Module 4

Network Security: Electronic Mail Security, Pretty Good Privacy, S/MIME, IP Security Overview, IP Security Architecture, Authentication Header, Encapsulating Security Payload.

Web Security: Web Security considerations- Secure Socket Layer -Transport layer Security- Secure electronic transaction. Firewalls-Packet filters- Application Level Gateway- Circuit Level Gateway.

Electronic Mail Security

To provide authentication and confidentiality in e-mail we use two methods

- Pretty Good Privacy
- S/MIME

Pretty Good Privacy (PGP)

PGP provides a confidentiality and authentication service that can be used for electronic mail and file storage applications. This is the effort of a single person Zimmerman.

PGP has grown explosively and is now widely used. A number of reasons can be cited for this growth.

- It is available free worldwide in versions that run on a variety of platforms, including Windows, UNIX, Macintosh, and many more.
- It is based on algorithms that have survived extensive public review and are considered extremely secure. Specifically, the package includes RSA, DSS, and Diffie-Hellman for public-key encryption; CAST-128, IDEA, and 3DES for symmetric encryption; and SHA-1 for hash coding.
- It has a wide range of applicability, from corporations that wish to select and enforce a standardized scheme for encrypting files and messages to individuals who wish to communicate securely with others worldwide over the Internet and other networks.
- It was not developed by, nor is it controlled by, any governmental or standards organization.

- PGP is now on an Internet standards track (RFC 3156).

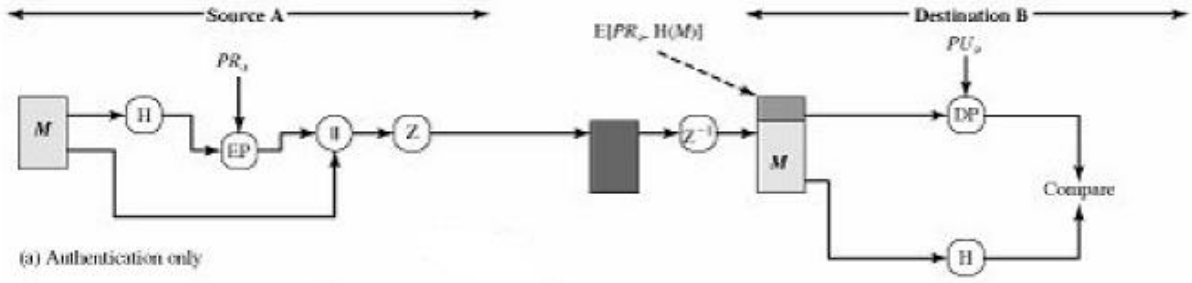
The actual operation of PGP consists of five operations

- Authentication
- Confidentiality
- Compression
- E-mail compatibility
- Segmentation

Authentication

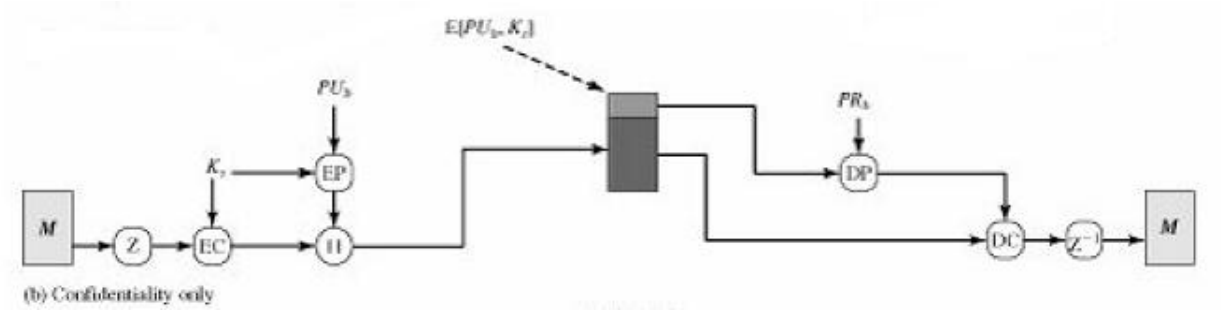
The sequence in authentication is as follows

- The sender creates a message.
- SHA-1 is used to generate a 160-bit hash code of the message.
- The hash code is encrypted with RSA using the sender's private key, and the result is prepended to the message.
- The receiver uses RSA with the sender's public key to decrypt and recover the hash code.
- The receiver generates a new hash code for the message and compares it with the decrypted hash code. If the two match, the message is accepted as authentic.



The combination of SHA-1 and RSA provides an effective digital signature scheme. Because of the strength of RSA, the recipient is assured that only the possessor of the matching private key can generate the signature. Because of the strength of SHA-1, the recipient is assured that no one else could generate a new message that matches the hash code and, hence, the signature of the original message.

Confidentiality

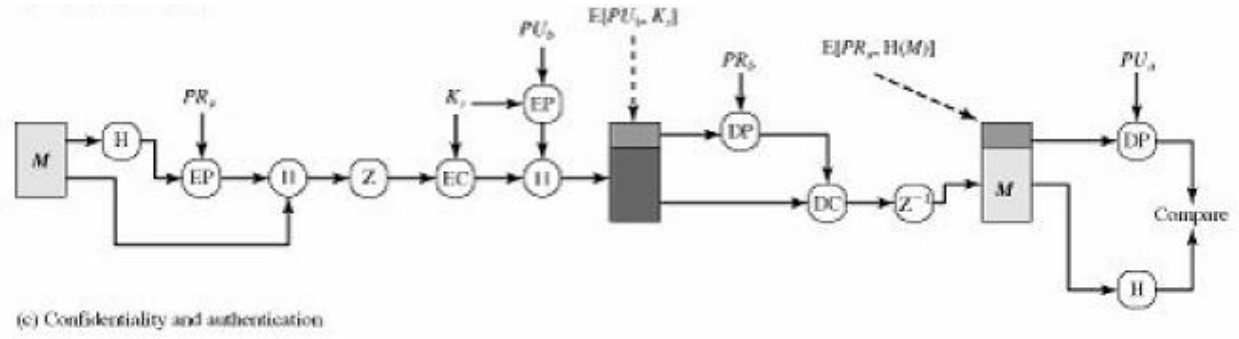


(b) Confidentiality only

Another basic service provided by PGP is confidentiality, which is provided by encrypting messages to be transmitted or to he stored locally as files. In both cases, the symmetric encryption algorithm CAST-128 may be used. The sequence can be described as follows.

- The sender generates a message and a random 128-bit number to be used as a session key for this message only.
- The message is encrypted, using CAST-128 (or IDEA or 3DES) with the session key.
- The session key is encrypted with RSA, using the recipient's public key, and is prepended to the message.
- The receiver uses RSA with its private key to decrypt and recover the session key.
- The session key is used to decrypt the message.

Confidentiality and Authentication



(c) Confidentiality and authentication

Both services may be used for the same message. First, a signature is generated for the plaintext message and prepended to the message. Then the plaintext message plus signature is encrypted using CAST-128 (or IDEA or 3DES), and the session key is encrypted using RSA. This

sequence is preferable to the opposite: encrypting the message and then generating a signature for the encrypted message. It is generally more convenient to store a signature with a plaintext version of a message. Furthermore, for purposes of third-party verification, if the signature is performed first, a third party need not be concerned with the symmetric key when verifying the signature. In summary, when both services are used, the sender first signs the message with its own private key, then encrypts the message with a session key, and then encrypts the session key with the recipient's public key.

In summary, when both services are used, the sender first signs the message with its own private key, then encrypts the message with a session key, and then encrypts the session key with the recipient's public key. 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.

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 is critical:

The signature is generated before compression for two reasons:

- It is preferable to sign an uncompressed message so that one can store only the uncompressed message together with the signature for future verification If one signed a compressed document, then it would be necessary either to store a compressed version of the message for later verification or to recompress the message when verification is required.

- Even if one were willing to generate dynamically a recompressed message for verification, PGP's compression algorithm presents a difficulty. The algorithm is not deterministic; various implementations of the algorithm achieve different tradeoffs in running speed versus compression ratio and, as a result, produce different compressed

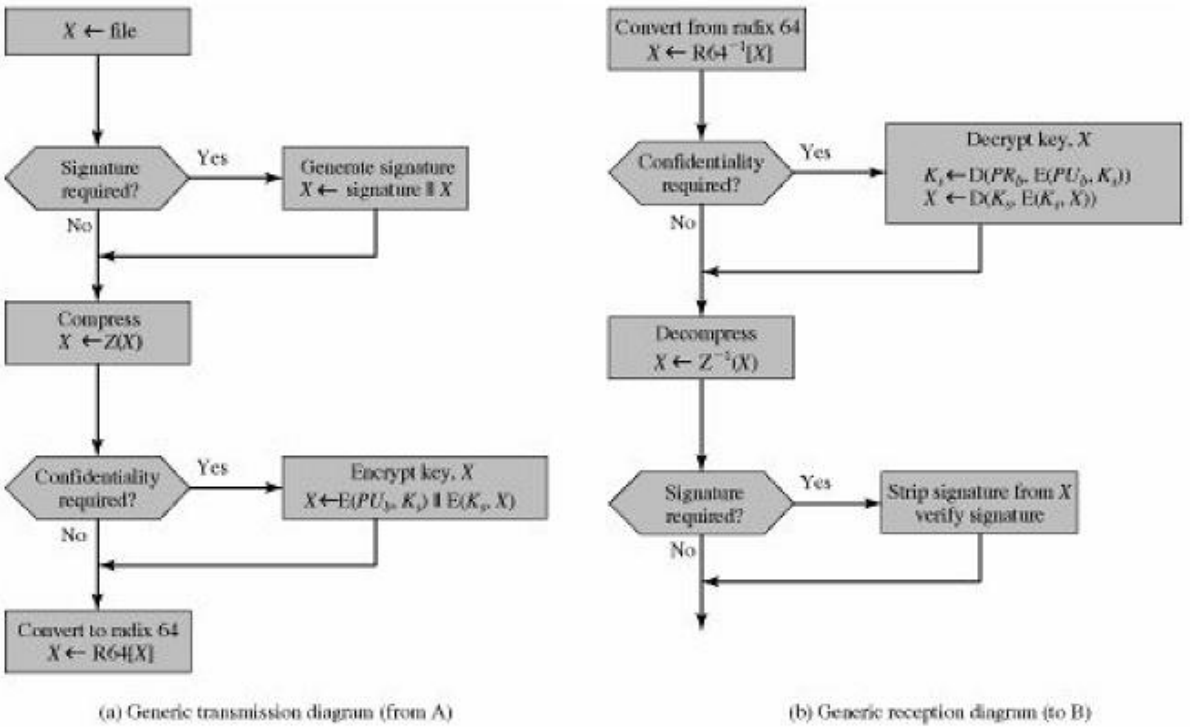
forms. However, these different compression algorithms are interoperable because any version of the algorithm can correctly decompress the output of any other version. Applying the hash function and signature after compression would constrain all PGP implementations to the same version of the compression algorithm.

Message encryption is applied after compression to strengthen cryptographic security. Because the compressed message has less redundancy than the plaintext, cryptanalysis is more difficult.

E-mail 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 signatures message plus signature (if present) are encrypted (with a one-time symmetric key).

Figure 15.2. Transmission and Reception of PGP Messages



(a) Generic transmission diagram (from A)

(b) Generic reception diagram (to B)

Thus, part or the entire 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 for 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 keys is octets of binary data are mapped into four ASCII characters. This format also appends a CRC to detect transmission errors. The relationship among the four operations is shown in figure.

Segmentation and Reassembly

E-mail facilities often are restricted to a maximum message length. For example, many of the facilities accessible through the Internet impose a maximum length of 50,000 octets. Any message longer than that must he broken up into smaller segments, each of which is mailed separately.

To accommodate this restriction, PGP automatically subdivides a message that is too large into segments that are small enough to send via e-mail. The segmentation done after all of the other processing, including the radix-64 conversion. Thus, the session key component and signature component appear only once, at the beginning of the first segment. At the receiving end, PGP must strip off all e-mail headers and reassemble the entire original block.

Cryptographic Keys and Key Rings

PGP makes use of four types of keys: one-time session symmetric keys, public keys private keys, and pass phrase-based symmetric keys. Three separate requirements can be identified with respect to these keys:

- A means of generating unpredictable session keys is needed.
- We would like to allow a user to have multiple public-key/private-key pairs. One reason is that the user may wish to change his or her key pair from time to time. When this happens, any messages in the pipeline will be constructed with an obsolete key. Furthermore, recipients will know only the old public key until an update reaches them. In addition to the need to change keys over time, a user may wish to have multiple key pairs at a given time to interact with different groups of correspondents or simply to enhance security by limiting the amount of material encrypted with any one key. The upshot of all this is that there is not a one-to-one correspondence between users and their public keys. Thus, some means is needed for identifying particular keys.

- Each PGP entity must maintain a file of its own public/private key pairs well as a file of public keys of correspondents.

Session Key Generation

Each session key is associated with a single message and is used only for the purpose of encrypting and decrypting that message.

Random 128 bit numbers are generated using CAST 128 itself. The input to the random number generator consists of a 128-bit key and two 64-bit blocks that are treated as plaintext to be encrypted. Using cipher feedback mode the CAST-128 encrypter produces two 64-bit cipher text blocks, which are concatenated to form the 128-bit session key.

The plain text input to the random number generator, consists of two 64-bit blocks is itself derived from a stream of 128 bit randomized numbers. These numbers are based on keystroke input from the user. Both the keystroke timing and the actual key struck are used to generate randomized stream. Thus if the user hits arbitrary keys at his or her normal pace, reasonably random input will be generated. This random input is then combined with previous session keys output from CAST-128 to form the key input to the generator. The result is to produce a sequence of session keys.

