [40], is a hybrid protocol. It is based on a framework defined by the Internet Security Association and Key Management Protocol (ISAKMP), defined in [41], and implements parts of two key management protocols, Oakley and SKEME [37]. In addition IKE defines two exchanges of its own.

- **ISAKMP** was developed by researchers at the National Security Agency (NSA) [41]. The NSA used to be a super secret organization whose existence was denied by the United States government. Recently, the NSA has come out of the shadows and its considerable expertise in cryptography and security has been put to visible use. ISAKMP is one such output.
- **Oakley** is a protocol developed by Hilarie Orman, a cryptographer from the University of Arizona [42]. It is a free-form protocol that allows each party to advance the state of the protocol at is own speed. From Oakley, IKE borrowed the idea of different modes, each party producing a similar result; an authenticated key exchange through the exchange of information at different speeds. In Oakley, there was no definition of information required to exchange with each message.

The modes were examples of how Oakley could be utilized to achieve a secure key exchange. IKE codified the modes into exchanges. By narrowing the flexibility of the Oakley model, IKE limits the wide range of possibilities that Oakley allows yet still provides multiple modes, albeit in a well-defined manner.

- **SKEME** is another key exchange protocol, designed by cryptographer Hugo Krawczyk [50]. SKEME defines a type of authenticated key exchange in which the parties use public key encryption to authenticate each other and "share" components of the exchange. Each side encrypts a random number in the public key of the peer and both random numbers (after decryption) contribute to the ultimate key. One can optionally do a Diffie-Hellman exchange along with the SKEME share technique for Perfect Forward Secrecy (PFS), or merely use another rapid exchange, which does not require public key operations, to refresh an existing key. IKE borrows this technique directly from SKEME for one of its authentication methods (authentication with public key encryption) and also borrows the notion of rapid key refreshment without PFS.

It is upon these three protocols ISAKMP, Oakley, and SKEME that IKE is based. It is a hybrid protocol: it uses the foundation of ISAKMP, the modes of Oakley, and the share and rekeying techniques of SKEME to define its own unique way of deriving authenticated keying material and negotiating shared policy, as is demonstrated in Figure 3.15. The contributions of Oakley and SKEME can be seen in the discussion of IKE itself, but the contributions of ISAKMP are sufficiently considerable to warrant a separate discussion.

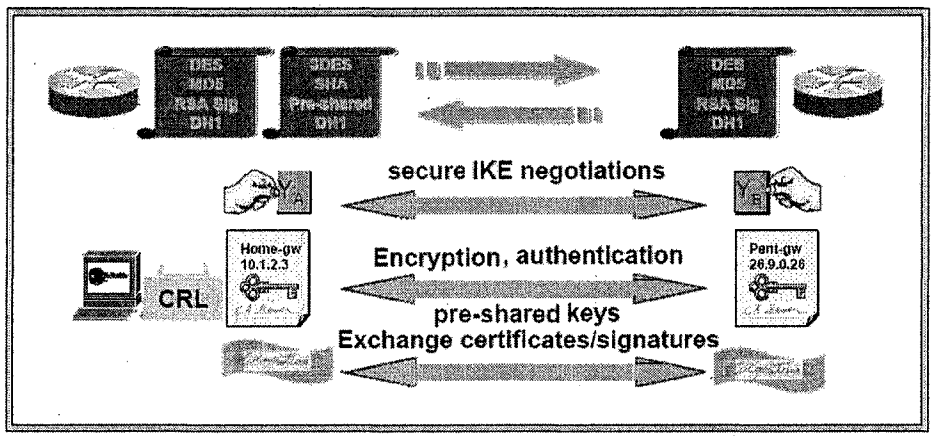


Figure 3.15: IKE parameters

3.4.2 ISAKMP

ISAKMP defines how two peers communicate, how the messages they use to communicate are constructed, and also defines the state transitions they go through to secure their communication. It provides the means to authenticate a peer, to exchange information for a key exchange, and to negotiate security services.

ISAKMP defines procedures and packet formats to negotiate, establish, modify and delete SAs.

ISAKMP provides a common framework for the format of SA attributes and the methodologies for negotiating, modifying and deleting SAs that different key exchange protocols can use. ISAKMP defines several payloads. These payloads provides modular building block for constructing ISAKMP messages. The ISAKMP header format and the Generic Payload header as well as the various payloads that IKE uses are given in Appendix A4 for interested reader.

It must be noted that IKE uses a variety of payloads as listed below:

1) Security Association Payload
2) Proposed Association Payload
3) Transform Association Payload
4) Key Exchange Association Payload
5) Identification Payload
6) Certificate Payload
7) Hash Payload
8) Signature Payload
9) Source Payload

For a description and format of the above payload consult the Appendix A4.

o Message Format

The ISAKMP specification has a complete description of payload formats and payload specific attributes.

Payloads are chained together in a message by using the next payload field in the generic header (see Figure 3.16). The ISAKMP header describes the first payload following the header, and each payload describes which payload comes next. The last payload in the message is zero.*

* ISAKMP communication is done via UDP port 500. Other port protocol combinations are possible, but UDP port 500 has been reserved by IANA (Internet Assigned Number Authority) for ISAKMP and all ISAKMP implementations are required to support communication over UDP port 500.

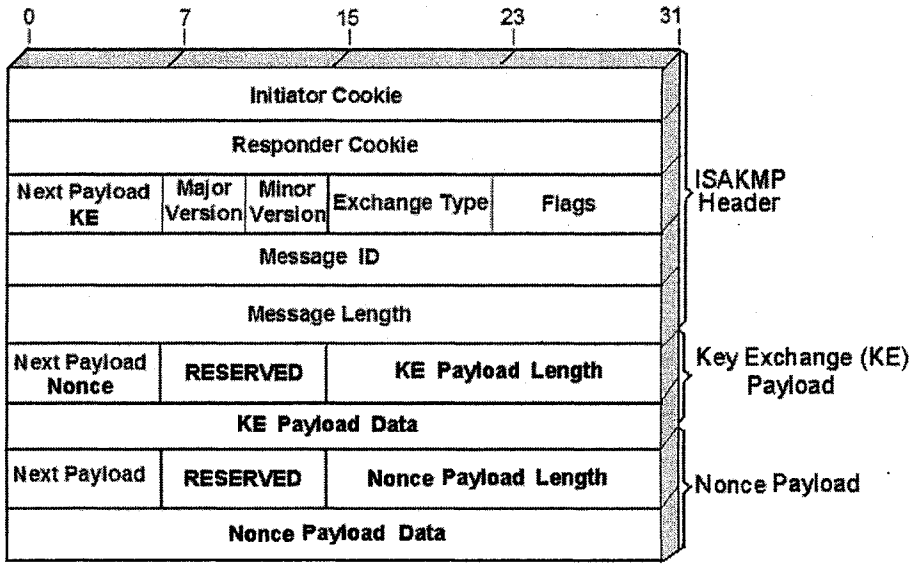


Figure 3. 16: ISAKMP Payloads Chained to form a Massage

3.4.3 Exchanges and Phases

ISAKMP describes two separate phases of negotiation. In the first phase, peers establish an authenticated and secure channel between themselves. In the second phase, the authenticated and secure channel is used to negotiate security services for a different protocol such as IPSec.

3.4.3.1 Exchange Phases

IKE presents different exchanges in modes which operate in one of two ISAKMP negotiation phases:

(1) Phase 1 Negotiation

In this phase, two ISAKMP communicating peers establish an ISAKMP SA which essentially is an agreement on how further negotiation traffic between the peers will be protected. This

ISAKMP SA is then used to protect the negotiation of SAs for other protocols, such as ESP or AH.

(2) Phase 2 Negotiation

This phase is used to establish SAs for other security services, such as AH or ESP. A single Phase 1 SA can be used to establish many Phase 2 SAs.

3.4.3.2 Exchange Modes

1. IKE currently defines four possible modes of exchange: Main Mode, Aggressive Mode, Quick Mode and New Group Mode. The first three negotiate SAs: whereas in the latter Transform Payload(s) encapsulated in Proposal Payload(s) which is/are encapsulated in Security Association Payload(s). These exchange modes will be discussed briefly in the subsequent subsection.

1. **Main Mode**

Main Mode is an adaptation of the ISAKMP "Identity Protection Exchange". It involves an authenticated Diffie-Hellman key exchange. It is used to negotiate Phase 1 ISAKMP SA. This exchange mode was designed to separate key exchange information from the identity and authentication information. The separation of these information provides protection for the identity of the communicating peers since the identity information can be exchanged under the protection of the previously generated Diffie-Hellman shared secret at the expense of two or three additional messages. Figure 3.17 shows an example of a Main Mode exchange.

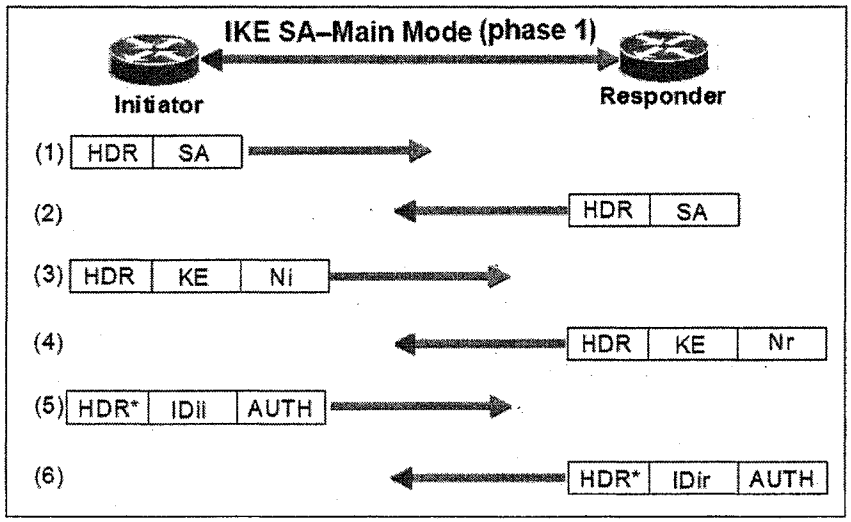


Figure 3.17: Example of a Main Mode Exchange

2. Aggressive Mode

Aggressive Mode is an implementation of ISAKMP "Aggressive Mode Exchange" it is used to negotiate ISAKMP Phase 1 SA where identity protection of the communicating peers is not required. This exchange mode allow the SA, Key Exchange and Authentication related payloads to be transmitted together. Combining these payloads into one message reduces the number of round-trips at the expense of not providing protection of the identity of the communicating peers. Figure 3.18 shows an example of an Aggressive Mode exchange.

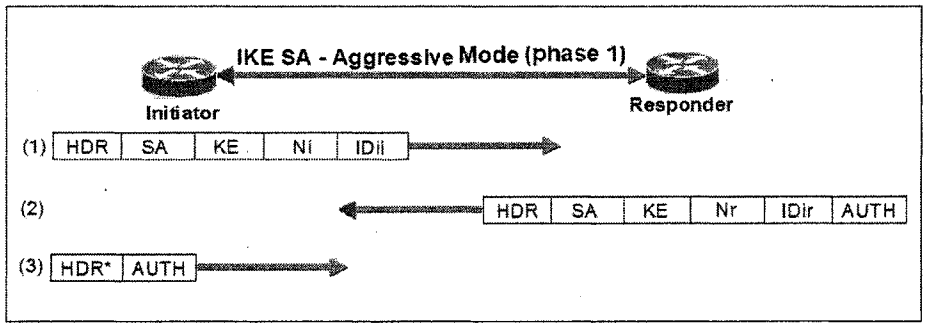


Figure 3.18: Example of an Aggressive Mode Exchange.

3. Quick Mode

Quick mode is used to negotiate phase 2 SA under the protection of the negotiated Phase 1 ISAKMP SA. All of the payloads in a Quick Mode exchange are encrypted. Figure 3.19 shows an example of a Quick Mode exchange.

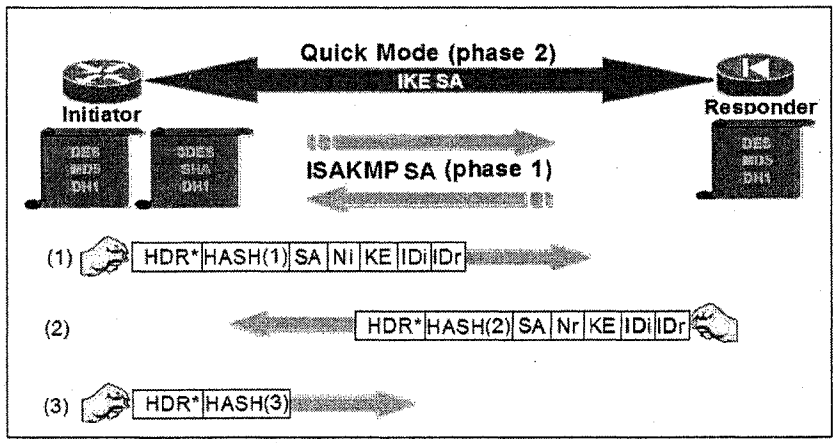


Figure 3.19 Example of a Quick Mode exchange.

