IP SECURITY OVERVIEW UNIT- 4

Definition: Internet Protocol security (IPSec) is a framework of open standards for protecting communications over Internet Protocol (IP) networks through the use of cryptographic security services. IPSec supports network-level peer authentication, data origin authentication, data integrity, data confidentiality (encryption), and replay protection.

Need for IPSec

In Computer Emergency Response Team (CERT)'s 2001 annual report it listed 52,000 security incidents in which most serious types of attacks included **IP spoofing**, in which intruders create packets with false IP addresses and exploit applications that use authentication based on IP and various forms of **eavesdropping and packet sniffing**, in which attackers read transmitted information, including logon information and database contents. In response to these issues, the IAB included authentication and encryption as necessary security features in the next-generation IP i.e. IPv6.

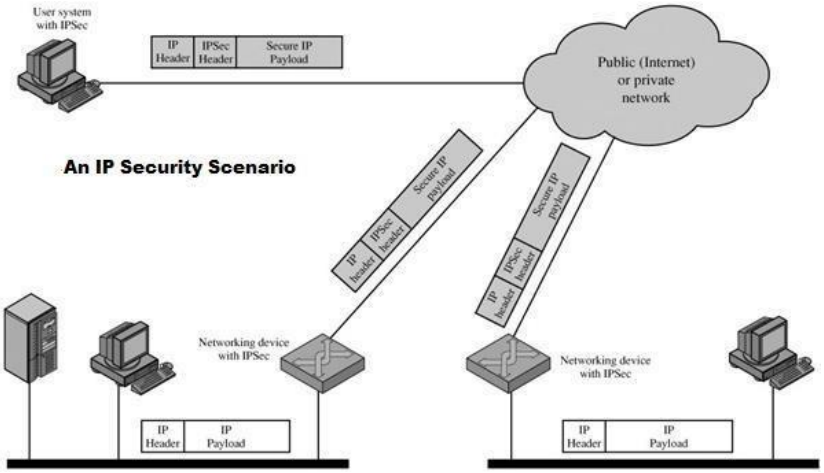
Applications of IPSec

IPSec provides the capability to secure communications across a LAN, across private and public wide area networks (WAN's), and across the Internet.

- ***Secure branch office connectivity over the Internet***: A company can build a 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 (ISP) and gain secure access to a company network. This reduces the cost of toll charges for travelling employees and telecommuters.
- ***Establishing extranet and intranet connectivity with partners***: IPSec can be used to secure communication with other organizations, ensuring authentication and confidentiality and providing a key exchange mechanism.
- ***Enhancing electronic commerce security***: Even though some Web and electronic commerce applications have built-in security protocols, the use of IPSec enhances that security.

The principal feature of IPSec enabling it to support varied applications is that it can encrypt and/or authenticate all traffic at IP level. Thus, all distributed applications, including remote logon, client/server, e-mail, file transfer, Web access, and so on, can be secured.

The following figure shows a typical scenario of IPSec usage. An organization maintains LANs at dispersed locations. Non secure IP traffic is conducted on each LAN.



The IPSec protocols operate in networking devices, such as a router or firewall that connect each LAN to the outside world. The IPSec networking device will typically encrypt and compress all traffic going into the WAN, and decrypt and decompress traffic coming from the WAN; these operations are transparent to workstations and servers on the LAN. Secure transmission is also possible with individual users who dial into the WAN. Such user workstations must implement the IPSec protocols to provide security.

Benefits of IPSec

The benefits of IPSec are listed below:

- IPSec in a firewall/router provides strong security to all traffic crossing the perimeter
- IPSec in a firewall is resistant to bypass
- IPSec is below transport layer(TCP,UDP), hence transparent to applications
- IPSec can be transparent to end users
- IPSec can provide security for individual users if needed (useful for offsite workers and setting up a secure virtual subnetwork for sensitive applications)

Routing Applications

IPSec also plays a vital role in the routing architecture required for internetworking. It assures that:

- router advertisements come from authorized routers
- neighbor advertisements come from authorized routers
- redirect messages come from the router to which initial packet was sent
- A routing update is not forged

IP Security Architecture

To understand IP Security architecture, we examine IPSec documents first and then move on to IPSec services and Security Associations.

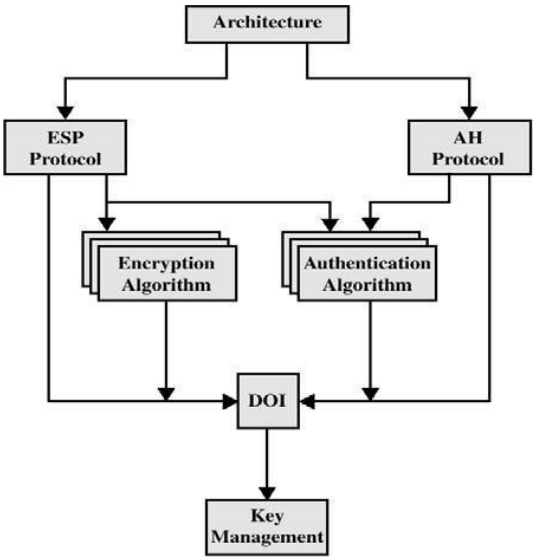
IPSec Documents

The IPSec specification consists of numerous documents. The most important of these, issued in November of 1998, are RFCs 2401, 2402, 2406, and 2408:

- RFC 2401: An overview of a security architecture
- RFC 2402: Description of a packet authentication extension to IPv4 and IPv6
- RFC 2406: Description of a packet encryption extension to IPv4 and IPv6
- RFC 2408: Specification of key management capabilities

Support for these features is mandatory for IPv6 and optional for IPv4. In both cases, the security features are implemented as extension headers that follow the main IP header.

The extension header for authentication is known as the Authentication header; that for encryption is known as the Encapsulating Security Payload (ESP) header. In addition to these four RFCs, a number of additional drafts have been published by the IP Security Protocol Working Group set up by the IETF. The documents are divided into seven groups, as depicted in following figure:



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

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

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

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

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

• **Key Management**: Documents that describe key management schemes.

• **Domain of Interpretation (DOI):** Contains values needed for the other documents to relate to each other. These include identifiers for approved encryption and authentication algorithms, as well as operational parameters such as key lifetime.

IPSec Services

IPSec architecture makes use of two major protocols (i.e., Authentication Header and ESP protocols) for providing security at IP level. This facilitates the system to beforehand choose an algorithm to be implemented, security protocols needed and any cryptographic keys required to provide requested services. The IPSec services are as follows:

Connectionless Integrity:- Data integrity service is provided by IPSec via AH which prevents the data from being altered during transmission.

Data Origin Authentication:- This IPSec service prevents the occurrence of

replay attacks, address spoofing etc., which can be fatal

- **Access Control**:- The cryptographic keys are distributed and the traffic flow is controlled in both AH and ESP protocols, which is done to accomplish access control over the data transmission.
- **Confidentiality**:- Confidentiality on the data packet is obtained by using an encryption technique in which all the data packets are transformed into ciphertext packets which are unreadable and difficult to understand.
- **Limited Traffic Flow Confidentiality**:- This facility or service provided by IPSec ensures that the confidentiality is maintained on the number of packets transferred or received. This can be done using padding in ESP.
- **Replay packets Rejection**:- The duplicate or replay packets are identified and discarded using the sequence number field in both AH and ESP.

	AH	ESP (encryption only)	ESP (encryption plus authentication)
Access control	✔	✔	✔
Connectionless integrity	✔		✔
Data origin authentication	✔		✔
Rejection of replayed packets	✔	✔	✔
Confidentiality		✔	✔
Limited traffic flow confidentiality		✔	✔

SECURITY ASSOCIATIONS

Since IPSEC is designed to be able to use various security protocols, it uses Security Associations (SA) to specify the protocols to be used. SA is a database record which specifies security parameters controlling security operations. They are referenced by the sending host and established by the receiving host. An index parameter called the Security Parameters Index (SPI) is used. SAs are in one direction only and a second SA must be established for the transmission to be bi-directional. 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 Database that defines the parameters associated with each SA. A security association is 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 of the Sequence Number Counter should generate an auditable event and prevent further transmission of packets on this SA (required for all implementations).

• **Anti-Replay Window:** Used to determine whether an inbound AH or ESP packet is a replay
• **AH Information:** Authentication algorithm, keys, key lifetimes, and related

parameters being used with AH (required for AH implementations).

• **ESP Information:** Encryption and authentication algorithm, keys, initialization values, key lifetimes, and related parameters being used with ESP (required for ESP implementations).

• **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 (required for all implementations).

• **IPSec Protocol Mode:** Tunnel, transport, or wildcard (required for all implementations). These modes are discussed later in this section.

