ignou
THE PEOPLE'S
UNIVERSITY

Indira Gandhi National Open University
School of Vocational Education and Training

MSEI-022
Network Security

Secure Protocols

2

''शिक्षा मानव को बन्धनों से मुक्त करती है और आज के युग में तो यह लोकतंत्र की भावना का आधार भी है। जन्म तथा अन्य कारणों से उत्पन्न जाति एवं वर्गगत विषमताओं को दूर करते हुए मनुष्य को इन सबसे ऊपर उठाती है।''

— *इन्दिरा गांधी*

"Education is a liberating force, and in our age it is also a democratising force, cutting across the barriers of caste and class, smoothing out inequalities imposed by birth and other circumstances."

– Indira Gandhi

ignou
THE PEOPLE'S
UNIVERSITY

MSEI-022
Network Security

Indira Gandhi National Open University
School of Vocational Education and Training

Block

2

SECURE PROTOCOLS

Programme Expert/ Design Committee of Post Graduate Diploma in Information Security (PGDIS)

Prof. K.R. Srivathsan,
Pro Vice-Chancellor, IGNOU

Mr. B.J. Srinath, Sr. Director & Scientist 'G',CERT-In, Department of Information Technology, Ministry of Communication and Information Technology, Govt of India

Mr. A.S.A Krishnan, Director, Department of Information Technology, Cyber-Laws and E-Security Group, Ministry of Communication and Information Technology, Govt of India

Mr. S. Balasubramony, Dy. Superintendent of Police, CBI, Cyber Crime Investigation Cell Delhi

Mr. B.V.C. Rao, Technical Director, National Informatics Centre, Ministry of Communication and Information Technology

Prof. M.N. Doja, Professor, Department of Computer Engineering, Jamia Milia Islamia New Delhi

Dr. D.K. Lobiyal, Associate Professor, School of Computer and Systems Sciences, JNU New Delhi

Mr. Omveer Singh, Scientist, CERT-In Department of Information Technology Cyber-Laws and E-Security Group Ministry of Communication and Information Technology Govt of India

Dr. Vivek Mudgil, Director, Eninov Systems Noida

Mr. V.V.Subrahmanyam, Assistant Professor School of Computer and Information Science IGNOU

Mr. Anup Girdhar, CEO, Sedulity Solutions & Technologies, New Delhi

Prof. A.K. Saini, Professor, University School of Management Studies, Guru Gobind Singh Indraprastha University, Delhi

Mr. C.S. Rao, Technical Director in Cyber Security Division, National Informatics Centre, Ministry of Communication and Information Technology

Prof. C.G. Naidu, Director, School of Vocational Education & Training, IGNOU

Prof. Manohar Lal, Director, School of Computer and Information Science, IGNOU

Prof. K. Subramanian, Director, ACIIL, IGNOU Former Deputy Director General National Informatics Centre, Ministry of Communication and Information Technology Govt of India

Prof. K. Elumalai, Director, School of Law IGNOU

Dr. A. Murali M Rao, Joint Director, Computer Division, IGNOU

Mr. P.V. Suresh, Sr. Assistant Professor School of Computer and Information Science IGNOU

Ms. Mansi Sharma, Assistant Professor, School of Law, IGNOU

Ms. Urshla Kant
Assistant Professor, School of Vocational Education & Training, IGNOU
Programme Coordinator

Block Preparation

Unit Writers

Prof. Gopi Krishna S Garge
Department of Electrical Communication Engineering, Indian Institute of Science (IISc), Bangalore

Dr. Malati Hegde, Department of Electrical Communication Engineering Indian Institute of Science (IISc) Bangalore (Unit 1, 2 &3)

Block Editor

Mr. Anup Girdhar, CEO
Sedulity Solutions & Technologies, New Delhi

Ms. Urshla Kant
Assistant Professor, School of Vocational Education & Training, IGNOU

Proof Reading

Ms. Urshla Kant
Assistant Professor
School of Vocational Education & Training
IGNOU

PRODUCTION

Mr. B. Natrajan
Dy. Registrar (Pub.)
MPDD, IGNOU

Mr. Jitender Sethi
Asstt. Registrar (Pub.)
MPDD, IGNOU

Mr. Hemant Parida
Proof Reader
MPDD, IGNOU

August 2011

ISBN: 978-81-266-5613-4

Further information on the Indira Gandhi National Open University courses may be obtained from the University's office at Maidan Garhi, New Delhi-110 068 or the website of IGNOU www.ignou.ac.in

Printed and Published on behalf of the Indira Gandhi National Open University, New Delhi, by the Registrar, MPDD.
Printed at: Berry Art Press A-9, Mayapuri Phase-I New Delhi-64

BLOCK INTRODUCTION

Secure protocols aim to establish one or more security goals, often a combination of integrity, authentication or confidentiality. Secure protocols are critical applications, since they are crucial to achieve trusted computing. So they are thoroughly studied to guarantee that does not exist an interleaving of protocols runs violating a security goal, called an attack. Designing correct secure protocols, however, has proven to be problematic, since it is difficult to anticipate what an adversary can learn from observing the simultaneous execution of an arbitrary number of protocols, given some initial knowledge. These are excellent candidates for rigorous analysis techniques. They are critical components of any distributed security architecture. They are very easy to express, and they are very difficult to evaluate by hand. This block comprises of three units and is designed in the following way;

The **Unit One** introduces the elements of security protocols. We have looked at the elements of security that are necessary to be provided with secure protocols. Security is built into protocols at all the layers. Application protocols have the security elements built into them and so do the other protocols at the link, network and transport layer.

The **Unit two** covers the detailed descriptions of the specific protocol at network layer. In this unit, we have looked at the various components that make up the secure sockets layer as well as how the components of the IPSEC protocol interwork together to provide the service. SSL provides the security enablement for the upper layer protocols such as http, smtp, imap, pop etc. IPSEC, in contrast, can provide two types of accesses to end users – the tunnel mode and the transport mode. Users can exchange data securely between two hosts using the transport mode or between sites in a completely encrypted fashion using the tunnel mode. Sharing of keys is an important concern in both the protocol implementations. Very often, shared secret keys are implemented and the keys are exchanged, manually.

The **Unit three** explains specific protocol at application layer. Pretty Good Privacy (PGP) provides security to messages that are encrypted and signed and either stored on a disk (as a file) or transmitted across the network. Similarly, two application layer protocols POP3s and https that use the lower level implementation of a secure transport protocol, SSL. SSL makes sure that there is an end-to-end secure channel available for transmission of data between the two end points. Protocols at the higher layers use protocols like SSL to transfer data securely.

Hope you benefit from this block.

ACKNOWLEDGEMENT

The material we have used is purely for educational purposes. Every effort has been made to trace the copyright holders of material reproduced in this book. Should any infringement have occurred, the publishers and editors apologize and will be pleased to make the necessary corrections in future editions of this book.

UNIT 1 INTRODUCTION TO SECURE PROTOCOLS

Structure

1.0 INTRODUCTION

In this unit, we will take a brief look at what secure protocols are. How normal protocols are made secure? What is the security element that is addressed and how they are addressed? Finally, we will list some of the commonly used secure protocols. These protocols will be discussed in some detail in the following units in this block.

1.1 OBJECTIVES

After going through this Unit, you should be able to:

- describe need to secure protocols;
- explain security protocols; and
- describe application protocols and protocols at the link, network and transport layer.

1.2 WHAT DO WE NEED TO SECURE?

Over the last 25 years, a number of networks have been set up and used. On these networks, several protocols have evolved for providing application services. The application level protocols have evolved over time and there are several of them, today. Typically there is one application protocol for each service. It is the data transferred between the applications that need to be secured.

What does security mean, then? In our context, security implies security provided for the transfer of data between the two communicating end points. There are three elements to this security, namely, Authentication, Integrity and Privacy. Each one of these elements is important. Security protocols should address all these aspects. We will look at each one of these characteristics in this context. In short, we require to know who we are communicating with, we need to ensure that the data we send is not altered along the way and finally ensure that even if our data is intercepted, it is unintelligible to anyone else other than the intended recipient.

1.2.1 Authentication

Authentication enables an entity to be verified as that which the entity claims itself to be. In this case, the entity can be an end user requiring a service, a program requiring access to specific information, an originator/recipient host that is accessing a service and so on. Each of these communicating entities needs to be authenticated to validate their claimed identity. There could be several means of authentication, starting with simple id/password pairs, several levels of such validation to bio-metric validation.

When dealing with an authentication service, the login id and password pair has to be transmitted. It is important that this information is transmitted in a form that is understandable only to the two communicating entities. Authentication is carried out by verifying the transmitted password with a password that is stored by the authentication service.

1.2.2 Integrity

Integrity of the transmitted message is necessary so that the two communicating entities receive exactly what one has sent to the other. The transmitted message or data could even be the login id and password that is sent to the authentication service. On the other hand, it could even be the email message that you sent to the recipient. Integrity checks ensure that the data has not been altered or tampered with, enroute to the recipient.

Digital signatures are used for the purpose of checking the integrity of the received message. The original message and its digital signature are transmitted together. The digital signature is generated using a forward hash function. This signature is called the message digest (MD). Both the message and its digest are transmitted together. Should either of them be tampered with, the digest will not match with a similar digest that is generated at the receiving end using the same algorithm used at the originating end. If the digests do not match, both the message and the digest are discarded and a retransmission is requested.