Key Identifier

If a user is having many public keys, how a user will identify which public key is used for encrypting the message? PGP assigns a unique key ID to each public key within a user. The Key ID of the public key PUa is **(PU_a** mod 264).

The format of the transmitted message is given in figure.

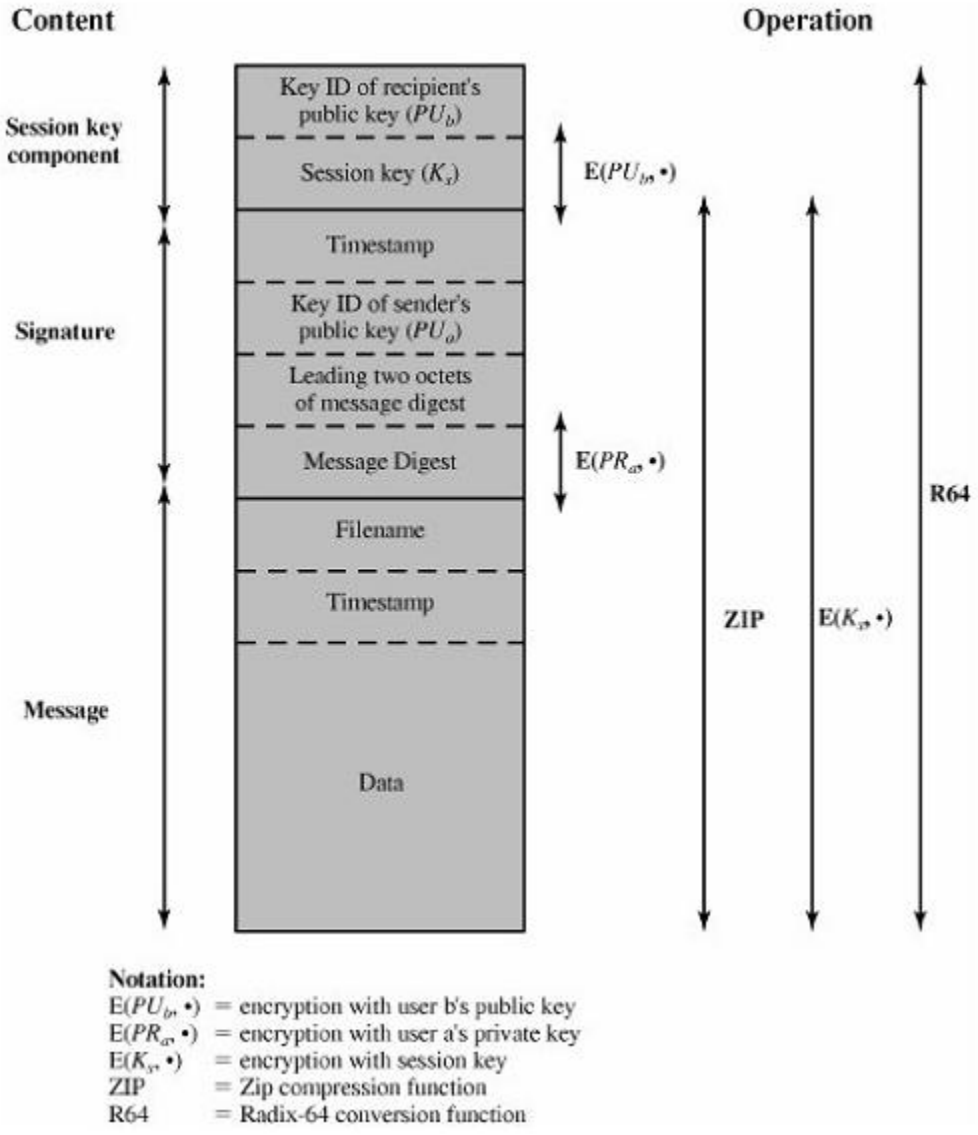
The message component includes the actual data to be stored or transmitted, as well as a filename and a timestamp that specifies the time of creation.
The **signature component** includes the following:

Timestamp:

The time at which the signature was made.

Message digest:

The 160-hit SHA-1 digest, encrypted with the sender's private signature key. The digest is calculated over the signature timestamp concatenated with the data portion of the message component. The inclusion of the signature timestamp in the digest assures against replay types of attacks.



The exclusion of the filename and timestamp portions of the message component ensures that detached signatures are exactly the same as attached signatures prefixed to the message. Detached signatures are calculated on a separate file that has none of the message component header fields.

Leading two octets of message digest:

To enable the recipient to determine if the correct public key was used to decrypt the message digest for authentication, by comparing this plaintext copy of the first two octets with the first two octets of the decrypted digest. These octets also serve as a 16-bit frame check **1** sequence for the message.

Key ID of sender's public key:

Identifies the public key that should be used to decrypt the message digest and, hence, identifies the private key that was used to encrypt the message digest.

The message component and optional signature component may be compressed using ZIP and may be encrypted using a session key. The session key **component** includes the session key and the identifier of the recipient's public key that was used by the sender to encrypt the session key. The entire block is usually encoded with radix-64 encoding.

Key Rings

We have seen how key IDs are critical to the operation of POP and that two key IDs are included in any POP message that provides both confidentiality and authentication. These keys need to be stored and organized in a systematic way for efficient and effective use by all parties. The scheme used in PGP is to provide a pair of data structures at each node, one to store the public/pi vate key pairs owned by that node and one to store the public keys of other users known at this node. These data structures are referred to, respectively, as the private-key ring and the public-key ring.

Figure shows the general structure of a **private-key ring.** We can view the ring as a table, in which each row represents one of the public/private key pairs owned by this user. Each row contains the following entries:

Timestamp: The date/time when this key pair was generated.

Key ID: The least significant 64 bits of the public key for this entry.

Public key: The public-key portion of the pair.

Private key: The private-key portion of the pair: this field is encrypted.

User ID: Typically, this will be the users e-mail address**.** However, the user may choose to associate a different name with each pair or to reuse the same User ID more than once.

Private-Key Ring

Timestamp	Key ID*	Public Key	Encrypted Private Key	User ID*
. . .	. . .	. . .	. . .	. . .
T_i	$PU_i \bmod 2^{64}$	PU_i	$E(H(P_i), PR_i)$	User i
. . .	. . .	. . .	. . .	. . .

Public-Key Ring

Timestamp	Key ID*	Public Key	Owner Trust	User ID*	Key Legitimacy	Signature(s)	Signature Trust(s)
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
T_i	$PU_i \bmod 2^{64}$	PU_i	$trust\_flag_i$	User i	$trust\_flag_i$		
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .

* = field used to index table

Either User ID or Key ID can index the private-key ring. Although it is intended that the private-key ring be stored only on the machine of the user that created and owns the key pairs, and that it be accessible only to that user, it makes sense to make the value of the private key as secure as possible. Accordingly, the private key itself is not stored in the key ring. Rather, this key is encrypted using CAST-128 (or IDEA or 3DES). The procedure is as follows:

- The user selects a passphrase to be used for encrypting private keys.
- When the system generates a new public/private key pair using RSA, it asks the user for the passphrase. Using SHA-1, a 160-bit hash code is generated from the passphrase, and the passphrase is discarded.
- The system encrypts the private key using CAST-128 with the 128 bits of the hash code as the key. The hash code is then discarded, and the encrypted private key is stored in the private-key ring.

Subsequently, when a user accesses the private-key ring to retrieve a private key, he or she must supply the passphrase. PGP will retrieve the encrypted private key, generate the hash code of the passphrase, and decrypt the encrypted private key using CAST-128 with the hash code.

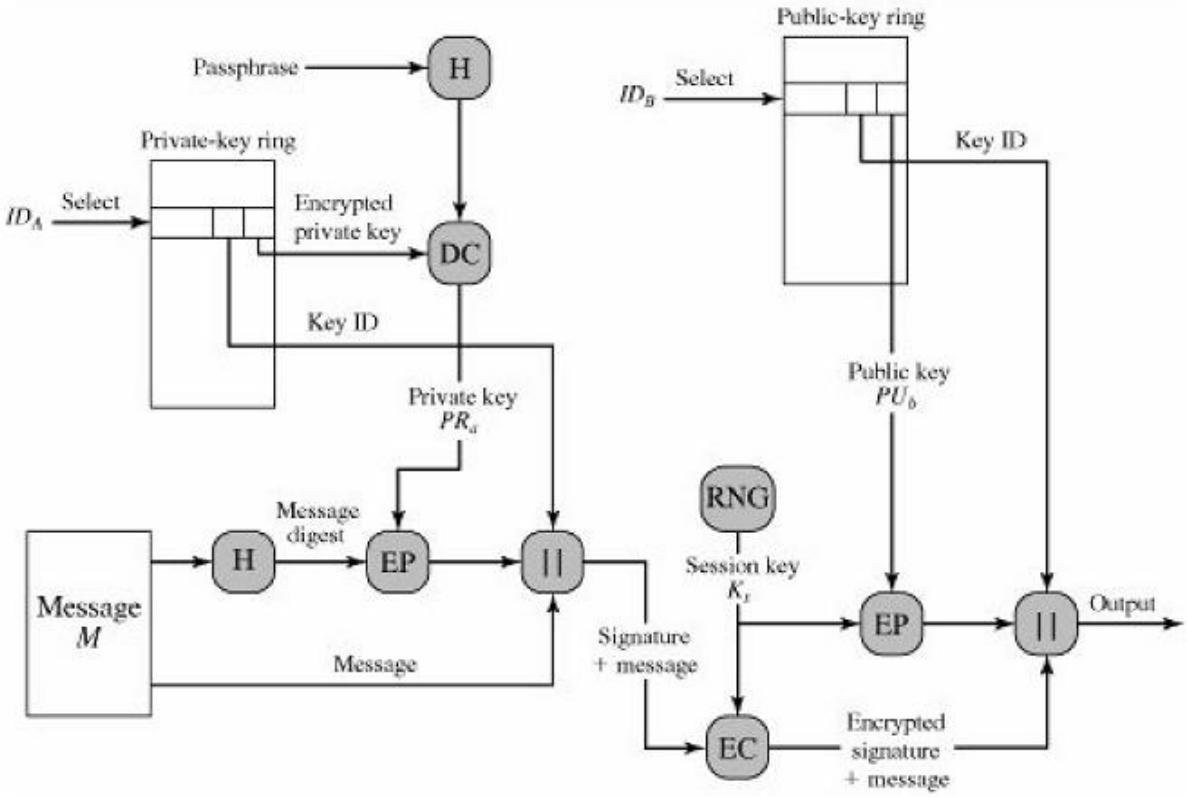
As in any system based on passwords, the security of this system depends on the security of the password. To avoid the temptation to write it down, the user should use a passphrase that is not easily guessed but that is easily remembered. Figure also shows the general structure of a **public-key ring.** This data structure is used to store public keys of other users that are known to this use:, For the moment, let us ignore some fields shown in the table and describe the following fields:

- **Timestamp:** The date/time when this entry was generated.
- **Key ID:** The least significant 64 bits of the public key for this entry.
- **Public Key:** The public key for this entry.
- **User ID:** Identifies the owner of this key. Multiple user IDs may be with a single public key.

We are now in a position to show how these key rings are used in message transmission and reception. For simplicity, we ignore compression and radix-64 conversion in the following

discussion. First consider message transmission and assume that the message is to be both signed and encrypted. The sending PGP entity performs the following steps:

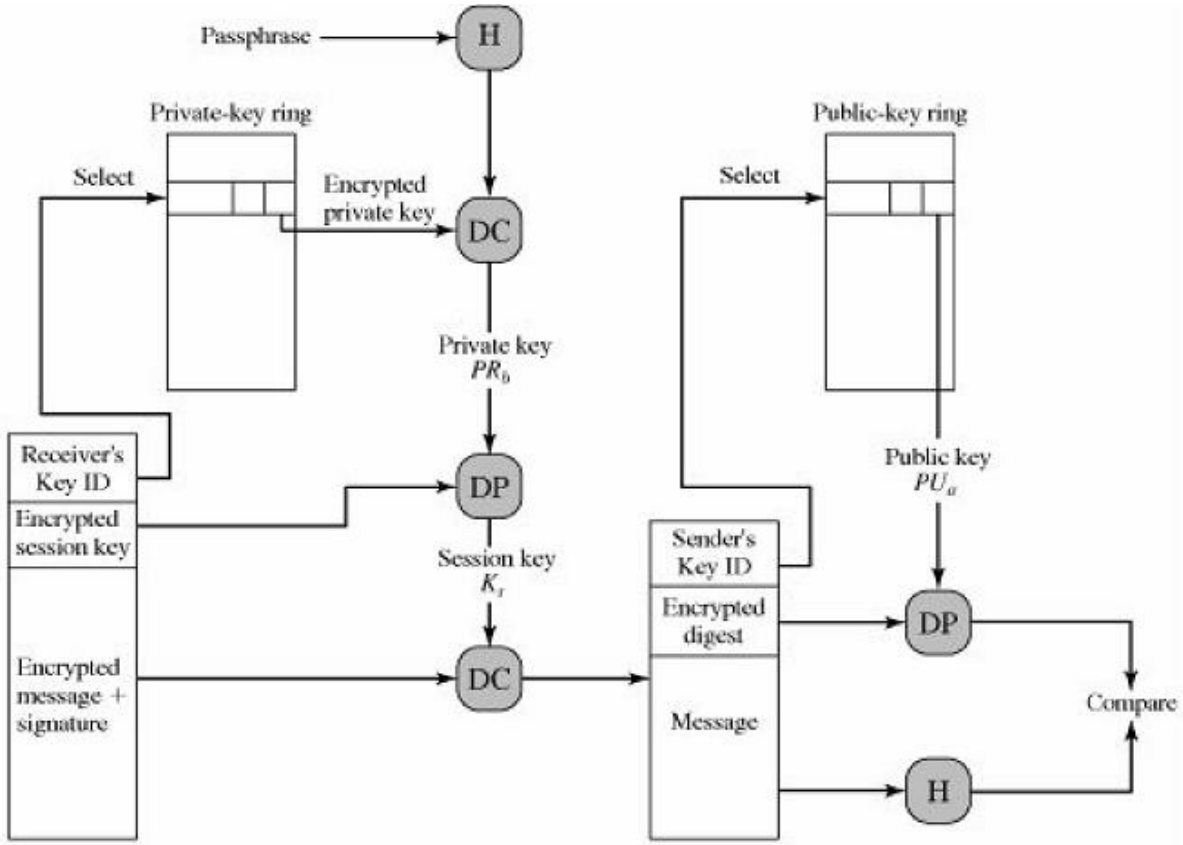
Figure 15.5. PGP Message Generation (from User A to User B; no compression or radix 64 conversion)



- Signing the message
  - PGP retrieves the sender's private key from the private-key ring using your user id as an index. If your user id was not provided in the command, the first private key on the ring is retrieved.
  - PGP prompts the user for the passphrase to recover the unencrypted private key.
  - The signature component of the message is constructed.
- Encrypting the message
  - PGP generates a session key and encrypts the message.
  - PGP retrieves the recipient's public key from the public-key ring using her user id as an index.

