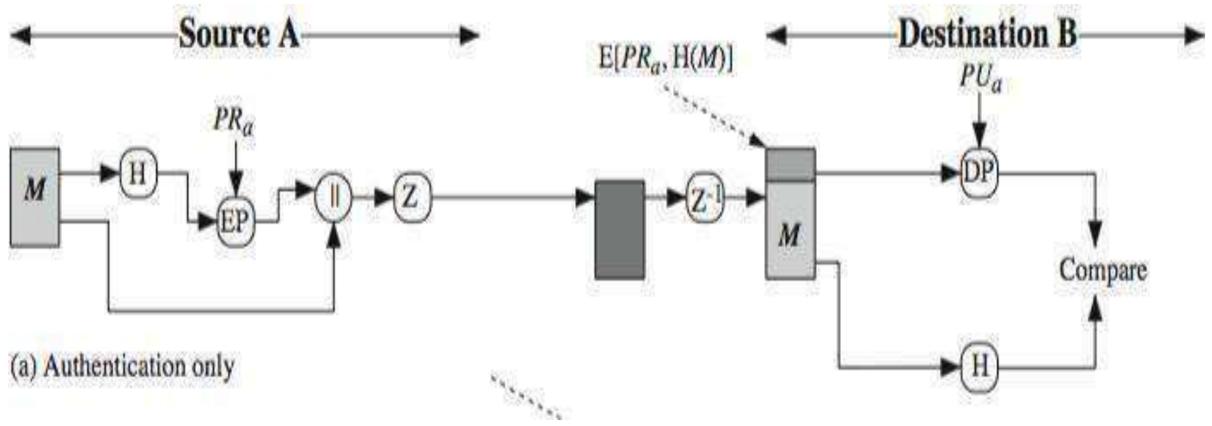
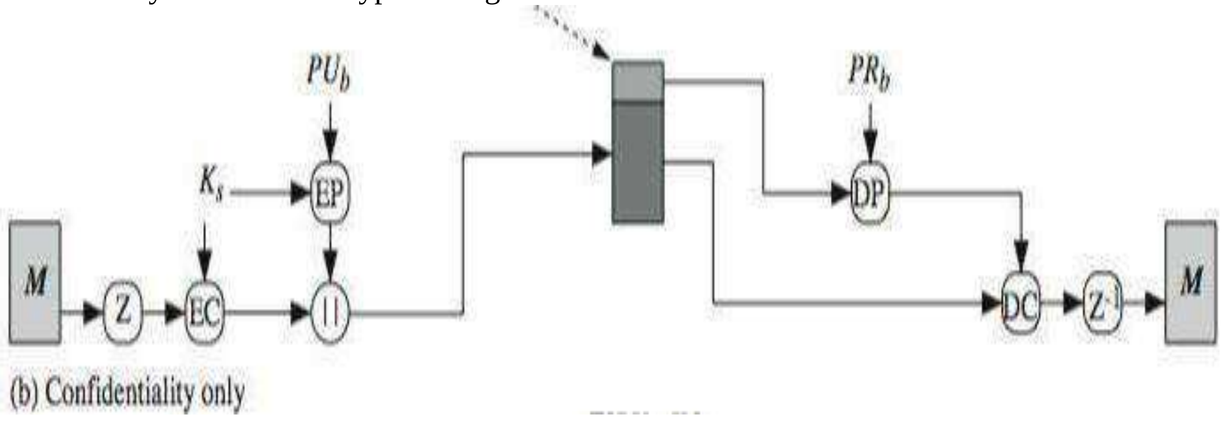


PGP Operation – Confidentiality:

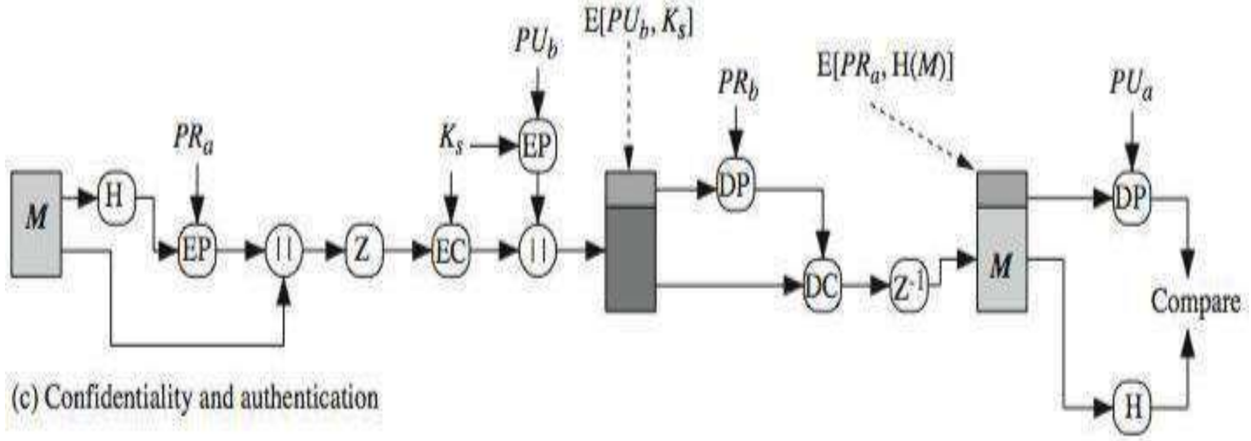
1. sender generates message and random 128-bit number to be used as session key for this message only
2. message is encrypted, using CAST-128 / IDEA/3DES with session key
3. session key is encrypted using RSA with recipient's public key, then attached to message
4. receiver uses RSA with its private key to decrypt and recover session key
5. session key is used to decrypt message



PGP Operation – Confidentiality & Authentication

Uses both services on same message

Create signature & attach to message o encrypt both message & signature o attach RSA encrypted session key



PGP Operation – Compression

As a default, PGP compresses the message after applying the signature but before encryption. This has the benefit of saving space both for e-mail transmission and for file storage.

The placement of the compression algorithm, indicated by Z for compression and Z^{-1} for decompression. So can store uncompressed message & signature for later verification & because compression is non deterministic uses ZIP compression algorithm

PGP Operation – Email Compatibility

- When PGP is used, at least part of the block to be transmitted is encrypted. If only the signature service is used, then the message digest is encrypted (with the sender's private key). If the confidentiality service is used, the message plus signature (if present) are encrypted (with a one-time symmetric key).
- Thus, part or all of the resulting block consists of a stream of arbitrary 8-bit octets.
- However, many electronic mail systems only permit the use of blocks consisting of ASCII text.
- To accommodate this restriction, PGP provides the service of converting the raw 8 -bit binary stream to a stream of printable ASCII characters. The scheme used for this purpose is radix-64 conversion.
- Each group of three octets of binary data is mapped into four ASCII characters. This format also appends

S/MIME (Secure/Multipurpose Internet Mail Extensions)