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

Transport and Tunnel Modes

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

	Transport Mode SA	Tunnel Mode SA
AH	**Authenticates** IP payload and selected portions of IP header and IPv6 extension headers	**Authenticates** entire inner IP packet plus selected portions of outer IP header
ESP	**Encrypts** IP payload and any IPv6 extesion header	**Encrypts** inner IP packet
ESP with authentication	**Encrypts** IP payload and any IPv6 extesion header. **Authenticates** IP payload but no IP header	**Encrypts** inner IP packet. **Authenticates** inner IP packet

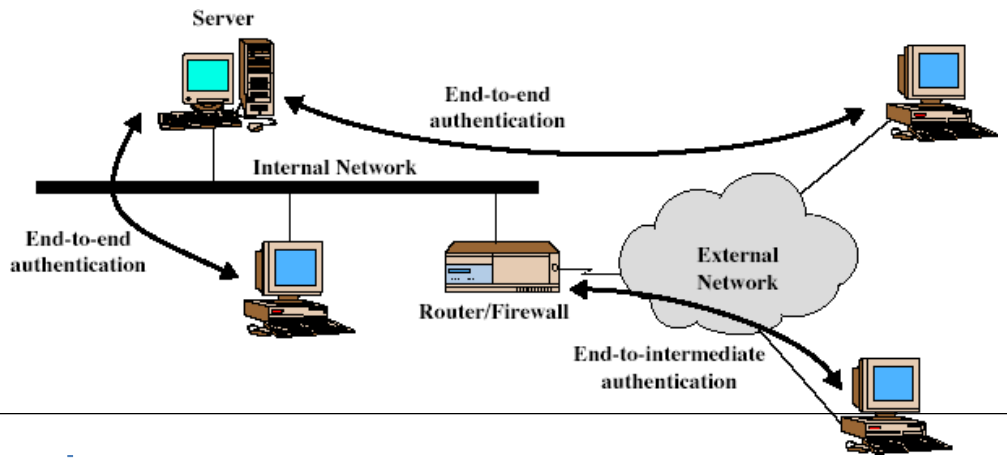
IP sec can be used (both AH packets and ESP packets) in two modes

• **Transport mode**: the IP sec header is inserted just after the IP header –this contains the security information, such as SA identifier, encryption, authentication

- Typically used in end-to-end communication IP header not protected

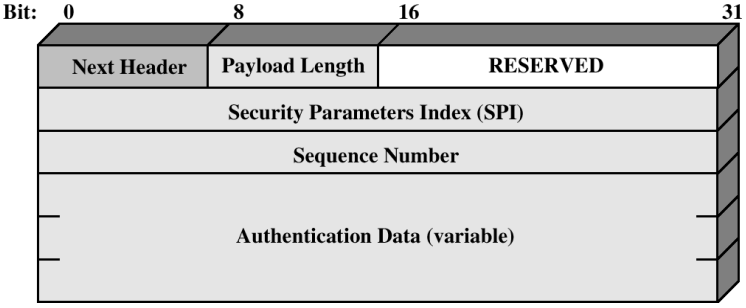
• **Tunnel mode**: the entire IP packet, header and all, is encapsulated in the body of a new IP packet with a completely new IP header

- Typically used in firewall-to-firewall communication Provides protection for the whole IP packet
- No routers along the way will be able (and will not need) to check the content of the packets



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 AH also guards against the replay attack. Authentication is based on the use of a message authentication code (MAC), hence the two parties must share a secret key. The Authentication Header consists of the following fields:



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, discussed later.
- **Authentication Data (variable):** A variable-length field (must be an integral number of 32-bit words) that contains the Integrity Check Value (ICV), or MAC, for this packet.

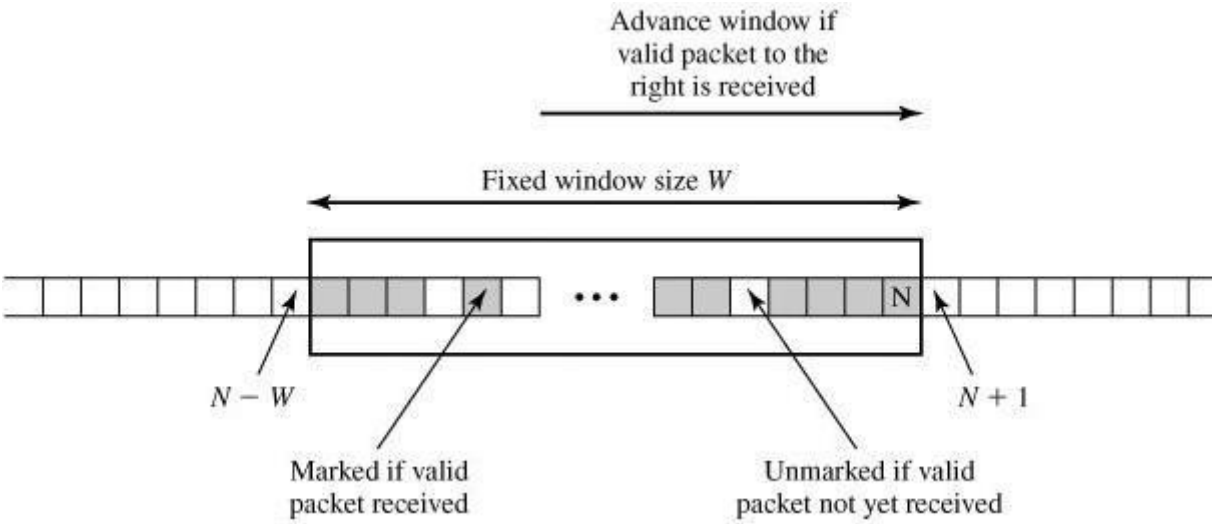
Anti-Replay Service

Anti-replay service is designed to overcome the problems faced due to replay attacks in which an intruder intervenes the packet being transferred, make one or more duplicate copies of that authenticated packet and then sends the packets to the desired destination, thereby causing inconvenient processing at the destination node. The Sequence Number field is designed to thwart such attacks.

When a new SA is established, the sender initializes a sequence number counter to 0. Each time that a packet is sent on this SA, the sender increments the counter and places the value in the Sequence Number field. Thus, the first value to be used is 1. This value goes on increasing with respect to the number of packets being transmitted. The sequence number field in each packet represents the value of this counter. The maximum value of the sequence number field can go up to 2_{32}-1. If the limit of 2_{32}-1 is reached, the sender should terminate this SA and negotiate a new SA with a new key.

The IPSec authentication document dictates that the receiver should implement a

window of size W, with a default of W = 64. The right edge of the window represents the highest sequence number, N, so far received for a valid packet. For any packet with a sequence number in the range from N-W+1 to N that has been correctly received (i.e., properly authenticated), the corresponding slot in the window is marked as shown. Inbound processing proceeds as follows when a packet is received:



Antireplay Mechanism

1. If the received packet falls within the window and is new, the MAC is checked. If the packet is authenticated, the corresponding slot in the window is marked.
2. If the received packet is to the right of the window and is new, the MAC is checked. If the packet is authenticated, the window is advanced so that this sequence number is the right edge of the window, and the corresponding slot in the window is marked.
3. If the received packet is to the left of the window, or if authentication fails, the packet is discarded; this is an auditable event.

Integrity Check Value

ICV is the value present in the authenticated data field of ESP/AH, which is used to determine any undesired modifications made to the data during its transit. ICV can also be referred as MAC or part of MAC algorithm. MD5 hash code and SHA-1 hash code are implemented along with HMAC algorithms i.e.,

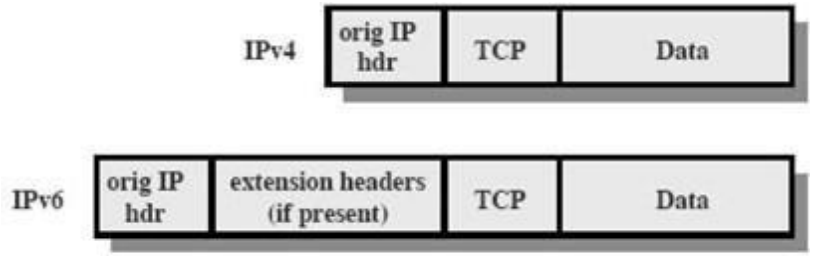
- HMAC-MD5-96
- HMAC-SHA-1-96

In both cases, the full HMAC value is calculated but then truncated by using the first 96 bits, which is the default length for the Authentication Data field. The MAC is calculated over

- IP header fields that either do not change in transit (immutable) or that are predictable in value upon arrival at the endpoint for the AH SA. Fields that may change in transit and whose value on arrival is unpredictable are set to zero for purposes of calculation at both source and destination.
- The AH header other than the Authentication Data field. The Authentication Data field is set to zero for purposes of calculation at both source and destination.
- The entire upper-level protocol data, which is assumed to be immutable in transit (e.g., a TCP segment or an inner IP packet in tunnel mode).

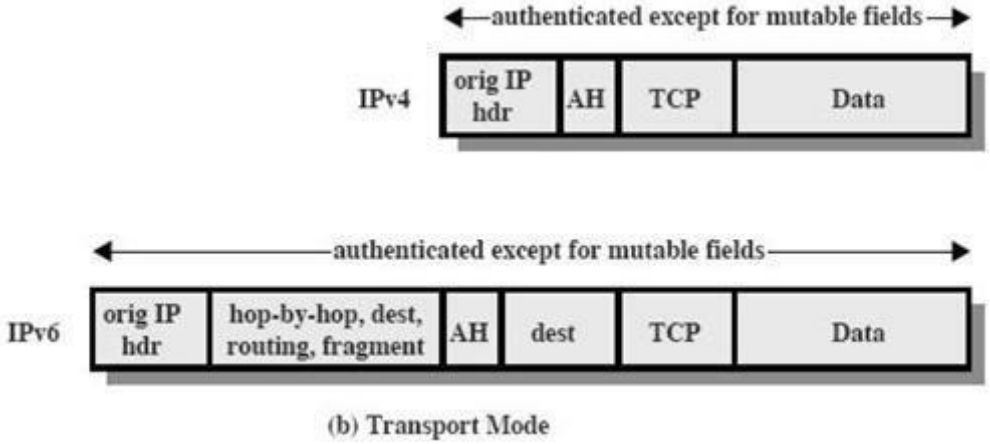
Transport and Tunnel Modes

The following figure shows typical IPv4 and IPv6 packets. In this case, the IP payload is a TCP segment; it could also be a data unit for any other protocol that uses IP, such as UDP or ICMP.