1.2.3 Privacy

Privacy of communication implies two things – to ensure that the data transmitted on the link is readable only by the intended recipient/s and no one else. This is done by encrypting the transmitted message. Encryption requires keys that perform the encryption and the two end points require that they have the right keys to encrypt as well as decrypt. Exchange of keys between the two end points is a challenge.

Many privacy schemes involve either single key (shared) or dual key cryptography. In addition, there are schemes where the end points initially start a private session with a certain key and then use that private connection to negotiate a session key that is used for the session. All attempts are made to ensure that the data transferred is kept private.

1.3 SECURITY PROTOCOLS

Security protocols aim to establish one or more security goals, often a combination of integrity, authentication or confidentiality. Security protocols are critical applications, since they are crucial to achieve trusted computing. So they are thoroughly studied to guarantee that does not exist an interleaving of protocols runs violating a security goal, called an attack.

Designing correct security protocols, however, has proven to be problematic, since it is difficult to anticipate what an adversary can learn from observing the simultaneous execution of an arbitrary number of protocols, given some initial knowledge.

Security protocols are infact excellent candidates for rigorous analysis techniques. They are critical components of any distributed security architecture. They are very easy to express, and they are very difficult to evaluate by hand. They are deceptively simple looking: the literature is full of protocols that appear to be secure but have subsequently been found to fall prey to some subtle attack, sometime years later.

Protocol flaws can usually be understood as violation to well-known design guidelines. These kinds of principles capture prudent practises in security protocol design, while pinpointing features that make protocols difficult to verify or possibly susceptible to an attack, and should honest principal a camouflaged message whose implementation equals that of the message being spoofed.

The call for and desire for security and privacy has led to the advent of several proposals for security protocols and standards. Among these are Secure Socket

Layer (SSL) and Transport Layer Security (TLS) Protocols,; secure IP (IPSec); Secure HTTP (S-HTTP or HTTPS), secure E-mail (PGP and S/MIME), DNDSEC, SSH and others. Before we walk through all these protocols, we need a firm understanding of the network protocol stack.

- Application level security – PGP, S/MIME, HTTPS, SET and KERBEROS
- Transport level security - SSL and TLS
- Network level security – IPSec and VPN
- Link level security – PPP and RADUIS

1.3.1 Application Level Security

All the protocols in this section are application layer protocols, which means that they reside on both ends of the communication link. They are all communication protocols ranging from simple text to multimedia including graphics, video, audio, and so on. In the last ten years, there has been almost an explosion in the use of electronic communication, both mail and multimedia content, that has resulted in booming e-commerce and almost unmanageable personal e-mails, much of in private or intended to be private anyway, especially e-mails. Along with this explosion, there has been a growing demand for confidentiality and authenticity of private communications.

1.3.2 Security in Transport Layer

Unlike the last five protocols, this layer covers little deeper in the network infrastructure. Although several protocols are found in this layer, the two most used are Secure Socket Layer (SSL) and Transport Layer Security (TLS). Currently, however these two are no longer considered as two separate protocols but one under the name SSL/TLS, after the SSL standardization was passed over to IETF, by the Netscape consortium, and Internet Engineering Task Force (IETF) renamed it.

1.3.3 Security in Network Layer

IPSec is a collection of protocols and mechanisms that provide confidentiality, authentication, message integrity, and replay detection at the IP layer. Because cryptography forms the basis for these services, the protocols also include a key management scheme.

Conceptually, think of messages being sent between two hosts as following a path between the hosts. The path also passes through other intermediate hosts. IPSec mechanisms protect all messages sent along a path. If the IPSec mechanisms reside on an intermediate host (for example firewall/ gateway), that host is called a security gateway. This is what happens with a VPN access.

The secure access is established between a remote host and the VPN server/gateway.

1.3.4 Security in Link Layer

Although there are several protocols including those applied in the LAN technology, the PPP and RADIUS are the two protocol standards operational at Data Link Layer. PPP is a connection oriented protocol that works on point-to-point links. RADIUS is used for authentication of the remote connecting host. Although RADIUS is listed here, it is not specifically a link layer protocol. It is a generic authentication protocol that is used to authenticate user accesses, typically on a point-to-point link as mentioned above or to authenticate user accesses to devices or hosts.

Check Your Progress 1

Note: a) Space is given below for writing your answer.

b) Compare your answer with the one given at the end of the Unit.

1) What are the three elements to security?

....

....

....

....

....

2) Explain Security in network layer.

....

....

....

....

....

1.4 LET US SUM UP

This unit introduces the elements of security protocols. We have looked at the elements of security that are necessary to be provided with secure protocols. Security is built into protocols at all the layers. Application protocols have the security elements built into them and so do the other protocols at the link, network and transport layer.

1.5 CHECK YOUR PROGRESS: THE KEY

Check Your Progress 1

1) The three elements to security are:

- **Authentication**

  Authentication enables an entity to be verified as that which the entity claims itself to be. In this case, the entity can be an end user requiring a service, a program requiring access to specific information, an originator/recipient host that is accessing a service and so on. Each of these communicating entities needs to be authenticated to validate their claimed identity. There could be several means of authentication, starting with simple id/password pairs, several levels of such validation to bio-metric validation.

  When dealing with an authentication service, the login id and password pair has to be transmitted. It is important that this information is transmitted in a form that is understandable only to the two communicating entities. Authentication is carried out by verifying the transmitted password with a password that is stored by the authentication service.

- **Integrity**

  Integrity of the transmitted message is necessary so that the two communicating entities receive exactly what one has sent to the other. The transmitted message or data could even be the login id and password that is sent to the authentication service. On the other hand, it could even be the email message that you sent to the recipient. Integrity checks ensure that the data has not been altered or tampered with, enroute to the recipient.

  Digital signatures are used for the purpose of checking the integrity of the received message. The original message and its digital signature are transmitted together. The digital signature is generated using a forward hash function. This signature is called the message digest (MD). Both the message and its digest are transmitted together. Should either of them be tampered with, the digest will not match with a similar digest that is generated at the receiving end using the same algorithm used at the originating end. If the digests do not match, both the message and the digest are discarded and a retransmission is requested.

- **Privacy**

  Privacy of communication implies two things – to ensure that the data transmitted on the link is readable only by the intended recipient/s and no one else. This is done by encrypting the transmitted message. Encryption requires keys that perform the encryption and the two end points require that they have the right keys to encrypt as well as decrypt. Exchange of keys between the two end points is a challenge.

Many privacy schemes involve either single key (shared) or dual key cryptography. In addition, there are schemes where the end points initially start a private session with a certain key and then use that private connection to negotiate a session key that is used for the session. All attempts are made to ensure that the data transferred is kept private.

2) **Security in Network Layer**

IPSec is a collection of protocols and mechanisms that provide confidentiality, authentication, message integrity, and replay detection at the IP layer. Because cryptography forms the basis for these services, the protocols also include a key management scheme.

Conceptually, think of messages being sent between two hosts as following a path between the hosts. The path also passes through other intermediate hosts. IPSec mechanisms protect all messages sent along a path. If the IPSec mechanisms reside on an intermediate host (for example firewall/ gateway), that host is called a security gateway. This is what happens with a VPN access. The secure access is established between a remote host and the VPN server/gateway.

UNIT 2 SPECIFIC PROTOCOL-1

Structure

2.0 INTRODUCTION

This unit deals with secure protocols that exist in the layers below the application layer. Secure Sockets Layer (SSL) is a protocol that is included as a sublayer above the transport layer. This led to the evolution of the Transport Layer Security (TLS) standard and implementations of both SSL and TLS are prevalent in various product implementations. Following this, we look at another important utility that provides end-to-end secure application transport – the Virtual Private Network (VPN) utility. The VPN servers and clients are used to access enterprise networks from the Internet. Users use VPN clients to login to the enterprise networks and access data, using the end-to-end encrypted channel provided by the VPN client.

2.1 OBJECTIVES

After going through this Unit, you should be able to:

- know Secure Sockets Layer; and
- understand SSL record protocol, Alert Protocol, handshake protocol and IPsec.

2.2 SECURE SOCKETS LAYER (SSL)

In 1996 the IETF task force attempted to develop an Internet's standardised secure method to communicate over the web. They adopted the SSL 3.0 as the starting point and in 1999 released the document RFC 2246 that defined the new Transport Layer Security (TLS) protocol in its version 1.0. The basic goal placed by the task force to be achieved through TLS, was similar to that associated with SSL's standards, namely to provide security and data integrity features at the transport layer between two web applications.

The SSL protocol was originally developed by Netscape, to ensure security of data transported and routed through HTTP, LDAP or POP3 application layers. SSL is designed to make use of TCP as a communication layer to provide a reliable end-to-end secure and authenticated connection between two points over a network (for example between the service client and the server). SSL is widely used on the Internet by an increasing number of varied applications, especially for interactions that involve exchanging confidential information such as credit card numbers. SSL evolved into the Transport Layer Security (TLS) Version 1 standard. Today, almost each available HTTP server can support an SSL session, whilst IE or Netscape Navigator browsers are provided with SSL-enabled client software.

SSL is positioned as a protocol layer between the Transmission Control Protocol (TCP) layer and the application to form a secure connection between clients and servers so that they can communicate in a secure manner over a network by providing:

- Privacy, where data messages are encrypted so that only the two application endpoints understand the data.
- Integrity, where message digests detect if any data was altered in flight.
- Authentication, which verifies the identity of the remote node, application, or user by using digital certificates.

This type of secure connection ensures that all data exchanged between clients and servers is encrypted, and is therefore not readable by a third party on the Internet. SSL has gained popularity in the Internet industry primarily because of its use of public-key certificates as a means of authenticating principles.