Secure/Multipurpose Internet Mail Extension (S/MIME) is a security enhancement to the MIME Internet e-mail format standard based on technology from RSA Data Security. it appears likely that S/MIME will emerge as the industry standard for commercial and organizational use, while PGP will remain the choice for personal e-mail security for many users. S/MIME is defined in a number of documents—most importantly RFCs 3370, 3850, 3851, and 3852.
S/MIME support in many mail agents eg MS Outlook, Mozilla, Mac Mail etc

To understand S/MIME, we need first to have a general understanding of the underlying e -mail format that it uses, namely MIME. We have to learn about RFC5322(internet Message Format)

RFC 5322:

- Defines a format for text messages that are sent using electronic mail
- Messages are viewed as having an envelope and contents
  - The envelope contains whatever information is needed to accomplish transmission and delivery
  - The contents compose the object to be delivered to the recipient
  - RFC 5322 standard applies only to the contents

The content standard includes a set of header fields that may be used by the mail system to create the envelope

The overall structure of a message that conforms to RFC 5322 is very simple. A message consists of some number of header lines (*the header*) followed by unrestricted text (*the body*). The header is separated from the body by a blank line. Put differently, a message is ASCII text, and all lines up to the first blank line are assumed to be header lines used by the user agent part of the mail system.

A header line usually consists of a keyword, followed by a colon, followed by the keyword's arguments; the format allows a long line to be broken up into several lines. The most frequently used keywords are *From*, *To*, *Subject*, and *Date*. Here is an example message:

Date: October 8, 2009 2:15:49 PM EDT

From: "William Stallings"
<ws@shore.net> Subject: The
Syntax in RFC 5322
To: Smith@Other-host.com
Cc: Jones@Yet-Another-Host.com

Hello. This section begins the actual message body, which is delimited from the message heading by a blank line.

Multipurpose Internet Mail Extensions (MIME):

An extension to the RFC 5322 framework that is intended to address some of the problems and limitations of the use of Simple Mail Transfer Protocol (SMTP) lists the following limitations of the SMTP/5322 scheme.

1. SMTP cannot transmit executable files or other binary objects.
2. SMTP cannot transmit text data that includes national language characters, because these are represented by 8-bit codes with values of 128 decimal or higher, and SMTP is limited to 7-bit ASCII.
3. SMTP servers may reject mail message over a certain size.
4. SMTP gateways that translate between ASCII and the character code EBCDIC do not use a consistent set of mappings, resulting in translation problems.

MIME is intended to resolve these problems in a manner that is compatible with existing RFC 5322 implementations. The specification is provided in RFCs 2045 through 2049.

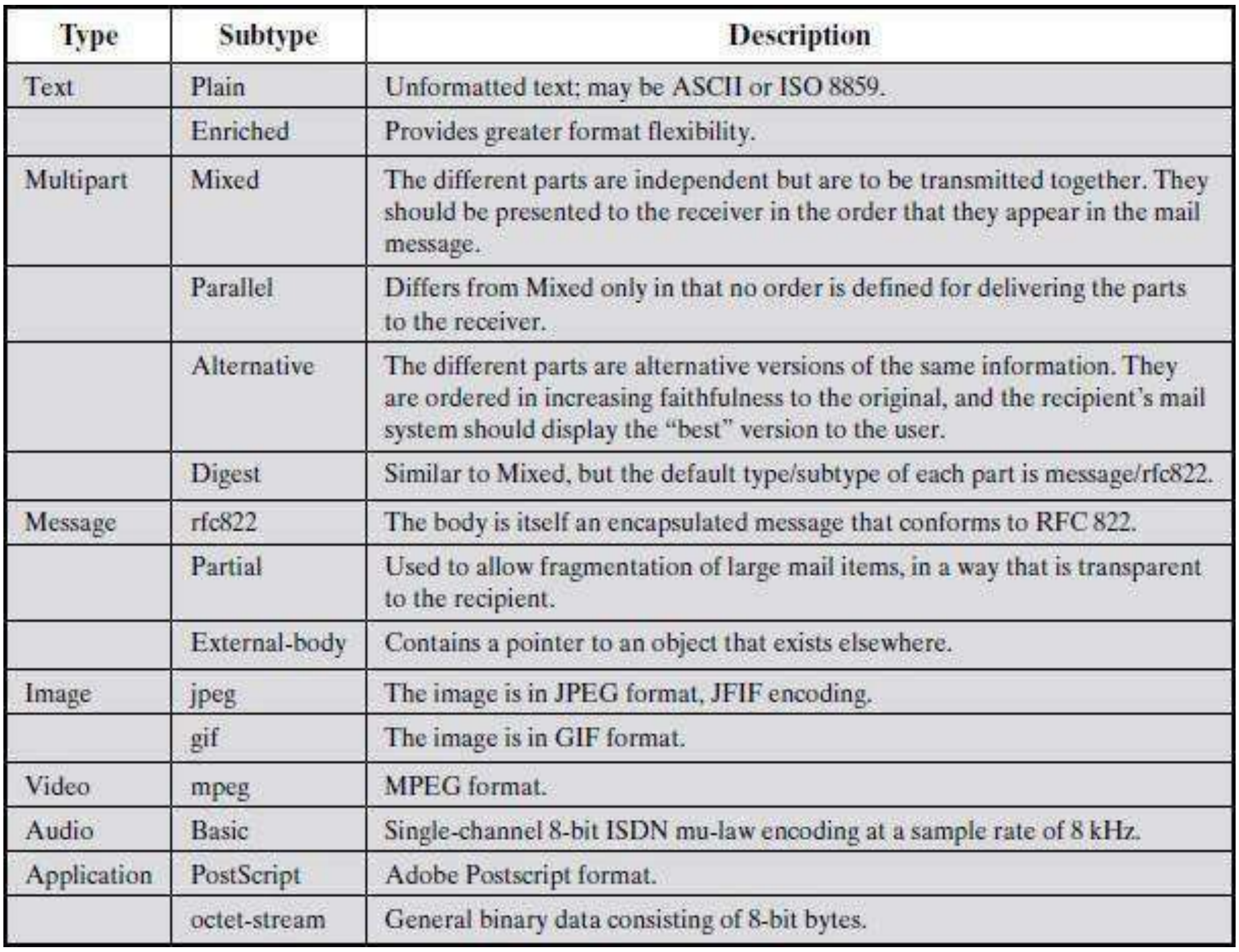
The MIME specification includes the following elements.

1. **Five new message header fields** are defined, which may be included in an RFC 5322 header. These fields provide information about the body of the message.
2. A number of content formats are defined, thus standardizing representations that support multimedia electronic mail.
3. Transfer encodings are defined that enable the conversion of any content format into a form that is protected from alteration by the mail system.