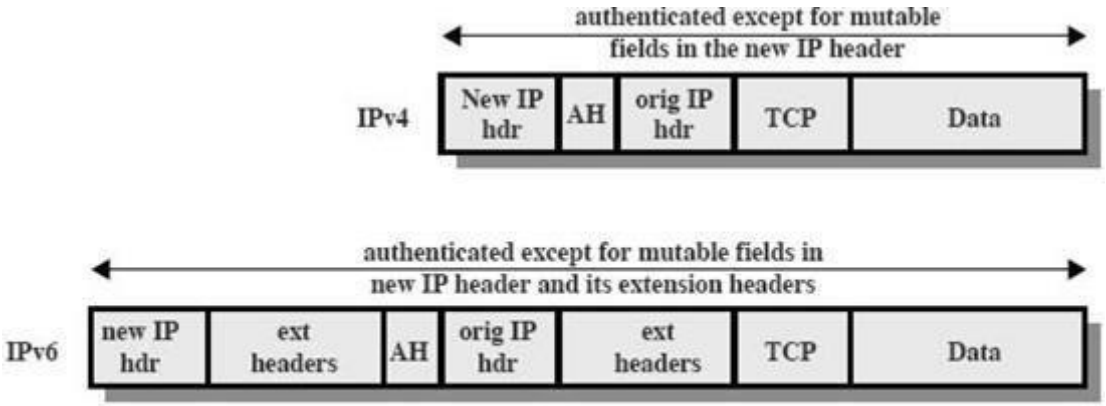


(a) Before Applying AH

For transport mode AH using IPv4, the AH is inserted after the original IP header and before the IP payload (e.g., a TCP segment) shown below. Authentication covers the entire packet, excluding mutable fields in the IPv4 header that are set to zero for MAC calculation. In the context of IPv6, AH is viewed as an end-to-end payload; that is, it is not examined or processed by intermediate routers. Therefore, the AH appears after the IPv6 base header and the hop-by-hop, routing, and fragment extension headers. The destination options extension header could appear before or after the AH header, depending on the semantics desired. Again, authentication covers the entire packet, excluding mutable fields that are set to zero for MAC calculation.



(b) Transport Mode



(c) Tunnel Mode

For tunnel mode AH, the entire original IP packet is authenticated, and the AH is inserted between the original IP header and a new outer IP header. The inner IP header carries the ultimate source and destination addresses, while an outer IP header may contain different IP addresses (e.g., addresses of firewalls or other security gateways). With tunnel mode, the entire inner IP packet, including the entire

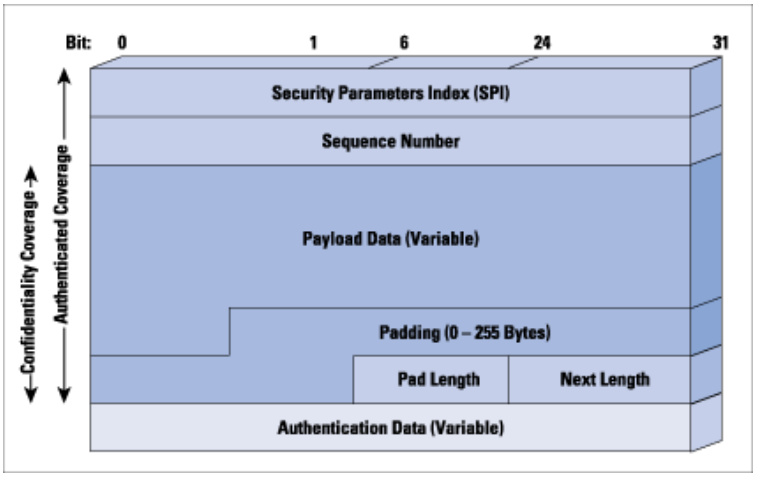
inner IP header is protected by AH. The outer IP header (and in the case of IPv6, the outer IP extension headers) is protected except for mutable and unpredictable fields.

ENCAPSULATING SECURITY PAYLOAD

The Encapsulating Security Payload provides confidentiality services, including confidentiality of message contents and limited traffic flow confidentiality. As an optional feature, ESP can also provide an authentication service.

ESP Format

The following 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, as discussed for AH.

• **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):** This field is used to make the length of the plaintext to be a multiple of some desired number of bytes. It is also added to provideconfidentiality.

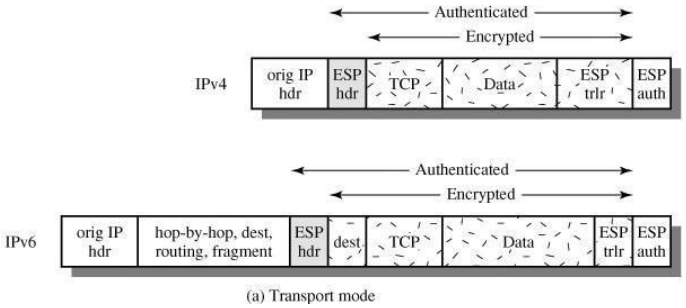
• **Pad Length (8 bits):** Indicates the number of pad bytes immediately preceding this field.

• **Next Header (8 bits):** Identifies the type of data contained in the payload data field by identifying the first header in that payload (for example, an extension header in IPv6, or an upper-layer protocol such as TCP).

• **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.

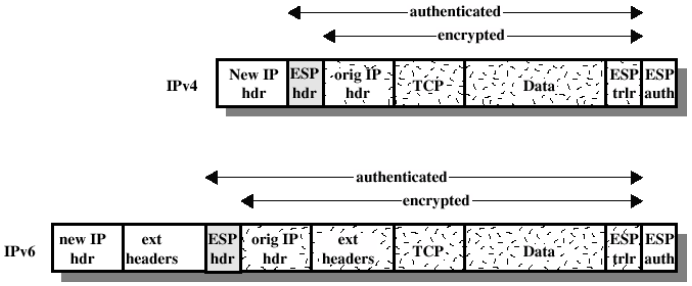
Adding encryption makes ESP a bit more complicated because the encapsulation *surrounds* the payload rather than *precedes* it as with AH: ESP includes header and trailer

Transport Mode ESP



(a) Transport mode

Tunnel Mode ESP

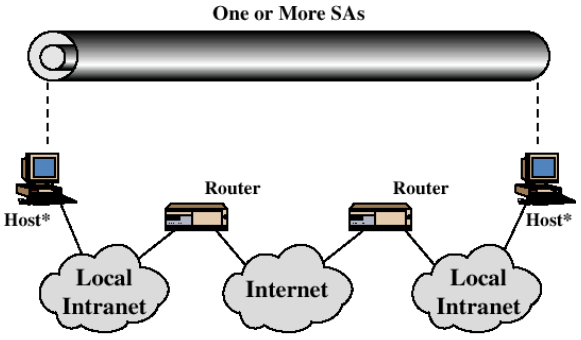


(b) Tunnel Mode

Basic Combinations of Security Associations

The IPSec Architecture document lists four examples of combinations of SAs that must be supported by compliant IPSec hosts (e.g., workstation, server) or security gateways (e.g. firewall, router).

case:-1



(a) Case 1

All security is provided between end systems that implement IPSec. For any two end systems to communicate via an SA, they must share the appropriate secret keys. Among the possible combinations:

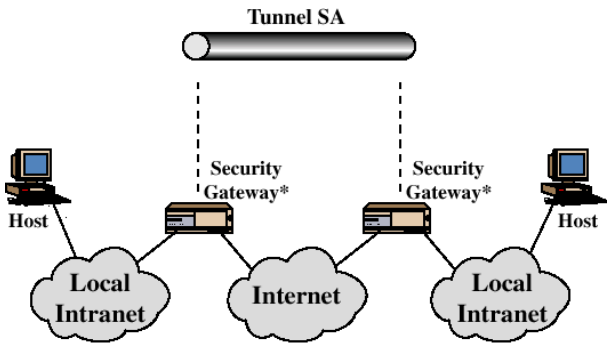
a) AH in transport mode

b) ESP in transport mode

c) ESP followed by AH in transport mode (an ESP SA inside an AH SA)

d) Any one of a, b, or c inside an AH or ESP in tunnel mode

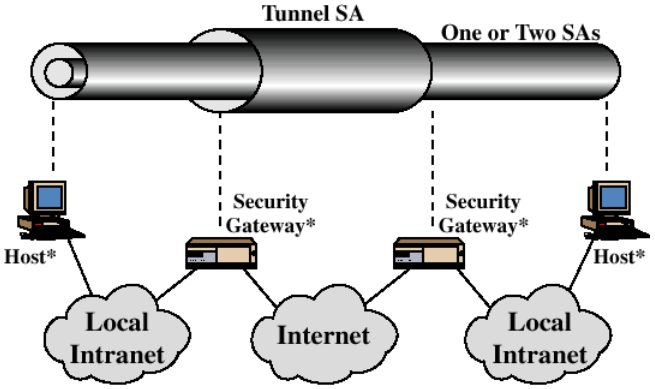
Case:-2



(b) Case 2

Security is provided only between gateways (routers, firewalls, etc.) and no hosts implement IPSec. This case illustrates simple virtual private network support. The security architecture document specifies that only a single tunnel SA is needed for this case. The tunnel could support AH, ESP, or ESP with the authentication option. Nested tunnels are not required because the IPSec services apply to the entire inner packet.