4. New Group Mode

The New Group Mode (NGM) is used to negotiate new group for Diffie-Hellman key exchange. NGM is carried out under the protection of ISAKMP phase 1 exchange.

3.4.4 Oakley Groups

The IKE protocol allows negotiation of the group for Diffie-Hellman exchange. The original specification of IKE defined four groups that originated from the Oakley Protocol. These groups are presented below.

CHAPTER IV: A FRAMEWORK FOR IPSEC FUNCTIONAL ARCHITECTURE

4.1 Introduction

IPSec provides the essential components to codify on IP-based Virtual Private Network (VPN) infrastructure [42]. Widespread implementation is likely because of the security required in Internet communications. IPSec as described in Chapters II and III is composed of a number of different pieces that makes its one-piece implementation extremely difficult, if not impossible. The major issues in implementing IPSec are: 1) performance, 2) applicability to different operation environment that have special requirements, 3) integration with network management systems, 4) expandability to adapt to network changes, enhancements, and new policies, and finally 5) modular implementation that can easily be modified. Although, IPSec enabled routers and gateways has been available commercially by venders such as Lucent IPSec* VPN, Cisco IOS IPSec/SSL† VPN, and Nortel‡ VPN router, no clear functional design and/or architecture for realization of IPSec system is supplied in the literature.

---

* Lucent IPSec Virtual Private Network: A Technical Review, White paper, 2001

† IPSec Features in Cisco IOS Software Releases 12.3T, 12.3, and 12.2T, White paper, Sept. 2004.

‡ Nortel VPN Router Portfolio

Free S/WAN IPSec is a freely available commercial off-the-shelf implementation of IPSec, which is installed in all Linux gateways. This implementation lacks some functionality. In practice, integrating network management, routing, QoS, and security requires the interoperability of IPSec and key management over multiple platform (e.g. Cisco, Microsoft Windows 2000, and Red Hat Linux). Keromytis, etal implemented IPSec protocols for BSD/OS, Linux, OpenBSD, and NetBSD operating environment.[46] This implementation includes only kernel patches for processing incoming and outgoing IP datagrams, maintaining the database and policy table as well as a set of tools for configuring the policy table. In Ref [47], the authors propose an IPSec implementation on Xilinx Virtex II Pro FPGA. However, this can implement only some functions of IPSec features, and is based on processing the IPSec inbound and outbound traffic using FPGAs. Ref [48] proposes an FPGA-based adaptive cryptographic engine for IPSec architectures, and focuses on the encryption/decryption aspects only.

With security service proliferation, and as authentication and encryption algorithms become stronger, so does IPSec. It is important to note that IPSec is only as strong as the methods chosen for its implementation.

We are now in a position to propose a framework on a layered security functional architecture to implement the IPSec full range of functionality for enterprise networks. The proposed functional architecture is modular and makes it possible to compose IPSec software applications from products developed by different supplies at different times. The design requirements and specifications for the accompanying Security Management Information Base (SMIB) is describe in Sec.4.3.

4.2 IPSec Functional Architecture

IPSec provides security services at IP layer (Network layer) by enabling a system to select required security mechanisms, determine the algorithms to use for the services, and put in place any cryptographic keys required to provide the requested services. We propose structuring the IPSec functionality based on the framework of the logical security management system and method, as described in [44]. In this architecture basic functionality of IPSec is organized in logical function layers. The basic IPSec data is stored and managed by an IP security management information base (IPSMIB). The developed architecture, shown in Fig.4.1,

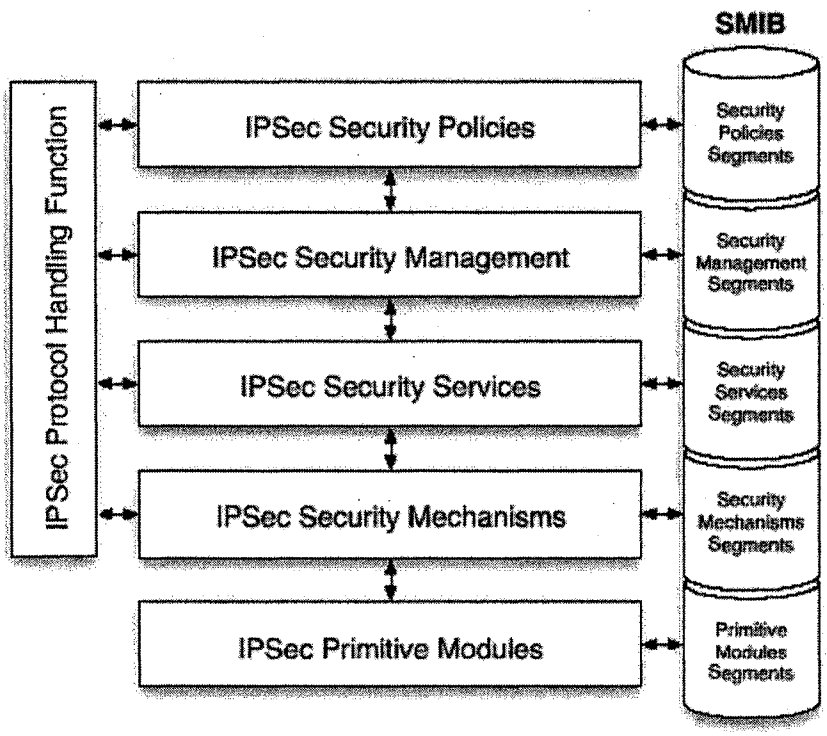


Figure 4.1: IPSec functional five logical layers.

implements IPSec functionality through five logical layers. Each layer has a modular architecture and is made of one or several "modules". Modules use the services of an infrastructure of independent lowerlayer functions to support their own offered functions and to communicate among themselves. It is seen that IPSec physical realization requires multiple modules at each layer. The five functional layers of IPSec are described in the following subsections.

4.2.1 – IPSec Policy & Business Management Function layer

4.2.1.1 - IPSec Policy Concepts

A policy defines a definite goal, course, or method of action to guide and determine present and future decisions. In the context of IPSec management, it can be seen that an IPSec policy consists of rules for providing security protection to IP traffic. As an example, a policy may call for all traffic originated from a given IP address to be protected by AH in tunnel mode using an integrity protection service based on the key hashing algorithm SHA-1.

A complicated IPSec policy may include multiple nested SAs.

a) Policy Definition Requirement

The design for a policy definition and management system must not constrain the richness of IPSec. Therefore, there are several requirements imposed on any policy system. These requirements are very high level because there is no single mechanism for policy definition and management. Different IPSec implementation will have different requirement. These are not all the requirements set of policy but merely a large set that any system will be abide by:

- Ability to configure multiple policies per host.
- Ability to configure policy per network.
- Ability to configure security at the granularity of an application.
- Ability to specify multiple protocols per flow.
- Ability to specify the desired action for each flow.
- Ability to configure tunnels of more than one level deep (nested tunnel)
- Ability to supply different IKE identities for access and authentication.
- Ability to selectively disable automated key management.
- Ability to support multiple security options.

b) Policy representation

Policy representation involves two aspects. The physical representation – defining the format in which the policy is represented, where it is stored, how it is updated, modified, and what protocols are used to update the policy;

- The interface representation – defining how the various IPSec components acquire and manage a policy.

Figure 4.2 shows the most popular paradigm that currently exists for policy configuration and distribution.

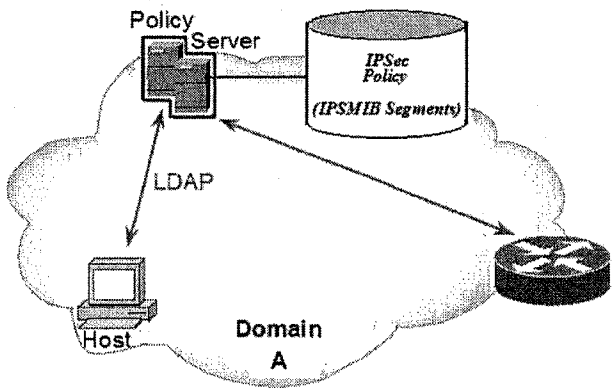


Figure 4.2: Policy deployment architecture.

The policy is stored centrally in a policy server. The policy server is responsible for maintaining the policy for all nodes (host and router) in the domain. The policy server provides the capability to configure security for all the nodes. The policy is either downloaded into the various nodes in the network or is fetched dynamically by the nodes using directory services protocol such as LDAP (Lightweight Directory Access Protocol).

c) Policy management

Policy management defines the following interfaces:

- Interface between the directory service and the IPSec Policy database located in IPSMIB segments;
- Interface between the User Interface application and the local policy database that allows read, write, and modify access to the database;
- Interface between IKE and policy database;
- Interface between kernel and the policy database.

4.2.1.2- IPSec Policy Management Function layer

This is the upper most functions in implementing IPSec in an enterprise environment. This function determines the general goals and policy for each security service implemented through IPSec, as set by the corporation's security vision. The current IPSec specifications don't identify any function in this layer. This policy management layer significantly simplifies the task of defining, deploying, and maintaining security policies across a network, therely significantly simplifying layer-scale IPSec deployment. In addition, the IPSecc Policy Management function can include the following functions: are as follows:

d) **Prevention and detection of IPSec security violation** – A set of rules can be Implemented in a rule – based system to detect intrusion by observing IPSec events and making a decision whether a given IPSec pattern of activities is suspicious.

e) **Network wide IPSec implementation policy** – Once this layer is in place, multiple IP security policy requirements can be accessed to support the many aspects of IPSec security across the enterprise network.

f) **Disaster Recovery**

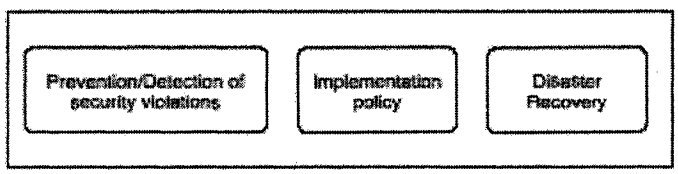


Figure 4.3: IPSec Policy and Business Management function Layer.

Currently, separate Policy-based IPSec management systems are employed to manage a network as a whole.

4.2.2 IPSec Management Function Layer

The IPSec security management function layer provides a completely integrated and centralized management option for IPSec implementation. This function will support multiple IPSec security services, VPN, and QoS with one effective configuration. This layer is concerned with security aspects which are outside normal scope of IPSec security services, but which are needed to support and control IPSec security services and mechanisms. The IPSec security management function, in general, are as follows:

a) **Policy Control and management of security services** – A set of security-related parameters and information to provision and control IPSec security services for centrally provisioned and managed for large – scale IPSec networks.

b) **Event Logging** – A function that maintains logs of connection attempts and traffic flow, including detailed IKE and IPSec negotiation.

c) **IPSec services monitoring** – A group of software functions to monitor IPSec tunnels activity, and can provide continuous traffic statistics (in real – time).

d) **User Interface** – A set of functions to manage the IPSec GUI (Graphical User Interface) and provide online help search capabilities.

e) **Interoperability** – A set of functions needed if the IPSec server is required to exchange network management information automatically with other network IP servers and IPSec servers. Note that not all IPSec management features are currently available with IP services routers.

f) **Recovery and Backup** – A set of functions for key and security state recovery in case of security breaches and backup software based on IPSec event log

massages. These tasks may be implemented monolithically or as individual software modules.

In addition to the above functions, this layer can include a module that implements the task of performing online secure upgrades of the modules in the lower layers of the architecture. The IPSec security management function layer is shown in Figure4.4 below.

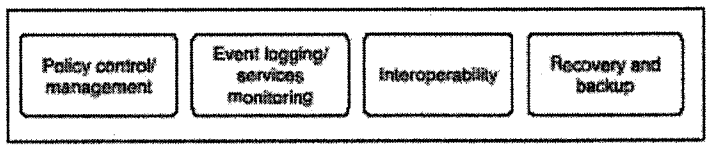


Figure 4.4: IPSec Security Management Layer

4.2.3 – IPSec Security Services Layer

The services provided by IPSec are implemented through modules in the security services layer of the proposed architecture. These services are as follows [48]:

1. Access control
2. Integrity
3. Authentication
4. Confidentiality (encryption) and privacy (limited traffic flow confidentiality)
5. Rejection of replayed packets

This layer is illustrated in Figure 4.5 below.