- The session key component of the message is constructed. The receiving POP entity performs the following steps

Figure 15.6. PGP Message Reception (from User A to User B; no compression or radix 64 conversion)



- Decrypting the message
  - PGP retrieves the receiver's private key from the private-key ring, using the Key ID field in the session key component of the message as an index.
  - PGP prompts the user for the passphrase to recover the unencrypted private key.
  - PGP then recovers the session key and decrypt the message.
- Authentication Message

- PGP retrieves the sender's public key from the public key ring, using the key id field in the signature key component of the message as an index.
- PGP recovers the transmitted message digest.
- PGP computes the message digest for the received message and compares it to the transmitted message digest to authenticate.

S/MIME

Secure/Multipurpose Internet Mail Extension is an e-mail security standard. PGP is used for personal e-mail security and S/MIME is for commercial purpose.

The five header fields defined in MIME are as follows:

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 within the body.

MIME Content Type

Type	Subtype	Description
Text	Plain	Unformatted ASCII text
	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 in which 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.
Message	Rfc822	Indicates that the body is an entire message including header and the body
	Partial	Enable fragmentation of a large message into a number of parts that must be reassembled at the destination.
	External body	It indicates that the actual data is not in the body, but it contains some information to access the data.

MIME transfer Encoding

The objective is to provide reliable delivery across the largest environment.

7 bit	The data are all represented by short lines of ASCII characters
8bit	The lines are short, but there may be non ASCII characters
Binary	The lines are not necessary to be short enough for SMTP transport
Quoted printable	Encodes the data in such a way that if the data being encoded are ASCII text the encoded form of the data remains largely recognizable

	by humans
Base64	Encodes data by mapping 6 bit block of input to 8 bit block of output, all of which are printable ascii characters
x-token	A named non standard 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 recipient can only view a signed data message with S/MIME capability.
- Clear-signed data: As with signed data, a digital signature of the content is formed. However, 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

Securing a MIME Entity

S/MIME secures a MIME entity with a signature, encryption, or both. An entity may be an entire message, or if the content type is multipart, then a MIME entity is one or more of the subparts of the message. The MIME entity is prepared according to the normal rules for MIME sage preparation. Then the MIME entity plus some security-related data such as algorithm identifiers and certificates are processed by S/MIME to produce what is

known as a PKCS object. A PKCS object is then treated as message Content at B wrapped in MIME.

Enveloped Data

An applicationlpkcs7-mime subtype is used for one of four categories of S/MIME processing, each with a unique smime-type parameter. In all cases, the resulting entity, referred to as an ***object,*** is represented in a form known as Basic. Encoding Rules (BER), which is defined in ITU-T Recommendation X.209? The BER format consists of arbitrary octet strings and is therefore binary data. Such an object should be transfer encoded with base64 in the outer MIME message. The steps for preparing an envelopedData MIME entity are as follows.

- Generate a pseudorandom session key for a particular symmetric encryption algorithm.
- For each recipient, encrypt the session key with the recipient's pub ***C***
- For each recipient, prepare a block known as RecieptientInfo that contains an identifier of the recipient's public-key certificate an identifier of the algorithm used to encrypt the session key, and the encrypted session key.
- Encrypt message content with the session key.

Signed Data

The signedData smime-type can actually be used with one or more signers. The steps for preparing a signedData MIME entity are as follows:

- Select a message digest algorithm (SHA or MD5).
- Compute the message digest, or hash function, of the content to be signed.
- Encrypt the message digest with the signer's private key.
- Prepare a block known as Signerlnfo that contains the signer's public-key certificate, an identifier of the message digest algorithm, an identifier of the algorithm used to encrypt the message digest, and the encrypted message digest.

Clear Signing

Clear signing is achieved using the multipart content type with a signed subtype. This signing process does not involve transforming the message to be signed, so that the message is sent "in the clear." Thus, recipient with MIME capability but not S/MIME capability are able to read the incoming message.

A multipart/signed message has two parts. The first part can be any type but must be prepared so that it will not be altered during transfer from source to destination. This means that if the first part is not 7bit, then it needs to be encoded using base64 or quoted-printable. Then this part is processed in the same manner as signed Data, but in this case an object with signed Data format is created that has an empty message content field. This object is a detached signature. It then transfer encoded using base64 to become the second part of the multipart signed message. This second part has a MIME content type of application and a subtype of pkcs7-signature.

IP Security Overview

IPSec provides the capability to secure communication across a LAN, across private and Public WANs and across the internet.

Applications of IPSec

- Secure branch office connectivity over the Internet: A company can I secure virtual private network over the Internet or over a public WAN. This enables a business to rely heavily on the Internet and reduce its need for private networks, saving costs and network management overhead.
- Secure remote access over the internet: An end user whose system **is** equipped with IP security protocols can make a local call to an Internet service provider and gain secure access to the company network. This reduces the toll charges for traveling employees and telecommuters.

- Establishing extranet and intranet connectivity with partners: IPSec 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 an electronic commerce applications have built-in security protocols, the use of IPSec enhances that security.

Benefits of IPSec

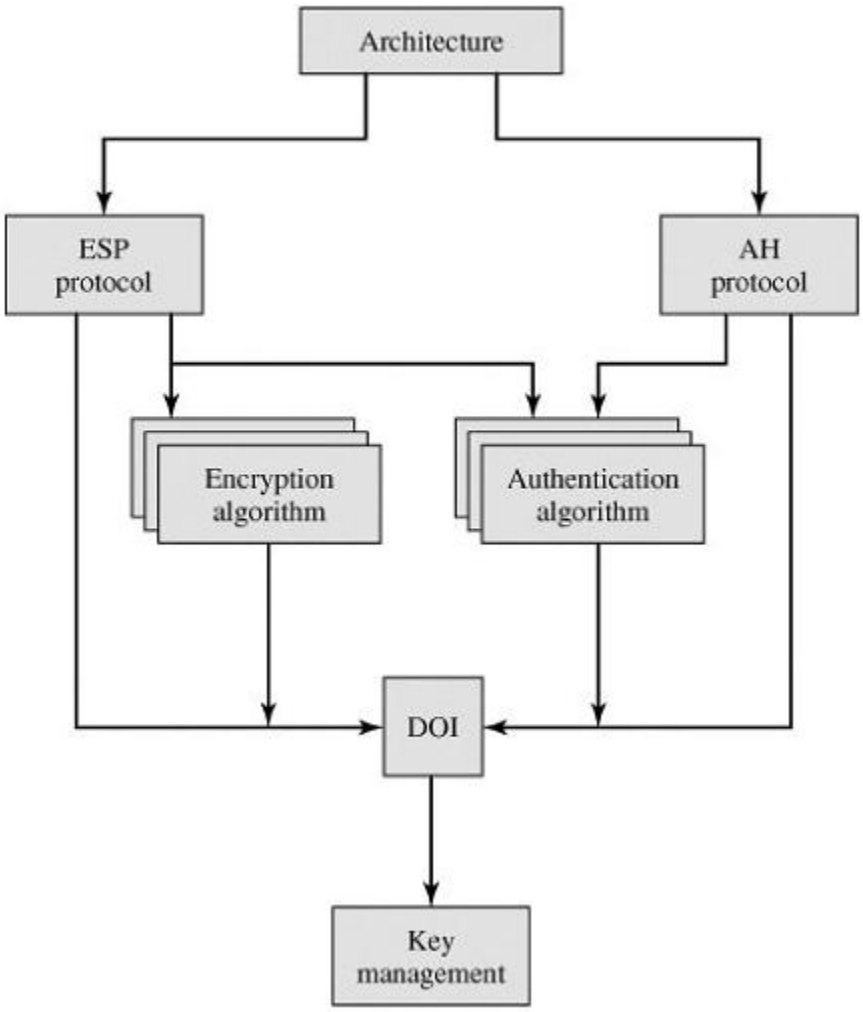
- When IPSec is implemented on a firewall or a router it provides strong security that can be applied to all traffic crossing the perimeter.
- IPSec in a firewall is resistant to bypass if all traffic from the outside must us IP, and the firewall is the only means of entrance from the Internet into the organization.
- 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. There is no need to train users on security mechanisms, issue keying material on a per-user basis, or revoke keying material when users leave the organization.
- IPSec can provide security for individual users if needed. This is useful offsite workers and for setting up a secure virtual sub network within an organization for sensitive applications.

IP Security Architecture

IP security consists of a number of documents. The documents are divided in to seven groups which are given below.

Architecture: Covers the general concepts, security requirements, definitions, and mechanisms defining IPSec technology.

Figure 16.2. IPSec Document Overview



Encapsulation Security Payload (ESP): Covers the packet format and general issues related to the use of the ESP for packet encryption and, authentication.

Authentication Header (AH): Covers the packet format and general issues related to the use of AH for packet authentication.

Encryption **Algorithm: A** set of documents that describe how various encryption algorithms are used for ESP.

Authentication Algorithm A set of documents that describe how various authentication algorithms are for AH for packet authentication and for the authentication option of ESP.

Key Management: Documents that describe key management schemes.

Domain of Interpretation: Contains values needed for the other documents relate to each other.

IPSec Services

IPSec provides security services at the IP layer by enabling a system to required security protocols, determine the algorithm(s) to use for the services and put in place any cryptographic keys required to provide, the sevices. Two protocols are used to provide security: An authentication protocol designated by the header of the protocol, Authentication Header (AH) combined encryption / authentication protocol designated by the format of packet for that protocol, Encapsulating Security Payload (ESP). The service as follows:

- Access control
- Connectionless integrity
- Data origin authentication
- Confidentiality
- Limited traffic flow confidentiality

Security Associations

A key concept that appears in both the authentication and confidentiality mechanisms for IP is the security association (SA). An association is a one-way relationship between a sender and a receiver that affords security services to the traffic carried on it. If a peer relationship is needed, for two-way secure exchange, then two security associations are required. Security services are afforded to an SA for the use of AH or ESP, but not both.
A security association is uniquely identified by three parameters:

- Security Parameters Index (SPI): A bit string assigned to this SA and having local significance only. The SPI is carried in AH and ESP headers to enable the receiving system to select the SA under which a received packet will be processed.

- **IP** Destination Address: Currently, only unicast addresses are allowed; this is the address of the destination endpoint of the SA, which may be an end user system or a network system such as a firewall or router.
- Security Protocol Identifier. This indicates whether the association is an AH or ESP security association.

SA Parameters

In each IPSec implementation, there is a nominal Security Association base that defines the parameters associated with each SA. A security association normally defined by the following parameters:

Sequence Number Counter: A 32-bit value used to generate the Sequence Number field in AH or ESP headers

Sequence Counter Overflow: A flag indicating whether overflow Sequence Number Counter should generate an auditable event and prevent further transmission of packets on this SA

Anti-Replay Window: Used to determine whether an inbound AR packet is a replay

AH Information: Authentication algorithm, keys, key lifetimes, and related parameters being used with AH

ESP Information: Encryption and authentication algorithm, keys, initialization values, key lifetimes, and related parameters being used with ESP

Lifetime of this Security Association: A time interval or byte count after which an SA must be replaced with a new SA (and new SPI) or terminated, plus an indication of which of these actions should occur

IPSec Protocol Mode: Tunnel, transport, or wildcard.

Path MTU: Any observed path maximum transmission unit (maximum size of a packet that can be transmitted without fragmentation) and aging variables.

SA Selectors

IPSec provides the user with considerable flexibility in the way in which services are applied to IP traffic. SAs can be combined in a number of ways to yield the desired user configuration. Furthermore provides a high degree of granularity in discriminating between traffic that is a related IPSec protection and traffic that is allowed to bypass IPSec, in the former case IP traffic to specific SAs.

The means by which IP traffic is related to specific SAs is the nominal Security Policy Database (SPD). In its simplest form, an SPD contains entries, each of which defines a subset of IP j 'traffic and points to an SA for that traffic. In more complex environments, there *r* may be multiple entries that potentially relate to a single SA or multiple SAs associated with a single SPD entry.