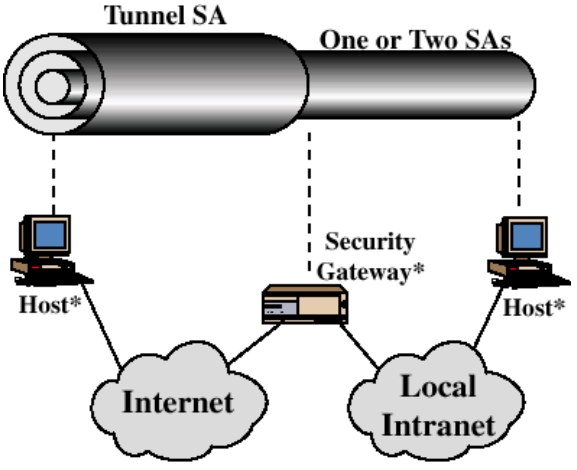
Case-3:-



(c) Case 3

The third combination is similar to the second, but in addition provides security even to nodes. This combination makes use of two tunnels first for gateway to gateway and second for node to node. Either authentication or the encryption or both can be provided by using gateway to gateway tunnel. An additional IPSec service is provided to the individual nodes by using node to node tunnel.

Case:-4



(d) Case 4

This combination is suitable for serving remote users i.e., the end user sitting anywhere in the world can use the internet to access the organizational workstations via the firewall. This combination states that only one tunnel is needed for communication between a remote user and an organizational firewall.

KEY MANAGEMENT

The key management portion of IPSec involves the determination and distribution of secret keys. The IPSec Architecture document mandates support for two types of key management:

• **Manual:** A system administrator manually configures each system with its own keys and with the keys of other communicating systems. This is practical for small, relatively static environments.

• **Automated:** An automated system enables the on-demand creation of keys for SAs and facilitates the use of keys in a large distributed system with an evolving configuration.

The default automated key management protocol for IPSec is referred to as ISAKMP/Oakley and consists of the following elements:

• **Oakley Key Determination Protocol**: Oakley is a key exchange protocol based on the Diffie-Hellman algorithm but providing added security. Oakley is generic in that it does not dictate specific formats.

• **Internet Security Association and Key Management Protocol (ISAKMP):** ISAKMP provides a framework for Internet key management and provides the specific protocol support, including formats, for negotiation of security attributes.

Oakley Key Determination Protocol

Oakley is a refinement of the Diffie-Hellman key exchange algorithm. The Diffie-Hellman algorithm has two attractive features:

• Secret keys are created only when needed. There is no need to store secret keys for a long period of time, exposing them to increased vulnerability.

• The exchange requires no pre-existing infrastructure other than an agreement on the global parameters.

However, Diffie-Hellman has got some weaknesses:

• No identity information about the parties is provided.

• It is possible for a man-in-the-middle attack

• It is computationally intensive. As a result, it is vulnerable to a clogging attack, in which an opponent requests a high number of keys.

Oakley is designed to retain the advantages of Diffie-Hellman while countering its weaknesses.

Features of Oakley

The Oakley algorithm is characterized by five important features:

1. It employs a mechanism known as cookies to thwart clogging attacks.
2. It enables the two parties to negotiate a group; this, in essence, specifies the global parameters of the Diffie-Hellman key exchange.

3. It uses nonces to ensure against replay attacks.
4. It enables the exchange of Diffie-Hellman public key values.
5. It authenticates the Diffie-Hellman exchange to thwart man-in-the-middle attacks.
In clogging attacks, an opponent forges the source address of a legitimate user and sends a public Diffie-Hellman key to the victim. The victim then performs a modular exponentiation to compute the secret key. Repeated messages of this type can clog the victim's system with useless work. The **cookie exchange** requires that each side send a pseudorandom number, the cookie, in the initial message, which the other side acknowledges. This acknowledgment must be repeated in the first message of the Diffie- Hellman key exchange. The recommended method for creating the cookie is to perform a
fast hash (e.g., MD5) over the IP Source and Destination addresses, the UDP Source and Destination ports, and a locally generated secret value. Oakley supports the use of different **groups** for the Diffie-Hellman key exchange. Each group includes the definition of the two global parameters and the identity of the algorithm. Oakley employs **nonces** to ensure against replay attacks. Each nonce is a locally generated pseudorandom number. Nonces appear in responses and are encrypted during certain portions of the exchange to secure their use. Three different authentication methods can be used with Oakley are digital signatures, public-key encryption and Symmetric-key encryption.

Aggressive Oakley Key Exchange

Aggressive key exchange is a technique used for exchanging the message keys and is so called because only three messages are allowed to be exchanged at any time.

Example of Aggressive Oakley Key Exchange

$\mathbf{I \rightarrow R:}$ $CKY_I, OK\_KEYX, GRP, g^x, EHAO, NIDP, ID_I, ID_R, N_I, S_{KI}[ID_I \| ID_R \| N_I \| GRP \| g^x \| EHAO]$

$\mathbf{R \rightarrow I:}$ $CKY_R, CKY_I, OK\_KEYX, GRP, g^y, EHAS, NIDP, ID_R, ID_I, N_R, N_I, S_{KR}[ID_R \| ID_I \| N_R \| N_I \| GRP \| g^y \| g^x \| EHAS]$

$\mathbf{I \rightarrow R:}$ $CKY_I, CKY_R, OK\_KEYX, GRP, g^x, EHAS, NIDP, ID_I, ID_R, N_I, N_R, S_{KI}[ID_I \| ID_R \| N_I \| N_R \| GRP \| g^x \| g^y \| EHAS]$

Notation:

I	=	Initiator
R	=	Responder
CKY_I, CKY_R	=	Initiator, responder cookies
OK_KEYX	=	Key exchange message type
GRP	=	Name of Diffie-Hellman group for this exchange
g^x, g^y	=	Public key of initiator, responder; g^{xy} = session key from this exchange
EHAO, EHAS	=	Encryption, hash authentication functions, offered and selected
NIDP	=	Indicates encryption is not used for remainder of this message
ID_I, ID_R	=	Identifier for initiator, responder
N_I, N_R	=	Random nonce supplied by initiator, responder for this exchange
$S_{KI}[X], S_{KR}[X]$	=	Indicates the signature over X using the private key (signing key) of intiator, responder

In the first step, the initiator (I) transmits a cookie, the group to be used, and I's public Diffie-Hellman key for this exchange. I also indicates the offered public-key encryption, hash, and authentication algorithms to be used in this exchange. Also included in this message are the identifiers of I and the responder (R) and I's nonce for this exchange. Finally, I appends a signature using I's private key that signs the two identifiers, the nonce, the group, the Diffie-Hellman public key, and the offered algorithms. When R receives the message, R verifies the signature using I's public signing key. R acknowledges the message by echoing back I's cookie, identifier, and nonce, as well as the group. R also includes in the message a cookie, R's Diffie-Hellman public key, the selected algorithms (which must be among the offered algorithms), R's identifier, and R's nonce for this exchange. Finally, R appends a

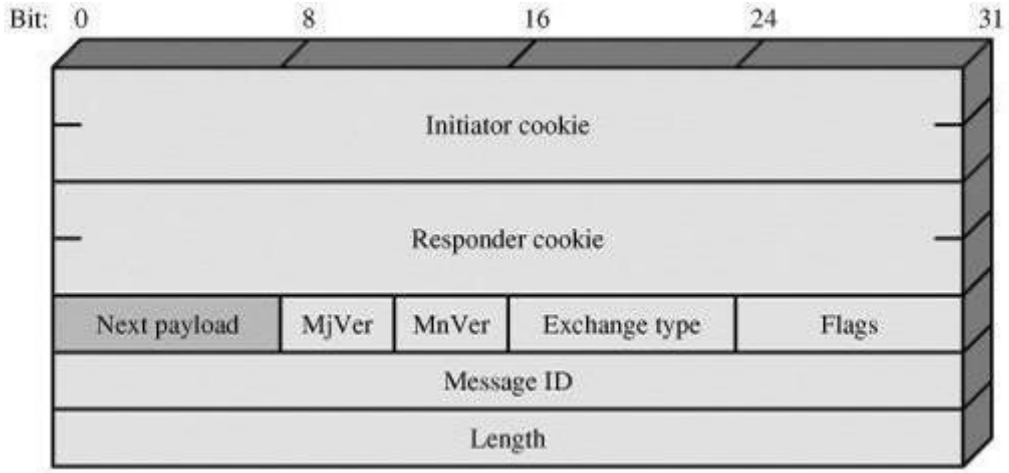
signature using R's private key that signs the two identifiers, the two nonces, the group, the two Diffie-Hellman public keys, and the selected algorithms.

When I receives the second message, I verifies the signature using R's public key. The nonce values in the message assure that this is not a replay of an old message. To complete the exchange, I must send a message back to R to verify that I has received R's public key.

ISAKMP

ISAKMP defines procedures and packet formats to establish, negotiate, modify, and delete security associations. As part of SA establishment, ISAKMP defines payloads for exchanging key generation and authentication data.

ISAKMP Header Format



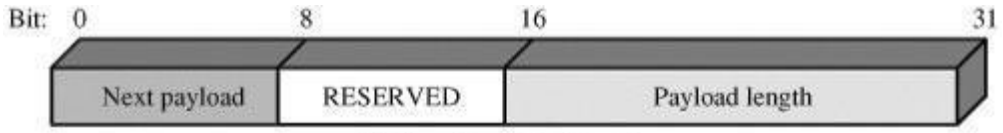
(a) ISAKMP header

An ISAKMP message consists of an ISAKMP header followed by one or more payloads and must follow UDP transport layer protocol for its implementation. The header format of an ISAKMP header is shown below:

- Initiator Cookie (64 bits): Cookie of entity that initiated SA establishment, SA notification, or SA deletion.
- Responder Cookie (64 bits): Cookie of responding entity; null in first message from initiator.
- Next Payload (8 bits): Indicates the type of the first payload in the message
- Major Version (4 bits): Indicates major version of ISAKMP in use.
- Minor Version (4 bits): Indicates minor version in use.
- Exchange Type (8 bits): Indicates the type of exchange. Can be informational, aggressive, authentication only, identity protection or base exchange (S).
- Flags (8 bits): Indicates specific options set for this ISAKMP exchange. Two bits so far defined: The Encryption bit is set if all payloads following the header are encrypted using the encryption algorithm for this SA. The Commit bit is used to ensure that encrypted material is not received prior to completion of SA establishment.
- Message ID (32 bits): Unique ID for this message.
- Length (32 bits): Length of total message (header plus all payloads) in octets.