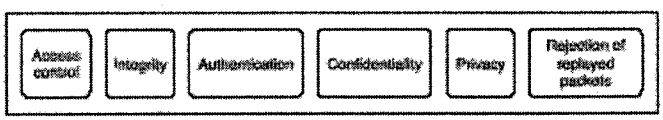


Figure 4.5: IPSec Security Services Layer

As per the IPSec specification, these services are provided by two protocols, namely Authentication Header (AH) [49] and Encapsulating Security Payload (ESP) [25]. The services provided by each protocol are shown in Table 1 below [8]. Note that ESP can be applied in two manners: with and without authentication.

Table 4.1: Security Services Provided by AH and ESP

	AH	ESP (encryption only)	ESP (encryption and authentication)
Access control	Y	Y	Y
Integrity	Y		Y
Authentication	Y		Y
Confidentiality		Y	Y
Privacy		Y	Y
Rejection of replayed packets	Y	Y	Y

4.2.4 – IPSec security Mechanism Layer

In some literature, \security services" and \security mechanisms" are interchangeable. In the proposed architecture we make a clear distinction between services and mechanisms, with a service in the previous layer implementing one or more mechanisms in this layer. The manner in which mechanisms are used to implement the security services is dependent on the specific IPSec implementation [37].

The security mechanism layer is composed of stand-alone modules that are needed to support the IPSec security services. Of the IPSec security services in use, cryptographic techniques form the majority [43].

The IPSec security mechanisms can be divided into five groups:

1. *Encryption*: Symmetric encryption techniques are used to provide confidentiality, while public-key encryption techniques are used in authentication, certificates, and key management mechanisms.

2. *Message authentication*: Hashed Message Authentication Codes (HMACs) [5] are used to provide authentication and data integrity [43].

3. *Key management*: The key management mechanism involves the determination and distribution of secret keys. Two types of key management are required as per the IPSec specification: manual key management where a system is manually configured with its own keys and the keys of any communicating systems, and automated key management where keys are generated on-demand [48]. The default automated key management protocol for IPSec is Internet Security Association Key Management Protocol (ISAKMP)/Oakley [46]. ISAKMP provides the framework for key management, while Oakley is a secure adaptation of the Diffe-Hellman key exchange algorithm [46].

ISAKMP does not specifically dictate a key exchange algorithm; Oakley is the key exchange algorithm that was required for use with the first version of ISAKMP [43]. Other key exchange algorithms may be used, such as RSA [41] or Diffe-Hellman.

4. *Certificates*: Public-key certificates are used by key management services. The types of certificates or certificate related information may include [43]: PGP certificates, Kerberos tokens, X.509 certificates, DNS signed of keys, and SPKI certificates.

5. *Digital signatures*: Digital signatures are used in certificates to verify the authenticity the certificate and the identity of the entity using the certificate. DSS [12] or RSA may be used. These mechanisms are illustrated in Figure 4.6 below.

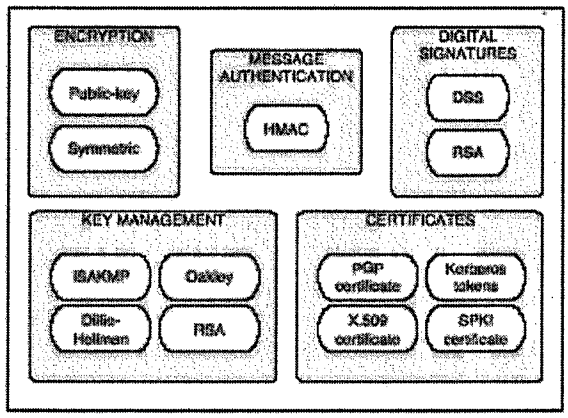


Figure 4.6: IPSec Security Mechanisms Layer

4.2.5 – IPSec Primitive Function Layer

The lowest layer of the proposed architecture consists of the mathematical operations and algorithms required by the security mechanisms in the previous layer [7].

We propose the implementation of each primitive mathematical module as one or more atomic functions; that is, each function is self-contained, with its performance not affecting any of the other functions in that module.

The rudimentary mathematical functions required by the services and mechanisms of IPSec can be divided into five groups:

1. *Prime number generation*: Some IPSec mechanisms, such key management, require the selection of one or more very large prime numbers at random; thus, the prime number generation module must contain functions to generate these numbers. At present there are no computationally simple techniques that definitively yield arbitrarily large primes; the procedure that is used is to generate at random, using a pseudo random number generator, an odd number of the desired order of magnitude, and then test whether this number is prime. A variety of tests and algorithms for primality have been developed; one of the more efficient and popular algorithms in use is the Miller-Rabin algorithm [32].

2. *Modular arithmetic*: The modular arithmetic module should contain functions for modular multiplication, multiplicative inverse, fast exponentiation, Chinese Remainder Theorem (CRT), and discrete logarithms.

3. *Encryption fundamentals*: The current specification of IPSec indicates that DES in cipher block chaining (CBC) mode must be supported for symmetric encryption [43]; however, other algorithms may be used for symmetric encryption, such as: three-key triple DES [15], RC5 [16, 17], IDEA [18], three-key triple IDEA [15], CAST [19], and Blowfish [20, 21].

4. *One-way hash codes*: The authentication mechanisms that must be supported in an IPSec implementation are HMAC-MD5-96 and HMAC-SHA-1-96 [43]; both use the first 96 bits of the value returned by the Hashed Message Authentication Code (HMAC) algorithm [5] using either the MD5 [10] or SHA-1 [23] hash codes.

5. *Elliptic curve arithmetic*: The elliptic curve arithmetic library contains the algorithms required to perform elliptic curve cryptography (ECC). Presently ECC is not used in IPSec implementations, but it is anticipated that it will be used in the future.

The modules in this layer are depicted in Figure 4.7 below.

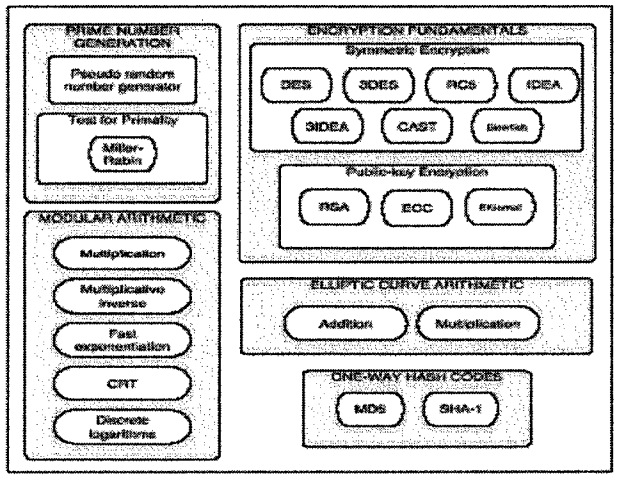


Figure 4.7: Primitive Modules Layer

4.3 IPSec Protocol Handling Function

The current IPSec specification employs three protocols:

1. Authentication Header (AH) [3]
2. Encapsulating Security Payload (ESP) [4]
3. Internet Security Association and Key Management Protocol (ISAKMP) [5]

Handling these protocols for both transport mode and tunnel mode in an IPSec network requires accessibility to the modules in the layers of the proposed architecture. To effectively realize IPSec, we proposed the implementation an \IPSec Protocol Handling Function" as shown previously in Figure 1. The IPSec protocol handling function performs these two overall functions:

1. For outgoing data, it interfaces with the layers to perform the necessary processing into IPSec packets, and then passes the packets to the operating system of the system on which IPSec is realized for transmission over the network.
2. It accepts incoming IPSec packets from the operating system, extracts the data, and passes the data on to the appropriate IPSec layer for processing into data usable by the system on which IPSec is implemented.

In a typical protocol handling process for IPSec we need a dispatcher and a message processing unit (MPU), as depicted in Figure 4.8 below. For outgoing packets, the dispatcher accepts outgoing data from the system, determines what protocol needs to be applied to the data and passes it on to the appropriate MPU. The MPU prepares the data for transmission by interfacing with the modules in the layers, wrapping the data in the appropriate IPSec header, and then returning the data to the dispatcher, which then

dispatches it to the operating system for transmission. The reverse process is carried out for incoming messages.

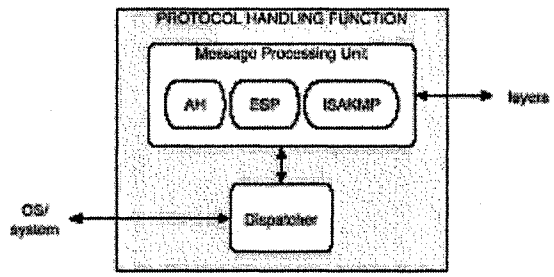


Figure4.8: IPSec Protocol Handling Function

4.4. IPSec Security Management Information Base (IPSec SMIB)

Security services applied to IP traffic between a sender and a receiver are defined by a security association (SA). The SA is defined for one direction only; if a two-way relationship is required, then two SAs are required. In each IPSec implementation there is a security management information base (SMIB) that defines and manages the parameters associated with each security association (SA). Multiple SMIB entries may relate to a single SA, or one SMIB entry may relate to multiple SAs [43]. The SMIB is divided into segments, one segment for use by each IPSec functional layer. Content or form for the storage, implementation, or usage of information is not explicitly defined since it is highly dependent on the specifics of each individual IPSec implementation [37].

Parameters that define a SA are [43]: sequence number counter, sequence number overflow, anti-replay window, AH/ESP information, lifetime of the SA, IPSec protocol mode (i.e., tunnel or transport), and path MTU.

Each SMIB entry is defined by SA selectors which enables the mapping of a particular SA to IP traffic. The following selectors are used [8]: destination/source IP

address, user ID, data sensitivity level, transport layer protocol, IPSec protocol (AH or ESP or AH/ESP), source and destination ports, IPv6 class, IPv6 flow table, and IPv4 type of service.

In addition to SA information, the SMIB may contain information such as control lists, IDs for networked resources, and session keys [37].

4.5 – IPSec Functional Layers Summary

The proposed IPSec layered architecture is a highly configurable architecture with the inherent flexibility needed to realize redundancy and adaptation needed for atypical distributed computing scenario consisting of a collection of machines connected by local or wide-area communication network. Application level processes communicate by using a communication subsystem that typically consists of IP, some transport-level protocol such as TCP or UDP, and potentially, some middle were- level protocols. The designed IPSec system provides the abstraction of a bidirectional point-to-point communication channel for each connection that is opened. The design uses the established algorithms in the security mechanism layer to provide security guarantees for the needed security services layer. Each of these security services is associated with one or more software modules that implement the corresponding security feature. The design is configurable in the sense that specific modules can be selected based on user and application requirements. First, it allows the user to determine which IPSec property to use if a certain property is not required or if it is already implemented by some other system layer, then no module that implements the property needs to

be included. Second, it allows the user to select which mechanism (or module) to use to implement a particular IPSec property. The user can evaluate the modules by various metrics, including the relative level of security and resource utilization.

All modules are synthetically independent from one another by virtue of the execution model supported by the IPSec processes, which means that any combination is synthetically valid. However, the different IPSec security properties have certain semantic dependencies and ordering constraints that are reflected as semantic dependencies and ordering constraints of intra-layers and inter-layers. In practice, these dependencies are recorded as a configuration graph that can be used to ensure any chosen IPSec configuration is semantically consistent.

4.6 Location of IPSec Architecture

Within a LAN, IP traffic is typically nonsecure since all of the systems in the network are part of the trusted network. IPSec operates in a networking device, such as a firewall or a router, that connects the LAN to the outside world [8]. The proposed architecture can be implemented on the networking device, or on a separate system that interfaces with the networking device.

A typical IPSec setup is depicted in Figure 4.9 below. In the figure, there is a LAN and a client system external to the LAN. Traffic within the LAN is nonsecure, but all traffic between the client system and LAN is secured using IPSec protocols. In this setup, the proposed architecture is implemented on its own system that interfaces with the network router of the LAN, while the architecture is part of the client system.