The Five Header Fields Defined in MIME: The five header fields defined in MIME are

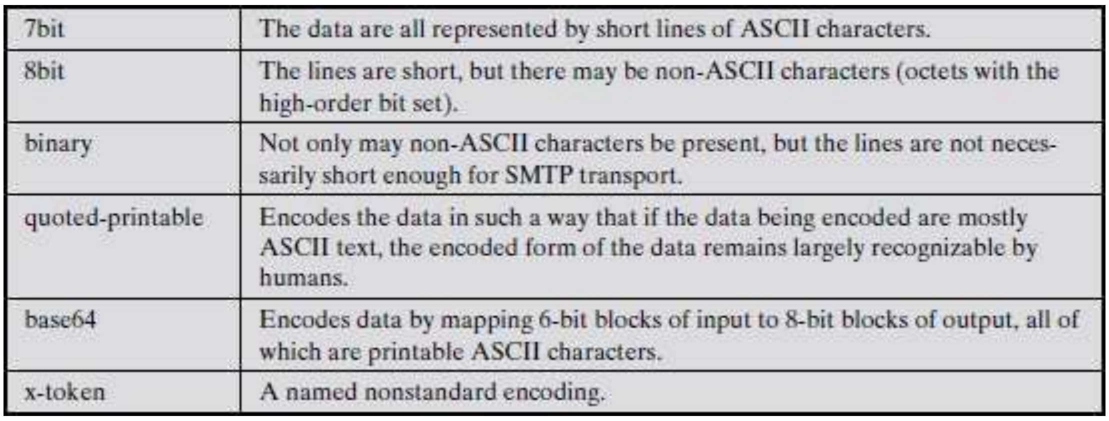
- **MIME-Version:** Must have the parameter value 1.0. This field indicates that the message conforms to RFCs 2045 and 2046.
- **Content-Type:** Describes the data contained in the body with sufficient detail that the receiving user agent can pick an appropriate agent or mechanism to represent the data to the user or otherwise deal with the data in an appropriate manner.
- **Content-Transfer-Encoding:** Indicates the type of transformation that has been used to represent the body of the message in a way that is acceptable for mail transport.
- **Content-ID:** Used to identify MIME entities uniquely in multiple contexts.
- **Content-Description:** A text description of the object with the body; this is useful when the object is not readable (e.g., audio data).

MIME Content Types:



Type	Subtype	Description
Text	Plain	Unformatted text; may be ASCII or ISO 8859.
	Enriched	Provides greater format flexibility.
Multipart	Mixed	The different parts are independent but are to be transmitted together. They should be presented to the receiver in the order that they appear in the mail message.
	Parallel	Differs from Mixed only in that no order is defined for delivering the parts to the receiver.
	Alternative	The different parts are alternative versions of the same information. They are ordered in increasing faithfulness to the original, and the recipient's mail system should display the "best" version to the user.
	Digest	Similar to Mixed, but the default type/subtype of each part is message/rfc822.
Message	rfc822	The body is itself an encapsulated message that conforms to RFC 822.
	Partial	Used to allow fragmentation of large mail items, in a way that is transparent to the recipient.
	External-body	Contains a pointer to an object that exists elsewhere.
Image	jpeg	The image is in JPEG format, JFIF encoding.
	gif	The image is in GIF format.
Video	mpeg	MPEG format.
Audio	Basic	Single-channel 8-bit ISDN mu-law encoding at a sample rate of 8 kHz.
Application	PostScript	Adobe Postscript format.
	octet-stream	General binary data consisting of 8-bit bytes.

MIME Transfer Encodings:



7bit	The data are all represented by short lines of ASCII characters.
8bit	The lines are short, but there may be non-ASCII characters (octets with the high-order bit set).
binary	Not only may non-ASCII characters be present, but the lines are not necessarily short enough for SMTP transport.
quoted-printable	Encodes the data in such a way that if the data being encoded are mostly ASCII text, the encoded form of the data remains largely recognizable by humans.
base64	Encodes data by mapping 6-bit blocks of input to 8-bit blocks of output, all of which are printable ASCII characters.
x-token	A named nonstandard encoding.

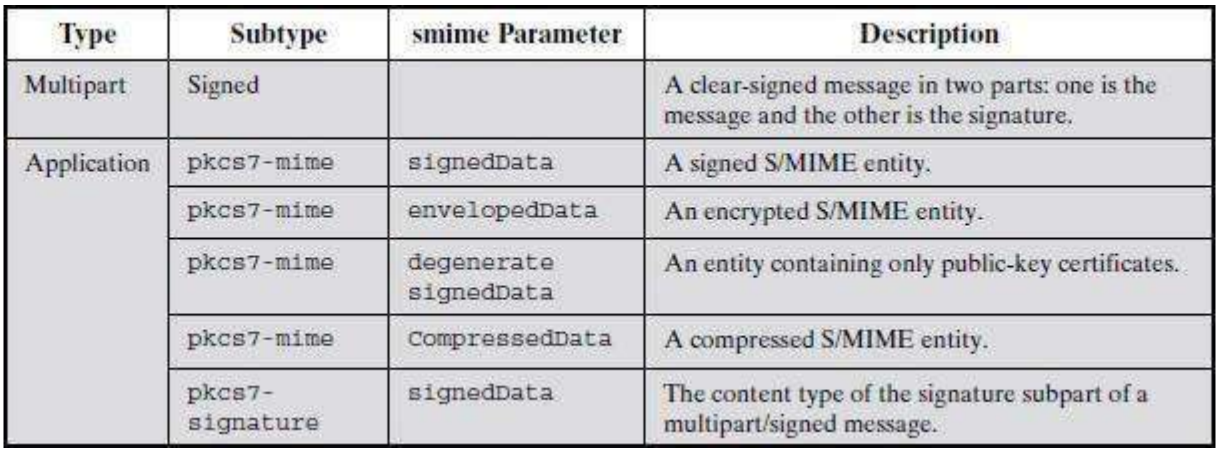
S/MIME Functionality: S/MIME provides the following functions.

- **Enveloped data:** This consists of encrypted content of any type and encrypted content encryption keys for one or more recipients.
- **Signed data:** A digital signature is formed by taking the message digest of the content to be signed and then encrypting that with the private key of the signer. The content plus signature are then encoded using base64 encoding. A signed data message can only be viewed by a recipient with S/MIME capability.
- **Clear-signed data:** As with signed data, a digital signature of the content is formed. However, in this case, only the digital signature is encoded using base64. As a result, recipients without S/MIME capability can view the message content, although they cannot verify the signature.
- **Signed and enveloped data:** Signed-only and encrypted-only entities may be nested, so that encrypted data may be signed and signed data or clear - signed data may be encrypted.

S/MIME Messages:

- S/MIME secures a MIME entity with a signature, encryption, or both. forming a MIME wrapped
  Public-Key Cryptography Standards(PKCS) object have a range of content-types:
  enveloped data o signed data, clear-signed data o registration request,certificate only message

  S/MIME Content Types



Type	Subtype	smime Parameter	Description
Multipart	Signed		A clear-signed message in two parts: one is the message and the other is the signature.
Application	pkcs7-mime	signedData	A signed S/MIME entity.
	pkcs7-mime	envelopedData	An encrypted S/MIME entity.
	pkcs7-mime	degenerate signedData	An entity containing only public-key certificates.
	pkcs7-mime	CompressedData	A compressed S/MIME entity.
	pkcs7-signature	signedData	The content type of the signature subpart of a multipart/signed message.