To establish the connection, SSL requires, at a minimum, a server certificate. As part of the initial SSL handshake process, the server presents its certificate to the client to authenticate the server's identity. The authentication process uses public-key encryption and digital signatures to confirm that the server is, in fact, who the server claims to be (that is, the server's certificate is valid). The Public Key does not need to be secret and is placed into a Certificate Signing Request (CSR) - a data file also containing your details. You should then submit the CSR. During the SSL Certificate application process, the Certification Authority will validate your details and issue an SSL Certificate containing your details and allowing you to use SSL.

Once the server has been authenticated (that is, the client determines that the server's certificate is valid), the client and server use techniques of public-key encryption to exchange a symmetric key, which is then used to encrypt all the information exchanged for the remainder of the SSL session. Message digests are used to detect data tampering. A different key is created for each client and server connection. As a result, if unauthorized users intercept and decrypt a session key (which is unlikely), they cannot use it to monitor later SSL sessions.

In a browser the complexities of the SSL protocol remain invisible to your customers. Instead their browsers provide them with a key indicator to let them know they are currently protected by an SSL encrypted session - the lock icon in the lower right-hand corner, clicking on the lock icon displays your SSL Certificate and the details about it. All SSL Certificates are issued to either companies or legally accountable individuals.

Typically, an SSL Certificate will contain most required information like your domain name, your company name, your address, your city, your state and your country. It will also contain the expiration date of the Certificate and details of the Certification Authority responsible for the issuance of the Certificate.

When a browser connects to a secure site it will retrieve the site's SSL Certificate and check that it has not expired, it has been issued by a Certification Authority the browser trusts, and that it is being used by the website for which it has been issued. If it fails on any one of these checks the browser will display a warning to the end user letting them know that the site is not secured by SSL.

Security of data in transit over the Internet becomes increasingly necessary because of steadily growing data volume and importance. Nowadays, every user of a public network sends various types of data, from e-mail to credit card details daily, and he would therefore like them to be protected when in transit over a public network. To this end, a practical SSL protocol has been adopted for protection of data in transit that encompasses all network services that use TCP/IP to support typical application tasks of communication between servers and clients.

The main objectives of SSL are

- Authenticating the client and server to each other: the SSL protocol supports the use of standard key cryptographic techniques (public key encryption) to authenticate the communicating parties to each other. Though the most frequent application consists in authenticating the service client on the basis of a certificate, SSL may also use the same methods to authenticate the client.
- Ensuring data integrity: during a session, data cannot be either intentionally or unintentionally tampered with.
- Securing data privacy: data in transport between the client and the server must be protected from interception and be readable only by the intended recipient. This prerequisite is necessary for both the data associated with the protocol itself (securing traffic during negotiations) and the application data that is sent during the session itself.

SSL is in fact not a single protocol but rather a set of protocols that can additionally be further divided in two layers:

- The protocol to ensure data security and integrity: this layer is composed of the SSL Record Protocol,
- The protocols that are designed to establish an SSL connection: three protocols are used in this layer: the SSL Handshake Protocol, the SSL Change Cipher Specification Protocol and the SSL Alert Protocol.

2.3 SSL SESSION AND CONNECTION

The concepts as mentioned above are fundamental for a connection between the client and the server, and they also encompass a series of attributes. Let's try to give some more details:

- **Connection:** this is a logical client/server link, associated with the provision of a suitable type of service. In SSL terms, it must be a peer-to-peer connection with two network nodes.
- **Session**: this is an association between a client and a server that defines a set of parameters such as algorithms used, session number etc. An SSL session is created by the Handshake Protocol that allows parameters to be shared among the connections made between the server and the client, and sessions are used to avoid negotiation of new parameters for each connection.

  This means that a single session is shared among multiple SSL connections between the client and the server. In theory, it may also be possible that multiple sessions are shared by a single connection, but this feature is not used in practice. The concepts of a SSL session and connection involve several parameters that are used for SSL-enabled communication between the client and the server. During the negotiations of the handshake protocol, the encryption methods are established and a series of parameters of the Session State are subsequently used within the session. A session state is defined by the following parameters:

- **Session Identifier:** this is an identifier generated by the server to identify a session with a chosen client,
- Peer certificate: X.509 certificate of the peer,
- compression method: a method used to compress data prior to encryption,
- Algorithm specification termed CipherSpec: specifies the bulk data encryption algorithm (for example DES) and the hash algorithm (for example MD5) used during the session,
- Master secret: 48-byte data being a secret shared between the client and server,
- "is resumable": this is a flag indicating whether the session can be used to initiate new connections.

According to the specification, the SSL connection state is defined by the following parameters:

- Server and client random: random data generated by both the client and server for each connection,
- Server write MAC secret: the secret key used for data written by the server,
- Client write MAC secret: the secret used for data written by the client,
- Server write key: the bulk cipher key for data encrypted by the server and decrypted by the client,
- Client write key: the bulk cipher key for data encrypted by the client and decrypted by the server,
- Sequence number: sequence numbers maintained separately by the server for messages transmitted and received during the data session.

The abbreviation MAC used in the above definitions means Message Authentication Code that is used for transmission of data during the SSL session. The role of MAC will be explained further when discussing the record protocols. A brief description of the terms was necessary to be able to explain the next issues connected with the functioning of the SSL protocol, namely the SSL record protocol.

2.3.1 SSL Record Protocol

The SSL record protocol involves using SSL in a secure manner and with message integrity ensured. To this end it is used by upper layer SSL protocols. The purpose of the SSL record protocol is to take an application message to be transmitted, fragment the data which needs to be sent, encapsulate it with appropriate headers and create an object just called a record, which is encrypted and can be forwarded for sending under the TCP protocol. The first step in the preparation of transmission of the application data consists in its fragmentation i.e. breaking up the data stream to be transmitted into 16Kb (or smaller) data fragments followed by the process of their conversion in a record. These data fragments may be further compressed, although the SSL 3.0 protocol specification includes no compression protocol, thus at present, no data compression is used.

At this moment, creation of the record is started for each data portion by adding a header to it, possible information to complete the required data size and the MAC. The record header that is added to each data portion contains two elementary pieces of information, namely the length of the record and the length of the data block added to the original data. In the next step, the record data constructed consists of the following elements:

- primary data,
- some padding to complete the datagram as required,
- MAC value.

MAC is responsible for the verification of integrity of the message included in the transmitted record. It is the result of a hash function that follows a specific hash algorithm, for example MD5 or SHA-1. MAC is determined as a result of a hash function that receives the following data:

MAC = **Hash function [secret key, primary data, padding, sequence number]**

A secret key in creation of MAC is either a client write MAC,secret or a server write MAC secret respectively, it depends on which party prepares the packet. After receiving the packet, the receiving party computes its own value of the MAC and compares it with that received. If the two values match, this means that data has not been modified during the transmission over the network. The length of the MAC obtained in this way depends on the method uses for its computing. Next, the data plus the MAC are encrypted using a preset symmetric encryption algorithm, for example DES or triple DES. Both data and MAC are encrypted. This prepared data is attached with the following header fields:

- Content type: identifies what payload is delivered by the packet to determine which higher protocols are to be used for processing of data included in the packet. The possible values are change_cipher_spec, alert, handshake, and application_data that refer to the appropriate protocols.
- Major version: establishes the main portion of the protocol version to be used. For SSL 3.0, the value is 3,
- Minor version: establishes the additional portion of the used version of the protocol. For SSL 3.0 the value is 0.

With the addition of fields, the process of record preparation is completed. Afterwards, the record is sent to the targeted point.

2.3.2 The Alert Protocol

The Alert Protocol is used by parties to convey session messages associated with data exchange and functioning of the protocol. Each message in the alert protocol consists of two bytes. The first byte always takes a value, "warning" (1) or "fatal" (2), that determines the severity of the message sent. Sending a message having a „fatal" status by either party will result in an immediate termination of the SSL session. The next byte of the message contains one of the defined error codes, which may occur during an SSL communication session.

2.3.3 The Change Cipher Specification Protocol

This protocol is the simplest SSL protocol. It consists of a single message that carries the value of 1. The sole purpose of this message is to cause the pending session state to be established as a fixed state, which results, for example, in defining the used set of protocols. This type of message must be sent by the client to the server and vice versa. After exchange of messages, the session state is considered agreed. This message and any other SSL messages are transferred using the SSL record protocol.

2.3.4 The Handshake Protocol

The handshake protocol constitutes the most complex part of the SSL protocol. It is used to initiate a session between the server and the client. Within the message of this protocol, various components such as algorithms and keys used for data encryption are negotiated. Due to this protocol, it is possible to authenticate the parties to each other and negotiate appropriate parameters of the session between them.

The process of negotiations between the client and the server can be divided into 4 phases separated with horizontal broken lines. During the first phase, a logical connection must be initiated between the client and the server followed by the negotiation on the connection parameters. The client sends the server a server_hello message containing data such as:

- Version: The highest SSL version supported by the client,
- Random: data consisting of a 32-bit timestamp and 28 bytes of randomly generated data. This data is used to protect the key exchange session between the parties of the connection.
- Session ID: a number that defines the session identifier. A nonzero value of this field indicates that the client wishes to update the parameters of an existing connection or establish a new connection on this session. A zero value in this field indicates that the client wishes to establish a new connection.
- Cipher Suite: a list of encryption algorithms and key exchange method supported by the client. The server, in response to the client_hello message sends a server_hello message, containing the same set of fields as the client message, placing the following data:
- Version: the lowest version number of the SSL protocol supported by the server,
- random data: the same fashion as used by the client, but the data generated is completely independent,
- session ID: if the client field was nonzero, the same value is sent back; otherwise the server's session ID field contains the value for a new session,
- Cipher Suite: the server uses this field to send a single set of protocols selected by the server from those proposed by the client. The first element of this field is a chosen method of exchange of cryptographic keys between the client and the server. The next element is the specification of encryption algorithms and hash functions, which will be used within the session being initiated, along with all specific parameters.