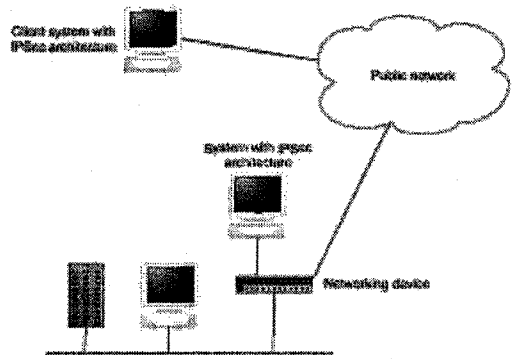


Figure 4.9: Location of IPSec Architecture

CHAPTER V: IPSec SCENARIOS

5.1 – IPSec SCENARIOS

5.1.1 Scenario 1 (An IPSec Session Scenario)

Let us assume a remote IPSec client system wants to establish an IPSec transport connection to a remote server. The client chooses to use ESP encryption and authentication. However, before ESP can be used keys need to be exchanged between the client system and the remote server using ISAKMP.

It is assumed that the base ISAKMP message exchange type is used. Four messages are exchanged between the client and server in this scenario; the first two establish an SA between the two systems, and the second two exchange key information [43]. Also, it is assumed that Diffe-Hellman is the key exchange algorithm used, and HMAC-MD5-96 is used to authenticate messages where appropriate.

The following steps and Figure 5.1 illustrate how the first two messages that establish theSA would be threaded throught the proposed architecture.

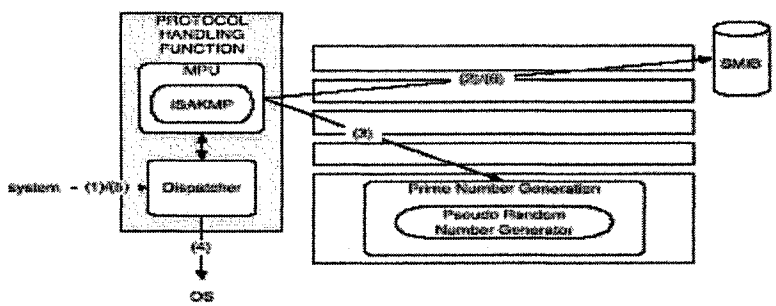


Figure 5.1: SA Negotiation

1. The client system determines the parameters for the SA between itself and the server. The client system interfaces with the IPSec protocol handling function, and the dispatcher

determines that ISAKMP will be used and passes the SA information to the ISAKMP MPU.

2. The ISAKMP MPU sends the SA data to the SMIB.

3. The ISAKMP MPU invokes the random number generator in layer 1 to obtain a nonce that is included in the initial ISAKMP message.

4. The message is passed on to the dispatcher for transmission to the remote server.

5. The client system receives a response (second message) from the server and passes it on to the IPSec protocol handling function. The dispatcher determines the data is a ISAKMP message and passes it on to the ISAKMP MPU.

6. The MPU verifies that the parameters of the SA have been accepted; any changes are written to the SMIB.

The third message is from the client to the server and contains the key exhange material of the client. Creation of this message using the proposed architecture is explained in the following steps and illustrated in Figure 5.2.

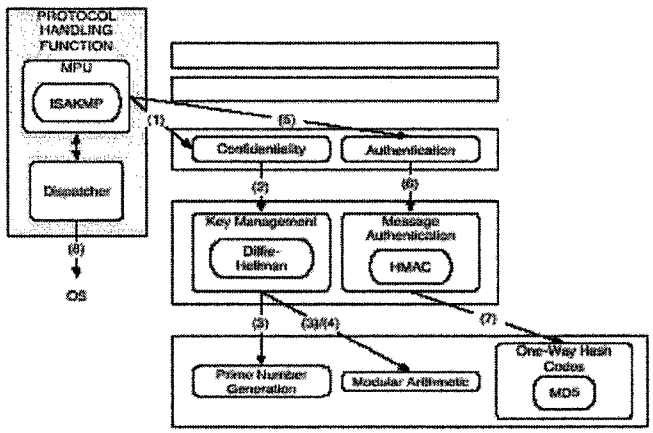


Figure 5.2: Third Message of the ISAKMP Base Exchange

1. The ISAKMP MPU invokes the confidentiality module in layer 3.

2. The confidentiality module invokes the Diffe-Hellman sub module of the key management module of layer 3.

3. The Diffe-Hellman module invokes the prime number generation and the modular arithmetic modules in layer 1 to choose a prime number and private key that is a primitive root of the prime number.

4. The Diffe-Hellman module invokes the modular arithmetic module in layer 1 to calculate the client's public key.

5. The ISAKMP MPU invokes the authentication module in layer 3.

6. The authentication module invokes the HMAC sub module of the message authentication module of layer 2.

7. The HMAC module invokes the MD5 submodule of the one-way hash module of layer 1 to calculate the authentication value over the message.

8. The ISAKMP MPU passes the message that contains the public key and identity of the client system and the authentication data to the dispatcher. The dispatcher passes the message to the client system for transmission to the remote server.

The fourth message is from the server to the client. The client uses this message to obtain the key information of the server to establish the session key. The processing of this message using the proposed architecture is explained in the following steps and illustrated in Figure 5.3.

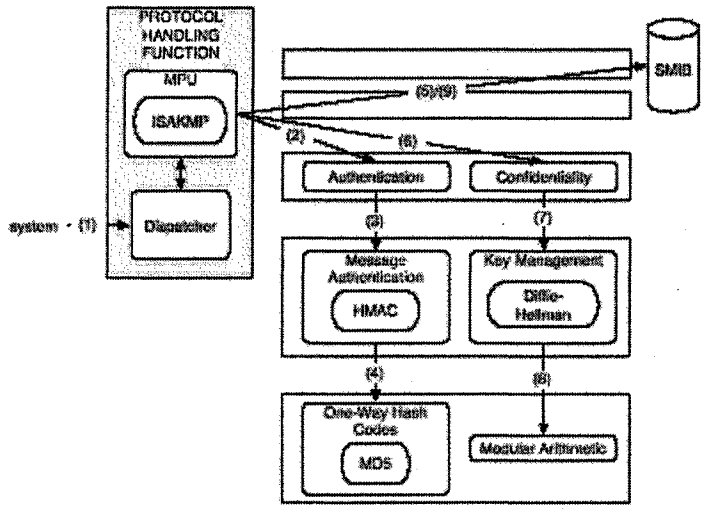


Figure 5.3: Fourth Message of the ISAKMP Base Exchange

1. The client system receives a response from the server and passes it on to the IPSec protocol handling function. The dispatcher determines the data is a ISAKMP message and passes it on to the ISAKMP MPU.

2. The MPU verifies the authentication data of the message by invoking the authentication module in layer 3.

3. The authentication module invokes the HMAC sub module of the message authentication module in layer 2.

4. The HMAC sub module invokes the MD5 sub module of the one-way hash module in layer 1.

5. The MPU verifies that the calculated HMAC-MD-96 matches that contained in the message. The MPU verifies the identity of the server by comparing the ID in the message to that in the SMIB entry.

6. The MPU retrieves the server public key from the message and invokes the confidentiality module in layer 3.

7. The confidentiality module invokes the Diffe-Hellman sub module of the key management module in layer 2.

8. The Diffe-Hellman submodule invokes the modular arithmetic module in layer 1 to generate the secret key.

9. The secret key is stored in the SMIB; this key will be used as the session key to encrypt all subsequent messages.

Now that a session key has been generated, ESP authentication and encryption can now be applied using the proposed architecture. This process is illustrated in Figure 5.4 and explained in the following steps.

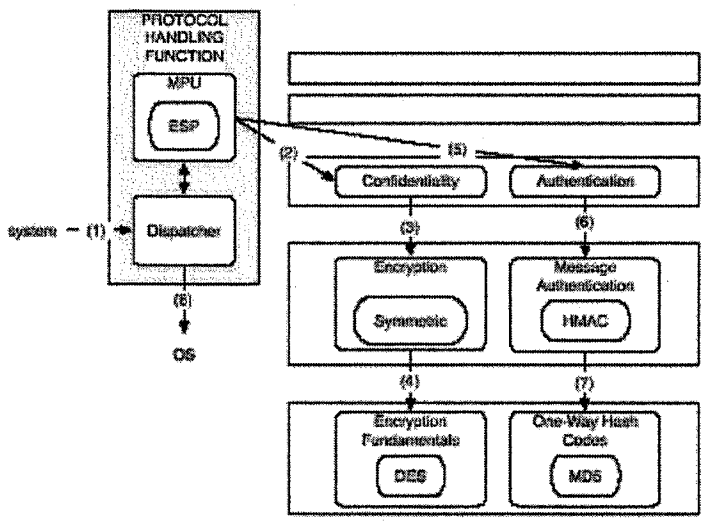


Figure 5.4: IPSec Example

1. The client system interfaces with the IPSec protocol handling function. The dispatcher determines the data will use ESP and passes the data onto the ESP MPU.

2. The ESP MPU invokes the confidentiality service in layer 3.

3. The IPSec confidentiality service in layer 3 invokes the symmetric encryption mechanism in layer 2. The encryption is applied to the IP payload.

4. Let us assume that the algorithm used for encryption is DES; thus the DES function in the encryption fundamentals module in layer 1 is invoked. The key used for encryption is retrieved from the SMIB.

5. The ESP MPU now invokes the authentication service in layer 3.

6. The authentication service relies on the IPSec authentication mechanism in layer 2. The HMAC module of the message authentication module is invoked.

7. Asssume HMAC-MD5-96 is used for authentication. The HMAC module in layer 2 calls the MD5 function of the one-way hash codes module in layer 1; the first 96 bits of the HMAC value are taken as the authentication data.

8. The authenticated and encrypted ESP message is passed on to the dispatcher which then interfaces with the operating system to transmit the data over the connection.

5.1.2 Scenario 2

Two hosts, A and B, with IP address 1.1.1.1 and 2.2.2.2 have established two SAs, SA_1, SA_2. (The SADB shown in the Figure 5.5 does not have all the fields. The sequence number, lifetime, and other fields have been left out). SA_1 is configured for ESP with 3DES and SA_2 ESP with DES. This information is stored in the SPD.

SA_1 is used for very secure applications; such as banking, which runs on a TCP using port 1000. After IPSec processing, if the policy is not checked to determine if the banking application did in fact use 3DES, then security is compromised. The policy can be checked only after the IPSec processing as the selectors are not available before because they may be encrypted.

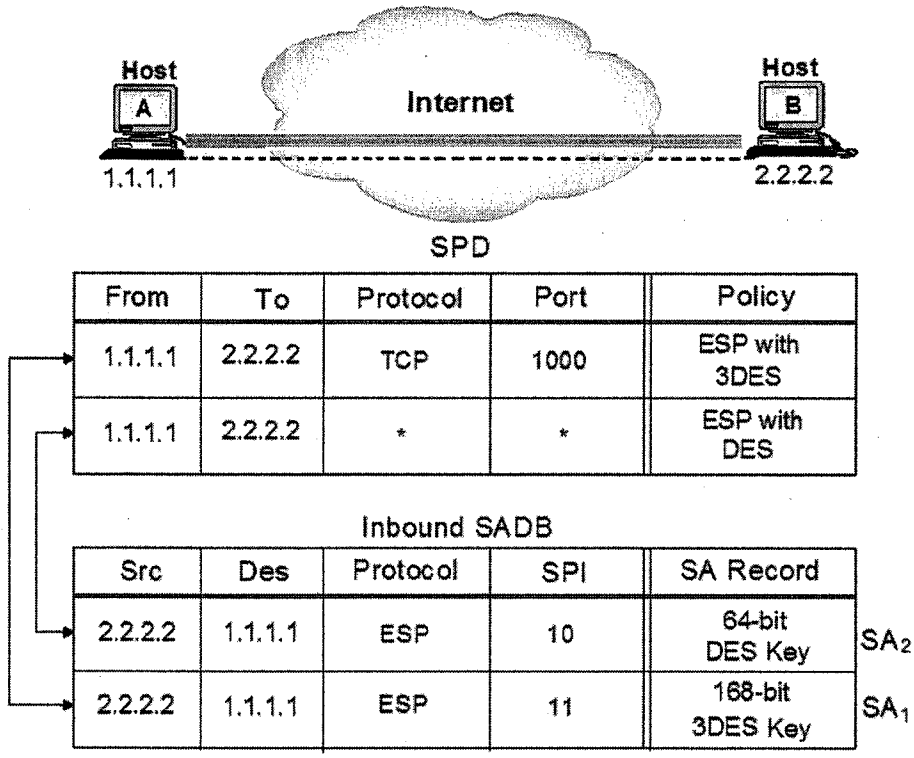
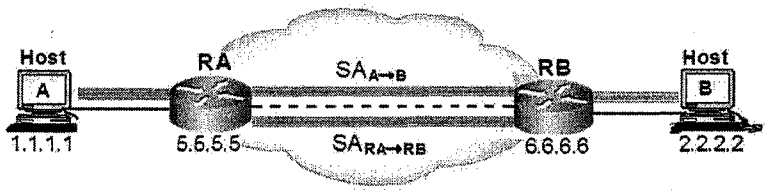


Figure 5.5: Flow of specific IPSec SMIB.