S/MIME Certificate Processing:

- S/MIME uses public-key certificates that conform to version 3 of X.509
- The key-management scheme used by S/MIME is in some ways a hybrid between a strict X.509 certification hierarchy and PGP's web of trust
- S/MIME managers and/or users must configure each client with a list of trusted keys and with certificate revocation lists.
  The responsibility is local for maintaining the certificates needed to verify incoming signatures and to encrypt outgoing messages
- The certificates are signed by certification authorities

User Agent Role An S/MIME user has several key-management functions to perform

• **Key generation:** The user of some related administrative utility (e.g., one associated with LAN management) MUST be capable of generating separate Diffie-Hellman and DSS key pairs and SHOULD be capable of generating RSA key pairs. Each key pair MUST be generated from a good source of nondeterministic random input and be protected in a secure fashion. A use agent SHOULD generate RSA key pairs with a length in the range of 768 to 1024 bits and MUST NOT generate a length of less than 512 bits.
• **Registration:** A user's public key must be registered with a certification authority in order to receive an X.509 public-key certificate.
• **Certificate storage and retrieval:** A user requires access to a local list of certificates in order to verify incoming signatures and to encrypt outgoing messages. Such a list could be maintained by the user or by some local administrative entity on behalf of a number of users.

VeriSign Certificates There are several companies that provide certification authority (CA) services. For example, Nortel has designed an enterprise CA solution and can provide S/MIME support within an organization. There are a number of Internet-based CAs, including VeriSign, GTE, and the U.S. Postal Service.

Enhanced Security Services : three enhanced security services have been proposed in an Internet draft. The three services are : **Signed receipts, Security labels, Secure mailing lists**

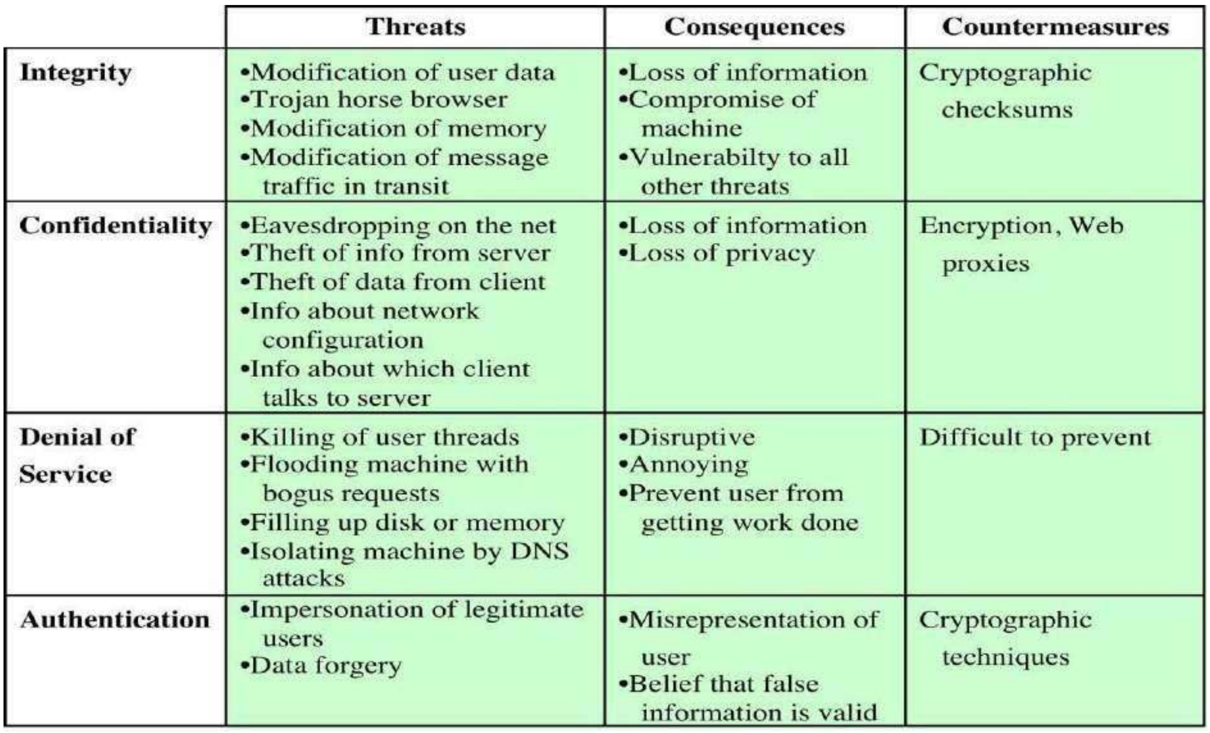
Transport Level Security:

Web security considerations:

The World Wide Web is fundamentally a client/server application running over the Internet and TCP/IP intranets
The following characteristics of Web usage suggest the need for tailored security tools:

- The Internet is two-way. Unlike traditional publishing environments—even electronic publishing systems involving teletext, voice response, or fax-back— the Web is vulnerable to attacks on the Web servers over the Internet.
- The Web is increasingly serving as a highly visible outlet for corporate and product information and as the platform for business transactions. Reputations can be damaged and money can be lost if the Web servers are subverted.
- Although Web browsers are very easy to use, Web servers are relatively easy to configure and manage, and Web content is increasingly easy to develop, the underlying software is extraordinarily complex. This complex software may hide many potential security flaws.
- A Web server can be exploited as a launching pad into the corporation's or agency's entire computer complex. Once the Web server is subverted, an attacker may be able to gain access to data and systems not part of the Web itself but connected to the server at the local site.
- Casual and untrained (in security matters) users are common clients for Web-based services. Such users are not necessarily aware of the security risks that exist and do not have the tools or knowledge to take effective countermeasures.

Web security Threats:



	Threats	Consequences	Countermeasures
Integrity	•Modification of user data •Trojan horse browser •Modification of memory •Modification of message traffic in transit	•Loss of information •Compromise of machine •Vulnerabilty to all other threats	Cryptographic checksums
Confidentiality	•Eavesdropping on the net •Theft of info from server •Theft of data from client •Info about network configuration •Info about which client talks to server	•Loss of information •Loss of privacy	Encryption, Web proxies
Denial of Service	•Killing of user threads •Flooding machine with bogus requests •Filling up disk or memory •Isolating machine by DNS attacks	•Disruptive •Annoying •Prevent user from getting work done	Difficult to prevent
Authentication	•Impersonation of legitimate users •Data forgery	•Misrepresentation of user •Belief that false information is valid	Cryptographic techniques

Table. A Comparison of Threats on the Web

Web Traffic Security Approaches:

A number of approaches to providing Web security are possible.