ISAKMP Payload Types

All ISAKMP payloads begin with the same generic payload header shown below.



(b) Generic payload header

The Next Payload field has a value of 0 if this is the last payload in the message; otherwise its value is the type of the next payload. The Payload Length field indicates the length in octets of this payload, including the generic payload header. There are many different ISAKMP payload types. They are:

a. The SA payload is used to begin the establishment of an SA. The Domain of Interpretation parameter identifies the DOI under which negotiation is taking place. The Situation parameter defines the security policy for this negotiation; in essence, the levels of security required for encryption and confidentiality are specified (e.g., sensitivity level, security compartment).

b. The Proposal payload contains information used during SA negotiation. The payload indicates the protocol for this SA (ESP or AH) for which services and mechanisms are being negotiated. The payload also includes the sending entity's SPI and the number of transforms. Each transform is contained in a transform payload.

c. The Transform payload defines a security transform to be used to secure the communications channel for the designated protocol. The Transform # parameter serves to identify this particular payload so that the responder may use it to indicate acceptance

of this transform. The Transform-ID and Attributes fields identify a specific transform (e.g., 3DES for ESP, HMAC-SHA-1-96 for AH) with its associated attributes (e.g., hash length).

d. The Key Exchange payload can be used for a variety of key exchange techniques, including Oakley, Diffie-Hellman, and the RSA-based key exchange used by PGP. The Key Exchange data field contains the data required to generate a session key and is dependent on the key exchange algorithm used.

e. The Identification payload is used to determine the identity of communicating peers and may be used for determining authenticity of information. Typically the ID Data field will contain an IPv4 or IPv6 address.

f. The Certificate payload transfers a public-key certificate. The Certificate Encoding field indicates the type of certificate or certificate-related information, which may include SPKI, ARL, CRL, PGP info etc. At any point in an ISAKMP exchange, the sender may include a Certificate Request payload to request the certificate of the other communicating entity.

g. The Hash payload contains data generated by a hash function over some part of the message and/or ISAKMP state. This payload may be used to verify the integrity of the data in a message or to authenticate negotiating entities.

h. The Signature payload contains data generated by a digital signature function over some part of the message and/or ISAKMP state. This payload is used to verify the

integrity of the data in a message and may be used for nonrepudiation services.

i.

The Nonce payload contains random data used to guarantee liveness during an exchange and protect against replay attacks.

j. The Notification payload contains either error or status information associated with this SA or this SA negotiation. Some of the ISAKMP error messages that have been defined are Invalid Flags, Invalid Cookie, Payload Malformed etc

k. The Delete payload indicates one or more SAs that the sender has deleted from its database and that therefore are no longer valid.

ISAKMP Exchanges

ISAKMP provides a framework for message exchange, with the payload types serving as the building blocks. The specification identifies five default exchange types that should be supported.

1. Base Exchange: allows key exchange and authentication material to be transmitted together. This minimizes the number of exchanges at the expense of not providing identity protection.

The first two messages provide cookies and establish an SA with agreed protocol and

(a) Base Exchange

(1)**I**⟶**R**: SA; NONCE	Begin ISAKMP-SA negotiation
(2)**R**⟶**E**: SA; NONCE	Basic SA agreed upon
(3)**I**⟶**R**: KE; ID_I AUTH	Key generated; Initiator identity verified by responder
(4)**R**⟶**E**: KE; ID_R AUTH	Responder identity verified by initiator; Key generated; SA established

transforms; both sides use a nonce to ensure against replay attacks. The last two messages exchange the key material and user IDs, with an authentication mechanism used to authenticate keys, identities, and the nonces from the first two messages.

2. Identity Protection Exchange: expands the Base Exchange to protect the users' identities.

(b) Identity Protection Exchange

(1)**I**⟶**R**: SA	Begin ISAKMP-SA negotiation
(2)**R**⟶**E**: SA	Basic SA agreed upon
(3)**I**⟶**R**: KE; NONCE	Key generated
(4)**R**⟶**E**: KE; NONCE	Key generated
(5)*__I__⟶**R**: ID_I; AUTH	Initiator identity verified by responder
(6)*__R__⟶**E**: ID_R; AUTH	Responder identity verified by initiator; SA established

The first two messages establish the SA. The next two messages perform key exchange, with nonces for replay protection. Once the session key has been computed, the two parties

exchange encrypted messages that contain authentication information, such as digital signatures and optionally certificates validating the public keys.

3. Authentication Only Exchange: used to perform mutual authentication, without a key exchange

The first two messages establish the SA. In addition, the responder uses the second message to convey its ID and uses authentication to protect the message. The initiator sends the third message to transmit its authenticated ID.

4. Aggressive Exchange: minimizes the number of exchanges at the expense of not providing identity protection.

(d) Aggressive Exchange

(1)**I**⟶**R**: SA; KE; NONCE; ID_I;	Begin ISAKMP-SA negotiation and key exchange
(2)**R**⟶**E**: SA; KE; NONCE; ID_R; AUTH	Initiator identity verified by responder; Key generated; Basic SA agreed upon
(3)*I⟶**R**: AUTH	Responder identity verified by initiator; SA established

In the first message, the initiator proposes an SA with associated offered protocol and transform options. The initiator also begins the key exchange and provides its ID. In the second message, the responder indicates its acceptance of the SA with a particular protocol and transform, completes the key exchange, and authenticates the transmitted information. In the third message, the initiator transmits an authentication result that covers the previous information, encrypted using the shared secret session key.

5. Informational Exchange: used for one-way transmittal of information for SA management.

(e) Informational Exchange

(1)*I⟶**R**: N/D	Error or status notification, or deletion

UNIT-5

Web Security: *Web Security Considerations, Secure Socket Layer (SSL) and Transport Layer Security (TLS), Secure Electronic Transaction (SET).* ***Intruders, Viruses and Firewalls:*** *Intruders, Intrusion Detection, Password Management, Virus and related threats, Countermeasures, Firewall Design Principles, Types of Firewalls.* ***Case Studies on Cryptography and Security****: Secure Inter Branch Transactions, Cross Site Vulnerability, Virtual Elections.*

Usage of internet for transferring or retrieving the data has got many benefits like speed, reliability, security etc. Much of the Internet's success and popularity lies in the fact that it is an open global network. At the same time, the fact that it is open and global makes it not very secure. The unique nature of the Internet makes exchanging information and transacting business over it inherently dangerous. The faceless, voiceless, unknown entities and individuals that share the Internet may or may not be who or what they profess to be. In addition, because the Internet is a global network, it does not recognize national borders and legal jurisdictions. As a result, the transacting parties may not be where they say they are and may not be subject to the same laws or regulations.

For the exchange of information and for commerce to be secure on any network, especially the Internet, a system or process must be put in place that satisfies requirements for confidentiality, access control, authentication, integrity, and nonrepudiation. These requirements are achieved on the Web through the use of encryption and by employing digital signature technology. There are many examples on the Web of the practical application of encryption. One of the most important is the SSL protocol.

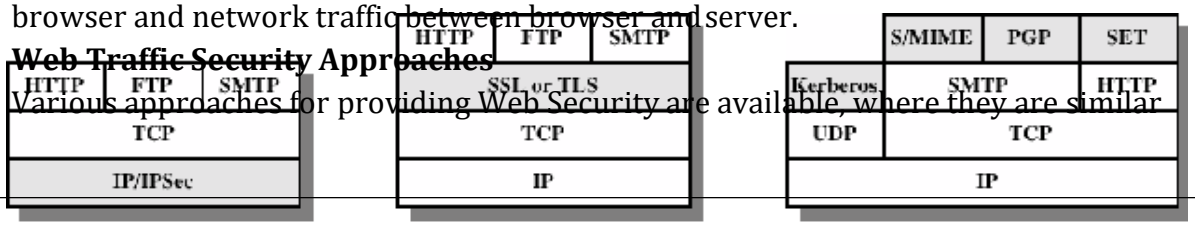
A summary of types of security threats faced in using the Web is given below:

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

One way of grouping the security threats is in terms of passive and active attacks. *Passive attacks* include eavesdropping on network traffic between browser and server and gaining access to information on a website that is supposed to be restricted. *Active attacks* include impersonating another user, altering messages in transit between client and server and altering information on a website. Another way of classifying these security threats is in terms of location of the threat: Web server, Web browser and network traffic between browser and server.

Web Traffic Security Approaches

Various approaches for providing Web Security are available, where they are similar



(a) Network Level (b) Transport Level (c) Application Level

in the services they provide and also similar to some extent in the mechanisms they use. They differ with respect to their scope of applicability and their relative location within the TCP/IP protocol stack. The main approaches are IPSec, SSL or TLS and SET.