5.1.3 Scenario 3

Let us consider the example network diagram shown in Figure 5.6 For outbound processing, we will consider an HTTP packet (TCP, port 80) generated by host A destined to host B (a Web server) traversing routers RA and RB. The policy on A for packets destined to host B mandates using AH in transport mode using HMAC-MD5. The policy on router RA mandates that all packets destined to the network 2.2.2/24 be encrypted with ESP using 3DES and tunneled to RB.



A's SPD

From	To	Protocol	Port	Policy
1.1.1.1	2.2.2.2	Any	Any	Transport AH with HMAC MD5

A's Outbound SADB

Src	Des	Protocol	SPI	SA Record	
1.1.1.1	2.2.2.2	AH	10	MD5 Key	$SA_{A \to B}$

RA's SPD

From	To	Protocol	Port	Policy	Tunnel dst
1.1.1/24	2.2.2/24	Any	Any	Tunnel ESP with 3DES	6.6.6.6

RA's Outbound SADB

Src	Des	Protocol	SPI	SA record	
5.5.5.5	6.6.6.6	ESP Tunnel	11	168-bit 3DES Key	$SA_{RA \to RB}$

Figure 5.6: Example of IPSec outbound processing.

5.2 – IPSec Deployment

In this section, we are going to illustrate how the security services can of designed IPSec system be used to provide different IPSec solutions.

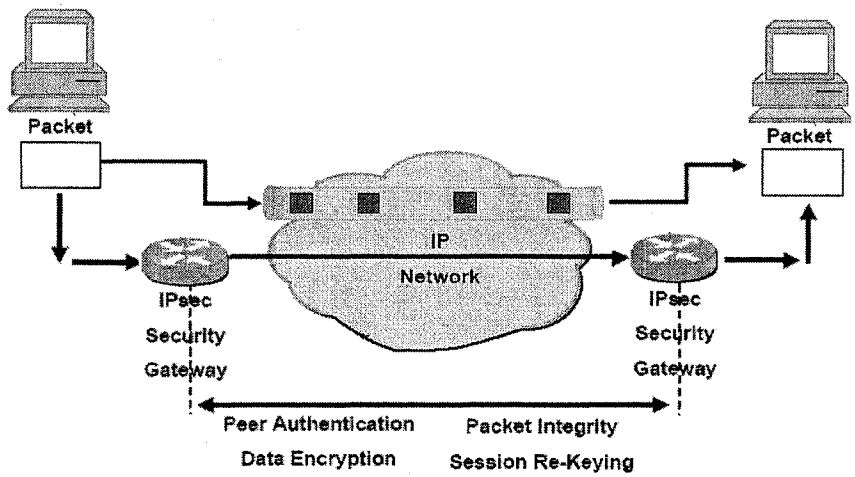


Figure 5.7: Schematic of A security Gateway

5.2.1- Virtual Private Network (VPN)

How do we ensure the safe passage of data across a shared infrastructure?

The answer is to deploy a secured virtual private network (VPN), VPNs are networks deployed on a public network infrastructure that utilize the same security, management, and quality of service policies that are applied in a private network. VPNs provide an alternative to building a private network for site to site communication over a public network or the Internet. Because they operate across a shared infrastructure rather than a private network, companies can cost effectively extend the corporate WAN to telecommuters, mobile users, and remote offices as well as to new constituencies, such as customers, suppliers, and business partners. Figure 5.4 shows the schematic of IPSec VPN configuration.

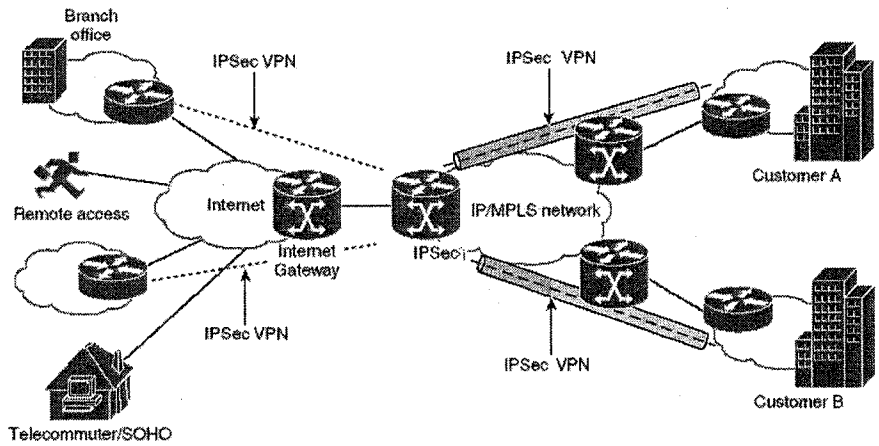


Figure 5.8: Schematic of IPSec VPN Configuration

Traditional private WANs connect customer sites via dedicated point to point links. This means that multiple independent circuits have to terminate at the corporate

network egress, making the deployment nonscalable and difficult to maintain. VPNs extend the classic WAN by replacing the physical point to point links with logical point to point links sharing a common infrastructure, allowing all the traffic to be aggregated into a single physical connection. This scenario results in potential bandwidth and cost savings at the network egress. Because customers no longer need to maintain a private network, and because a VPN itself cheaper to own and offers significant cost savings over private WANs, operation costs are reduced.

CHAPTER VI: CONCLUSIONS

6.1 Conclusion

The IPSec protocol provides the security functionality for interconnecting end systems across multiple networks. For this reason, IPSec is implemented in each end system and in network gateways, which provide connection between network. The implementation of IPSec should effectively support public key cryptography, open standards and interfaces, and bring together an otherwise dispersed security services into an integrated solution under a complete security management system.

In this thesis a framework for IPSec implementation architecture was developed. For our purposes, the developed IPSec implementation provides a useful overview of many of the concepts of network security that IPSec deals with, and it is flexible to allow for enhancement and future modifications. The developed architecture treats each security function as a "block box". This has two benefits:

First, an existing implementation of a security "function" piece can be used as a "module" in implementing IPSec. In this way, the bulk of the IPSec required software code is prepackaged and ready to use with no or slight modification. Second, if it is ever desired to replace a given functionality in an IPSec implementation all that is required is to remove the existing "function module" and drop in the new module. This could be done if a new more efficient "function piece" were desired. More important, if the security of one of the embedded function pieces were compromised, the security of the overall IPSec implementation could be retained simply by replacing the affected embedded function piece with a more secure one.

The proposed IPSec functional architecture is useful to network security engineers as a way of managing and administering the task of provisioning the network IPSec requirements. Furthermore, because the development is a systematic way of structuring the IPSec functionality and standards, computer and communication vendors can benefit from it by developing products that related to this structured implementation of IPSec

services and mechanisms. The IPSec services and the applications it supports are implemented as a collection of discrete modules. A certain set of modules may be developed by a particular vendor, while a different set of modules may be developed by another vendor. In addition, the modular structure of the implementation architecture lends itself to defining different version of each module. This in turn makes it possible to:

1) define alternative or enhanced capabilities for certain aspects of IPSec without needing to go to a new version of the entire standard (e.g., IPv6 security), and
2) clearly specify coexistence and transition strategies.

6.2 Concluding Remarks

The realization approaches outlined for the IPSec functions in this thesis can provide customizable communication security by allowing the user to choose which security attributes are required for a connection and which algorithms are used to implement these attributes.

The proposed IPSec functional architecture can provide a highly configurable secure communication service with the inherent flexibility needed to realize redundancy and adaptation needed for a typical distributed computing environment consisting of a collection of machines connected by a local or wide-area communication network[49].

6.3 Future Work

Future work shall include the development of a lab-grade prototype. Experimentation with the prototype and the development of a family of secure communication scenarios with different execution characteristics are to be performed. A special focus should be developing modules that are scalable and that minimize the synchronization required to support coordinated execution behavior. Performance analysis experiments should be preformed.

References:

[1] Tanenbaum, Andrew S., Computer Networks, Prentice-Hall Inc., 3rd Eddition 1996.

[2] Carl- Mitchell, Smoot and Quarterman, John S., Practical Internetwroking with TCP/IP and UNIX, Addison- Wesley Publishing Company, 1993.

[3] Baran, Paul, "On Distributed Communications: I Introduction to Distributed Communications Network" *MEMORANDUM RM-3420-PR*, RAND Corporation, August 1964.

[4] Braden R., Crocker S., Huitema C., "Security in the internet Architecture", MIT Laboratory for Compute, Trusted Information Systems, Inc. Request for Comments: 1636 (RFC – 1636), June1994.

[5] Hintema, C., IPv6: The New Internet Protocol Upper Saddle River, NJ: Prentice Hall 1998.

[6] Erfani, S., Security Management System and Method, US Patent 6,542,993 B1, April 1, 2003.

[7] Stevens, W. Richards, TCP/IP Illustrated Vol. 1 The Protocols, Addison Wesley Publishing Company, 1994.

[8] Metcalfe, R. M and Boggs, DR, "Ethernet: Distributed Packet Switching for local Computer Networks," *Commun. of the ACM.* Vol 19 pp. 395-104. july 1976.

[9] Latif, A, Rowlance, EJ, and Adams, R.H "IBM 8209 LAN Bridge, "IEEE Network Magazine, Vol.6, pp, 28-37. May/ June 1992.

[10] Ischer, W, Wallmeier, E, Worster, T, Davis, S.P. Hayter, A, "Data Communication Using ATM: Architecture, Protocols and Resource Management. "IEEE Communication Magazine, Vol,32, pp,24-33, August 1994.

[11] Goralski, W.J,. Introduction to ATM Networking," New York: McGraw Hill, 1995.

[12] Kyas, O. ATM Networks, London: International Thompson Publishing, 1995.

[13] Plummer, David C., "Ethernet Address Resolution Protocol." Netwrok Working Group Request for Comments (RFC826), November 1982.

[14] Finalyson, R, Timothy, M, Mogul, J, Theimer, M, "A Reverse Address Resolution Protocol, Network Working" Group Request for Comments (RFC 903), June 1984.

[15] Postel, J.B., "Internet Protocol," Reguest for comments (RFC 791), September 1981.

[16] Deering, S.E, "Host Extensions for IP Multicasting, "Request for Comments (RFC 1112), August 1989.

[17] Postel, J.B., "Transmission Control Protocol," Request for comments (RFC 793), September 1981.

[18] Postel, J.B., "User Datagram Protocol," Request for comments (RFC 768), August 1980.

[19] Doraswamy, N. and Harkins, D. "IPSec the New Security Standard for Internet, Intranet, and VPNs" Prentice Hall PTR Internet Infrastructure Series2nd edition, 2003.

[20] National Institute of Standard and Technology, "Data Encryption Standard (DES)," Federal Information Processing Standards Publication 46-3 (FIPS PUB 46-3), October 25, 1999.

[21] Schneier, B., "Description of a New Variable- Length Key, 64-bit Block Cipher (Blowfish)," *Proceedings Workshop on Fast Software Encryption* Published by Springer-Verlag, December 1993.

[22] Adams, C, "The CAST-128 Encryption Algorithm," Request for Comments (RFC 2144), May 1997.

[23] Lia, X and Massey, J, "Markov Cipher and Cryptanalysis" *Proceedings EUROCRYPTO 91*, Published by Springer Verlag, 1991.

[24] Lia, X and Massey, J, "A Proposal for a New Block Encryption Standard," Proceedings, *EUROCRYPTO 99*, Published by springer- Verlag, 1999.

[25] Kent Stephen, Atkinson Randall. IP Encapsulating Security Payload (ESP), "Description of a packet encryption extension to IPv4 and IPv6", IETF RFC 2406.

[26] Rivest, R., "RC5 Encryption Algorithm," *Proceedings Second International Workshop on Fast Software Encryption*, Published by Springer-Verlag, December 1994.

[27] Baldwin, R., Rivest, R., " The RC5, RC5- CBC, RC5- CBC-Pad, and RC5-CTS Algorithms, " Request for Comments (RFC 2040), October 1996.

[28] Riverst, R., Shamir, A., and Adleman, L., " A Method for Obtaining Digital Signatures and Public Key Cryptosystems," Communications of the ACM, February 1978.