Each SPD entry is defined by a set of IP and upper-layer protocol field values, called *selectors.* In effect, these selectors are used to filter outgoing traffic in order to map it into a particular SA. Outbound processing obeys the following general sequence for each IP packet:

- Compare the values of the appropriate fields in the packet against the SPD to find a matching SPD entry, which will point to zero or more SAs.
- Determine the SA if any for this packet and its associated SPI.
- Do the required IPSec processing.

The following selectors determine an SPD entry:

Destination **IP** Address: This may be a single IP address, an enumerated list or range of addresses, or a wildcard (mask) address.

Source **IP** Address: This may be a single IP address, an enumerated list or range of addresses, or a wildcard (mask) address. The latter two are required to support more than one source system sharing the same SA

UseriD: A user identifier from the operating system. This is not a field in the IP or upper-layer headers but is available if IPSec is running on the same operating system as the user.

Data Sensitivity Level: Used for systems providing information flow security

Transport Layer Protocol: Obtained from the IPv4 Protocol or IPv6 Next Header field. This may be an individual protocol number, a list of protocol numbers, or a range of protocol numbers.

IPSec **Protocol (AH or ESP** or ***AHIESP):*** If present, this is obtained from the IPv4 Protocol or IPv6 Next Header field.

Source and Destination Ports: These may be individual TCP or UDP port values, an enumerated list of ports, or a wildcard port.

IPv6 Class: Obtained from the IPv6 header. This may be a specific IPv6 Class value or a wildcard value.

IPv6 Flow Label: Obtained from the IPv6 header. This may be a specific IPv6, FIow Label value or a wildcard value.

IPV4 Type of Service (TOS): Obtained from the IPv4 header.

Transport mode and tunnel mode

Both AH and ESP support two modes of use: transport and tunnel mode

Transport Mode

Transport mode provides protection primarily for upper-layer protocols is, transport mode protection extends to the payload of an IP packet. Transport mode is used for end to end communication between two hosts. When a host runs AH or ESP over IPv4, the payload is the data that normally follow the IP header. For IPv6, the payload is the data that normally follow both the IP header and any IPv6 extensions headers that are present, with the possible extension of the destination options header, which may be included in the protection.

ESP in transport mode encrypts and optionally authenticates the IP payload but not the IP header. AH in transport mode authenticates the IP payload and selected portions of the IP header.

Tunnel Mode

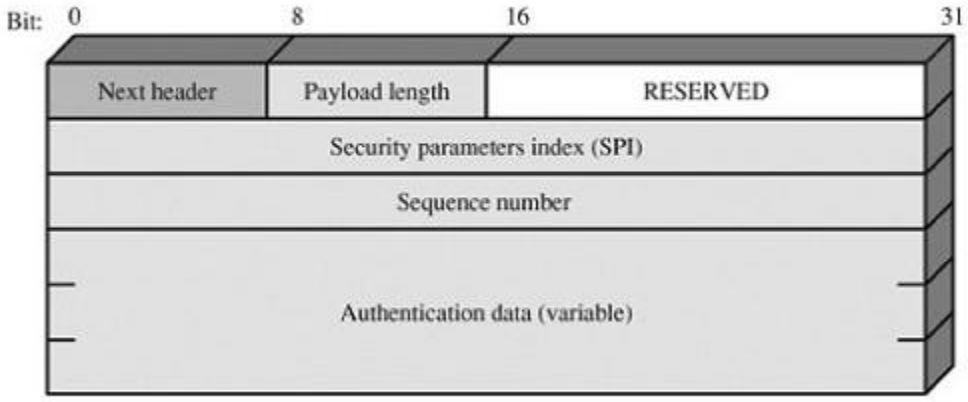
Tunnel mode provides protection to the entire IP packet. To achieve this, after the AH or ESP fields are added to the IP packet, the entire packet plus security fields is treated as the payload of new "outer" IP packet with a new outer IP header. The entire original, or inner, packet travels through a "tunnel" from one point of an IP network to another, no routers along the way are able to examine the inner IP header. Because the original packet is encapsulated, the new, larger packet may have totally different source and destination addresses, adding to the security. Tunnel mode is used when one or both ends of an SA is a security gateway, firewall or router that implements IPSec. With tunnel mode, a number of hosts on networks behind firewalls may engage in secure communications without implementing IPSec. The unprotected packets generated by such hosts are tunneled through external networks by tunnel mode SAs set up by the IPSec software in the firewall or secure router at the boundary of the local network.

Authentication Header

The Authentication Header provides support for data integrity and authentication of IP packets. The data integrity feature ensures that undetected modification to a packet's content in transit is not possible. The authentication feature enables an end system or network device to authenticate the user or application and filter traffic accordingly; it also prevents the address spoofing attacks observed in today's Internet.

The Authentication Header consists of the following fields.

Figure 16.3. IPSec Authentication Header



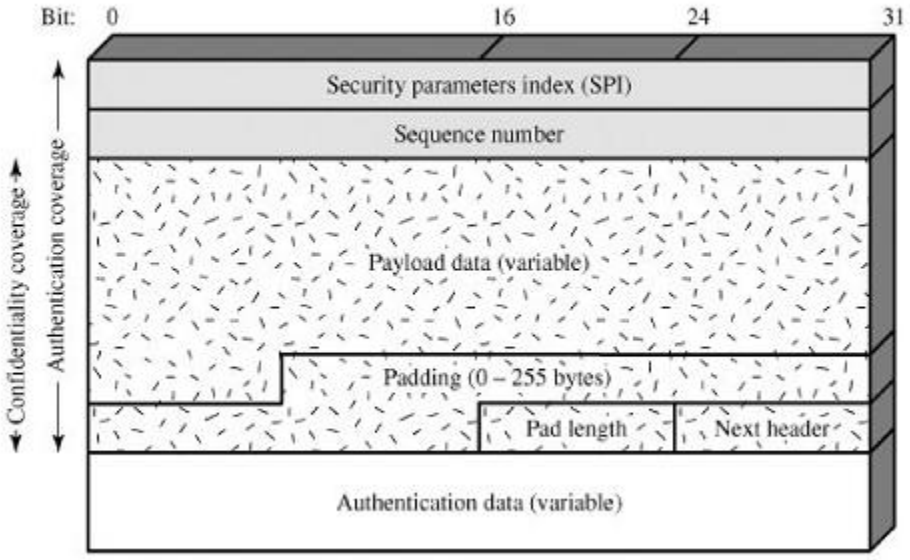
- Next Header (8 bits): Identifies the type of header immediately following this header.
- Payload Length (8 bits): Length of Authentication Header in 32-bit words, minus 2. For example, the default length of the authentication data field is 96 bits, or three 32-bit words. With a three-word fixed header, there are a total of six words in the header, and the Payload Length field has a value of 4.
- Reserved (16 bits): For future use.
- Security Parameters Index (32 bits): Identifies a security association.
- Sequence Number (32 bits): A monotonically increasing counter value.
- Authentication data: A variable length field that contains the integrity check value.

Encapsulating Security Payload

The Encapsulating Security Payload provides confidentiality services, including confidentiality of message contents and limited traffic flow confidentiality. As a optional feature, ESP can also provide the same authentication services as AH ESP Format. Figure shows the format of an ESP packet. It contains the following fields:

- Security Parameters **Index (32** bits): Identifies a security association.
- **Sequence Number** (32 bits): A monotonically increasing counter value; this provides **an** anti-replay function.
- Payload Data (variable): This is a transport-level segment (transport mode) or IP packet (tunnel mode) that is protected by encryption.
- Padding (0—255 bytes): to make the plain text into a fixed size.
- **Pad Length (8** bits): Indicates the number of pad bytes immediately preceding this field.
- Next Header (8 bits): Identifies the type of data contained in the payload data field by identifying the first header in that payload.

Figure 16.7. IPSec ESP format



- Authentication Data (variable): A variable-length field (must be an integral number of 32-bit words) that contains the Integrity Check Value computed over the ESP packet minus the Authentication Data field.

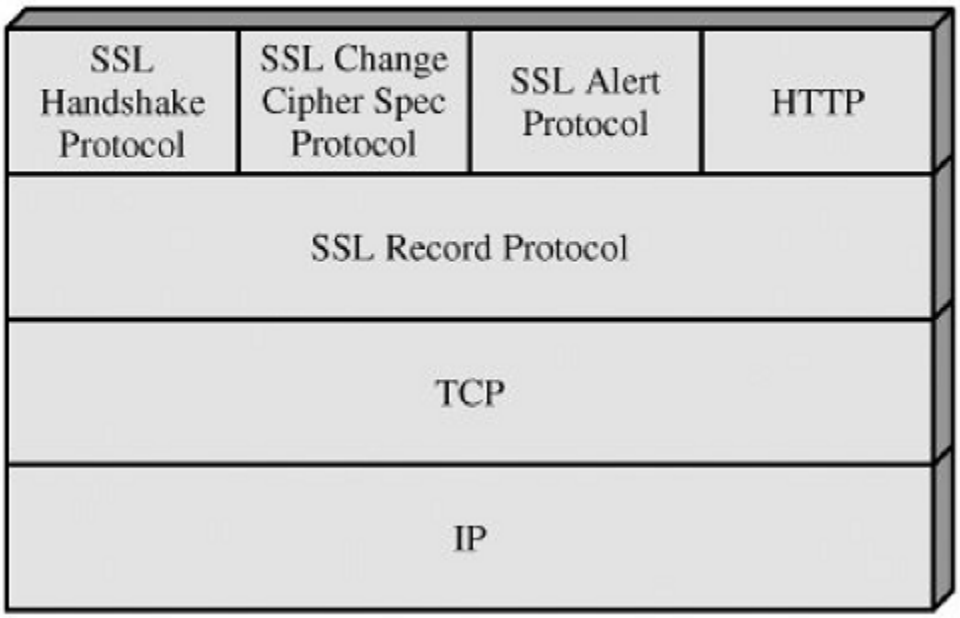
Secure Socket Layer

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.

SSL Architecture

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

Figure 17.2. SSL Protocol Stack



The SSL Record Protocol provides basic security services to various higher- layer protocols. In particular, the Hypertext Transfer Protocol (HTTP), which provides the transfer service for Web client/server interaction, can operate on top of SSL. Three higher-layer protocols are defined as part of SSL:

- The Handshake protocol
- The Change Cipher Spec Protocol
- The Alert Protocol

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

Connection: A connection is a transport that provides a suitable type of service. For SSL, such connections are peer- to-peer relationships. The connections are transient. Every connection is associated with one session.

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 graphic security parameters, which can be shared among multiple connections. Sessions are used to avoid the expensive negotiation of new security parameter or each connection.

Between any pair of parties there may be multiple secure connections. There are actually a number of states associated with each session. Once a session is established, there is a current operating state for both read and write. In addition, during the Handshake Protocol, pending read write states are created. Upon successful conclusion of the Handshake Protocol pending states become the current states.

A session state is defined by the following parameters

Session identifier: An arbitrary byte sequence chosen by the server to identify an active or resumable session state.

Peer certificate: An X509.v3 certificate of the peer. This element of the stat may be null.

Compression method: The algorithm used to compress data prior to encryption.

Cipher spec: Specifies the bulk data encryption algorithm (DES) and a hash algorithm (such as MD5 or SHA-1) used for MAC calculation. It also defines cryptographic attributes such as the hash_size.

Master secret: 48-byte secret shared between the client and server.

Is resumable: A flag indicating whether the session can be used to initiate new connections.

A connection state is defined by the following parameters:

Server and client random: Byte sequences that are chosen by the server and client for each connection.

Server write **MAC** secret: The secret key used in MAC operations on data sent by the server.

Client write **MAC** secret: The secret key used in MAC operation data S by the client.

Server write key: The conventional encryption key for data encrypted by server and decrypted by the client.

Client write key: The conventional encryption key for data encrypted by the client and decrypted by the server.

Initialization vectors: When a block cipher in CBC mode is used, an initialization vector (VI) is maintained for each key. This field is first initialized by the SSL Handshake Protocol. Thereafter the final ciphertext block from record is preserved for use as the IV with the following record.

Sequence numbers: Each party maintains separate sequence numbers transmitted and received messages for each connection. When a party sends or receives a change cipher spec message, the appropriate sequence number is set to zero. Sequence number may not exceed 2^6.

SSL Record Protocol

The SSL Record Protocol provides two services for SSL connections:

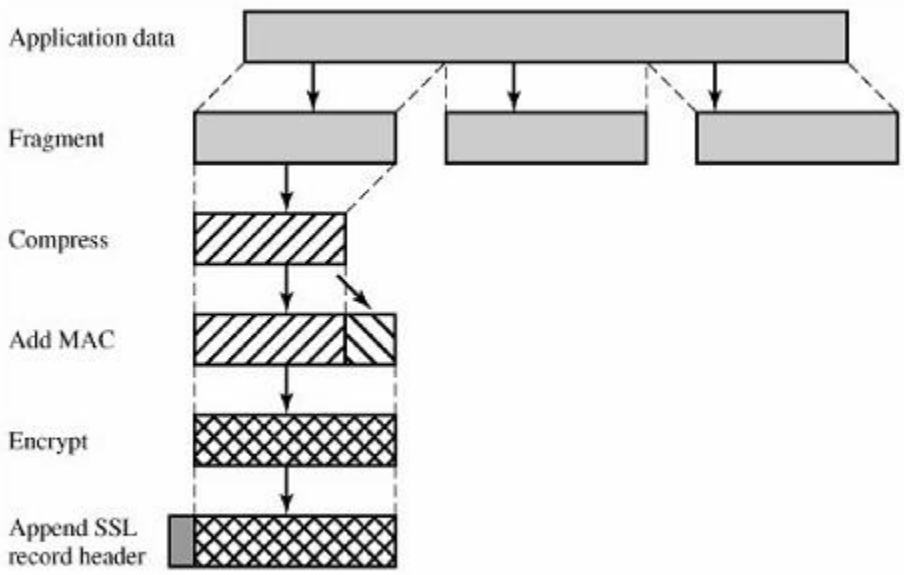
Confidentiality: The Handshake Protocol defines a shared secret key that is used for conventional encryption of SSL payloads.