1. One way to provide Web security is to use **IP security** (IPsec) (Figure(a)). The advantage of using IPsec is that it is transparent to end users and applications and provides a general-purpose solution. It includes filtering capability that filters the unwanted data.
2. Another relatively general-purpose solution is to implement security just above TCP (Figure (b)). The example of this approach is the **Secure Sockets Layer (SSL)** and the follow-on Internet standard known as **Transport Layer Security (TLS).** At this level, there are two implementation choices. For full generality, SSL (or TLS) could be provided as part of the underlying protocol suite and therefore be transparent to applications. Alternatively, SSL can be embedded in specific packages. For example, Netscape and Microsoft Explorer browsers come equipped with SSL, and most Web servers have implemented the protocol.
3. Application-specific security services are embedded within the particular application. Figure (c) shows examples of this architecture. The advantage of this approach is that the service can be tailored to the specific needs of a given application.

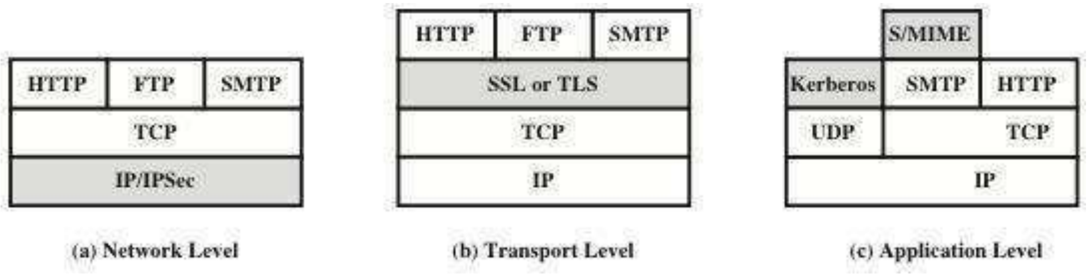


Figure: relative location of security facilities in the TCP/IP Protocol stack **5.4.**

SSL (Secure Socket Layer):

SSL probably most widely used Web security mechanism, and it is implemented at the Transport layer.

SSL is designed to make use of TCP to provide a reliable end-to-end secure service.

Netscape originated SSL. Version 3 of the protocol was designed with public review and input from industry and was published as an Internet draft document. Subsequently, became Internet standard known as TLS (Transport Layer Security)

SSL Architecture:

- SSL is designed to make use of TCP to provide a reliable end-to-end
- secure service. SSL is not a single protocol but rather two layers of protocols.

Two important SSL concepts are the SSL session and the SSL connection, which are defined in the specification as follows.

1. **Connection:** A connection is a transport that provides a suitable type of service. For SSL, such connections are peer-to-peer relationships. Every connection is associated with one session.
2. **Session:** An SSL session is an association between a client and a server. Sessions are created by the Handshake Protocol. Sessions define a set of cryptographic security parameters which can be shared among multiple connections.

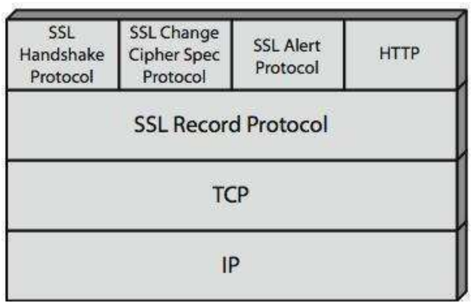


Figure: SSL Protocol stack

SSL Record Protocol:

SSL Record Protocol defines two services for SSL connections:

1. **Confidentiality:** The Handshake Protocol defines a shared secret key that is used for conventional encryption of SSL payloads. The message is compressed before being concatenated with the MAC and encrypted, with a range of ciphers being supported as shown.
2. **Message Integrity:** The Handshake Protocol also defines a shared secret key that is used to form a message authentication code (MAC).

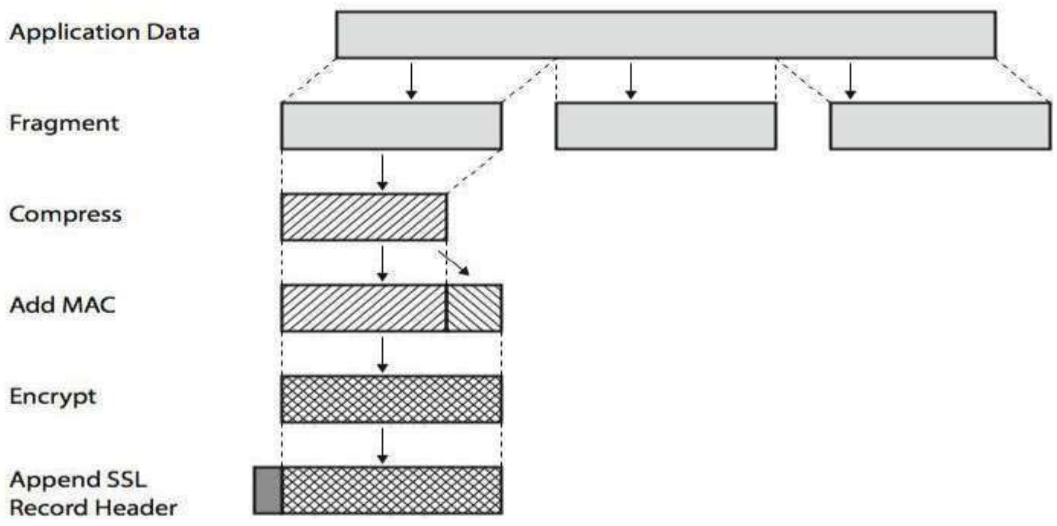


Figure: SSL Record Protocol Operation

Figure shows the overall operation of the SSL Record Protocol. The Record Protocol takes an application message to be transmitted, fragments the data into manageable blocks, optionally compresses the data, applies a MAC, encrypts, adds a header, and transmits the resulting unit in a TCP segment. Received data are decrypted, verified, decompressed, and reassembled before being delivered to higher-level users.

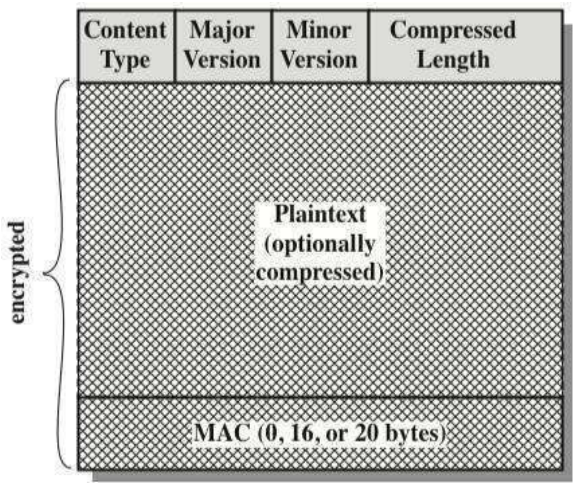


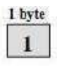
Figure: SSL Record Format

The final step of SSL Record Protocol processing is to prepare a header consisting of the following fields:

- **Content Type (8 bits):** The higher-layer protocol used to process the enclosed fragment.
- **Major Version (8 bits):** Indicates major version of SSL in use. For SSLv3, the value is 3.
- **Minor Version (8 bits):** Indicates minor version in use. For SSLv3, the value is 0.
- **Compressed Length (16 bits):** The length in bytes of the plaintext fragment (or compressed fragment if compression is used). The maximum value is $2^{14} + 2048$.

