1. What is PGP. Discuss about its services

PGP

- Pretty Good Privacy (PGP) is a protocol used to secure E-mail applications. It is developed by Phil Zimmermann. It is open source software.
- PGP provides a confidentiality and authentication services that can be used for electronic mail and file storage applications
- It is available on Unix, PC, Macintosh and Amiga operating systems.

Note: In e-mail security, the encryption/decryption is done using a symmetric-key algorithm, but the secret key to decrypt the message is encrypted with the public key of the receiver and is sent with the message

PGP services/Operations:

The operation of PGP consists of four services:

- **Authentication** – provided by using digital signature.
- **Integrity** –"
- **Confidentiality** – provided by using symmetric block encryption algorithms.
- **Compression** – provided by using compressed methods.
- **Code conversion**– provided by using radix64 encoding scheme.
- **Segmentation** – Automatically done. No specific method used.

PGP services/operations

SENDER: Alice RECEIVER: Bob

1. **Authentication and Integrity** –

To provide Authentication and Message Integrity

- Alice creates a digest of the message and signs it with her private-key Signing means -----digest is encrypted using senders private-key.
- When Bob receives the message, he verifies the message by using Alice's public-key. Here, two keys are needed – Alice's private- key and Bob's public-key.
- Here, two keys are needed – Alice's private- key and Bob's public-key

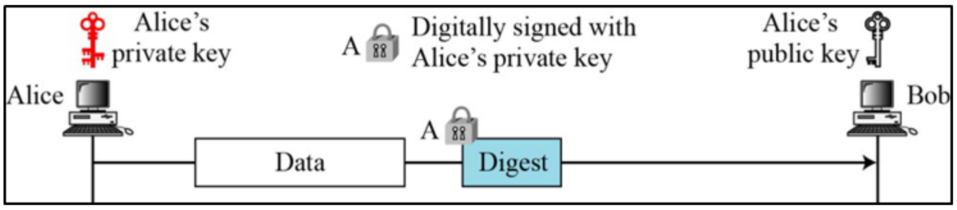


Figure 16.3 An authenticated message

2. Confidentiality – provided by using symmetric block encryption algorithms.

- Confidentiality in an e-mail system can be achieved using conventional encryption with a one-time session-key.
- Alice can create a session key, use session key to encrypt the message and the digest, and send the key itself with the message.
- However, to protect the session key, Alice encrypts it with Bob's public key. Figure 16.5 shows the situation
- When Bob receives the packet, he first decrypts the key, using his private key to remove the key.
- He then uses the session key to decrypt the rest of the message.

- After decomposing the rest of the message, Bob creates a digest of the message and checks to see if it is equal to the digest sent by Alice.
- If it is, then the message is authentic.

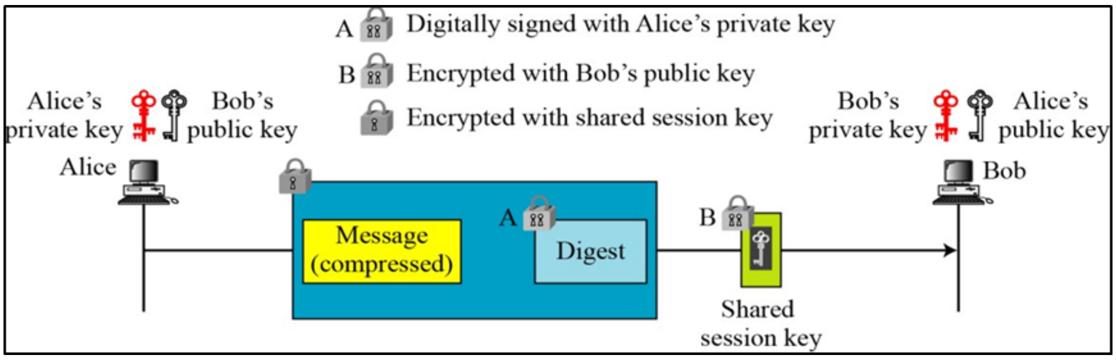


Figure 16.5 A confidential message

3. Compression

- After encrypting the digest, Alice compresses the Data using compression method. Then transmits encrypted digest and compressed data to Bob.
- There is no security benefit. but, it eases the traffic since the packet size is reduced.

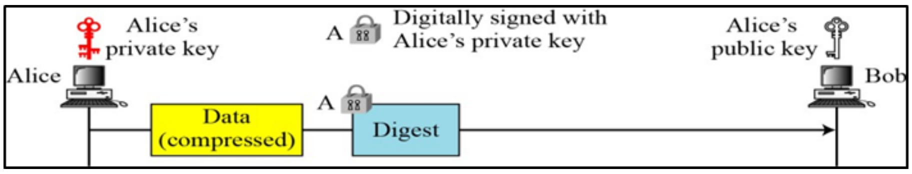


Figure 16.4 A compressed message

- 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.
- Z for compression and Z–1 for decompression
- 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 you generate signature after compression then there is a need recompression for message verification, PGP's compression algorithm presents a difficulty.
- Message encryption is applied after compression to strengthen cryptographic security. Therefore, cryptanalysis is more difficult.
- The compression algorithm used here is ZIP Algorithm

4. Code conversion

- All the characters in e-mail message are converted into ASCII characters
- To translate other characters not in ASCII set PGP uses Radix-64 conversion.