Message Integrity: The Handshake Protocol also defines a shared secret key that is used to form a message authentication code (MAC).

Figure indicates 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 and then delivered to higher level users. The first step is fragmentation. Each upper-layer message is fragmented into blocks of 2 bytes (1634 bytes) or less. Next, compression is optionally applied.

Compression must be lossless and may not increase the content length by more than 1024 bytes.2 In SLLv3 (as well as the current version of TLS), no compression algorithm is specified, so the default compression algorithm is null.

Figure 17.3. SSL Record Protocol Operation



The next step in processing is to compute the message authentication code over the compressed data. A shared key is used for this purpose.

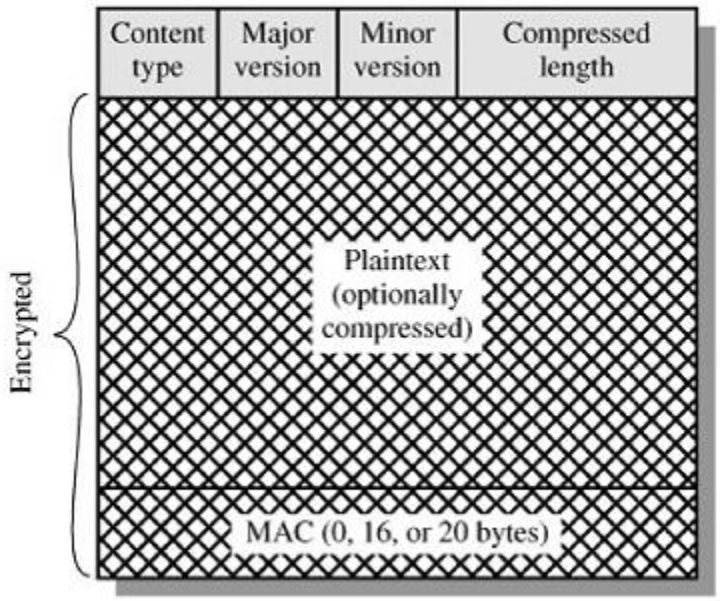
The compressed message plus the MAC are encrypted using symmetric encryption. Encryption may not increase the content length by more than 1024 bytes. For stream encryption, the compressed message plus the MAC are encrypted.

The final step of SSL Record Protocol processing is to prepend 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.

The SSL record format

Figure 17.4. SSL Record Format



Change Cipher Spec Protocol

The Change Cipher Spec Protocol is one of the three SSL-specific protocols that use the SSL Record Protocol, and it is the simplest. This protocol consists 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

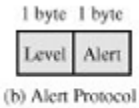
to be copied into the current state, which updates the cipher suite to be used on this connection.

(a) Change Cipher Spec Protocol

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 byte takes the value warning (l) or fatal (2) to convey the severity of the message. If the level is fatal, SSL immediately terminates the connection. Other connections on the same session may continue, but no new connections on this session may be established. The second byte contains a code that indicates the specific alert.



(b) Alert Protocol

The common errors are listed below.

unexpected_message: An inappropriate message was received.

bad_record_mac: An incorrect MAC was received.

decompression_failure: The decompression function received improper input.

handshake_failure: Sender was unable to negotiate an acceptable set of security parameters given the options available.

illegal_parameter: A field in a handshake message was out of range or inconsistent with other fields.

The remainder of the alerts are the following:

close_notify: Notifies the recipient that the sender will not send any more messages on this connection. Each party is required to send a close_notify alert before closing the right side of a connection.
no_certificate: May be sent in response to a certificate request if no appropriate certificate isavailable.

bad_certificate: A received certificate was corrupt.

unsupported_certificate: The type of the received certificate is not supported.

certificate_revoked: A certificate has been revoked by its signer.

certificate_expired: A certificate has expired.

certiflcate.unknown: Some other unspecified issue arose in processing the certificate, rendering it unacceptable.

Handshake Protocol

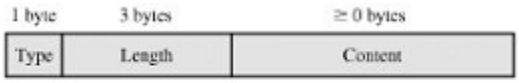
The most complex part of SSL is the Handshake Protocol. This protocol allows the server and client to authenticate each other and to negotiate an encryption and MAC algorithm and cryptographic keys to be used to protect data sent in an SSL record. The Handshake Protocol is used before any application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

The Handshake Protocol consists of a series of messages exchanged by client and server. All of these have the format shown in Figure. Each message has three fields:

Type **(1 byte):** Indicates one of 10 messages. Table 17.2 lists the defined message types.

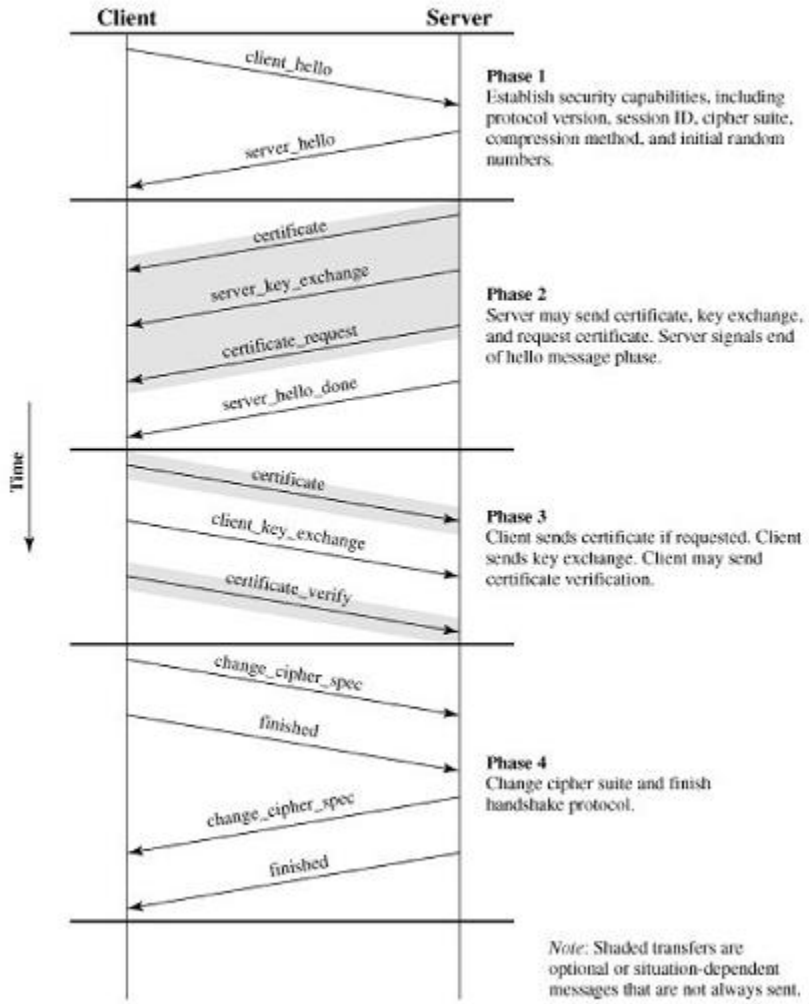
Length (3 bytes): The length of the message in bytes.

Content (>=**1 byte):** The parameters associated with this message;

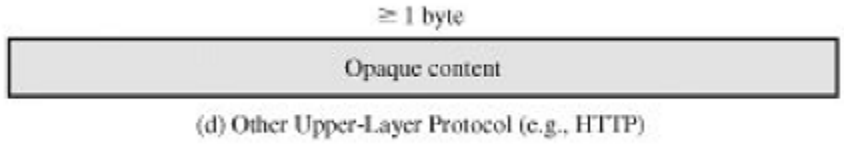


(c) Handshake Protocol

Figure 17.6. Handshake Protocol Action



The initial exchange needed to establish a logical connection between the client and server is shown in figure.



(d) Other Upper-Layer Protocol (e.g., HTTP)

The exchange can be viewed as having four phases.

Phase 1. Establish Security Capabilities

This phase is used to initiate a logical connection and to establish the security capabilities that will be associated with it. The exchange is initiated by the client, which sends a client_hello message with the following parameters:

Version: The highest SSL version understood by the client.

Random: A client-generated random structure, consisting of a 32-bit timestamp and 28 bytes generated by a secure random number generator. These values serve as nonces and are used during key exchange to prevent replay attacks.

Session ID: A variable-length session identifier. A nonzero value indicates that the client wishes to update the parameters of an existing connection or create a new connection on this session. A zero value indicates that the client wishes to establish a new connection on a new session.

CipherSuite: This is a list that contains the combinations of cryptographic algorithms supported by-the client, in decreasing order f preference. Each element of the list (each cipher suite) defines both a key exchange algorithm and aCipherSpec;

Compression Method: This is a list of the compression methods the client supports.

After sending the client_hello message, the client waits for the server_hello message, which contains the same parameters as the client_hello message. For the server_hello message, the following conventions apply. The Version field contains the lower of the version suggested by the client and the highest supported by the server. The Random field is generated by the server and is independent of the client's Random field. If the SessionlD field of the client was nonzero,

the same value is used by the server; otherwise the server's SessionlD field contains the value for a new session. The CipherSuite field contains the single cipher suite selected by the server from those proposed by the client. The Compression field contains the compression method selected by the server from, those proposed by the client. The first element of the Cipher Suite parameter is the key exchange method. The following key exchange methods are supported:

RSA: The secret key is encrypted with the receiver's RSA public key. A public-key certificate for the receiver's key must be made available.

Fixed Diffie-Hellman: This is a Diffie-Heilman key exchange in which the server's certificate contains the Diffie-Heilman public parameters signed by the certificate authority (CA). That is, the public-key certificate contains the Diffie-Heliman public-key parameters. The client provides its Diffie-Heliman public key parameters either in a certificate, if client authentication is required, or in a key exchange message. This method results in a fixed secret key between two peers, based on the Diffie-Heliman calculation using the fixed public keys.

Ephemeral Diffie-Hellman: This technique is used to create ephemeral secret keys. In this case, the Diffle-Heilman public keys are exchanged, signed using the sender's private RSA or DSS key. The receiver can use the corresponding public key to verify the signature. Certificates are used to authenticate the public keys. This would appear to be the most secure of the three Diffie-Heliman options because it results in a temporary, authenticated key.

Anonymous Diffie-Hellman: The base Diffie-Hellman algorithm is used, with no authentication. That is, each side sends its public Diffie-Heilman parameters to the other, with no authentication. This approach is vulnerable to man-in-the-middle attacks, in which the attacker conducts anonymous Diffice_Hellman with both parties.

- Fortezza The technique defined for the Fortezza scheme.

Phase 2: Server Authentication and Key Exchange

The server begins this phase by sending its certificate, if it needs to be authenticated; the message contains one or a chain of *X.509* certificates. The certificate message is required for any agreed-on key exchange method except anonymous Diffie-Heilman. Note that if fixed Diffie-Hellman is used, this certificate message functions as the server's key exchange message because it contains the server's public Diffie-Heilman parameters.

Next, a server_key_exchange message may be sent if it is required. It is not required in two instances:

(1) The server has sent a certificate with fixed Diffie. Hellman parameters

(2) RSA key exchange is to be used. The server_key exchange message is needed for the following:
Anonymous Diffie-Hellman: The message content consists of the two global Diffie-Hellman values (a prime number and a primitive root of that number) plus the server's public Diffie-Heilman key.

Ephemeral Diffie-Heliman: The message content includes the three Diffie. Heliman parameters provided for anonymous Diffie-Hellman, plus **a** signature of those parameters.

RSA key exchange, in which the server is using RSA but has a signature-only RSA key: Accordingly, the client cannot simply send a secret key encrypted with the server's public key. Instead, the server must create a temporary RSA public/private key pair and use the server_key_exchange message to send the public key. The message content includes the two parameters of the temporary RSA public key plus a signature of those parameters.

Fortezza

The final message in Phase 2, and one that is always required, is the server_done message, which is sent by the server to indicate the end of the end of the server

hello and associated messages. After sending this message, the server will wait for a client response. This message has no parameters.

Phase 3: Client Authentication and Key Exchange

Upon receipt of the server_done message, the client should verify that the server provided a valid certificate if required and check that the server_hello parameters are acceptable. If all is satisfactory, the client sends one or more messages back to the server.

If the server has requested a certificate, the client begins this phase by sending a certificate message. If no suitable certificate is available, the client sends a no_certificate alert instead.

Next is the client_key_exchange message, which must be sent in this phase. The content of the message depends on the type of key exchange, as follows:

RSA: The client generates a 48-byte *pre-master secret* and encrypts with the public key from the server's certificate or temporary RSA key from a server_key_exchange message.

Ephemeral or Anonymous Diffle-Hellman: The client's public Diffie-Hellman parameters are sent.

Fixed Diffie-Hellman: The client's public Diffie-Hellman parameters were sent in a *certificate* message, so the content of this message is null.

Fortezza: The client's Fortezza parameters are sent.
Finally, in this phase, the client may send a certificate_verify message to provide explicit verification of a client certificate. This message is only sent following any client certificate that has signing capability.

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.

Note that this message is not considered part of the Handshake Protocol but is sent using the Change Cipher Spec Protocol. The client then immediately sends the finished message under the new algorithms, keys, and secrets. The finished message verifies that the key exchange and authentication processes were successful.