Relative location of Security Faculties in the TCP/IP Protocol Stack

IPSec provides security at the network level and the main advantage is that it is transparent to end users and applications. In addition, IPSec includes a filtering capability so that only selected traffic can be processed. **Secure Socket Layer or Transport Layer Security (SSL/TLS)** provides security just above the TCP at transport layer. Two implementation choices are present here. Firstly, the SSL/TLS can be implemented as a

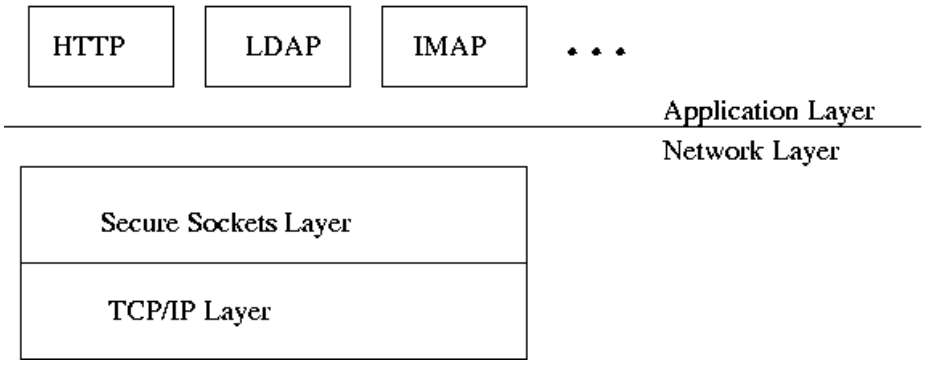
part of TCP/IP protocol suite, thereby being transparent to applications. Alternatively, SSL can be embedded in specific packages like SSL being implemented by Netscape and Microsoft Explorer browsers. **Secure Electronic Transaction (SET)** approach provides application-specific services i.e., according to the security requirements of a particular application. The main advantage of this approach is that service can be tailored to the specific needs of a given application.

SECURE SOCKET LAYER/TRANSPORT LAYER SECURITY

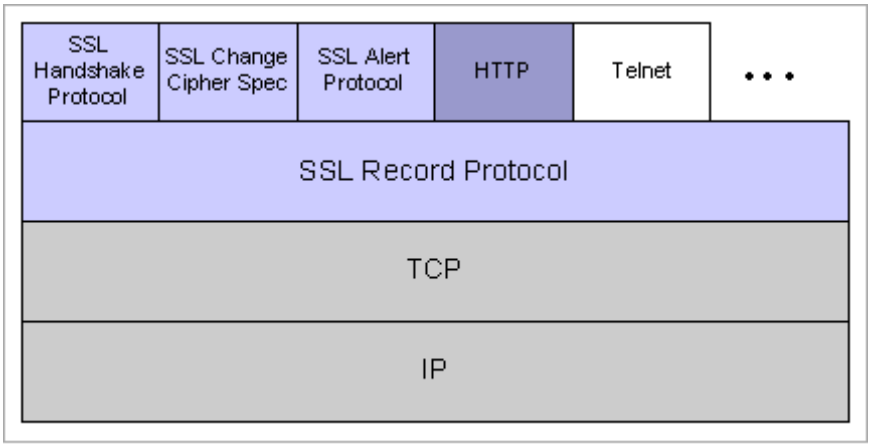
SSL was developed by Netscape to provide security when transmitting information on the Internet. The Secure Sockets Layer protocol is a protocol layer which may be placed between a reliable connection-oriented network layer protocol (e.g. TCP/IP) and the application protocol layer (e.g. HTTP).

SSL provides for secure communication between client and server by allowing mutual

SSL runs above TCP/IP and below high-level application protocols



authentication, the use of digital signatures for integrity and encryption for privacy. SSL protocol has different versions such as SSLv2.0, SSLv3.0, where SSLv3.0 has an advantage with the addition of support for certificate chain loading. SSL 3.0 is the basis for the Transport Layer Security [TLS] protocol standard. SSL is designed to make use of TCP to provide a reliable end-to-end secure service. SSL is not a single protocol, but rather two layers of protocols as shown below:



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, and 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 (in the OSI layering model definition) 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 cryptographic security parameters, which can be shared among multiple connections. Sessions are used to avoid the expensive negotiation of new security parameters for each connection.

An SSL session is *stateful*. Once a session is established, there is a current operating state for both read and write (i.e., receive and send). In addition, during the Handshake Protocol, pending read and write states are created. Upon successful conclusion of the Handshake Protocol, the pending states become the current states. An SSL session may include multiple secure connections; in addition, parties may have multiple simultaneous sessions.

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 state may be null.
- ***Compression method***: The algorithm used to compress data prior to encryption.
- ***Cipher spec:***Specifies the bulk data encryption algorithm (such as null, AES, etc.) 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 operations on data sent by the client.
- ***Server write key:***The conventional encryption key for data encrypted by the 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 (IV) is maintained for each key. This field is first initialized by the SSL Handshake Protocol. Thereafter the final ciphertext block from each record is preserved for use as the IV with the following record.
- ***Sequence numbers***: Each party maintains separate sequence numbers for 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 numbers may not exceed 264-1.

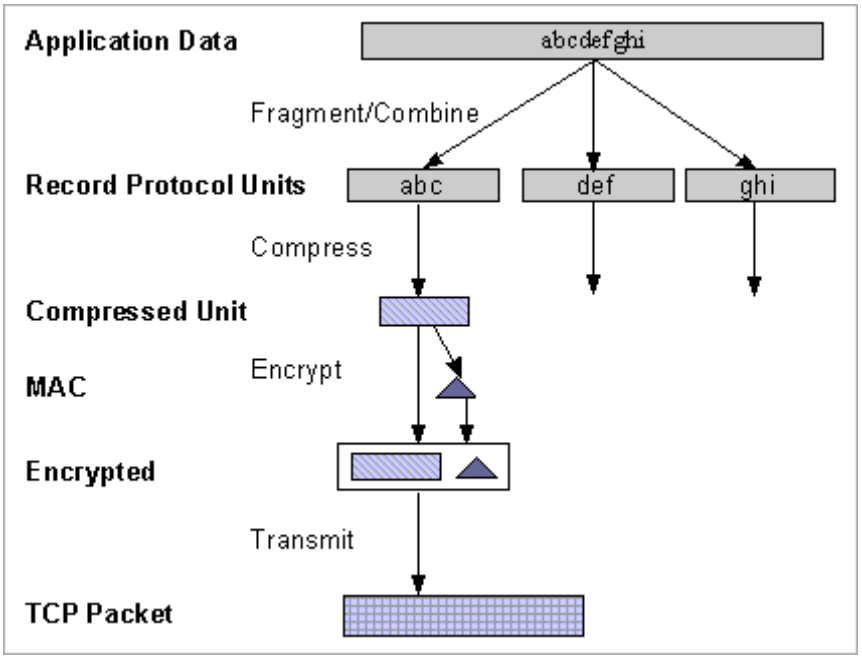
SSL Record Protocol

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).

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 overall operation of the SSL Record Protocol is shown below:



The first step is fragmentation. Each upper-layer message is fragmented into blocks of 214 bytes (16384 bytes) or less. Next, compression is optionally applied. Compression must be lossless and may not increase the content length by more than 1024 bytes. The next step in processing is to compute a message authentication code over the compressed data. For this purpose, a shared secret key is used. The calculation is defined as:

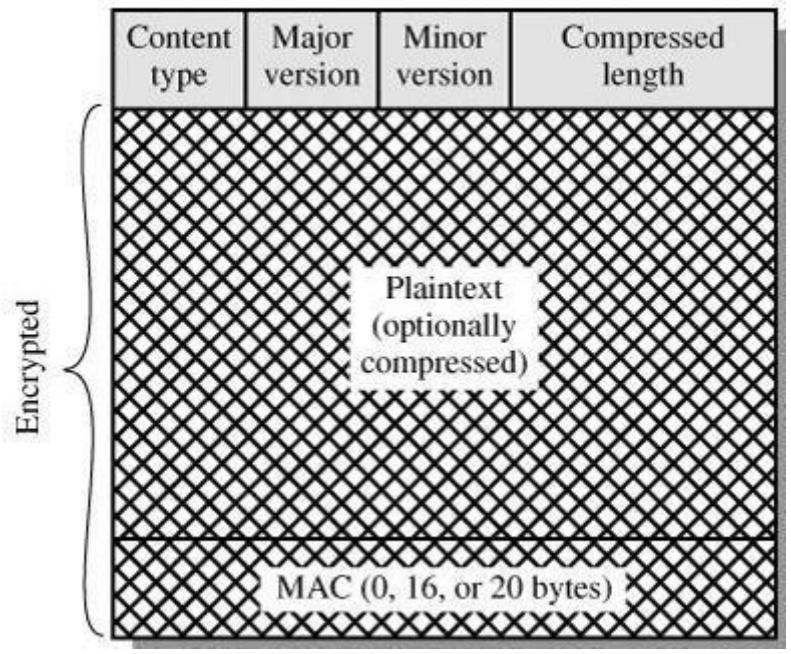
hash(MAC_write_secret ||
pad_2 ||
hash(MAC_write_secret ||
pad_1 || seq_num ||
SSLCompressed.type ||
SSLCompressed.length || SSLCompressed.fragment)) Where,

MAC_write_secret = Secret shared key pad_1 = the byte 0x36 (0011 0110) repeated 48 times (384 bits) for MD5 and 40 times for

pad_2 = the byte 0x5C (0101 1100) repeated 48 times for MD5 and 40 times for SHA-1

The main difference between HMAC and above calculation is that the two pads are concatenated in SSLv3 and are XORed in HMAC. Next, the compressed message plus the MAC are encrypted using symmetric encryption. Encryption may not increase the content length by more than 1024 bytes, so that the total length may not exceed 2_{14} + 2048. The encryption algorithms allowed are AES-128/256, IDEA-128, DES-40, 3DES-168, RC2-40, Fortezza, RC4-40 and RC4-128. For stream encryption, the compressed message plus the MAC are encrypted whereas, for block encryption, padding may be added after the MAC prior to encryption.