The set of encryption algorithms and key exchange method sent in the Cipher Suite field establishes three components:

1. the method of key exchange between the server and client,
2. the encryption algorithm for data encryption purposes,
3. a function used for obtaining the MAC value.

The server begins the next phase of negotiations by sending its certificate to the client for authentication. The message sent to the client contains one or a chain of X509 certificates. These are necessary for authentication of both the server and the certification path towards a trusted certification official of the certificating body for the server.

This step is not obligatory and may be omitted, if the negotiated method of key exchange does not require sending the certificate (in the case of anonymous Diffie-Hellman method). Depending on the negotiated method of key exchange, the server may send an additional server_key_exchange message, which is however not required in the case when the fixed Diffie-Hellman method or RSA key exchange technique has been negotiated. Moreover, the server can request a certificate from the client. The final step of Phase 2 is the server_done message, which has no parameters and is sent by the server merely to indicate the end of the server messages.

After sending this message, the server waits for a client response. Upon receipt of the message, the client should verify the server's certificate, the certificate validation data and path, as well as any other parameters sent by the server in the server_hello message. The client's verification consists of:

- Validation date check of the certificate and comparison with the current date, to verify whether the certificate is still valid,
- checking whether the certifying body is included in the list of trusted Certifying Authorities in possession of the client. If the CA, which has issued the server's certificate is not included in the CAs list, the client attempts to verify the CA signature. If no information about the CA can be obtained, the client terminates the identification procedure by either returning the error signal or signalling the problem for the user to solve it.
- Identifying the authenticity of the public key of the CA which has issued the certificate: if the Certifying Authority is included in the client's list of trusted CAs, the client checks the CA's public key stated in the server's certificate with the public key available from the list. This procedure verifies the authenticity of the certifying body.
- checking whether the domain name used in the certificate matches the server name shown in the server's certificate.

Upon successful completion of all steps the server is considered authenticated. If all parameters are matched and the server's certificate correctly verified, the client sends the server one or multiple messages. Next is the client__key_exchange message, which must be sent to deliver the keys. The content of this message depends on the negotiated method of key exchange. Moreover, at the server's request, the client's certificate is sent along with the message enabling verification of the certificate. This procedure ends Phase 3 of negotiations.

Phase 4 is to confirm the messages so far received and to verify whether the pending data is correct. The client sends a change_cipher_spec message (in accordance with the pending SSL ChangeCipher Spec), and then sets up the pending set of algorithm parameters and keys into the current set of the same. Then the client sends the finished message, which is first protected with just negotiated algorithms, keys and secrets. This is to confirm that the negotiated parameters and data are correct. The server in response to the client sends the same message sequence. If the finished message is correctly read by either party this confirms that the transmitted data, negotiated algorithms and the session key are correct. This indicates that the session has been terminated and that it is possible to send the application data between the server and the client, via SSL. At this point the TCP session between the client and the server is closed, however a session state is maintained, allowing it to resume communications within the session using the retained parameters.

It is worth noticing that both Phases 2 and 3 are used by both parties to verify the authenticity of the server's certificate and possibly the client's certificate during the handshake step. If the server cannot be successfully authenticated by the client on the basis of the delivered certificate, the handshake terminates and the client will generate an error message. The same will occur at the server if the client's certificate authenticity cannot be confirmed.

2.4 SSL IN USE

What is the reason for these complicated things associated with an SSL connection? SSL is used in many services but mostly SSL protects the HTTP communication channel over the Internet and therefore the SSL protocol is seen quite often as associated only with WWW pages. As it has been already mentioned, the SSL protocol can be used to protect the transmission for any TCP/IP service. Apart from the WWW accessing, the second most likely application of this protocol is associated with e-mail sending and receiving.

SSL is practically used in the system of HTTP and SMTP server services that work in conjunction with web servers. These servers allow an appropriate request to be generated with the user's own certificate (certificate generating utilities are usually available with SSL implementations) or by obtaining the certificate from one of the trusted CAs such as VeriSign (www.verisgn.com) or Thawte (www.thawte.com). In the same manner as for the WWW server it is possible to obtain and install the certificate on an SMTP server and enable secure SMTP transfers as well as IMAP and POP transfers for purposes of exchanging e-mail. A possible application of the SSL to support other services depends on the possibility of configuration of such a connection by the server.

2.5 IPSEC

IPsec is a suite of protocols for securing network connections, but the details and many variations quickly become overwhelming. This is particularly the case when trying to interoperate between disparate systems.

It is a complex suite of protocols. One cause of the complexity is that IPsec provides mechanism, not policy; rather than define an encryption algorithm or a certain authentication function, it provides a framework that allows an implementation to provide nearly anything that both ends agree upon.

2.6 THE IP DATAGRAM

Since we're looking at IPsec from the bottom up, we must first take a brief detour to revisit the IP Header itself, which carries all of the traffic we'll be considering.

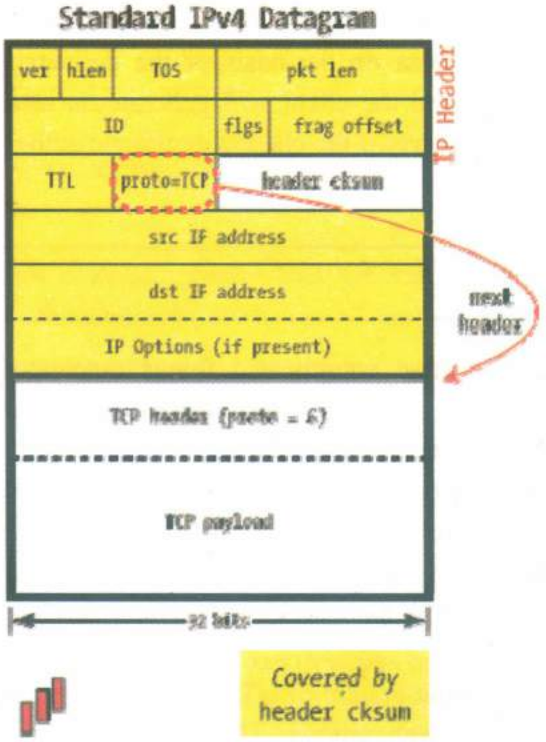

Fig. 1: The IP Datagram

ver: This is the version of the protocol, which is now 4=IPv4

hlen: IP Header length, as a four-bit number of 32-bit words ranging from 0..15. A standard IPv4 header is always 20 bytes long (5 words), and IP Options — if any — are indicated by a larger hlen field up to at most 60 bytes. This header length never includes the size of payload or other headers that follow.

TOS: Type of Service

This field is a bitmask that gives some clues as to the type of service this datagram should receive: optimize for bandwidth? Latency? Low cost? Reliability?

pkt len: Overall packet length in bytes, up to 65535. This count includes the bytes of the header, so this suggests that the maximum size of any payload is at least 20 bytes less. The vast majority of IP datagrams are much, much smaller.

ID: The ID field is used to associate related packets that have been fragmented (large packets broken up into smaller ones).

Flgs: These are small flags that mainly control fragmentation: one marks the packet as ineligible for fragmentation, and the other says that more fragments follow.

frag offset: When a packet is fragmented, this shows where in the overall "virtual" packet this fragment belongs.

TTL: This is the Time to Live, and is decremented by each router that passes this packet. When the value reaches zero, it suggests some kind of routing loop, so it's discarded to prevent it from running around the Internet forever.

Proto: This represents the protocol carried within this packet, and it's going to be central to most of our discussions. Though the datagram itself is IP, it always encapsulates a subsidiary protocol (TCP, UDP, ICMP, etc. — see the chart below) within. It can be thought of as giving the type of the header that follows.

header cksum: This holds a checksum of the entire IP header, and it's designed to detect errors in transit. This is not a cryptographicchecksum, and it doesn't cover any part of the datagram that follow the IP header.

src IP address: The 32-bit source IP address, which the recipient uses to reply to this datagram. Generally speaking, it's possible to spoof these addresses (i.e., lie about where the datagram is coming from).

dst IP address: The 32-bit destination IP address, which is where the packet is intended to arrive.

IP Options: These are an optional part of the IP header that contains application-specific information, though they are not commonly used for routine traffic. The presence of IP options is indicated by a hlen greater than 5, and they (if present) are included in the header checksum.

Payload: Each protocol type implies its own format for what follows the IP header, and we've used TCP here just to show an example.