[29] Doraswamy, N. and Harkins, D. "IPSec the New Security Standard for Internet, Intranet, and VPNs" Prentice Hall PTR Internet Infrastructure Series2nd edition, 2003.

[30] Rivest, R., "The MD4 Message Digest Algorithm, "Request for comments (RFC 1320), April 1992.

[31] Stallings, William, "Cryptography and Network Security: Principle and Practice", Prentice Hall Inc., 3rd ed. 1999.

[32] H. Krawczyk, M. Bellare, R. Canetti, "HMAC: Keyed-Hashing for Message Authentication" IBM UCSD Request for Comments: RFC – 2104. February 1997.

[33] Diffie, W. and Hellman, M., "Multiusers Cryptographic Techniques," Procedings of the AFIPS National Computer Conference, June 1976.

[34] Baldwin, R., Rivest, R., "The RC5, RC5- CBC, RC5- CBC-Pad, and RC5-CTS Algorithms, "Request for Comments (RFC 2040), October 1996.

[35] Stallings, William, "Cryptography and Network Security: Principle and Practice", Prentice Hall Inc., 3rd ed. 1999.

[36] Kent S., and Atkinson R., "An overview of security architecture", Request for Comments: RFC – 2401, November 1998.

[37] Manghan Douglas, Schertler Mark; Schneider Mark; Turner Jeff. Internet Security Association and Key Management Protocol (ISAKMP) IETF RFC – 2408, November 1998.

[38] Diffie, W., and Helman, M.,"Multiuser Cryptographic Techniques," IEEE Trans. Info., Theory, Vol. November 1976.

[39] Kent Stephen, Atkinson Randall. IP Encapsulating Security Payload (ESP), "Description of a packet encryption extension to IPv4 and IPv6", IETF RFC 2406.

[40] Harkins D., Carrel D. The Internet Key Exchange (IKE), IETF RFC 2409, November 1998.

[41] Maughan D., Schertler M. "Internet Security Association and Key Management Protocol (ISAKMP)," RFC 2408 Request for Comments: 2408 National Security Agency, November 1998.

[42] Orman, H, "Oakley Key Determination Protocol," Request for comments (RFC 2412), November 1998.

[43] Stallings, w., Network Security Essentials: Applications and StandRDS, 2nd Ed. Upper Saddle River, NJ: Prentice Hall 2003.

[44] Erfani, S., Security Management System and Method, US Patent 6,542,993 B1, April 1, 2003.

[45] Krawczyk, H," SKEME: A versatile Secure Key Exchange Mechanism for Internet", IEEE Proceedings of Symposium on Network and Distributed Systems Security, 1996.

[46] Keromytis, A.D., and Ioannidis, J., and Smith, J.M., "Implementing IPSec," *Proc. IEEE Globe com 97*, pp.1948-1952.

[47] Lu, J., and Lockwook, J., "IPSec Implementation on Xilinx Virtex – II Pro FPGA and its Allications," *Proc. 19th IEEE International Panallel and Distributed Processing Symposium (IPDPS'OS)* April 4-8, 2005, pp.158b-165b.

[48] Dandalis, A. and Prasanna, V.K. and Rolim, J.D.P., "An Adaptive Cryptographic Engine for IPSec Architectures," *Proc. 2000 IEEE Syposuim on Field-Programmable Custom Computing Machines*, April 17-19, 2000, pp. 132-141.

[49] Hiltune, M.A and Schlichting, R.D. and Ugarte, C.A, "Building Survivable Services Using Redundancy and Adaptation," *IEEE Trans. On Computers*, Vol. 52, No. 2, February 2003, pp. 181-194.

Appendices

Appendix A1

- **IPv4 Datagrams Format**

An IP datagram consists of a header portion and a data portion. The header portion consists of a 20 bytes fixed part and a variable length optional part. The data portion is of variable length. Figure A.1 shows the format of an IPv4 datagram.

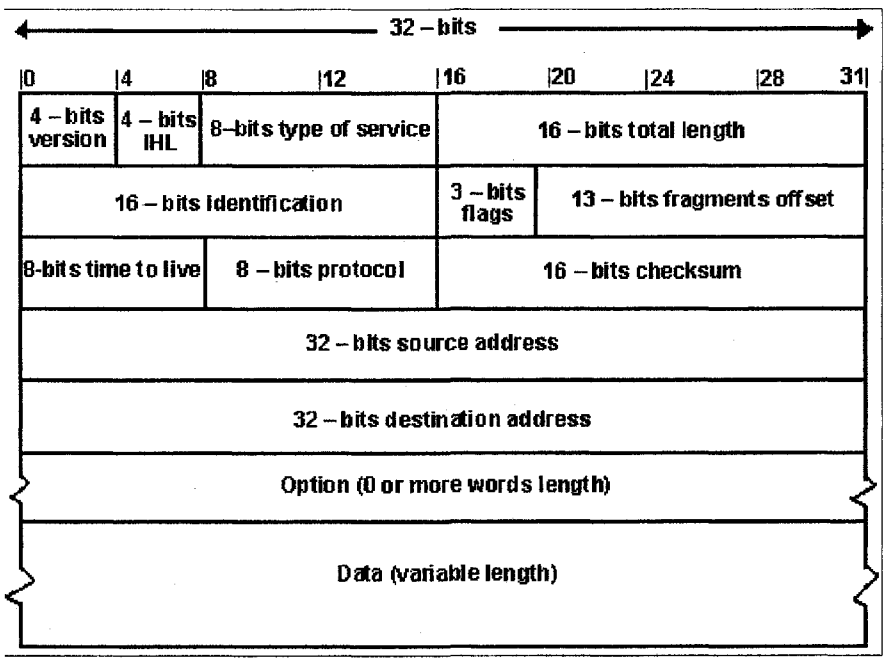


Figure A.1: IPv4 datagram.

An IP datagram is transmitted in big endian byte order: from left to right, that is, lower order bytes are transmitted first. This is the byte ordering required for all binary integers in the TCP/IP headers as they traverse a network. This is called

network byte order. Machines such as Pentiums, which uses little endian byte ordering format must convert header values to network byte order before transmitting the data.

- **The version field** indicates the version of the IP protocol that the data belongs. This field is usually use for backward compatibility.

- **The IHL field** is the length of the header in 32-bits words. The minimum value is 5 words(20 bytes) which is the case when no options are present: and the maximum value is 15 words (60 bytes) which applies when the options field is 40 bytes.

- **The type of service (TOS) field** indicates the traffic requirement of the datagram in bytes. The maximum length is 65,535 bytes. This field is used in combination with the IHL field to indicate where the data portion of the IP datagram begins.

- **The identification field** allows a host to determine which datagram that a newly arrived fragment belongs to. Each datagram has a unique identification number, and each fragment of a datagram has the same identification number.

- **Only 2 of the 3 bits flags** are used. The first of the two bits is called the DF (don't fragment) bit: it indicates whether or not the IP datagram should be fragmented. If this bit is set, it indicates that the datagram should not be fragmented. When a router receives a datagram with DF bit set, it usually sends back an ICMP message to the host from which the datagram originated indicating its MTU (Maximum Transfer Unit). This bit is therefore used to determine path

- **MTU (path Maximum Transfer Unit)**: the maximum size of an IP datagram

that will be allowed to pass through a given network path on route to the destination without been subjected to fragmentation.

- **The next bit is called MF (more fragment) bit**. This bit indicates when the last fragment of a datagram arrives. This bit is set for all of the fragments of a datagram except the last fragment.

The fragment offset tells where in the current datagram this fragment belongs. All fragments except the last one in a datagram must be a multiple of 8 bytes.

- **The time to live (TTL)** field is used to limit the life time of a datagram, thus preventing datagram from looping infinitely within a network segment. The TTL field is set to a default value by the host. Each router that the datagram passes through, decrements the TTL value by one. If a router receives a datagram with a TTl value of 1, it discards the datagram and sends an ICMP message to the source indicating that the TTL value of the datagram has expired.

- **The protocol field** tells which transport protocol is used for the data encapsulated within the IP datagram. It allows destination hosts to demultiplex IP datagrams among the different transport protocols.

- **The header checksum** is computed on the IP header and is used to determine the integrity of the IP header. It should be noted that the checksum is not encrypted and it can easily be forged.

- **The source and destination address fields** indicate the 4 bytes IP address of the host that generated the datagram, and the destination host respectively.

- **The variable length options field** carries optional information about a datagram such as the security and handling restriction. This field is rarely used and is usually ignored by most routers.

- **The data portion of the IP datagram** is of variable length and it contains the IP payload. For further detail regarding the format of the data or header portion of an IPv4 datagram refer to [30] which contains the official specification of IPv4 protocol.

o IPv6 Datagram Format

IPv4 suffers from a major limitation :it limits IP address to 32 bits. With the current rate of growth of the Internet, there is a real possibility that if the size of IP addresses does not increase, then there might not be enough IP addresses to meet the demand. The designated successor of IPv4: IPv6 overcomes this limitation as well as simplifies the IP header and adds more flexibility to the IP datagram. Some of the changes from IPv4 to IPv6 are outlined below.

- IPv6 increases the IP address size from 32 bits to 128 bits, thus increasing the addressable nodes by many folds.
- The number of header fields is reduced from 13 in IPv4 to 7 in IPv6. The smaller number of headers allows routers to process packets faster and therefore, increases throughput.
- IPv6 provides better support for options. The options are treated as separate headers instead of being a part to the IP header. This change allows routers to skip over headers that are not intended for them. This feature speeds up packet processing time; it also allows for less stringent limits on the length of options and provides greater flexibility for the introduction of new options in the future.
- IPv6 provides new capability to label packets belonging to particular traffic stream for which the sender requests special handling, such as non default quality of service.
- IPv6 does not support any fragmentation for packets in transit. The host that generates the packet must perform path MTU to ascertain the maximum size of the IP

datagram that will be allowed to pass through the network segment on route to the destination host.

- IPv6 specified extension to support authentication, data integrity and(optional) data confidentiality.
- Figure A.1.2 shows the required IPv6 fixed headers.

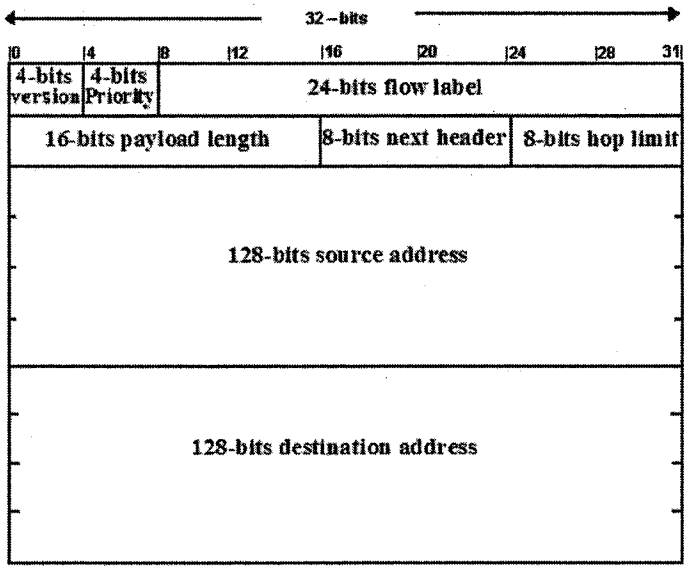


Figure A.1.2: IPv6 Header Format.

- **The version field** is similar to IPv4 version field. The priority field is used to indicate the quality of service that a packet requires. The flow label field is still experimental, but is likely that it will be used in the future to set up a pseudo-connection with particular properties and requirements [31].

- **The payload length** is a 16 bit unsigned integer which indicates the length of the IP payload, that is, the rest of the packet following the IPv6 headers.

- **The next header field** identifies the type of header immediately following the IPv6 header. This field facilitates the reduction in the number of fields in the IPv6 header compared to that of IPv4. It tells which extension header, if any, follows this one; if this header is the last IP header, it tells which transport protocol the packet should be passed to. For information on the number assigned to each protocol see [33]

- **The hop limit field** is the same as the TTL field in IPv4.

The source and destination address fields represent the 128-bits IPv6 addresses of the source and intended recipient of the packet, respectively.

- **Extension Headers**

IPv6 introduced the concept of optional extension headers. These headers can be supplied to provide additional information. There are currently 6 extension headers defined by IPv6. and each has a unique identification number (described in [33]).

The extension headers, if present, are inserted between the IPv6 header and the transport header. If more than one extension header is present, the order of the headers is important and should be as detailed in [32] (the original specification of IPv6).

Appendix A2

Fields for the SA:

- **Sequence Number**

  " This is a 32-bit number that is incremented by 1 every time SA is used to secure the communication" [5]. It is used by both parties for outbound processing and it is contained in the AH and ESP headers.