SSL Record Format



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 (or compressed fragment if compression is used). The maximum value is $2_{14} + 2048$.

The content types that have been defined are change_cipher_spec, alert, handshake, and application_data.

SSL 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.

SSL Alert Protocol

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

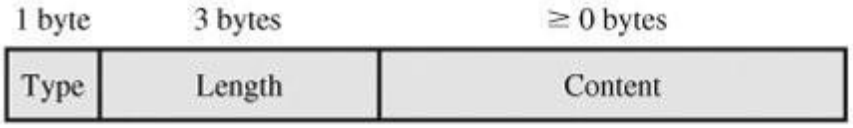
- unexpected_message: An inappropriate message was received.
- bad_record_mac: An incorrect MAC was received.
- decompression_failure: The decompression function received improper input (e.g., unable to decompress or decompress to greater than maximum allowable length).
- handshake_failure: Sender was 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 given below:
• 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 write side of a connection.
• no_certificate: May be sent in response to a certificate request if no appropriate certificate is available.
• bad_certificate: A received certificate was corrupt (e.g., contained a signature that did not verify).
• 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.
• certificate_unknown: Some other unspecified issue arose in processing the certificate, rendering it unacceptable.

SSL Handshake Protocol

SSL Handshake protocol ensures establishment of reliable and secure session between client and server and also allows server & client to:
• authenticate each other
• to negotiate encryption & MAC algorithms
• to negotiate cryptographic keys to be used

The Handshake Protocol consists of a series of messages exchanged by client and server. All of these have the format shown below and each message has three fields:
• **Type (1 byte**): Indicates one of 10 messages.



(c) Handshake Protocol

• **Length (3 bytes):** The length of the message in bytes.
• **Content (>=0 bytes):** The parameters associated with this message

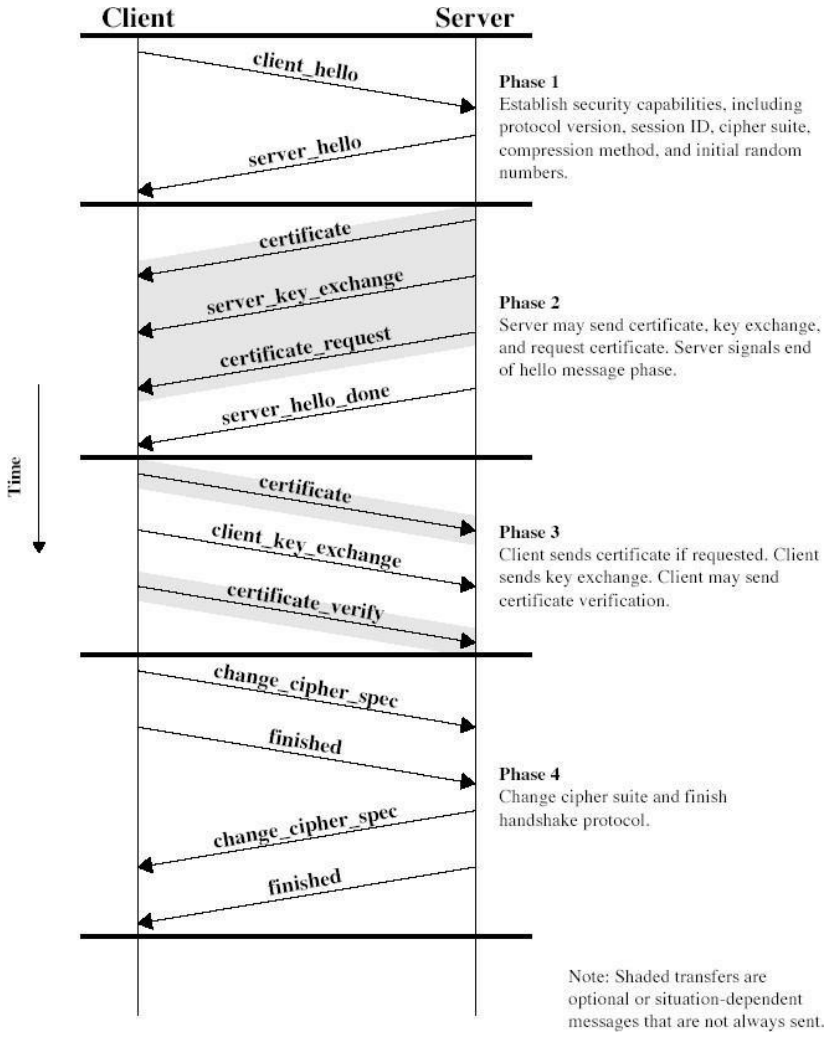
The following figure shows the initial exchange needed to establish a logical connection between client and server. The exchange can be viewed as having four phases.in phases o Establish Security Capabilities
o Server Authentication and Key Exchange
o Client Authentication and Key Exchange
o Finish

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 of preference. Each element of the list (each cipher suite) defines both a key exchange algorithm and a CipherSpec.
- Compression Method: This is a list of the compression methods the client supports.



Phase 2. Server Authentication and Key Exchange

The server begins this phase by sending its certificate via a certificate message, which contains one or a chain of X.509 certificates. The **certificate message** is required for any agreed-on key exchange method except anonymous Diffie-Hellman. 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, or

(2) RSA key exchange is to be used.

Phase 3. Client Authentication and Key Exchange

Once the server_done message is received by client, it should verify whether a valid certificate is provided 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, for which the

content of the message depends on the type of key exchange.

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. 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 was released in response to the Internet community's demands for a standardized protocol. TLS (Transport Layer Security), defined in RFC 2246, is a protocol for establishing a secure connection between a client and a server. TLS (Transport Layer Security) is capable of authenticating both the client and the server and creating a encrypted connection between the two. Many protocols use TLS (Transport Layer Security) to establish secure connections, including HTTP, IMAP, POP3, and SMTP. The TLS Handshake Protocol first negotiates key exchange using an asymmetric algorithm such as RSA or Diffie-Hellman. The TLS Record Protocol then begins opens an encrypted channel using a symmetric algorithm such as RC4, IDEA, DES, or 3DES. The TLS Record Protocol is also responsible for ensuring that the communications are not altered in transit. Hashing algorithms such as MD5 and SHA are used for this purpose. RFC 2246 is very similar to SSLv3. There are some minor differences ranging from protocol version numbers to generation of key material.

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: Two differences arise one being the actual algorithm and the other being scope of MAC calculation. TLS makes use of the HMAC algorithm defined in RFC 2104. SSLv3 uses the same algorithm, except that the padding bytes are concatenated with the secret key rather than being XORed with the secret key padded to the block length. 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 PRF is based on the following data expansion function:

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

define

d 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. As can be seen, P_hash can be iterated as many times as necessary to produce the required quantity of data. each iteration involves two executions of HMAC, each of which in turn involves two executions of the underlying hash algorithm.

SET (SECURE ELECTRONIC TRANSACTION)

SET is an open encryption and security specification designed to protect credit card transactions on the Internet. SET is not itself a payment system. Rather it is a set of security protocols and formats that enables users to employ the existing credit card payment infrastructure on an open network, such as the Internet, in a secure fashion. In essence, SET provides three services:

- 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 Requirements

- Provide confidentiality of payment and ordering information Ensure the integrity of all transmitted data
- Provide authentication that a cardholder is a legitimate user of a credit card account Provide authentication that a merchant can accept credit card transactions through its relationship with a financial institution
- Ensure the use of the best security practices and system design techniques to protect all legitimate parties in an electronic commerce transaction
- Create a protocol that neither depends on transport security mechanisms nor prevents their use
- Facilitate and encourage interoperability among software and network providers

SET Key Features

To meet the requirements, SET incorporates the following features:

- Confidentiality of information
- Integrity of data
- Cardholder account authentication
- Merchant authentication

SET Participants

- Cardholder: purchasers interact with merchants from personal computers over the Internet
- Merchant: a person or organization that has goods or services to sell to the cardholder Issuer: a financial institution, such as a bank, that provides the cardholder with the

payment card.

- Acquirer: a financial institution that establishes an account with a merchant and processes payment card authorizations and payments
- Payment gateway: a function operated by the acquirer or a designated third party that processes merchant payment messages
- Certification authority (CA): an entity that is trusted to issue X.509v3 public-key certificates for cardholders, merchants, and payment gateways

Events in a transaction

1. The customer obtains a credit card account with a bank that supports electronic payment and SET
2. The customer receives a X.509v3 digital certificate signed by the bank.
3. Merchants have their own certificates
4. The customer places an order
5. The merchant sends a copy of its certificate so that the customer can verify that it's a valid store
6. The order and payment are sent
7. The merchant requests payment authorization
8. The merchant confirms the order
9. The merchant ships the goods or provides the service to the customer
10. The merchant requests payment