These proto codes are defined by IANA — the Internet Assigned Numbers Authority — and there are many more than would ever be used by any single installation, but most will ring a bell with a network-savvy technician. These representative types are taken from the IANA website listing protocols:

Table 1: Some IP protocol codes

Protocol code	Protocol Description
1	ICMP — Internet Control Message Protocol
2	IGMP — Internet Group Management Protocol
4	IP within IP (a kind of encapsulation)
6	TCP — Transmission Control Protocol
17	UDP — User Datagram Protocol
41	IPv6 — next-generation TCP/IP
47	GRE — Generic Router Encapsulation (used by PPTP)
50	IPsec: ESP — Encapsulating Security Payload
51	IPsec: AH — Authentication Header

2.7 AH: AUTHENTICATION HEADER - AUTHENTICATION ONLY

AH is used to authenticate — but not encrypt — IP traffic, and this serves the treble purpose of ensuring that we're really talking to who we think we are, detecting alteration of data while in transit, and (optionally) to guard against replay by attackers who capture data from the wire and attempt to re-inject that data back onto the wire at a later date.

Authentication is performed by computing a cryptographic hash-based message authentication code over nearly all the fields of the IP packet (excluding those which might be modified in transit, such as TTL or the header checksum), and stores this in a newly-added AH header and sent to the other end.

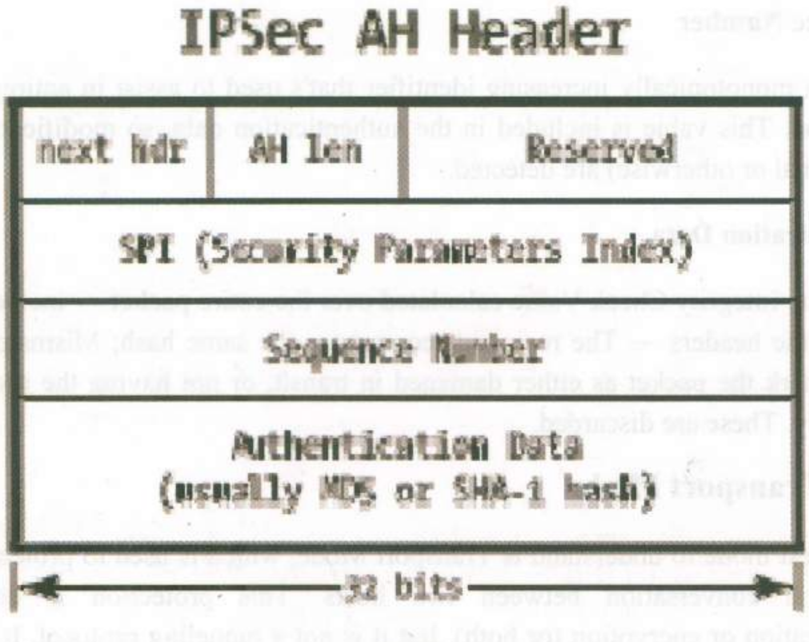

Fig. 2

This AH header contains just five interesting fields, and it's injected between the original IP header and the payload. We'll touch on each of the fields here, though their utility may not be fully apparent until we see how they're used in the larger picture.

next hdr

This identifies the protocol type of the following payload, and it's the original packet type being encapsulated: this is how the IPsec header(s) are linked together.

AH len

This defines the length, in 32-bit words, of the whole AH header, minus two words (this "minus two words" proviso springs from the format of IPv6's RFC 1883 Extension Headers, of which AH is one).

Reserved

This field is reserved for future use and must be zero.

Security Parameters Index

This is an opaque 32-bit identifier that helps the recipient select which of possibly many ongoing conversations this packet applies. Each AH-protected connection implies a hash algorithm (MD5, SHA-1, etc.), some kind of secret data, and a host of other parameters. The SPI can be thought of as an index into a table of these settings, allowing for easy association of packet with parameter.

Sequence Number

This is a monotonically increasing identifier that's used to assist in antireplay protection. This value is included in the authentication data, so modifications (intentional or otherwise) are detected.

Authentication Data

This is the Integrity Check Value calculated over the entire packet — including most of the headers — The recipient recomputes the same hash; Mismatched values mark the packet as either damaged in transit, or not having the proper secret key. These are discarded.

2.7.1 Transport Mode

The easiest mode to understand is Transport Mode, which is used to protect an end-to-end conversation between two hosts. This protection is either authentication or encryption (or both), but it is not a tunneling protocol. It has nothing to do with a traditional VPN: it's simply a secured IP connection.

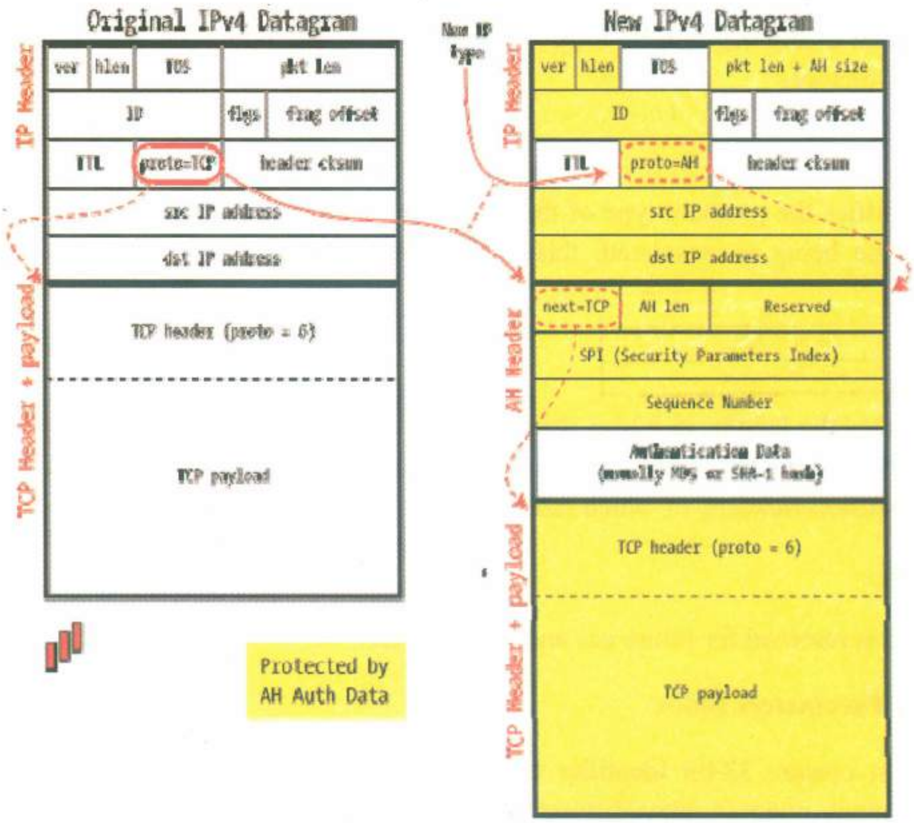

Fig. 3: IPSec in AH Transport Mode

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

This protocol shuffling is required to allow the original IP packet to be reconstituted at the other end: after the IPsec headers have been validated upon receipt, they're stripped off, and the original protocol type (TCP, UDP, etc.) is stored back in the IP header. We'll see this chain of next header fields again and again as we examine IPsec.

When the packet arrives at its destination and passes the authentication check, the AH header is removed and the Proto=AH field in the IP header is replaced with the saved "Next Protocol". This puts the IP datagram back to its original state, and it can be delivered to the waiting process.

2.7.2 Tunnel Mode

Tunnel Mode forms the more familiar VPN functionality, where entire IP packets are encapsulated inside another and delivered to the destination.

Like Transport mode, the packet is sealed with an Integrity Check Value to authenticate the sender and to prevent modification in transit. But unlike Transport mode, it encapsulates the *full IP header* as well as the payload, and this allows the source and destination addresses to be different from those of the encompassing packet: This allows formation of a tunnel.

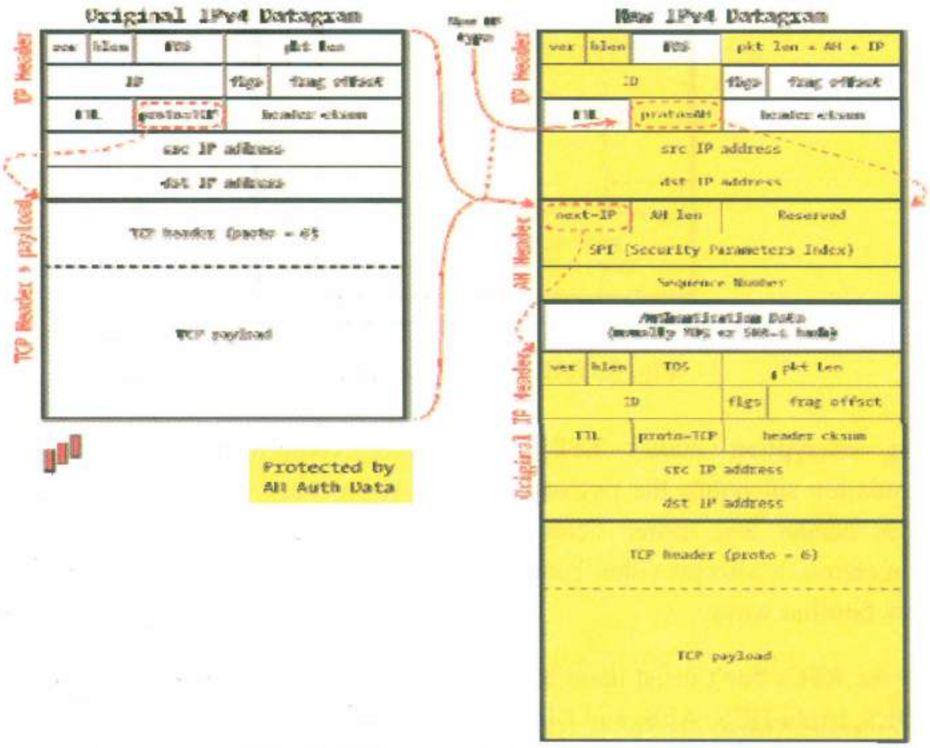

Fig. 4: IPSec in AH Tunnel Mode

When a Tunnel-mode packet arrives at its destination, it goes through the same authentication check as any AH-type packet, and those passing the check have their entire IP and AH headers stripped off. This effectively reconstitutes the original IP datagram, which is then injected into the usual routing process.