Transport Layer Security

TLS is an IETF standardization initiative whose goal is to produce an Internet standard version of SSL. TLS is defined as a Proposed Internet Standard in RFC 2246. RFC 2246 is very similar to SSLv3.

The difference between TLS and SSL is shown below

Version Number: - The TLS Record Format is the same as that of the SSL Record Format and the fields in the header have the same meanings. The one difference is in version values. For the current version of TLS, the Major Version is 3 and the Minor Version is 1.

Message Authentication Code: - There are two differences between the SSLv3 and TLS MAC schemes: the actual algorithm and the scope of the MAC calculation. TLS makes use of the HMAC algorithm defined in RFC 2104. HMAC is defined as follows:

$$HMAC_K(M) = H[(K^+ \oplus opad)||H[(K^+ \oplus ipad)||M]]$$

where

H	=	embedded hash function (for TLS, either MD5 or SHA-1)
M	=	message input to HMAC
K^+	=	secret key padded with zeros on the left so that the result is equal to the block length of the hash code(for MD5 and SHA-1, block length = 512 bits)
ipad	=	00110110 (36 in hexadecimal) repeated 64 times (512 bits)
opad	=	01011100 (5C in hexadecimal) repeated 64 times (512 bits)

SSLv3 uses the same algorithm, except that the padding bytes are concatenated with the secret key rather than being XOR-ed with the secret key padded to the block length. The level of

security should be about the same in both cases. For TLS, the MAC calculation encompasses the fields indicated in the following expression:

```
HMAC_hash(MAC_write_secret, seq_num || TLSCompressed.type ||
      TLSCompressed.version || TLSCompressed.length ||
      TLSCompressed.fragment)
```

The MAC calculation covers all of the fields covered by the SSLv3 calculation, plus the field TLSCompressed.version, which is the version of the protocol being employed.

Pseudorandom Function: - TLS makes use of a pseudorandom function referred to as PRF to expand secrets into blocks of data for purposes of key generation or validation. The objective is to make use of a relatively small shared secret value but to generate longer blocks of data in a way that is secure from the kinds of attacks made on hash functions and MACs. The PRF is based on the data expansion function shown in the below figure.

```
P_hash(secret, seed) = HMAC_hash(secret, A(1)  || seed) ||
                       HMAC_hash(secret, A(2)  || seed) ||
                       HMAC_hash(secret, A(3) || seed) || ...
```

Where A( ) is defined as

A(0) = seed

```
A(i) = HMAC_hash (secret, A(i - 1))
```

The data expansion function makes use of the HMAC algorithm, with either MD5 or SHA-1 as the underlying hash function. P_hash can be iterated as many times as necessary to produce the required quantity of data. For example, if P_SHA-1 was used to generate 64 bytes of data, it would have to be iterated four times, producing 80 bytes of data, of which the last 16 would be discarded. In this case, P_MD5 would also have to be iterated four times, producing exactly 64 bytes of data. Each iteration involves two executions of HMAC, each of which in turn involves two executions of the underlying hash algorithm.

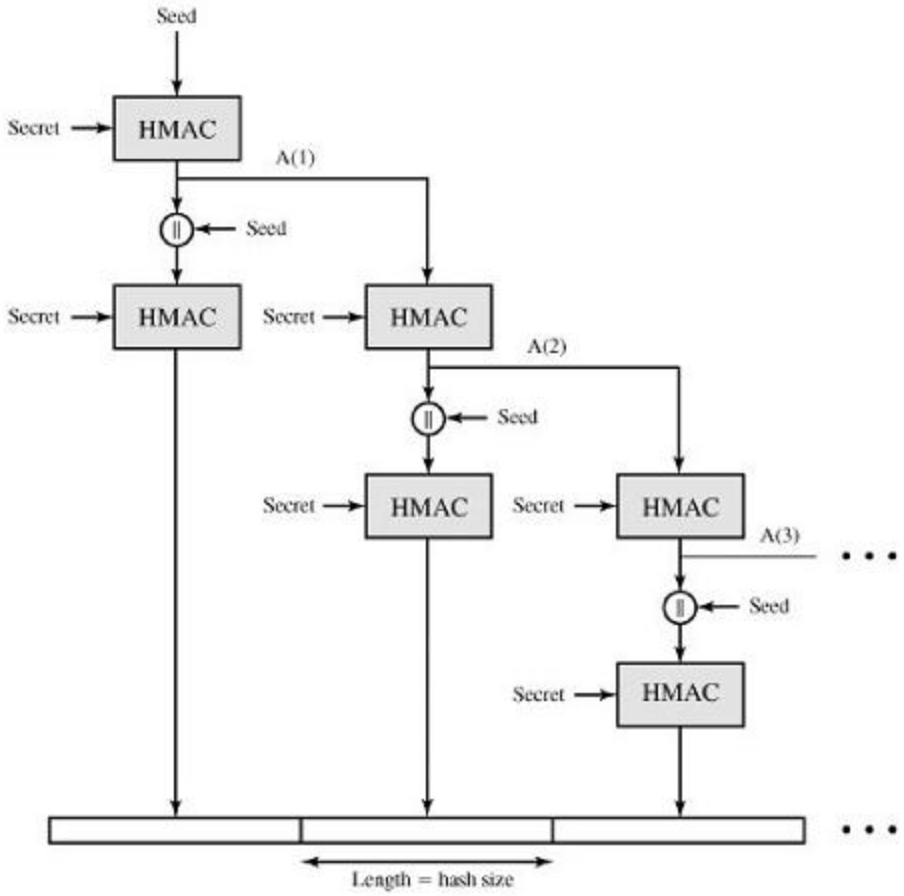


Figure: - TLS Function P_hash (secret, seed)

To make PRF as secure as possible, it uses two hash algorithms in a way that should guarantee its security if either algorithm remains secure. PRF is defined as

$$PRF(secret, label, seed) = P\_MD5(S1, label \| seed) \oplus P\_SHA\text{-}1(S2, label \| seed)$$

PRF takes as input a secret value, an identifying label, and a seed value and produces an output of arbitrary length. The output is created by splitting the secret value into two halves (S1 and S2) and performing P_hash on each half, using MD5 on one half and SHA-1 on the other half. The two results are exclusive-ORed to produce the output; for this purpose, P_MD5 will generally

have to be iterated more times than P_SHA-1 to produce an equal amount of data for input to the exclusive-OR function.

Alert Codes:- TLS supports all of the alert codes defined in SSLv3 with the exception of no_certificate. A number of additional codes are defined in TLS; of these, the following are always fatal:

decryption_failed: A ciphertext decrypted in an invalid way; either it was not an even multiple of the block length or its padding values, when checked, were incorrect.

record_overflow: A TLS record was received with a payload (ciphertext) whose length exceeds 2^{14} + 2048 bytes, or the ciphertext decrypted to a length of greater than 2^{14}+ 1024 bytes.

unknown_ca: A valid certificate chain or partial chain was received, but the certificate was not accepted because the CA certificate could not be located or could not be matched with a known, trusted CA.

access_denied: A valid certificate was received, but when access control was applied, the sender decided not to proceed with the negotiation.

decode_error: A message could not be decoded because a field was out of its specified range or the length of the message was incorrect.

export_restriction: A negotiation not in compliance with export restrictions on key length was detected.

protocol_version: The protocol version the client attempted to negotiate is recognized but not supported.

insufficient_security: Returned instead of handshake_failure when a negotiation has failed specifically because the server requires ciphers more secure than those supported by the client.

internal_error: An internal error unrelated to the peer or the correctness of the protocol makes it impossible to continue.

The remainder of the new alerts includes the following:

decrypt_error: A handshake cryptographic operation failed, including being unable to verify a signature, decrypt a key exchange, or validate a finished message.

user_canceled: This handshake is being canceled for some reason unrelated to a protocol failure.

no_renegotiation: Sent by a client in response to a hello request or by the server in response to a client hello after initial handshaking. Either of these messages would normally result in renegotiation, but this alert indicates that the sender is not able to renegotiate. This message is always a warning.

Cipher Suites: - There are several small differences between the cipher suites available under SSLv3 and under TLS:

- Key Exchange: TLS supports all of the key exchange techniques of SSLv3 with the exception of Fortezza.
- Symmetric Encryption Algorithms: TLS includes all of the symmetric encryption algorithms found in SSLv3, with the exception of Fortezza.

Client Certificate Types: - TLS defines the following certificate types to be requested in a certificate_request message: rsa_sign, dss_sign, rsa_fixed_dh, and dss_fixed_dh. These are all defined in SSLv3. In addition, SSLv3 includes rsa_ephemeral_dh, dss_ephemeral_dh, and fortezza_kea. Ephemeral Diffie-Hellman involves signing the Diffie-Hellman parameters with either RSA or DSS; for TLS, the rsa_sign and dss_sign types are used for that function; a separate signing type is not needed to sign Diffie-Hellman parameters. TLS does not include the Fortezza scheme.

Certificate_Verify and Finished Messages: - In the TLS certificate_verify message, the MD5 and SHA-1 hashes are calculated only over handshake_messages. For SSLv3, the hash calculation also included the master secret and pads. These extra fields were felt to add no additional security. As with the finished message in SSLv3, the finished message in TLS is a hash based on the shared master_secret, the previous handshake messages, and a label that identifies client or server. The calculation is somewhat different. For TLS, we have

```
PRF(master_secret, finished_label, MD5(handshake_messages)||
  SHA-1(handshake_messages))
```

where finished_label is the string "client finished" for the client and "server finished" for the server.

Cryptographic Computations: - The pre_master_secret for TLS is calculated in the same way as in SSLv3. As in SSLv3, the master_secret in TLS is calculated as a hash function of the pre_master_secret and the two hello random numbers. The form of the TLS calculation is different from that of SSLv3 and is defined as follows:

```
master_secret = PRF(pre_master_secret, "master secret",
            ClientHello.random || ServerHello.random)
```

The algorithm is performed until 48 bytes of pseudorandom output are produced. The calculation of the key block material (MAC secret keys, session encryption keys, and IVs) is defined as follows. As with SSLv3, the key_block is a function of the master_secret and the client and server random numbers, but for TLS the actual algorithm is different.

```
key_block = PRF(master_secret, "key expansion",
        SecurityParameters.server_random ||
        SecurityParameters.client_random)
```

Padding: - In SSL, the padding added prior to encryption of user data is the minimum amount required so that the total size of the data to be encrypted is a multiple of the cipher's block length. In TLS, the padding can be any amount that results in a total that is a multiple of the cipher's block length, up to a maximum of 255 bytes. For example, if the plaintext (or compressed text if compression is used) plus MAC plus padding length, in byte is 79 bytes long, then the padding length, in bytes, can be 1, 9, 17, and so on, up to 249. A variable padding length may be used to frustrate attacks based on an analysis of the lengths of exchanged messages.

Secure Electronic Transaction

SET is an open encryption and security specification designed to protect credit card transactions on the Internet. It is a set of security protocols and formats that enables users to employ the existing credit card payment infrastructure on an open network. SET provides three services. They are,

- Provides a secure communications channel among all parties involved in a transaction
- Provides trust by the use of *X.509v3* digital certificates
- Ensures privacy because the information is only available to parties in a transaction when and where necessary.

SET Overview

Requirements

- **Provide confidentiality of payment and ordering information:** It is necessary to assure cardholders that this information is safe and accessible only to the intended recipient. Confidentiality also reduces the risk of fraud by either party to the transaction or by malicious third parties. SET uses encryption to provide confidentiality.
- **Ensure the integrity of all transmitted data:** That is, ensure that no changes in content occur during transmission of SET messages. Digital signatures are used to provide integrity.
- **Provide authentication that a cardholder is a legitimate user of a credit card account: A** mechanism that links **a** cardholder to **a** specific account number reduces the incidence of fraud and the overall cost of payment processing. Digital signatures and certificates are used to verify that a cardholder is a legitimate user of a valid account.
- **Provide authentication that a merchant can accept credit card transaction through** its **relationship with a financial institution:** This is the complement the preceding requirement. Cardholders need to be able to identify merchants with whom they can conduct secure transactions.
- **Ensure the use of the best security practices and** system design techniques **to protect all legitimate parties in an electronic commerce transaction:** SET is a well-tested specification based on highly secure cryptographic algorithm and protocols.
- **Create** a **protocol that neither depends on transport security mechanisms nor prevents their use: SET** can securely operate over a "raw" TCP/IP stack. SET does not interfere with the use of other security mechanisms such as IPSec and SSL/TLS.

- **Facilitate and encourage interoperability among software and network providers:** The SET protocols and formats are independent of hardware platform operating system, and Web software.

Key Features of SET

To meet the requirements just outlined, SET incorporates the following features: **Confidentiality of** information: Cardholder account and payment information is secured as it travels across the network. An interesting and important feature of SET is that it prevents the merchant from learning the cardholder's credit card number; this is only provided to the issuing bank. Conventional encryption by DES is used to provide confidentiality.

- **Integrity of data:** Payment information sent from cardholders to merchants includes order information, personal data, and payment instructions. SET guarantees that these message contents are not altered in transit. RSA digital signatures, using SHA-1 hash codes, provide message integrity. Certain messages are also protected by HMAC using SHA-1.
- **Cardholder account authentication:** SET enables merchants to verify that a cardholder is a legitimate user of a valid card account number. SET uses X.509v3 digital certificates with RSA signatures for this purpose.
- **Merchant authentication:** SET enables cardholders to verify that a merchant has a relationship with a financial institution allowing it to accept payment car S. SET uses X.509v3 digital certificates with RSA signatures for this purpose.