ChangeCipherSpec Protocol:

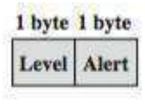
The Change Cipher Spec Protocol is one of the three SSL-specific protocols that use the SSL Record Protocol. It is the simplest, consisting of a single message, which consists of a single byte with the value 1. The sole purpose of this message is to cause the pending state to be copied into the current state, which updates the cipher suite to be used on this connection.



(a) Change Cipher Spec Protocol

SSL Alert Protocol:

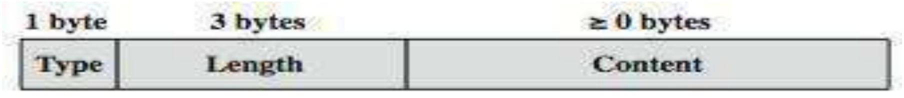
The Alert Protocol is used to convey SSL-related alerts to the peer entity. As with other applications that use SSL, alert messages are compressed and encrypted, as specified by the current state. Each message in this protocol consists of two bytes, the first takes the value warning (1) or fatal (2) to convey the severity of the message. The second byte contains a code that indicates the specific alert.



(b) Alert Protocol

SSL Handshake Protocol:

The most complex part of SSL is the Handshake Protocol. This protocol allows the server and client to authenticate each other and to negotiate an encryption and MAC algorithm and cryptographic keys to be used to protect data sent in an SSL record. The Handshake Protocol is used before any application data is transmitted. The Handshake Protocol consists of a series of messages exchanged by client and server.



(c) Handshake Protocol

The exchange can be viewed in 4 phases:

- **Phase 1. Establish Security Capabilities** - this phase is used by the client to initiate a logical connection and to establish the security capabilities that will be associated with it
- **Phase 2. Server Authentication and Key Exchange** - the server begins this phase by sending its certificate if it needs to be authenticated.
- **Phase 3. Client Authentication and Key Exchange** - the client should verify that the server provided a valid certificate if required and check that the server_hello parameters are acceptable
- **Phase 4. Finish** - this phase completes the setting up of a secure connection. The client sends a change_cipher_spec message and copies the pending CipherSpec into the current CipherSpec. At this point the handshake is complete and the client and server may begin to exchange application layer data.

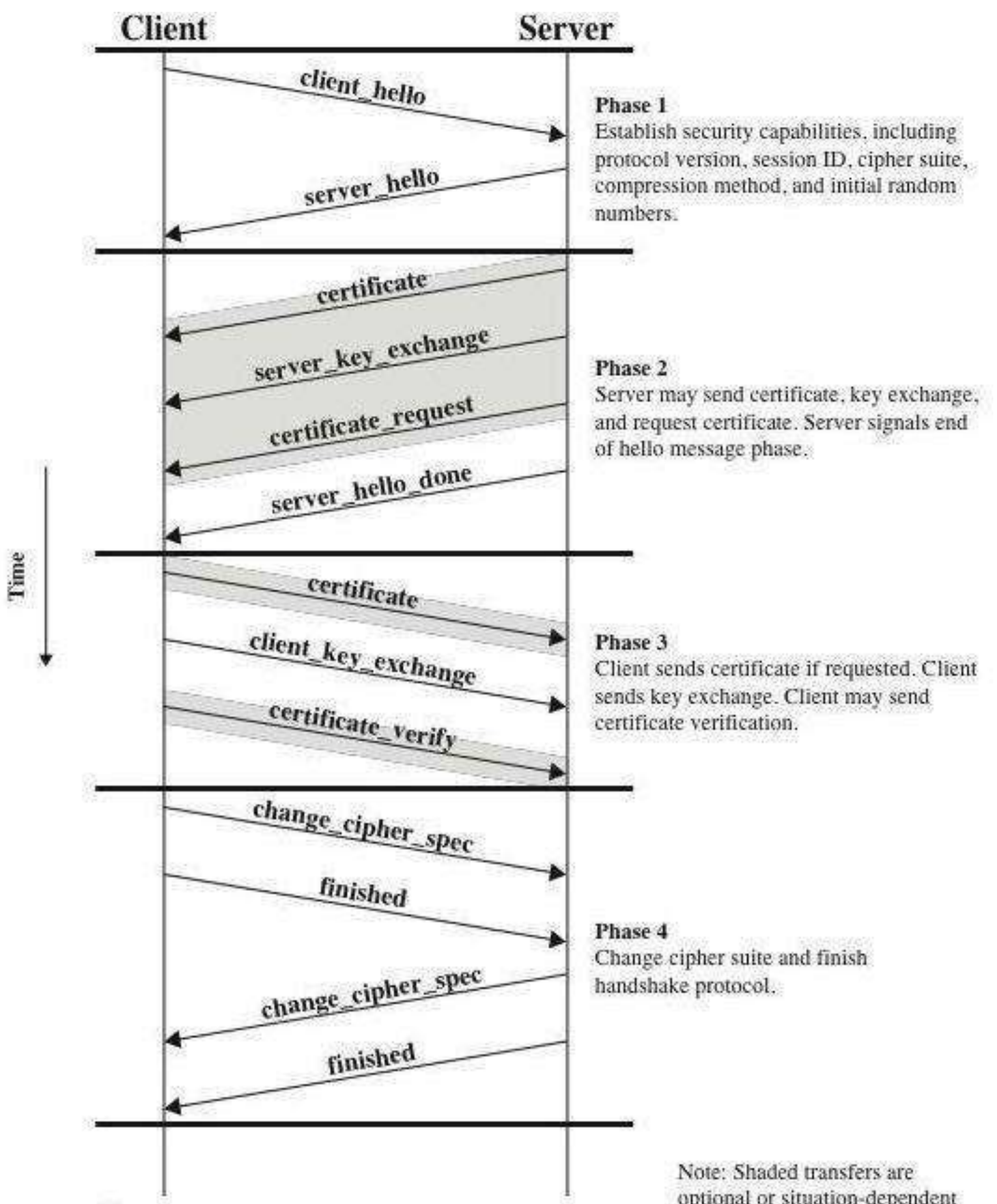

Client
Server
client_hello
server_hello
certificate
server_key_exchange
certificate_request
server_hello_done
certificate
client_key_exchange
certificate_verify
change_cipher_spec
finished
change_cipher_spec
finished
Time
Phase 1
Establish security capabilities, including protocol version, session ID, cipher suite, compression method, and initial random numbers.
Phase 2
Server may send certificate, key exchange, and request certificate. Server signals end of hello message phase.
Phase 3
Client sends certificate if requested. Client sends key exchange. Client may send certificate verification.
Phase 4
Change cipher suite and finish handshake protocol.

Figure Handshake Protocol Action

Note: Shaded transfers are optional or situation-dependent messages that are not always sent.