Most implementations treat the Tunnel-mode endpoint as a virtual network interface — just like an Ethernet interface or localhost — and the traffic entering or leaving it is subject to all the ordinary routing decisions.

The reconstituted packet could be delivered to the local machine or routed elsewhere (according to the destination IP address found in the encapsulated packet), though in any case is no longer subject to the protections of IPsec. At this point, it's just a regular IP datagram.

Though Transport mode is used strictly to secure an end-to-end connection between two computers, Tunnel mode is more typically used between gateways (routers, firewalls, or standalone VPN devices) to provide a Virtual Private Network (VPN).

2.8 ESP-ENCAPSULATING SECURITY PAYLOAD

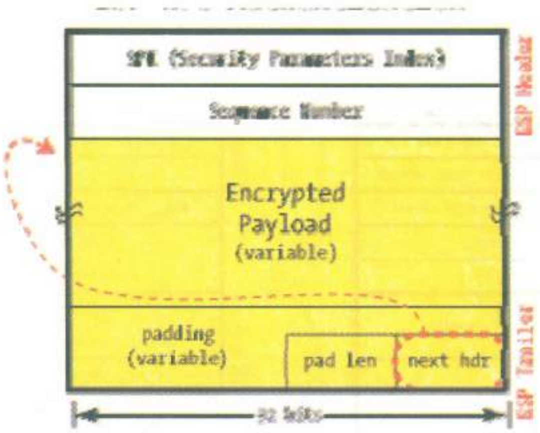

Fig. 5: ESP w/o Authentication

Adding encryption makes ESP a bit more complicated because the encapsulation surrounds the payload rather than precedes it as with AH: ESP includes header and trailer fields to support the encryption and optional authentication. It also provides Tunnel and Transport modes which are used in by-now familiar ways.

The IPsec RFCs don't insist upon any particular encryption algorithms, but we find DES, triple-DES, AES, and Blowfish in common use to shield the payload from prying eyes. The algorithm used for a particular connection is specified by the Security Association (covered in a later section), and this SA includes not only the algorithm, but the key used.

Unlike AH, which provides a small header before the payload, ESP surrounds the payload it's protecting. The Security Parameters Index and Sequence Number serve the same purpose as in AH, but we find padding, the next header, and the optional Authentication Data at the end, in the ESP Trailer.

It's possible to use ESP without any actual encryption (to use a NULL algorithm), which nonetheless structures the packet the same way. This provides no confidentiality, and it only makes sense if combined with ESP authentication. It's pointless to use ESP without either encryption or authentication (unless one is simply doing protocol testing).

Padding is provided to allow block-oriented encryption algorithms room for multiples of their blocksize, and the length of that padding is provided in the pad len field. The next hdr field gives the type (IP, TCP, UDP, etc.) of the payload in the usual way, though it can be thought of as pointing "backwards" into the packet rather than forward as we've seen in AH.

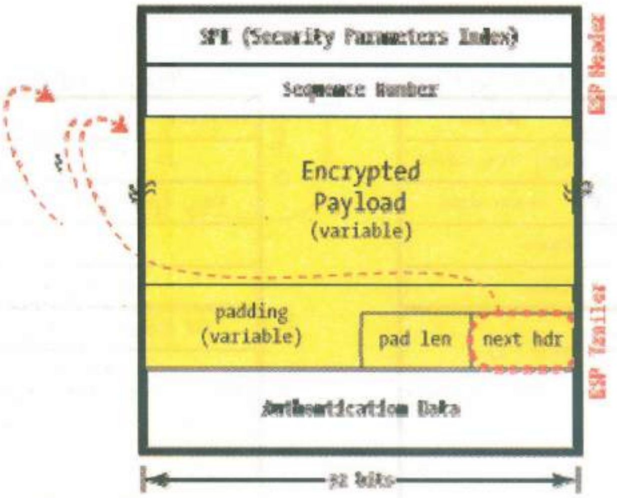

Fig. 6: ESP with Authentication

In addition to encryption, ESP can also optionally provide authentication, with the same HMAC as found in AH. Unlike AH, however, this authentication is only for the ESP header and encrypted payload: it does not cover the full IP packet. Surprisingly, this does not substantially weaken the security of the authentication, but it does provide some important benefits.

When an outsider examines an IP packet containing ESP data, it's essentially impossible to make any real guesses about what's inside save for the usual data found in the IP header (particularly the source and destination IP addresses). The attacker will certainly know that it's ESP data — that's also in the header — but the type of the payload is encrypted with the payload.

Even the presence or absence of Authentication Data can't be determined by looking at the packet itself (this determination is made by using the Security Parameters Index to reference the pre shared set of parameters and algorithms for this connection).

However, it should be noted that sometimes the envelope provides hints that the payload does not. With more people sending VoIP inside ESP over the internet, the QoS taggings are in the outside header and is fairly obvious what traffic is VoIP signaling (IP precedence 3) and what is RTP traffic (IP precedence 5). It's not a sure thing, but it might be enough of a clue to matter in some circumstances.

2.8.1 ESP in Transport Mode

As with AH, Transport Mode encapsulates just the datagram's payload and is designed strictly for host-to-host communications. The original IP header is left in place (except for the shuffled Protocol field), and it means that the source and destination IP addresses are unchanged.

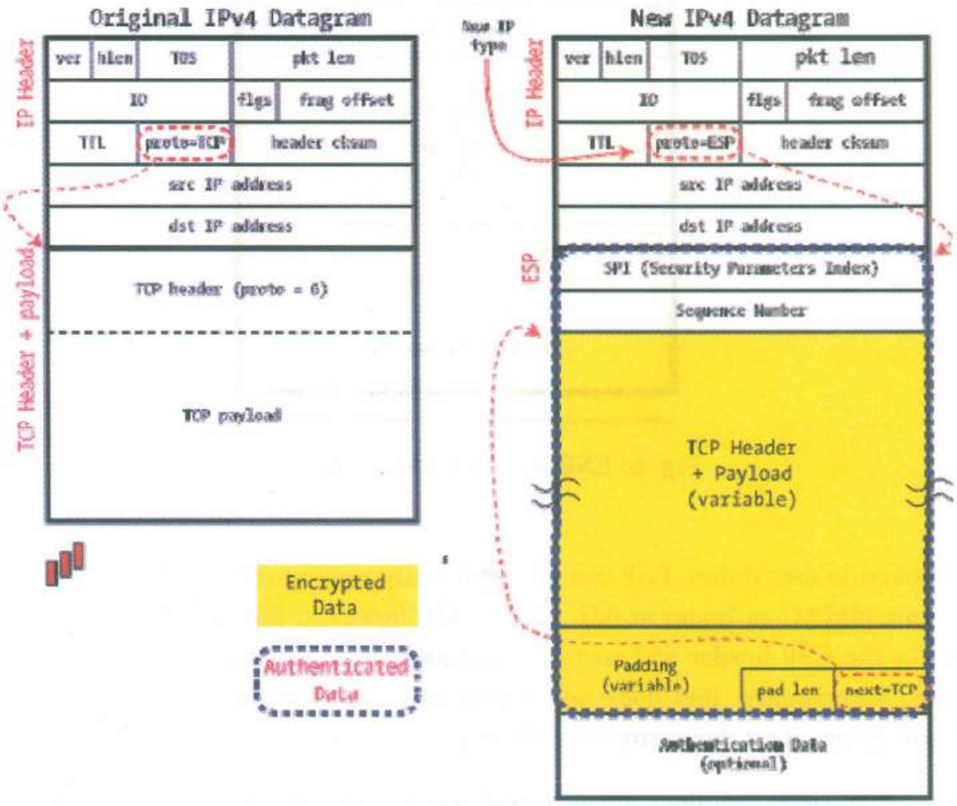

Fig. 7: IPSec in ESP Transport Mode

IPsec would be nearly useless without the cryptographic facilities of authentication and encryption, and these require the use of secret keys known to the participants but not to anyone else.

The most obvious and straightforward way to establish these secrets is via manual configuration: one party generates a set of secrets, and conveys them to all the partners. All parties install these secrets in their appropriate Security Associations in the SPD.

But this process does not scale well, nor is it always terribly secure: the mere act of conveying the secrets to another site's SPD may well expose them in transit. In a larger installation with many devices using the same preshared key, compromise of that key makes for a very disruptive re-deployment of new keys.

IKE — Internet Key Exchange — exists to allow two endpoints to properly set up their Security Associations, including the secrets to be used. IKE uses the ISAKMP (Internet Security Association Key Management Protocol) as a framework to support establishment of a security association compatible with both ends.