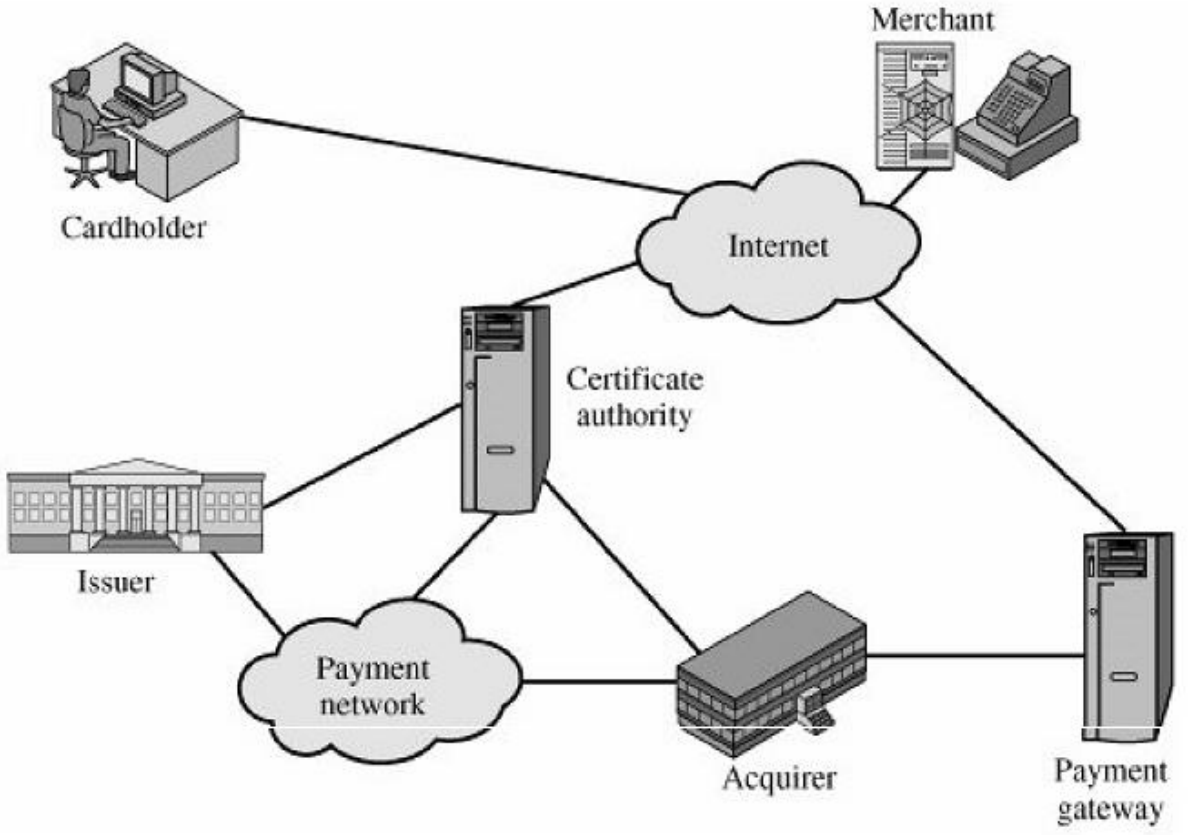
SET Participants

Figure indicates the participants in the SET system, which includes the following:

- **Cardholder:** In the electronic environment, consumers and corporate purchasers interact with merchants from personal computers over the Internet. A cardholder is an authorized holder of a payment card that has been issued by an issuer.

- Merchant: A merchant is a person or organization that has goods or services to sell to the cardholder. Typically, these goods and services are offered via a Web site or by electronic mail. A merchant that accepts payment cards must have a relationship with an acquirer.
- Issuer: This is a financial institution, such as a bank, that provides the cardholder with the payment card. Typically, accounts are applied for and opened by mail or in person. Ultimately, it is the issuer that is responsible for the payment of the debt of the cardholder.
- **Acquirer:** This is a financial institution that establishes an account with a merchant and processes payment card authorizations and payments. Merchants will usually accept more than one credit card brand but do not want to deal with multiple bankcard associations or with multiple individual issuers. The acquirer provides authorization to the merchant that a given card account is active and that the proposed purchase does not exceed the credit limit. The acquirer also provides electronic transfer of payments to the merchant's account. Subsequently, the issuer reimburses the acquirer over some sort of payment network for electronic funds transfer.
- **Payment** gateway: This is a function operated by the acquirer or a designated third party that processes merchant payment messages. The payment gateway interfaces between SET and the existing bankcard payment networks for authorization and payment functions. The merchant exchanges SET messages with the payment gateway over the Internet, while the payment gateway has Some direct or network connection to the acquirer's financial processing system.

- **Certification authority (CA):** This is an entity that is trusted to issue X.509v3 public-key certificates for cardholders, merchants, and payment gateways. The success of SET will depend on the existence of a CA infrastructure available for this purpose. A hierarchy of CAs issued, so that a root authority need not directly certify participants.



The sequence of events

1. **The customer** opens an account. The customer obtains a credit card account, such good as MasterCard or Visa, with a bank that supports electronic payment and SET.
2. **The customer receives** a certificate. After suitable verification of identity, the customer receives an X.509v3 digital certificate, which is signed by the bank. The certificate verifies the customer's RSA public key and its expiration date. It also establishes a relationship, guaranteed by the bank, between the customer's key pair and his or her credit card.
3. **Merchants have their** own **certificates.** A merchant who accepts a certain brand hash of card must be in possession of two certificates for two public keys owned by is the merchant: one for signing messages, and one for key exchange. The merchant also needs a copy of the payment gateway's public-key certificate.

4. **The customer places an order.** This is a process that may involve the customer first browsing through the merchant's Web site to select items and determine the price. The customer then sends a list of the items to be purchased to the merchant, who returns an order form containing the list of items, their price, is in a total price, and an order number.
5. **The merchant is verified.** In addition to the order form, the merchant sends a copy of its certificate, so that the customer can verify that he or she is dealing with a valid store.
6. **The order and payment are sent.** The customer sends both a order and payment information to the merchant, along with the customer's certificate. The then order confirms the purchase of the items in the order form. The payment contains credit card details. The payment information is encrypted in such a way that it cannot be read by the merchant. The customer's certificate enables the merchant to verify the customer.
7. **The merchant requests payment** authorization. The merchant sends the pay men information to the payment gateway, requesting authorization that the customer's available credit is sufficient for this purchase.
8. **The merchant confirms the order.** The merchant sends confirmation of the order to the customer.
9. **The merchant** provides **the goods or service.** The merchant ships the goods or provides the service to the customer.
10. **The merchant requests payment.** This request is sent to the payment gateway, which handles all of the payment processing.

Dual Signature

The purpose of the dual signature is to link two messages that are intended for two different recipients. In this case, the customer wants to send the order information (OI) to the merchant and the payment information (PI) to the bank. The merchant does not need to know the customer's credit card number, and the bank does not need to know the details of the customer's order. The customer is afforded extra protection in terms of privacy by keeping these two items

separate. However, the two items must be linked in a way that can be used to resolve disputes if necessary. The link is needed so that the customer can prove that this payment is intended for this order and not for some other goods or service.

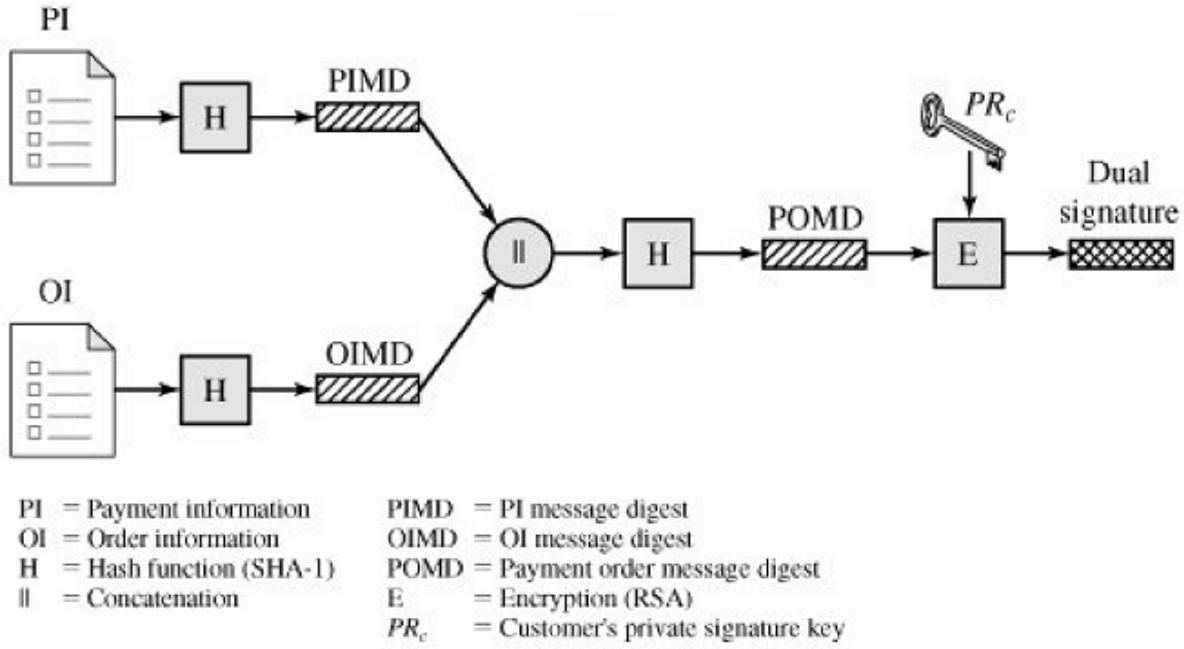
To see the need for the link, suppose that the customers send the merchant two messages—a signed OI and a signed PI—and the merchant passes the PI on to the bank. If the merchant can capture another OI from this customer, the merchant could claim that this OI goes with the PI rather than the original OI. The linkage prevents this.

Figure shows the use of a dual signature to meet the requirement of the preceding paragraph. The customer takes the hash (using SHA-1) of the PI and the hash of the OI. These two hashes are then concatenated and the hash of the result is taken. Finally, the customer encrypts the final hash with his or her private signature key, creating the dual signature. The operation can be summarized as

$$DS = E(PRc, [H(H(PI) \| H(OI))]$$

where *PRc* is the customer's private signature key. Now suppose that the merchant is in possession of the dual signature (DS), the OI, and the message digest for the PI (PIMD). The merchant also has the public key of the customer, taken from the customer's certificate.

Figure 9. Construction of Dual Signature



Then the merchant can compute the following two quantities:

H(H(PIMS || H(OI)) and D(Puc,DS)

where PUc is the customer's public signature key. If these two quantities are equal, then the merchant has verified the signature. Similarly, if the bank is in possession of DS, PI, the message digest for OI (OTMD), and the customer's public key, then the bank can compute the following:

H(H[OI] || OIMD) and D(PUc, DS)

Again, if these two quantities are equal, then the bank has verified the signature. In summary,

The merchant has received OI and verified the signature.

The bank has received PI and verified the signature.

The customer has linked the OI and PI and can prove the linkage.

Firewall

Firewalls can be an effective means of protecting a local system or network of systems from network based security threats while at the same time affording access to the outside world via wide area networks and internet.

Design Goals

- All traffic from inside to outside must pass through the firewall. This is achieved by physically blocking all access to the local network except via firewall.
- Only authorized traffic will be allowed to pass.
- The firewall itself is immune to penetration. This implies the use of trusted systems with a secure operating system.

The techniques used to control access

Service control

It determines the types of Internet services that can be accessed, inbound or outbound. The firewall may filter traffic on the basic of IP address and TCP port number. It may provide proxy software that receives and interprets each service request before passing it on or it may host the server software itself such as Web mail service.

Direction control

It determines the direction in which particular service request may be initiated and allowed to flow through firewall.

User control

It controls access to service according to which user is attempting to access it. The feature is applied to the users inside the firewall perimeter. It may also apply to the incoming traffic from external users.

Behavior control

Controls how particular service is used. For example firewall may filter email to eliminate Spam.

Scope of firewall

- A firewall defines a single choke point that keeps unauthorized users out of the protected network, prohibits potentially vulnerable services from entering or leaving the network,

and provides protection from various kinds of IP spoofing and routing attacks. The use of a single choke point simplifies security management because security capabilities are consolidated on a single system or set of systems.

- A firewall provides a location for monitoring security related events. Audits and alarms can be implemented on the firewall system.

- A firewall is a convenient platform for several Internet functions that are not security related. These include a network address translator, which maps local security address to Internet addresses and a network management function that audits or logs Internet usage.

- A firewall can serves as a platform for IPSec.