Transport Layer Security (TLS) Protocol

In order to provide an open Internet standard of SSL, Internet Engineering Task Force(IETF) released The Transport Layer Security (TLS) protocol in January 1999. TLS is defined as a proposed Internet Standard in RFC 5246.

Salient Features

TLS protocol has same objectives as SSL.

It enables client/server applications to communicate in a secure manner by authenticating, preventing eavesdropping and resisting message modification.
TLS protocol sits above the reliable connection-oriented transport TCP layer in the networking layers' stack.
The architecture of TLS protocol is similar to SSLv3 protocol. It has two sub protocols: the TLS Record protocol and the TLS Handshake protocol.

Though SSLv3 and TLS protocol have similar architecture, several changes were made in architecture and functioning particularly for the handshake protocol.

Comparison of TLS and SSL Protocols:

1. **Protocol Version −** The header of TLS protocol segment carries the version number 3.1 todifferentiate between number 3 carried by SSL protocol segment header.
2. **Message Authentication −** TLS employs a keyed-hash message authentication code (HMAC). Benefit is that H-MAC operates with any hash function, not just MD5 or SHA, as explicitly stated by the SSL protocol.
3. **Session Key Generation −** There are two differences between TLS and SSL protocol for generation of key material.
   1. Method of computing pre-master and master secrets is similar. But in TLS protocol, computation of master secret uses the HMAC standard and pseudorandom function (PRF) output instead of ad-hoc MAC.
   2. The algorithm for computing session keys and initiation values (IV) is different in TLS than SSL protocol.
4. **Alert Protocol Message −**
   1. TLS protocol supports all the messages used by the Alert protocol of SSL, except *No certificate* alert message being made redundant. The client sends empty certificate in case client authentication is not required.
   2. Many additional Alert messages are included in TLS protocol for other error conditions such as
      *record_overflow, decode_error*etc.
5. **Supported Cipher Suites −** SSL supports RSA, Diffie-Hellman and Fortezza cipher suites. TLS protocol supports all suits except Fortezza.
6. **Client Certificate Types −** TLS defines certificate types to be requested in a *certificate_request* message. SSLv3 support all of these. Additionally, SSL support certain other types of certificate such as Fortezza.
7. **CertificateVerify and Finished Messages −**

1. In SSL, complex message procedure is used for the*certificate_verify* message. With TLS, the verified information is contained in the handshake messages itself thus avoiding this complex procedure.
2. Finished message is computed in different manners in TLS and SSLv3.

8. **Padding of Data −** In SSL protocol, the padding added to user data before encryption is the minimum amount required to make the total data-size equal to a multiple of the cipher's block length. In TLS, the padding can be any amount that results in data-size that is a multiple of the cipher's block length, up to a maximum of 255 bytes.

Secure Shell Protocol (SSH):

The salient features of SSH are as follows −

- SSH is a network protocol that runs on top of the TCP/IP layer. It is designed to replace the TELNET which provided unsecure means of remote logon facility.
- SSH provides a secure client/server communication and can be used for tasks such as file transfer and e-mail.
- SSH2 is a prevalent protocol which provides improved network communication security over earlier version SSH1.

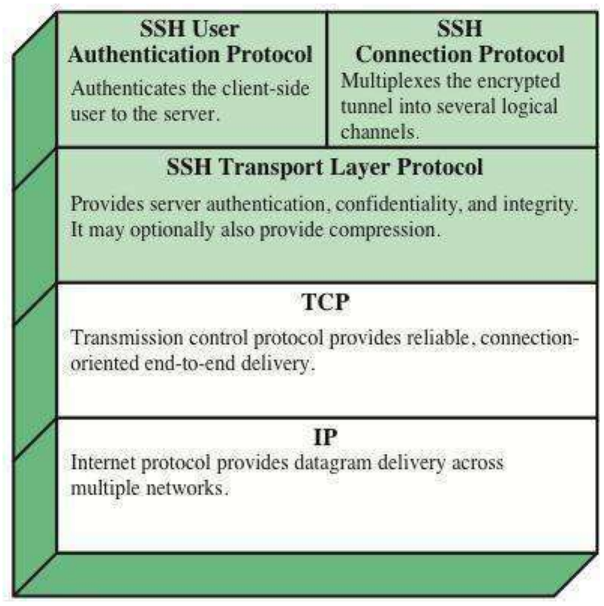

SSH User Authentication Protocol
Authenticates the client-side user to the server.
SSH Connection Protocol
Multiplexes the encrypted tunnel into several logical channels.
SSH Transport Layer Protocol
Provides server authentication, confidentiality, and integrity. It may optionally also provide compression.
TCP
Transmission control protocol provides reliable, connection-oriented end-to-end delivery.
IP
Internet protocol provides datagram delivery across multiple networks.

Figure: SSH Protocol stack

Transport Layer Protocol:
In this part of SSH protocol provides data confidentiality, server (host) authentication, and data integrity. It may optionally provide data compression as well.

- Server Authentication − Host keys are asymmetric like public/private keys. A server uses a public
key to prove its identity to a client. The client verifies that contacted server is a ―known‖ host from the database it maintains. Once the server is authenticated, session keys are generated.

- Session Key Establishment − After authentication, the server and the client agree upon cipher to be
used. Session keys are generated by both the client and the server. Session keys are generated before user authentication so that usernames and passwords can be sent encrypted. These keys are generally replaced at regular intervals (say, every hour) during the session and are destroyed immediately after use.
- Data Integrity − SSH uses Message Authentication Code (MAC) algorithms to for data integrity check. It is an improvement over 32 bit CRC used by SSH1.

User Authentication Protocol:

In this part of SSH authenticates the user to the server. The server verifies that access is given to intended users only. Many authentication methods are currently used such as, typed passwords, Kerberos, public-key authentication, etc.

Connection Protocol:

This provides multiple logical channels over a single underlying SSH connection

SSH Services:

SSH provides three main services that enable provision of many secure solutions. These services are briefly described as follows −

Secure Command-Shell (Remote Logon) − It allows the user to edit files, view the contents of directories, and access applications on connected device. Systems administrators can remotely start/view/stop services and processes, create user accounts, and change file/directories permissions and so on. All tasks that are feasible at a machine’s command prompt can now be performed securely from the remote machine using secure remote logon.

Secure File Transfer − SSH File Transfer Protocol (SFTP) is designed as an extension for SSH-2 for secure file transfer. In essence, it is a separate protocol layered over the Secure Shell protocol to handle file transfers. SFTP encrypts both the username/password and the file data being transferred. It uses the same port as the Secure Shell server, i.e. system port no 22.

Port Forwarding (Tunneling) − It allows data from unsecured TCP/IP based applications to be secured. After port forwarding has been set up, Secure Shell reroutes traffic from a program (usually a client) and sends it across the encrypted tunnel to the program on the other side (usually a server). Multiple applications can transmit data over a single multiplexed secure channel, eliminating the need to open many ports on a firewall or router.