2.8.2 ESP in Tunnel Mode

Our final look of standalone ESP is in Tunnel mode, which encapsulates an entire IP datagram inside the encrypted shell

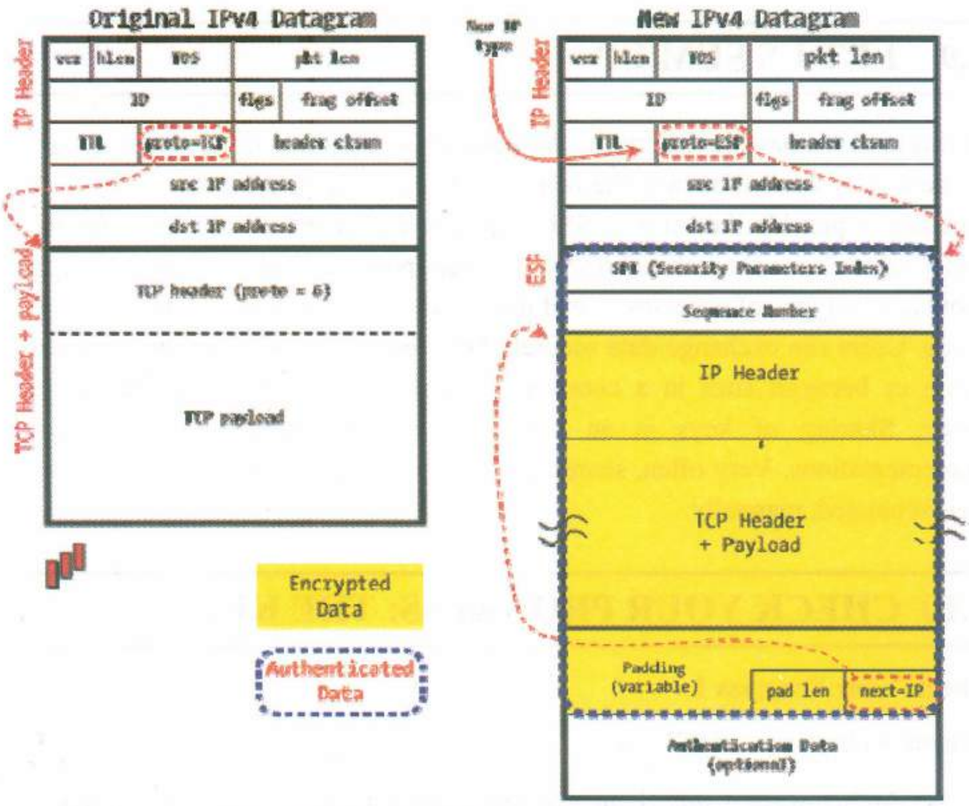

Fig. 8: IPSec in ESP Tunnel Mode

Providing an encrypted Tunnel Mode connection is getting very close to the traditional VPN that springs to mind when most of us think about IPsec, but we have to add authentication of one type or another to complete the picture: this is covered in the following section.

Unlike AH, where an onlooker can easily tell whether traffic is in Tunnel or Transport mode, this information is unavailable here: the fact that this is Tunnel mode (via next=IP) is part of the encrypted payload, and is simply not visible to one unable to decrypt the packet.

Check Your Progress 1

Note: a) Space is given below for writing your answer.

b) Compare your answer with the one given at the end of the Unit.

1) What are the main objectives of SSL?

....

....

....

....

....

2.9 LET US SUM UP

In this unit, we have looked at the various components that make up the secure sockets layer as well as how the components of the IPSEC protocol interwork together to provide the service. SSL provides the security enablement for the upper layer protocols such as http, smtp, imap, pop etc. IPSEC, in contrast, can provide two types of accesses to end users – the tunnel mode and the transport mode. Users can exchange data securely between two hosts using the transport mode or between sites in a completely encrypted fashion using the tunnel mode. Sharing of keys is an important concern in both the protocol implementations. Very often, shared secret keys are implemented and the keys are exchanged, manually.

2.10 CHECK YOUR PROGRESS: THE KEY

Check Your Progress 1

The main objectives of SSL are:

- Authenticating the client and server to each other: the SSL protocol supports the use of standard key cryptographic techniques (public key encryption) to authenticate the communicating parties to each other.

Though the most frequent application consists in authenticating the service client on the basis of a certificate, SSL may also use the same methods to authenticate the client.

- Ensuring data integrity: during a session, data cannot be either intentionally or unintentionally tampered with.
- Securing data privacy: data in transport between the client and the server must be protected from interception and be readable only by the intended recipient. This prerequisite is necessary for both the data associated with the protocol itself (securing traffic during negotiations) and the application data that is sent during the session itself.

2.11 SUGGESTED READINGS

- http://www.unixwiz.net/

UNIT 3 SPECIFIC PROTOCOL-II

Structure

3.0 INTRODUCTION

In this unit, we will take a look at several application layer protocols that enable us to provide security to the data that is in transit from one point on the network to another. These protocols are part of common applications that we use in our day-to-day use of the Internet as well.

Initially, we will look at protocols that help us secure transfer of email information and then go on to look at other similar protocols that help in secure transfer.

3.1 OBJECTIVES

After going through this Unit, you should be able to:

- understand secure shell;
- explain POP3 protocol and PGP protocol; and
- know Hypertext Transfer Protocol.

3.2 SECURE SHELL (SSH)

Secure Shell (SSH) is a protocol for secure remote login and other secure network services over an insecure network. It consists of three major components:

- The Transport Layer Protocol [SSH-TRANS] provides server authentication, confidentiality, and integrity. It may optionally also provide compression. The transport layer will typically be run over a TCP/IP connection, but might also be used on top of any other reliable data stream.

- The User Authentication Protocol [SSH-USERAUTH] authenticates the client-side user to the server. It runs over the transport layer protocol.
- The Connection Protocol [SSH-CONNECT] multiplexes the encrypted tunnel into several logical channels. It runs over the user authentication protocol.

The client sends a service request once a secure transport layer connection has been established. A second service request is sent after user authentication is complete. This allows new protocols to be defined and coexist with the protocols listed above.

The connection protocol provides channels that can be used for a wide range of purposes. Standard methods are provided for setting up secure interactive shell sessions and for forwarding ("tunneling") arbitrary TCP/IP ports and X11 connections.

ssh packets have message numbers in the range 1 to 255. These numbers have been allocated as follows:

1. Transport layer protocol

   1 to 19 Transport layer generic (e.g., disconnect, ignore, debug, etc.)

   20 to 29 Algorithm negotiations

   30 to 49 Key exchange methods specific (numbers can be reused for different authentication methods)

2. User authentication protocol

   50 to 59 User authentications generic

   60 to 79 User authentication methods specific (numbers can be reused for different authentication methods)

3. Connection protocol

   80 to 89 Connection protocol generic

   90 to 127 Channel related messages

4. Reserved for client protocols

   128 to 191 Reserved

5. Local extensions

   192 to 255 Local extensions

The transport protocol [SSH-TRANS] provides a confidential channel over an insecure network. It performs server host authentication, key exchange, encryption, and integrity protection. It also derives a unique session id that may be used by higher-level protocols.

The authentication protocol [SSH-USERAUTH] provides a suite of mechanisms that can be used to authenticate the client user to the server. Individual mechanisms specified in the authentication protocol use the session id provided by the transport protocol and/or depend on the security and integrity guarantees of the transport protocol.

The connection protocol [SSH-CONNECT] specifies a mechanism to multiplex multiple streams (channels) of data over the confidential and authenticated transport. It also specifies channels for accessing an interactive shell, for proxy-forwarding various external protocols over the secure transport (including arbitrary TCP/IP protocols), and for accessing secure subsystems on the server host.

3.3 IMPLEMENTATIONS AVAILABLE

Ssh client and server implementations are available both on MS Windows and Linux platforms. They are available for most Unix-like OS platforms. Openssh is an open-source implementation that is available freely for use. Most Linux distributions provide versions of Openssh.

On a typical Linux implementation, the server is available as sshd and the client is available as ssh. Additional utilities for copying and transferring files, scp and sftp are provided for convenience. On MS Windows platform, a GUI based client is available for ssh and scp. The latter allows drag-and-drop of files across the local and remote host.

The server runs sshd which initialises its startup parameters from a file named sshd.conf. Ssh, scp, sftp clients connect to the hosts that run this server program for interactive logins and file transfers.

3.4 POST OFFICE PROTOCOL 3 (POP3S)

Users typically download email from mail servers using the POP3 protocol. POP3, like so many other legacy internet protocols, exchanges all data and passwords in the clear. This means that anyone with access to the network path between the server and its clients can discover users' passwords simply by listening in on the network. Also, most Unix systems use the same password for email access and shell access, anyone able to get your password can access your account and in turn attempt to hack the server or plant trojans.

Therefore, POP3 should be operated in a secure mode. This is done by operating POP3 over a secure transport protocol such as SSL. All Linux distributions provide an implementation of SSL called OpenSSL. This is a free, open source implementation of the Secure Sockets Layer. It is the same encryption layer that is now the *de facto* standard for secure, Internet-based e-commerce transactions. Open SSL is used to secure POP3 transfers to protect them from eavesdroppers looking to get userid and passwords.

The POP3 protocol, when used over an SSL layer, is called POP3S. The Post Office Protocol version 3 (POP3) is the protocol that clients use to retrieve mail from servers. Users of POP generally store their mail locally on their machines and configure their mail client to retrieve new mail periodically.

POP3 servers require clients to authenticate with a username and password. This ensures that only authenticated clients are able to retrieve the user's email. POP3 is an unencrypted protocol, so the username and password are transported as clear text.

POP3S is POP3 over Secure Sockets Layer. The SSL provides an encrypted end-to-end connection for the Post Office Protocol. More importantly, it provides an encrypted connection over which the username and password may be transmitted for authentication purposes.