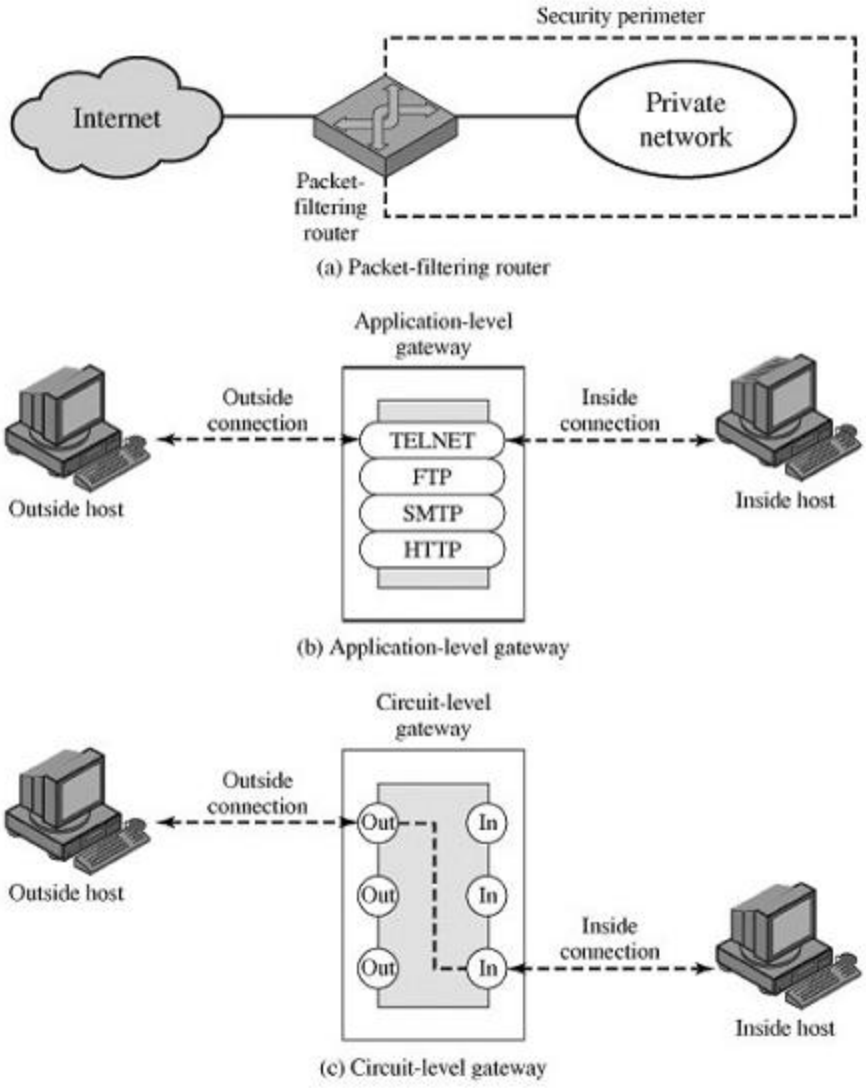
Limitations

- Firewall cannot protect against attack that bypass the firewall.
- The firewall does not protect against internal threats
- The firewall cannot protect against the transfer of virus infected programs or files.

Types of firewall

The three common types of firewalls are

- Packet Filters
- Application Level gateways
- Circuit level gateways



Packet Filtering Router

- It applies a set of rules to each incoming and outgoing IP packets and then forwards or discard the packet. The router is configured to filter packets going in both the directions. Filtering rules are based on information contained in a network packet.
- Source IP Address: The IP address of the system that originated the IP packet.
- Destination IP Address: The IP address of the system the IP packet is trying to reach.
- Source and destination transport-level address: The transport level (e.g., TCP or UDP) port number, which defines applications such as SNMP or TELNET
- IP protocol field: Defines the transport protocol

- Interface: For a router with three or more ports, which interface of the router the packet came from or which interface of the router the packet is destined for.
- The packet filter is typically set up as a list of rules based on matches to field in the IP or TCP header. If there is a match to one of the rules, that rule is invoked to determine whether to forward or discard the packet. If there is no match to any rule, then a default action is taken. Two default policies are possible:

Default = discard: That which is not expressly permitted is prohibited.

Default = forward: That which is not expressly prohibited is permitted.

The default discard policy is the more conservative. Initially, everything is blocked, and services must be added on a case-by-case basis. This policy is more visible to users, who are more likely to see the firewall as a hindrance. The default forward policy increases ease of use for end users but provides reduced security; the security administrator must, in essence, react to each new security threat as it becomes known.

One advantage of a packet-filtering router is its simplicity. Also, packet filters typically are transparent to users and are very fast.

Weaknesses of packet filter firewalls:

- Because packet filter firewalls do not examine upper-layer data, they cannot prevent attacks that employ application-specific vulnerabilities or functions.
- Because of the limited information available to the firewall, the logging functionality present in packet filter firewalls is limited.
- Most packet filter firewalls do not support advanced user authentication schemes. Once again, this limitation is mostly due to the lack of upper-layer functionality by the firewall.
- They are generally vulnerable to attacks and exploits that take advantage of problems within the TCP/IP specification and protocol stack, such as network layer address spoofing.
- Due to the small number of variables used in access control decisions, packet filter firewalls are susceptible to security breaches caused by improper configurations.

Some of the attacks that can be made on packet-filtering routers and the appropriate countermeasures are the following:

- **IP address spoofing:** The intruder transmits packets from the outside with a source **IP** address field containing an address of an internal host. The attacker hopes that the use of a spoofed address will allow penetration of systems that employ simple source address security, in which packets from specific trusted internal hosts are accepted. The countermeasure is to discard packets with an inside source address if the packet arrives on an external interface.
- **Source routing attacks:** The source station specifies the route that a packet should take as it crosses the Internet, in the hopes that this will bypass security measures that do not analyze the source routing information. The countermeasure is to discard all packets that use this option.
- **Tiny fragment attacks:** The intruder uses the IP fragmentation option to create extremely small fragments and force the TCP header information into a separate packet fragment. This attack is designed to circumvent filtering rules that depend on TCP header information. The attacker hopes that only the filtering router examines the first fragment and that the remaining fragments are passed through. Discarding all packets where the protocol type is TCP and the IP Fragment Offset iS equal to 1 can defeat a tiny fragment attack.

Application-Level Gateway

An application-level gateway, also called a proxy server, acts as a relay of application-level traffic. The user contacts the gateway **using** TCP/IP application, such as Telnet or FTP, and the gateway asks the User for the name of the remote host to be accessed. When the user responds and provides a valid user ID and authentication information, the gateway contacts the application on- the remote host and relays TCP segments containing the application data between the two endpoints. If the gateway does not implement the proxy code for a specific application, the service is not supported and cannot be forwarded across the firewall. Further, the gateway can be configured to support only specific features of an application that the network administrator considers acceptable while denying all other features.

Application-level gateways tend to be more secure than packet filters. Rather than trying to deal with the numerous possible combinations that are to be allowed and forbidden at the TCP and IP level, the application-level gateway need only scrutinize a few allowable applications. In addition, it is easy to log and audit all incoming traffic at the application level.

A prime disadvantage of this type of gateway is the additional processing overhead on each connection In effect, there are two spliced connections between the end users, with the gateway at the splice point, and the gateway must examine and forward all traffic in both directions.

Circuit-Level Gateway

A third type of firewall is the circuit-level gateway (Figure 20.lc). This can be a stand-alone system or it can be a specialized function performed by an application- level gateway for certain applications. A circuit-level gateway does not permit an end-to-end TCP connection; rather, the gateway sets up two TCP connections, one between itself and a TCP user on an inner host and one between itself and a TCP user on an outside host. Once the two connections are established, the gateway typically relays TCP segments from one connection to the other without examining the contents. The security function consists of determining which connections will be allowed.

A typical use of circuit-level gateways is a situation in which the system administrator trusts the internal users. The gateway can be configured to support application-level or proxy service on inbound connections and circuit-level functions or outbound connections. In this configuration, the gateway can incur the proceSs1n overhead of examining incoming application data for forbidden functions but not incur that overhead on outgoing data.

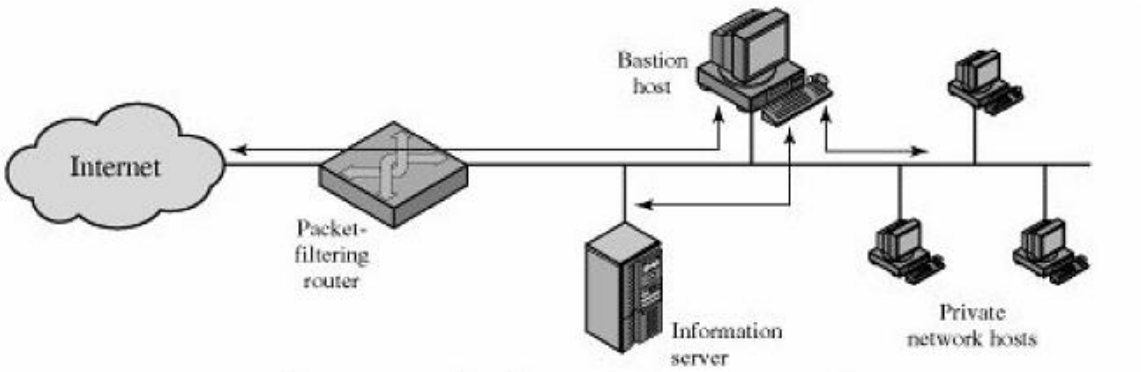
Firewall Configurations

In addition to the use of a simple configuration consisting of a single system, such as a single packet-filtering router or a single gateway more complex configurations are possible and indeed more common.

In the **screened host firewall, single-homed bastion** configuration, the firewall consists of two systems: a packet-filtering router and a bastion host. Typically, the router is configured so that

- For traffic from the Internet, only IP packets destined for the bastion host are allowed in.
- For traffic from the internal network, only IP packets from the bastion host are allowed out.

The bastion host performs authentication and proxy functions. This configuration has greater security than simply a packet-filtering router or an application- level gateway alone, for two reasons. First, this configuration implements both packet-level and application-level filtering, allowing for considerable flexibility in defining security policy. Second, an intruder must generally penetrate two separate systems before the security of the internal network is compromised.

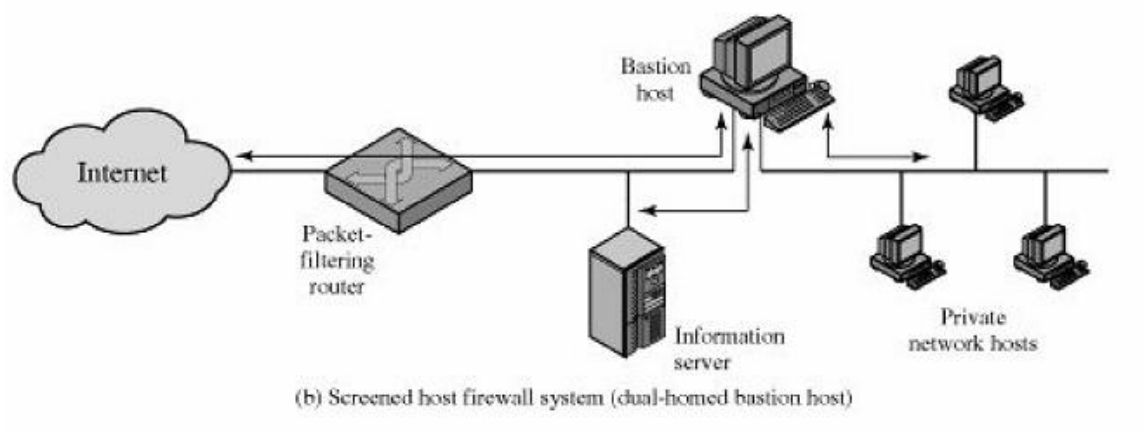


(a) Screened host firewall system (single-homed bastion host)

This configuration also affords flexibility in providing direct Internet access. For example, the internal network may include a public information server, such as a Web server, for which a high level of security is not required. In that case, the router can be configured to allow direct traffic between the information server and the Internet.

In the single-homed configuration just described, if the packetfilter1flg router is completely compromised, traffic could flow directly through the router between the Internet and other hosts on the private network. The **screened host firewall, dual-homed bastion** configuration physically prevents such a security breach. The advantages of dual layers of security that were present in the previous configuration are present here as well. Again, an information server or

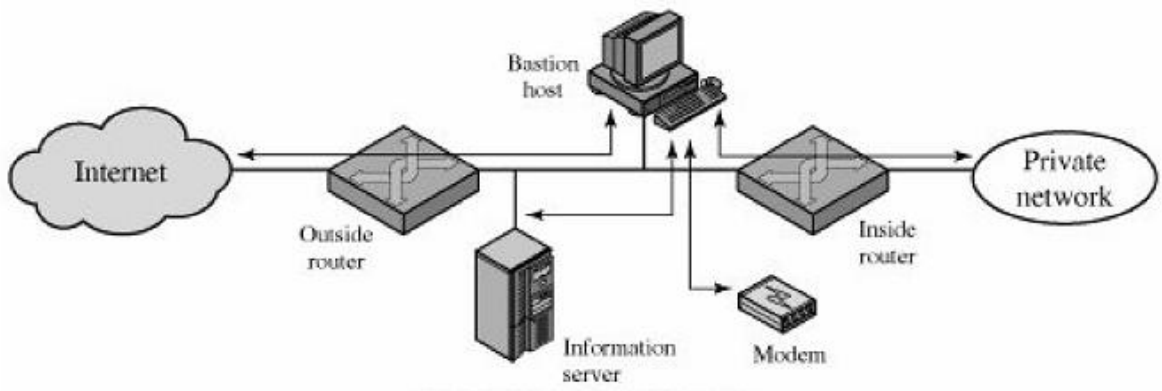
other hosts can be allowed direct communication with the router if this is in accord with the security policy.



(b) Screened host firewall system (dual-homed bastion host)

The **screened subnet firewall** configuration of is the most secure of those we have considered. In this configuration, two packet-filtering routers art used, one between the bastion host and the Internet and one between the bastion host and the internal network. This configuration creates an isolated sub network which may consist of simply the bastion host but may also include one or more information servers and modems for dial-in capability. Typically, both the Internet and the internal network have access to hosts on the screened subnet, but traffic across the screened subnet is blocked. This configuration offers several advantages:

- There are now three levels of defense to thwart intruders.
- The outside router advertises only the existence of the screened subnet to the Internet; therefore, the internal network is invisible to the Internet.
- The inside router advertises only the existence of the screened subnet to the internal network; therefore, the systems on the inside network cannot construct direct routes to the Internet.



(c) Screened-subnet firewall system