UNIT -VI:
Network Security-II : Security at the Network Layer: IPSec, System Security

1. IP SECURITY OVERVIEW

IPSec is an Internet Engineering Task Force (IETF) standard suite of protocols that provides data authentication, integrity, and confidentiality as data is transferred between communication points across IP networks.
IPSec provides data security at the IP packet level. A packet is a data bundle that is organized for transmission across a network, and it includes a header and payload (the data in the packet).

IPSec SECURITY FEATURES:

IPSec is the most secure method commercially available for connecting network sites.
IPSec was designed to provide the following security features when transferring packets across networks:
Authentication: Verifies that the packet received is actually from the claimed sender.
Integrity: Ensures that the contents of the packet did not change in transit.
Confidentiality: Conceals the message content through encryption.

IPSec ELEMENTS:

IPSec contains the following elements:
Encapsulating Security Payload (ESP): Provides confidentiality, authentication, and integrity.
Authentication Header (AH): Provides authentication and integrity.
Internet Key Exchange (IKE): Establish shared symmetric key. Provides key management and Security Association (SA) management.

APPLICATIONS OF IPSec:

IPSec provides the capability to secure communications across a LAN, across private and public WANs, and across the Internet.
Examples of its use include the following:

- Secure branch office connectivity over the Internet
- Secure remote access over the Internet

Establishing extranet and intranet connectivity with partners:

- IPSec can be used to secure communication with other organizations, ensuring authentication and confidentiality and providing a key exchange mechanism.

Enhancing electronic commerce security:

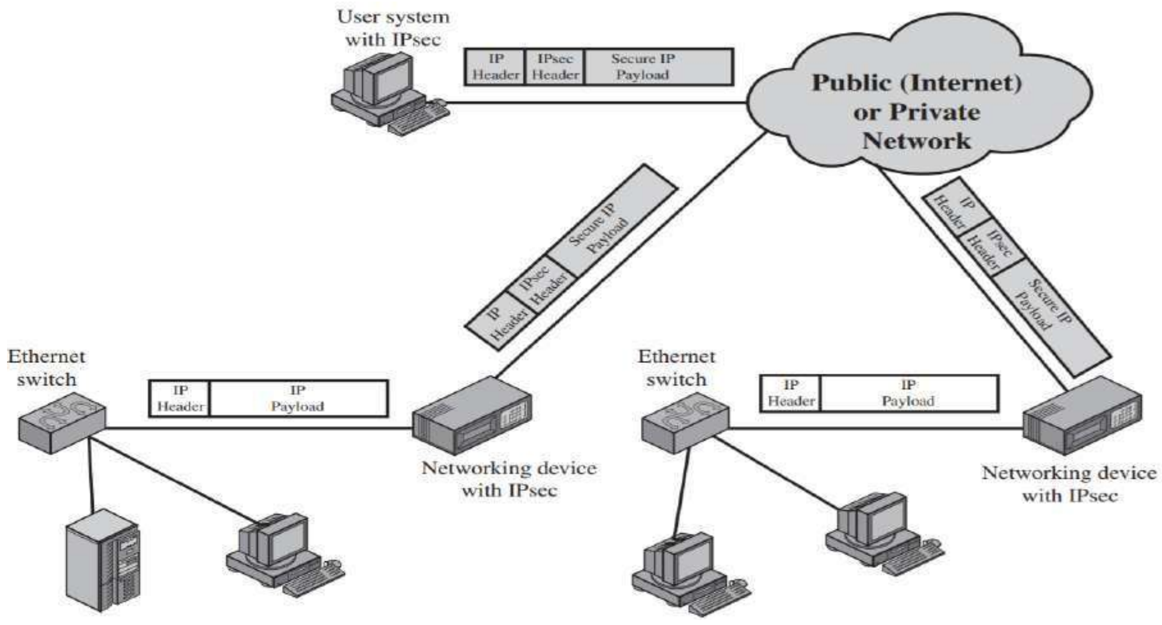
- Even though some Web and electronic commerce applications have built-in security protocols, the use of IPSec enhances that security.

BENEFITS OF IPSEC:

- IPSec provides strong security within and across the LANs.
- Firewall uses IPSec to restrict all those incoming packets which are not using IP. Since firewall is the only way to enter into an organization, restricted packets cannot enter.
- IPSec is below the transport layer (TCP, UDP) and so is transparent to applications.
- There is no need to change software on a user or server system when IPSec is implemented in the firewall or router.
- Even if IPSec is implemented in end systems, upper- layer software, including applications, is not affected. IPSec can be transparent to end users.

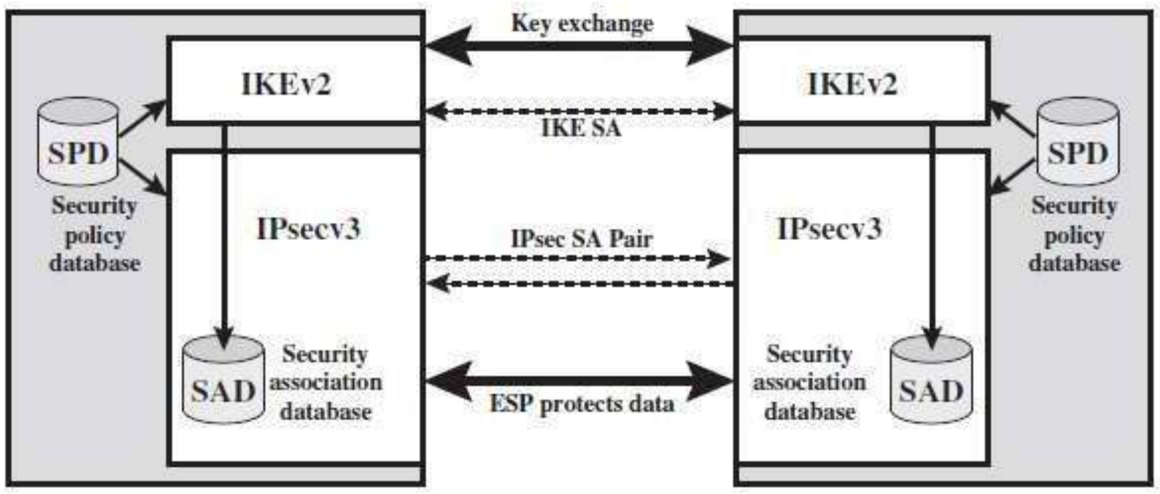
- IPSec can provide security for individual users if needed.

IPSec Scenario:


User system with IPsec
IP Header | IPsec Header | Secure IP Payload
Public (Internet) or Private Network
IP Header | IPsec Header | Secure IP Payload
IP Header | IPsec Header | Secure IP Payload
Ethernet switch
IP Header | IP Payload
Networking device with IPsec
Ethernet switch
IP Header | IP Payload
Networking device with IPsec

IPSec Architecture:

Architecture covers general concepts of security requirements, definitions, and mechanisms defining IPSec technology.


Key exchange
IKEv2
IKEv2
IKE SA
SPD
Security policy database
SPD
Security policy database
IPsecv3
IPsecv3
IPsec SA Pair
SAD
Security association database
Security association database
SAD
ESP protects data