- **Sequence Number Overflow**

  " This field is set when the sequence number overflows" [5]. It is used by both parties to process outbound traffic. Once the overflow is occurred, IPSec policy decides whether the current SA can still be used for additional communication or a new SA should be established.

- **Anti-replay Window**

  This field is used in inbound processing to control an important issue in networking which is replay attacks. IPSec defeats these attacks by detecting packets replayed by malicious hosts.

- **Lifetime**

  Each SA has a lifetime associated with it that indicates when it can not be used any more. This lifetime is specified either in terms of number of bytes that have been secured using this SA, or the duration for which the SA has been used, or both.

- **Mode**

This field identifies the mode of IPSec protocols which can be either a tunnel mode or transport mode. Table 2.1 shows an SA with example values.

Security Association Parameters	Example Values
SPI	3219011
IP Destination Address	172.16.16.10
AH Mode	Transport
AH Algorithm	SHA-1
AH Key	196 bit SHA-1 Key length
ESP Mode	Transport
ESP Algorithm	3DES
ESP key	168 bit 3DES key length
ESP Authentication Algorithm	MD5
ESP Authentication Algorithm Key	96 bit MD5 key length
Life time	3600 sec

Table A2.1: Example of SA

Appendix A3

The objects stored in Security Database:

1) Security Policy Database

The IPSec protocol mandates that the Security Policy Database (SPDB) must be consulted during the processing of all traffic, whether the traffic is inbound or outbound. The SPDB contains an order list of policy entries. Each entry is specified by the use of one or more selectors. The selectors the IPSec currently allows are:

- **Destination IP address:** The destination IP address can be a 32-bit IPv4 or a 128-bit IPv6 address. The address can be a host IP address, a broadcast, unicast, anycast, a multicast group, a range of addresses, address plus netmask or wild card address. The destination IP address is obtained from the destination IP address field of the Authentication Header (AH) or the Encapsulating Security Payload (ESP) header (s) or if IPSec is not applied to the packet the IP header.
- **Source IP address:** The source IP address can be a 32-bit IPv4 or a 128-bit IPv6 address. The address can be a host IP address, a broadcast, unicast, anycast, a multicast group, a range of addresses, address plus netmask or wild card address. The destination IP address is obtained from the source IP address field of the AH or the ESP header(s) or if IPSec is not applied to the packet the IP header.

- **Transport layer protocol:** Ther transport layer protocol is obtained from the IPv4"protocol" or the IPv6"next header" fields.
- System name: The system name can be a fully qualified DNS name such as eep.mega.uwindsor.ca, a X.500 distinguished name or a X.500 general name.
- **Use ID:** The user ID can be a fully qualified DNS user name such as fah1@ece.uwindsor.ca or a X.500 distinguished name.

2) Security Association Database

The Security Association Database (SADB) contains the active Security Association (SA) entries. Each SA entry is indexed by a triplet consisting of a Security Parameter Index (SPI), a source or destination IP address and a IPSec protocol. In addition, a SADB entry consists of the following fields:

- **Sequence Number Counter:** This is a 32-bit integer which is used to generate the sequence number field in AH or ESP headers.
- **Sequence Counter Overflow:** This is a flag indicating whether the overflow of the Sequence Number Counter should be audited and the transmission of additional traffic be blocked for the given SA.
- **Anti- Replay Window:** A 32-bit counter and a bit-map that is used to ascertain whether an inbound AH or ESP packet is replay.
- **AH authentication** algorithms and required keys.
- **ESP Authentication** algorithms and required keys.
- **ESP encryption** algorithms, keys, Initial Vector(IV) and IV mode.

- **IPSec protocol mode:** This field indicates which IPSec protocol mode (transport, tunnel or wild card) is applied to AH and ESP traffic.
- **Path Maximum Transfer Unit (PMTU):** Any observed PMTU and aging variables.
- **Lifetime of the SA:** This field contains the time interval within which a SA must be replaced by a new SA or be terminated, plus an indication of whether the SA should be replaced or terminated when it expires. The lifetime of a SA takes two forms: a time interval or byte count which represents the number of bytes that IPSec protocols has been applied to the parameter that expires first takes precedence.

Appendix A4

- **ISAKMP Header Format**

ISAKMP message consist of a fixed length header followed by a variable number of payloads. The fixed-length header contains the necessary information for the protocol to maintain state and process the payloads. The ISAKMP header format is illustrated in Figure 3.16.

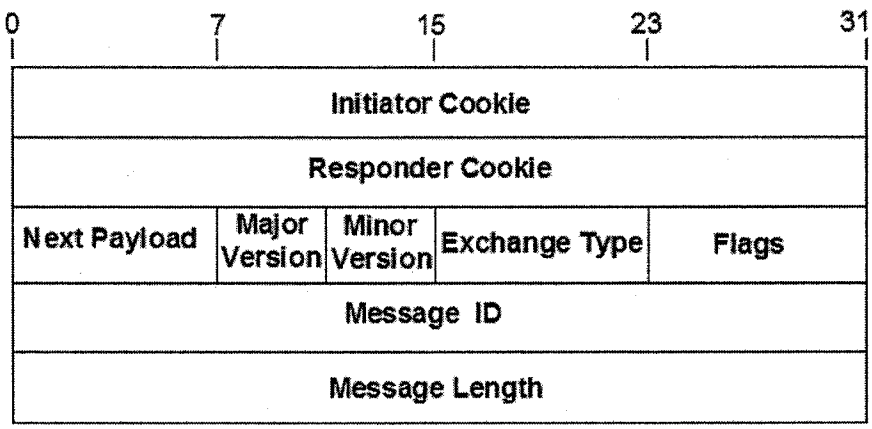


Figure A.4.1: The ISKAMP Header Format

A description of the ISAKMP header fields follows.

- **Initiator Cookie:** This field contains a unique 8-bit integer that the initiator of the ISAKMP exchanges generates. This cookie is created by each peer and are used, along with the message ID, to identify the state that defines an ISAKMP exchange in progress.

  The method of generating the cookies varies with different implementations of ISAKMP; however, the protocol specifies that the cookies, whether that of the initiator or responder, should be generated

using secret information that is unique to the respective ISAKMP communicating hosts, and it should not be possible to determine the secret information from the cookie. In addition the cookie for each SA should be unique. A possible method of generating cookies is to perform a hash (using MD5 or SHA-1 hash function) of the concatenation of the source and destination address, UDP source and destination ports, a locally generated secret random number and the current date and time.

- **Responder Cookie:** This field contains the responder's 8-bit cookie. The attributes of this cookie are similar to that of the initiator's.
- **Next Payload:** This is a 8-bit field that indicates the first payload in the message; it also indicates which of the various ISAKMP payloads immediately follows the header. Table A.4.1 shows the payload that ISAKMP currently defines.

Next Payload Type	Assigned Value
NONE	0
Security Association	1
Proposal	2
Transform	3
Key Exchange	4
Identification	5
Certificate	6
Certificate Request	7
Hash	8
Signature	9
Nonce	10
Notification	11
Delete	12
Vendor	13
RESERVED	14-127
Private USE	128-255

Table A.4.1: Assigned Values for ISAKMP Payloads.

- **Major Version:** This 4-bit field indicates the major version of the ISAKMP Protocol in use. The specification for ISAKMP specifies that an ISAKMP implementation should not accept packets with a Major or Minor Version that is larger than its own.
- **Minor Version:** This 4-bit field contains the protocol Minor Version number.
- **Exchange Type:** This 8-bit field indicates the type of exchange that the message comprises of. Table 3.2 shows the exchange types that ISAKMP currently defines.

Exchange Type	Assigned Value
NONE	0
Base	1
Identity Protection	2
Authentication Only	3
Aggressive	4
Informational	5
ISAKMP Future Use	6 - 31
DOI Specific Use	32-239
Private Use	240-255

Table A.4.2: ISAKMP Exchange Type and Assigned Values

- **Flags:** This 8-bit field indicates specific options that are set for ISAKMP exchanges. The first 3 bits of this field are currently used, the others are set to zero before transmission.
    - E(Encryption Bit): which signifies that the payloads following the header are encrypted.
    - C(Commit Bit): which signifies that a peer wishes a notification of exchange completion.

- A(Authentication-only Bit): used primarily by those who wish to add key recovery to ISAKMP.

- **Message ID:** This is a 4-byte field which contains a random value generated by the initiator of phase 2 (which will be discussed later in this chapter) negotiation. It serves as a unique Message Identifier that is used to identify the protocol state during phase 2 negotiations.
- **Message Length:** This is a 4-byte field which indicates the length of the total message (header + payloads) in bytes.

o **ISAKMP Payloads Formats**

Generic Payload Header

Each ISAKMP payload begins with a generic header. The Generic Payload Header clearly defines the boundaries of the payload and consequently allows the chaining of different payloads. Figure A.4.2 illustrates the Generic Payloads Header.

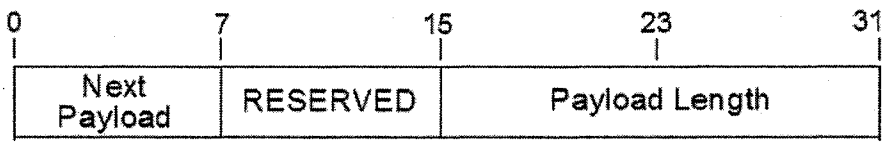


Figure A.4.2: Generic Payload Header Format

A description of the Generic Payload Header fields follows.

- **Next Payload:** This 8-bit field identifies the payload type that follows this payload. If the current payload is the last payload of the message, the Next Payload field will be set to 0.

- **RESERVED** : This field is unused and it is set to 0.
- **Payload Length:** Contains the length in bytes of the current payload including the Generic Payload Header. This is a 2-byte field.

We will now discuss the ISAKMP payloads that IKE uses.

2) Security Association Payload

The Security Association Payload is used to negotiate SA and to indicate the Domain of Interpretation (DOI) under which the negotiation takes place. Figure A.4.3 illustrates the format of this payload.

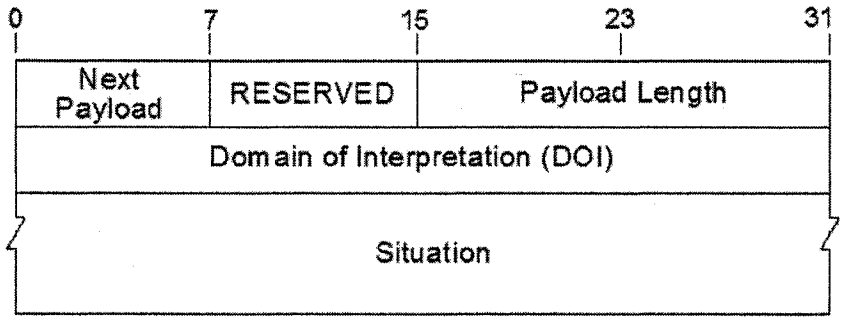


Figure A.4.3: Security Association Payload Format.

The "Next Payload", " RESERVED" and "Payload Length" fields are similar to those of Generic Payload Header fields.

- **Domain of Interpretation (DOI):** This 4-byte field contains a 4-byte unsigned integer which identifies the DOI under which the negotiation is taking place. A DOI defines payload formats, exchange types and convention for naming relevant information such as cryptographic algorithms, modes, etc.
- **Situation:** This variable length field identifies the Situation under which the negotiation is taking place. A Situation is the set of information that will be used to determine the required security service.

3) Proposal Payload

The Proposal Payload contains information used during the SA negotiation. This payload provides the framework for the initiator of the ISAKMP exchange to present to the recipient the preferred security protocol(s) and associated security mechanisms desired for the SA being negotiated. The Proposal Payload is illustrated in Figure A.4.4.

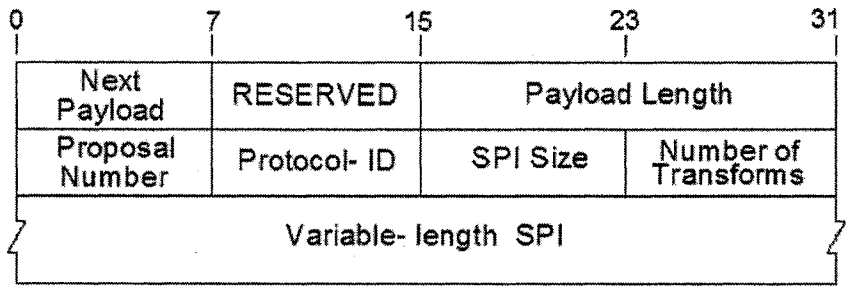


Figure A.4.4: Proposal Payload Format

A description of the Proposal Payload fields follows. The "Next Payload" RESERVED" and "Payload Length" fields are similar to those described in the payloads that we discussed previously.