3.5 PRETTY GOOD PRIVACY (PGP)

Pretty Good Privacy (PGP) is both a protocol and the name of the program that most widely implements that protocol. The PGP protocol, documented in RFC 1991 and expanded into the OpenPGP proposal (RFC 2440), is a presentation layer protocol that defines a standard to cryptographically secure email messages.

The program PGP (Pretty Good Privacy) was initially published by Phil Zimmermann in 1991, in response to U.S. Senate Bill 266 which was designed to force manufacturers of secure communications to provide a "back-door" by which the U.S. government would be able to read those communications. A group of open-source programmers related to the GNU project wrote GnuPG, a free software (GPL'd) version conforming to the OpenPGP standards.

The two basic encryption techniques are "symmetric" and "asymmetric". Symmetric encryption involves only one key, which is used by both the sender for encrypting and the recipient for decrypting. A number of symmetric algorithms exist, including blowfish, Triple-DES, CAST, IDEA. IDEA is, like RSA legally restricted, but the other algorithms may be freely used. An older algorithm, DES, was cracked in 1999 and should not be used now. A key size of 128 bits is currently considered to be sufficiently secure, key sizes of 56 bits or less can be considered crackable.

The obvious problem with symmetric encryption is the means of distributing the key. Asymmetric (public key) encryption solves this problem by using two keys, one public and the other private. A message is encrypted to a recipient using that person's public key, but it can only be decrypted using the corresponding private key. This means you cannot read a message that you yourself encrypted (unless you also encrypted it simultaneously with your own public key). Hence the public key may be freely distributed to all with no fear of compromising security. The private key, of course, should be carefully protected.

A key size of 2048 bits is sufficiently secure. Symmetric encryption at 128 bits is roughly equivalent is strength to asymmetric encryption at 2048 bits.

3.6 SECURE HTTP

Hypertext Transfer Protocol (http) is a means for transmitting and receiving information across the Internet. http is a request and response protocol. http is commonly used to access html pages, but other resources can be utilized as well through http. In many cases, clients may be exchanging confidential information with a server, which needs to be secured in order to prevent unauthorized access. For this reason, https, or *secure http*, was developed by Netscape corporation to allow authorization and secured transactions.

HTTPS encrypts and decrypts user page requests as well as the pages that are returned by the Web server. The use of HTTPS protects against eavesdropping and man-in-the-middle attacks.

There are some primary differences between http and https, however, beginning with the default port, which is 80 for http and 443 for https. Https works by transmitting normal http interactions through an encrypted system, so that in theory, the information cannot be accessed by any party other than the client and end server.

There are two common types of security layers:

- Transport Layer Security (TLS) and
- Secure Sockets Layer (SSL)

HTTPS and SSL support the use of X.509 digital certificates from the server so that, if necessary, a user can authenticate the sender.

There are two primary differences between an HTTPS and an HTTP connection work:

- HTTPS connects on port 443, while HTTP is on port 80 (by default, unless specified)
- HTTPS encrypts the data sent and received with SSL, while HTTP sends it all as plain text

For example, suppose you visit a Web site to view their online catalog. When you're ready to order, you will be given a Web page order form with a Uniform Resource Locator (URL) that starts with https://. When you click "Send," to send the page back to the catalog retailer, your browser's HTTPS layer will encrypt it. The acknowledgement you receive from the server will also travel in encrypted form, arrive with an https:// URL, and be decrypted for you by your browser's HTTPS sublayer.

The effectiveness of HTTPS can be limited by poor implementation of browser or server software or a lack of support for some algorithms. Furthermore, although HTTPS secures data as it travels between the server and the client, once the data is decrypted at its destination, it is only as secure as the host computer.

There are only a couple things you need in order to host secure pages on your Web site:

- A Web server such as Apache with mod_ssl that supports SSL encryption
- A Unique IP address - this is what the certificate providers use to validate the secure certificate
- An SSL Certificate from an SSL certificate provider

If you got HTTPS Certificate, your hosting provider will need to set up the certificate in your Web server so that every time a page is accessed via the https:// protocol, it hits the secure server. Once that is set up, you can start building your Web pages that need to be secure.

When using HTTPS:

- Point to all Web forms on the https:// server. Whenever you link to Web forms on your Web site, get in the habit of linking to them with the full server URL including the https:// designation. This will insure that they always are secured.
- Use relative paths to images on secured pages. If you use a full path (http://www...) for your images, and those images are not on the secure server, your customers will get error messages that say things like: "Insecure data found. Continue?" This can be disconcerting, and many people will stop the purchase process when they see that. If you use

relative paths, your images will be loaded from the same secure server as the rest of the page.

- Secure only the pages that request and collect data. It is possible to run your entire Web site on https://, but it slows down the connection and some SSL providers charge you on the bandwidth secured. You should only secure those pages that collect data.

Check Your Progress 1

Note: a) Space is given below for writing your answer.

b) Compare your answer with the one given at the end of the Unit.

1) What is Post Office Protocol 3 (PoP3s)?

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

2) What are two primary differences between an HTTPS and an HTTP connection work?

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

3.7 LET US SUM UP

In this unit, we have seen three secure protocols. PGP provides security to messages that are encrypted and signed and either stored on a disk (as a file) or transmitted across the network. Similarly, we have seen two application layer protocols POP3s and https that use the lower level implementation of a secure transport protocol, SSL. SSL makes sure that there is an end-to-end secure channel available for transmission of data between the two end points. Protocols at the higher layers use protocols like SSL to transfer data securely.

3.8 CHECK YOUR PROGRESS: THE KEY

Check Your Progress 1

1) Users typically download email from mail servers using the POP3 protocol. POP3, like so many other legacy internet protocols, exchanges all data and passwords in the clear. This means that anyone with access to the network path between the server and its clients can discover users' passwords simply by listening in on the network. Also, most Unix systems use the same password for email access and shell access, anyone able to get your password can access your account and in turn attempt to hack the server or plant trojans.

   Therefore, POP3 should be operated in a secure mode. This is done by operating POP3 over a secure transport protocol such as SSL. All Linux distributions provide an implementation of SSL called OpenSSL. This is a free, open source implementation of the Secure Sockets Layer. It is the same encryption layer that is now the *de facto* standard for secure, Internet-based e-commerce transactions. Open SSL is used to secure POP3 transfers to protect them from eavesdroppers looking to get userid and passwords.

   The POP3 protocol, when used over an SSL layer, is called POP3S. The Post Office Protocol version 3 (POP3) is the protocol that clients use to retrieve mail from servers. Users of POP generally store their mail locally on their machines and configure their mail client to retrieve new mail periodically.

   POP3 servers require clients to authenticate with a username and password. This ensures that only authenticated clients are able to retrieve the user's email. POP3 is an unencrypted protocol, so the username and password are transported as clear text.

   POP3S is POP3 over Secure Sockets Layer. The SSL provides an encrypted end-to-end connection for the Post Office Protocol. More importantly, it provides an encrypted connection over which the username and password may be transmitted for authentication purposes.

2) There are two primary differences between an HTTPS and an HTTP connection work:

   - HTTPS connects on port 443, while HTTP is on port 80 (by default, unless specified)
   - HTTPS encrypts the data sent and received with SSL, while HTTP sends it all as plain text

3.9 SUGGESTED READINGS

- http://help.ece.ubc.ca/POP3S
- http://searchsoftwarequality.techtarget.com/definition/HTTPS
- http://webdesign.about.com/od/ecommerce/a/aa070407.htmhttp://www.ietf.org/rfc/rfc4251.txt
- http://www.lugod.org/presentations/pgp/history.html

Student Satisfaction Survey

ignou
THE PEOPLE'S
UNIVERSITY

Student Satisfaction Survey of IGNOU Students

Enrollment No.	
Mobile No.	
Name	
Programme of Study	
Year of Enrolment	
Age Group	□ Below 30 □ 31-40 □ 41-50 □ 51 and above
Gender	□ Male □ Female
Regional Centre	
States	
Study Center Code	

Please indicate how much you are satisfied or dissatisfied with the following statements

Sl. No.	Questions	Very Satisfied	Satisfied	Average	Dissatisfied	Very Dissatisfied
1.	Concepts are clearly explained in the printed learning material	□	□	□	□	□
2.	The learning materials were received in time	□	□	□	□	□
3.	Supplementary study materials (like video/audio) available	□	□	□	□	□
4.	Academic counselors explain the concepts clearly	□	□	□	□	□
5.	The counseling sessions were interactive	□	□	□	□	□
6.	Changes in the counseling schedule were communicated to you on time	□	□	□	□	□
7.	Examination procedures were clearly given to you	□	□	□	□	□
8.	Personnel in the study centers are helpful	□	□	□	□	□
9.	Academic counseling sessions are well organized	□	□	□	□	□
10.	Studying the programme/course provide the knowledge of the subject	□	□	□	□	□
11.	Assignments are returned in time	□	□	□	□	□
12.	Feedbacks on the assignments helped in clarifying the concepts	□	□	□	□	□
13.	Project proposals are clearly marked and discussed	□	□	□	□	□
14.	Results and grade card of the examination were provided on time	□	□	□	□	□
15.	Overall, I am satisfied with the programme	□	□	□	□	□
16.	Guidance from the programme coordinator and teachers from the school	□	□	□	□	□

After filling this questionnaire send it to:
Programme Coordinator, School of Vocational Education and Training,
Room no. 19, Block no. 1, IGNOU, Maidangarhi, New Delhi- 110068

MPDD-IGNOU/P.O.2T/August,2011

ISBN-978-81-266-5513-4