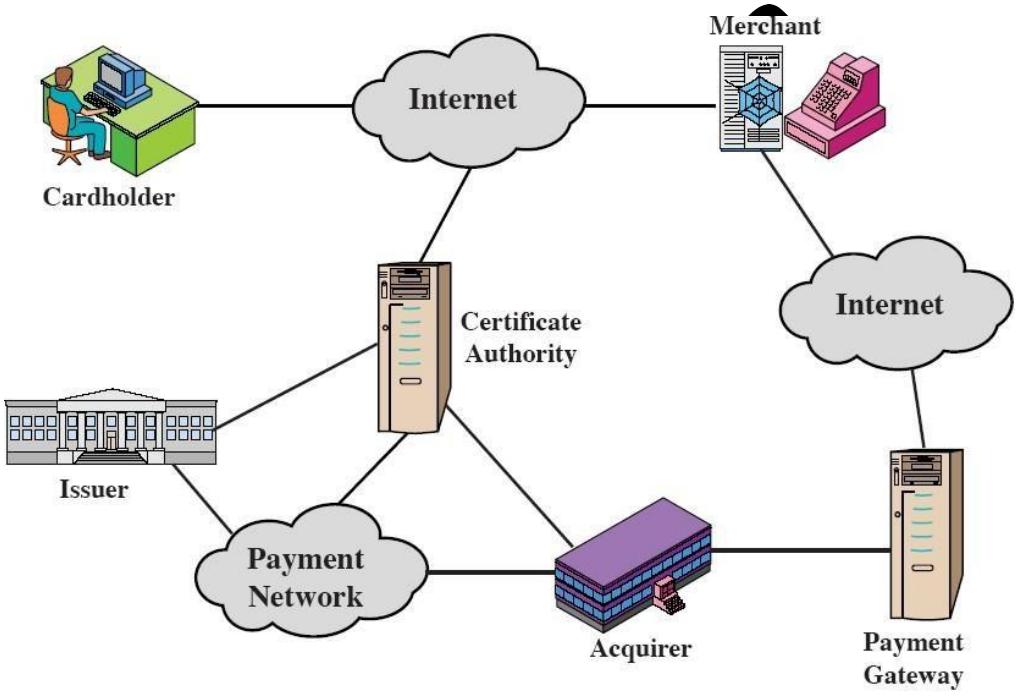


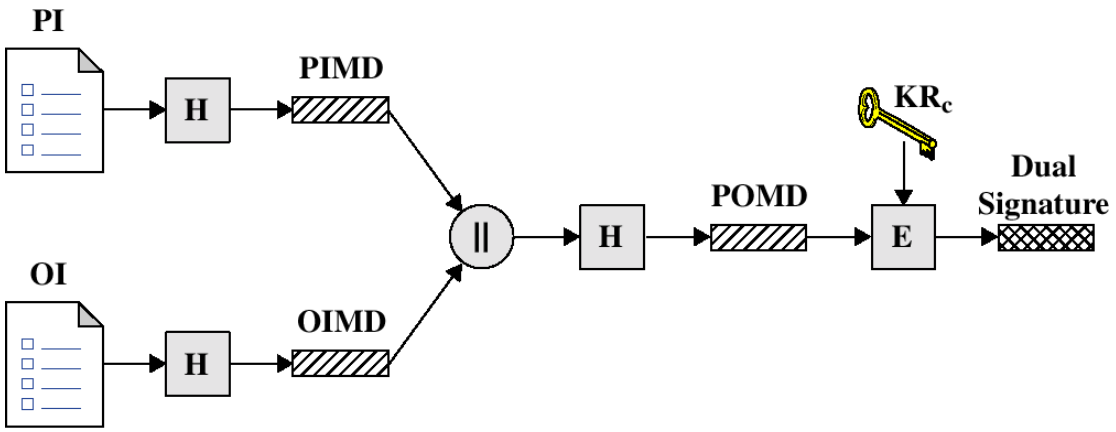
Figure 17.8 Secure Electronic Commerce Components

DUAL SIGNATURE

The purpose of the dual signature is to link two messages that are intended for two different recipients. 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. The two items must be linked and 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.



PI = Payment Information
OI = Order Information
H = Hash function (SHA-1)
|| = Concatenation

PIMD = PI message digest
OIMD = OI message digest
POMD = Payment Order message digest
E = Encryption (RSA)
KR_c = Customer's private signature key

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

$$DS = E_{KR_c}[H(H(PI) \| \mathrm{H}(\mathrm{OI}))]$$

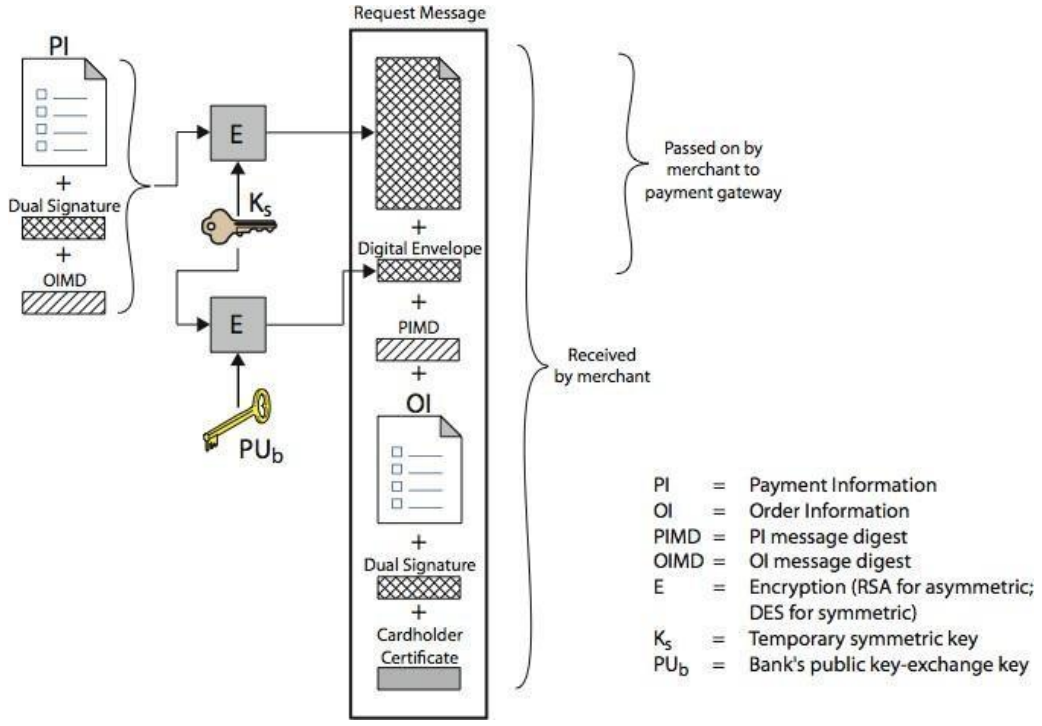
where KR_c is the customer's private signature key. Now suppose that the merchant is in possession of the dual signature (DS), the OI, and the message digest for the PI (PIMD). The merchant also has the public key of the customer, taken from the customer's certificate. Then the merchant can compute the quantities H(PIMS||H[OI]) and D_{KUc}(DS) where KU_c 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 (OIMD), and the customer's public key, then the bank can

compute H(H[OI]||OIMD) and D_{KUc}(DS). Again, if these two quantities are equal, then the bank has verified the signature. To summarize:

- 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.

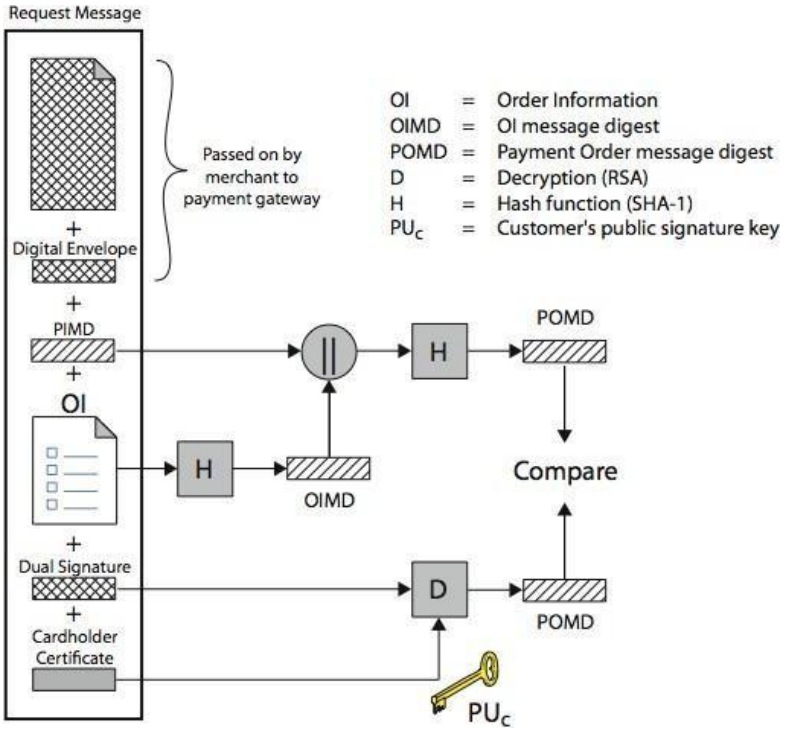
For a merchant to substitute another OI, he has to find another OI whose hash exactly matches OIMD, which is deemed impossible. So, the OI cannot be linked with another PI.

Purchase Request



The message includes the following:
1. Purchase-related information, which will be forwarded to the payment gateway by the merchant and consists of: PI, dual signature & OI message digest (OIMD). These are encrypted using Ks. A digital envelope is also present which is formed by encrypting Ks with the payment gateway's public key-exchange key.
2. Order-related information, needed by the merchant and consists of: OI, dual signature, PI message digest (PIMD). OI is sent in the clear.
3. Cardholder certificate. This contains the cardholder's public signature key. It is needed by the merchant and payment gateway.

Merchant receives the Purchase Request message, the following actions are done:
1. verifies cardholder certificates using CA sigs
2. verifies dual signature using customer's public signature key to ensure order has not been tampered with in transit & that it was signed using cardholder's private signature key
3. processes order and forwards the payment information to the payment gateway for authorization
4. sends a purchase response to cardholder



The Purchase Response message includes a response block that acknowledges the order and references the corresponding transaction number. This block is signed by the merchant using its private signature key. The block and its signature are sent to the customer, along with the merchant's signature certificate. Necessary action will be taken by cardholder's software upon verification of the certificates and signature.