- **Proposal Number:** This 8-bit field contains the identification number of the proposal for the current payload. The Proposal Number allows the peers that are involved in the SA establishment negotiation to present preferences to its communication peers in the form of logical AND or OR operations. For example, if the initiator of the exchange wishes to inform the other peer that it desires a combined protection suite consisting of ESP encryption with Triple-DES and authentication with HMAC-SHA-1, and AH authentication with HMAC-MD5, it would send a message that contains three Proposal Payloads, each with the same Proposal Number. If however, the peer needs to transmit the information that it

requires either ESP authentication with HMAC-MD5 or AH authentication with HMAC-SHA-1, it would send a message consisting of two Proposal Payloads, each with monotonically increasing Proposal Number. The Proposal Number here indicates the order of preference of the proposals: The greater the preference, the smaller the proposal number.

- **Protocol- ID:** This 8-bit field contains the protocol identifier (example 50 for ESP and 51 for AH).
- **SPI Size:** Contains the length in bytes of the Security Parameter Index (SPI) of the protocol specified by the Protocol-ID field. If the protocol is ISAKMP, the SPI is the initiator-responder cookie pair. This field is a 8-bit field.
- **Number of Transform:** This 8-bit field gives the number of Transforms for the proposal. Each of the transforms specified is embodied in a Transform Payload.
- **SPI:** This variable-length field contains the source node SPI.

4) Transform Payload

The Transform Payload provides the framework for the initiating peer to present different security mechanisms for a given protocol during the negotiation to establish a SA. The proposal payload identifies a protocol for which services are being negotiated. The Transform Payload allows the initiating peer to present the supported transform or mode of operation of the proposed protocol. Figure A.4.5 illustrates the format of the Transform Payload.

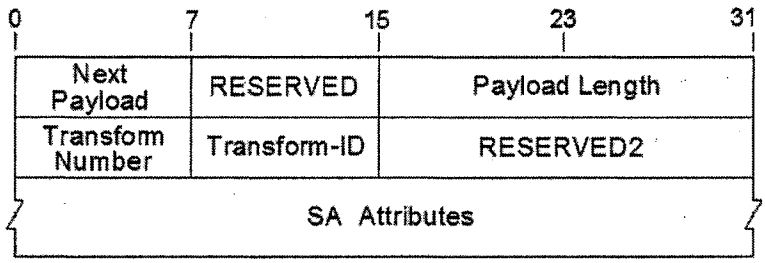


Figure A.4.5: Transform Payload Format

A description of the Transform Payload fields follows.

- **Next Payload:** The 8-bit Next Payload field identifies the payload type that follows the current payload. This field will either contains the value 3 or 0. If another Transform Payload follows the current Transform Payload, the value in this field will be 3; if however the current Transform Payload is the last one for the proposed protocol, the value in this field will be 0.
- **Reserved:** Similar to the "RESERVED" field in the payloads previously discussed.
- **Payload Length:** Similar to the "Payload Length" field in the payloads previously discussed.
- **Transform number:** This 8-bit field identifies the Transform Number for the current payload. If there are more than one transform for the proposed protocol, each transform will be embodied in a Transform Payload and each transform is identified by a monotonically increasing number. The transforms are ordered according to the initiating peer's preference: the most desired transform has the lowest Transform Number.

- **Transform-ID:** This 8-bit field specifies the transform identifier for the proposed protocol.
- **Reserved2:** This 16-bit field is unused and is set to 0.
- **SA Attributes:** This variable-length field contains the Security Association attributes for the Transform specified by the Transform-ID field.

5) Key Exchange Payload

The key Exchange Payload supports various key exchange protocols.

Figure A.4.6 illustrates the key Payload.

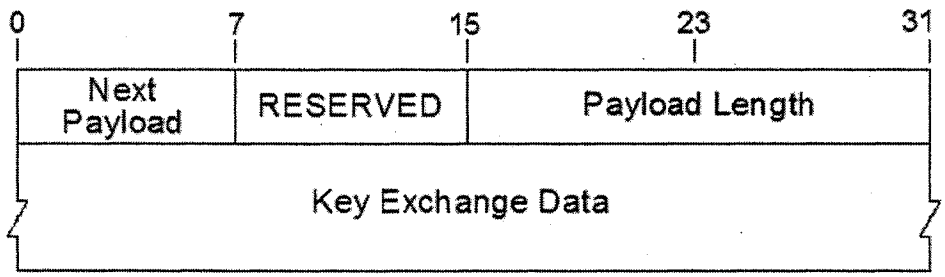


Figure A.4.6: Key Exchange Payload Format

The "Next payload", " RESERVED" and "Payload Length" fields are similar to those of the payload previously discussed. The key Exchange Data field is of variable length and it contains the data required to generate a session key. The DOI for the respective key exchange specifies the format and the interpretation of the data in this field.

6) Identification Payload

The Identification Payload allows communicating peers to exchange identity information. Figure A.4.7 illustrates the format of this payload.

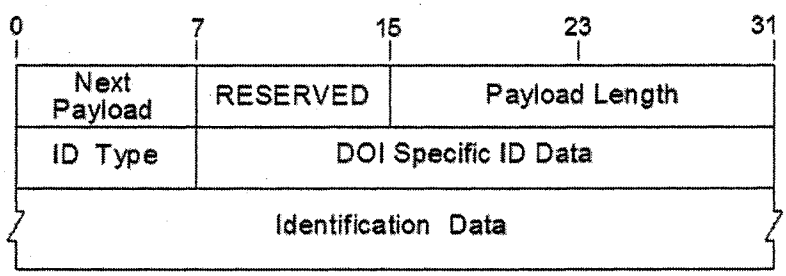


Figure A.4.7: Identification Payload Format.

The "Next Payload", "RESERVED" and "Payload Length" fields are similar to those of the payload previously discussed.

- **ID Type:** This 8-bit field contains the type of identification being used, and it depends on the DOI of the respective key exchange.
- **DOI Specific ID Data:** This 24-bit field contains DOI specific identification data.
- **Identification Data:** This variable-length field contains identification information specified by the DOI of the key exchange. The ID Type field specifies the format of the information in this field.

7) **Certificate Payload**

The Certificate Payload allows communicating peers to exchange certificate or certificate-related material. Figure A.4.8 illustrates the format of this payload.

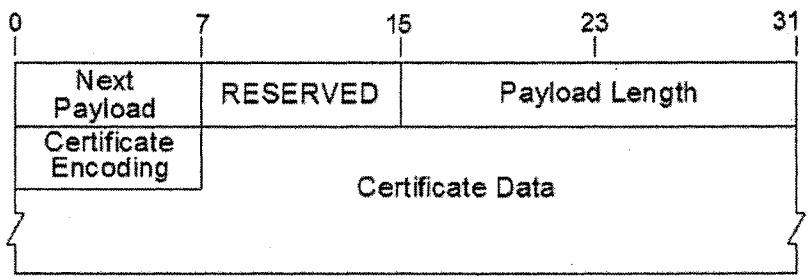


Figure A.4.8: Certificate Payload Format

The " Next Payload", " RESERVED" and " Payload Length" fields are again, similar to those of the payload previously discussed.

- **Certificate Encoding:** This 8-bit field identifies the kind of certificate or certificate-related information the " Certificate Data" field contains.
- **Certificate Data:** This variable-length field contains the actual certificate data for the certificate type specified in the "Certificate Encoding" field.

8) Hash Payload

The Hash Payload contains the data generated by the hash function selected during the SA negotiation. Figure A.4.9 illustrates the format of this payload.

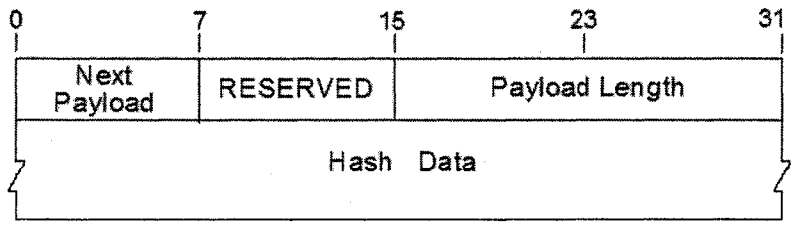


Figure A.4.9: Hash Payload Format

The "Hash Data" field contains the message digest that resulted from the application of the hash function to the input data.

9) Signature Payload

The Signature Payload contains the data generated by the signature function negotiated for the SA: and it is used to verify the integrity of the data in an ISAKMP message. The format of this payload is illustrated in figure A.4.10

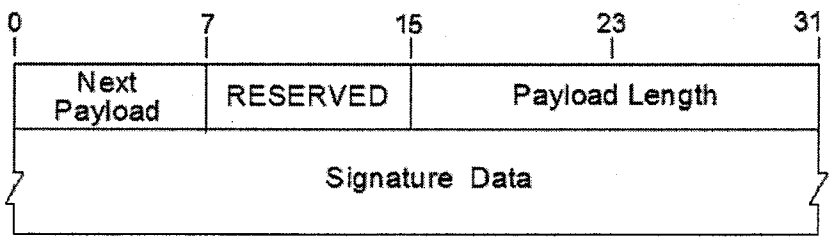


Figure A.4.10: Signature Payload Format

The "Signature Data "field is of variable length and it contains the signature data that resulted from the signing of the ISAKMP message using the digital signature algorithm negotiated for the SA.

10) Nonce Payload

The Nonce payload contains some pseudo - random information necessary used to protect the exchange data against replay. Figure A.4.11 illustrates this payload.

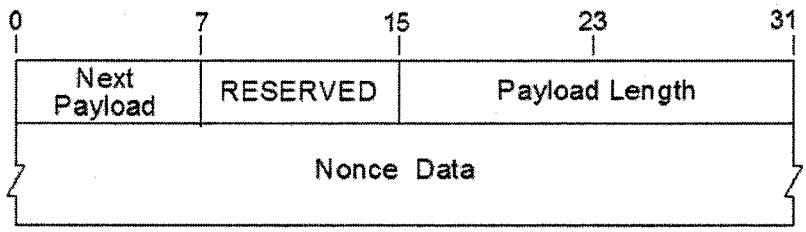


Figure A.4.11: Nonce Payload Format

The variable- length Nonce Data" field contains the random data generated by the communicating peer that is sending the message.

Appendix B

Preshared Keys Overview and Types

The same preshared key is configured on each IPSec peer. The preshared key is combined with the Diffie- Hellman key, initiator, and responder's nonce and cookie (this is an 8-byte pseudorandom number unique to each peer and the particular exchange in which it is defined) to form the authentication key. The authentication key is combined with device-specific information and sent through a hash algorithm to obtain HASH_I (initiator's hash value) and HASH_R (responder's hash value). IKE peers authenticate each other by exchanging and decoding HASH_I and HASH_R. If the receiving peer is able to independently recreate the same hash, the peer is authenticated. Each local peer must authenticate its remote peer before the tunnel is considered secure.

The preshared key is used during IKE (ISAKMP) phase 1 (main mode) to authenticate the Diffie-Hellman (D-H) shared secret exchange. There are three types of preshared keys: unique, group, and wildcard. Unique preshared keys are fixed to a specific IP address. They are defined for paris of peers when the IP addresses of the peers are known initially. Group preshared keys are tied to a group name identity; these are applicable only to remote access. Wildcard preshared keys are not coupled with any unique information to determine a peer's identity. They are valid with any other device, as long as that device is also defined to use the same key value. Any devices that have the key will authenticated successfully. When using wildcard preshared keys, every device in the network uses the same key. If a single device in your network is compromised and the wildcard preshared key has been gathered, then all devices are compromised. Readers are

strongly advised to avoid deploying wildcard preshared keys for site-to-site device authentication.

Preshared keys are simple to implement if your secured network is small (fewer than ten nodes). However, they do not scale well with a large or growing network. Depending on how strong the preshared keys are and how often they are replaced, they may not provide strong device authentication. In fact, preshared key is the least secure of the three different authentication methods.

VITA AUCTORIS

NAME:	Mehdi Fahandezh
PLACE OF BIRTH:	Shiraz, Iran
YEAR OF BIRTH:	1953
EDUCATION:	Razy High School, Shiraz, Iran 1966-1972
	Tehran University Tehran Iran 1972-1974 School of Business
	Iowa State University Ames Iowa USA 1974-1979 B.Sc.
	University of Windsor Windsor, Ontario 2002-2005 M.Sc.