5. Segmentation

- E-mail facilities often are restricted to a maximum length. To accommodate this, PGP automatically subdivides a message that is too large into segments that are small enough to send via e-mail.
- The segmentation is done after all of the other processing, including the radix-64 conversion.

2. Discuss how PGP key rings are maintained by the user.

PGP Key Rings

Each PGP user has a pair of key rings:

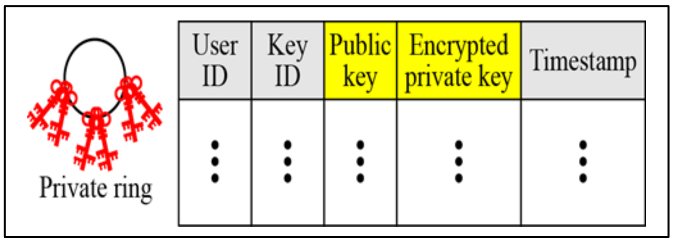
- Public-key ring contains all the public-keys of other PGP users known to this user, indexed by key ID
- Private-key ring contains the public/private key pair(s) for this user, indexed by key ID & encrypted keyed from a hashed passphrase

Example:

Alice has several pairs of private/public keys belonging to her and public keys belonging to other people.

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 entries:
  - **User ID:** It is the e-mail address of the user. Apart from e-mail address it may store alias names for each key pair. (e.g., stallings@acm.org).
  - **Key ID:** The least significant 64 bits of the public key. i.e. (public-key mod 264). It is used to uniquely identify a public-key among the use's public-keys. The **Key ID** is sent along with the message to the recipient, to use the public-key of the sender from public key ring of the recipient. Sending the public key along with the message is tough task.
  - **Public key:** It lists the public-key of a particular private/public key pair.
  - **Encrypted Private key:** Holds the encrypted value of the private-key in the private/public key pair.
  - **Timestamp:** The date and time of the key pair was generation.



User ID	Key ID	Public key	Encrypted private key	Timestamp
⋮	⋮	⋮	⋮	⋮

Figure 16.7 Format of private key ring table

Example:

Table 16.5 *Private key ring table for Example 1*

User ID	*Key ID*	*Public Key*	*Encrypted Private Key*	*Timestamp*
alice@anet.net	AB13...45	AB13...45...59	**32452398...23**	031505-16:23
alice@some.com	FA23...12	FA23...12...22	**564A4923...23**	031504-08:11

Public-key ring:

This ring is used to store public keys of other users that are known to this user.

- **User ID:** Identifies the owner of this key. Multiple user IDs may be associated with a single public key.
- **Key ID:** The first 64 bits of the public key.
- **Public Key:** The public key for the entity.

- **Producer Trust:** Defines producer level of trust. It can be one of 3 possible values: none partial or full.
- **Certificate(s):** This column holds the certificate or certificate(s) signed by other entities for this entity. One user ID may have than one certificate.
- **Certificate trust(s):** This column holds the value of user's trust on the certificate introducer to this entity.
- **Key Legitimacy**: This value this calculated based on the value of the certificate trust and predefined weight for each certificate trust.
- **Timestamp:** The date/time of the column creation.

The public key can be indexed by either User ID or Key ID.

User ID	Key ID	Public key	Producer trust	Certificate(s)	Certificate trust(s)	Key Legitimacy	Timestamp
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Example:

Table 16.10 *Example 2, after John is added to the table*

User ID	*Key ID*	*Public key*	*Prod. Trust*	*Certificate*	*Cert. trust*	*Key legit.*	*Time-stamp*
Alice...	AB...	AB........	F			F	....
Bob...	12...	12........	F			F	....
Ted...	48...	48........	F	Bob's	F	F	....
Anne...	71...	71........	P	Bob's	F	F	....
John...	31...	31........	N	Anne's	P	P	....

3. Describe how trust in PGP is achieved using web of trust model.

Trusted Model in PGP

• **PGP** is widely used for exchanging secure e -mail over Internet. **Trust** in **PGP** is achieved using the web of **trust model**. The underlying idea of this **model** is that you accept the public key of a **PGP** user if it has been signed by one or more other trustworthy **PGP** users.

• We can create a trust model for any user based on a public-key ring table. The trust model may change with any changes in the public-key ring table.

• When a new public key is inserted on public-key ring, it assigns the trust value of the key's owner given by the user. The trust value for each signature attached at the key is given to the OwnerTrust value for the owner of the signature;

If the owner for the signature is not in the key ring, it is given to a unknown user value.

Example: Consider web of trust model of the user Alice.

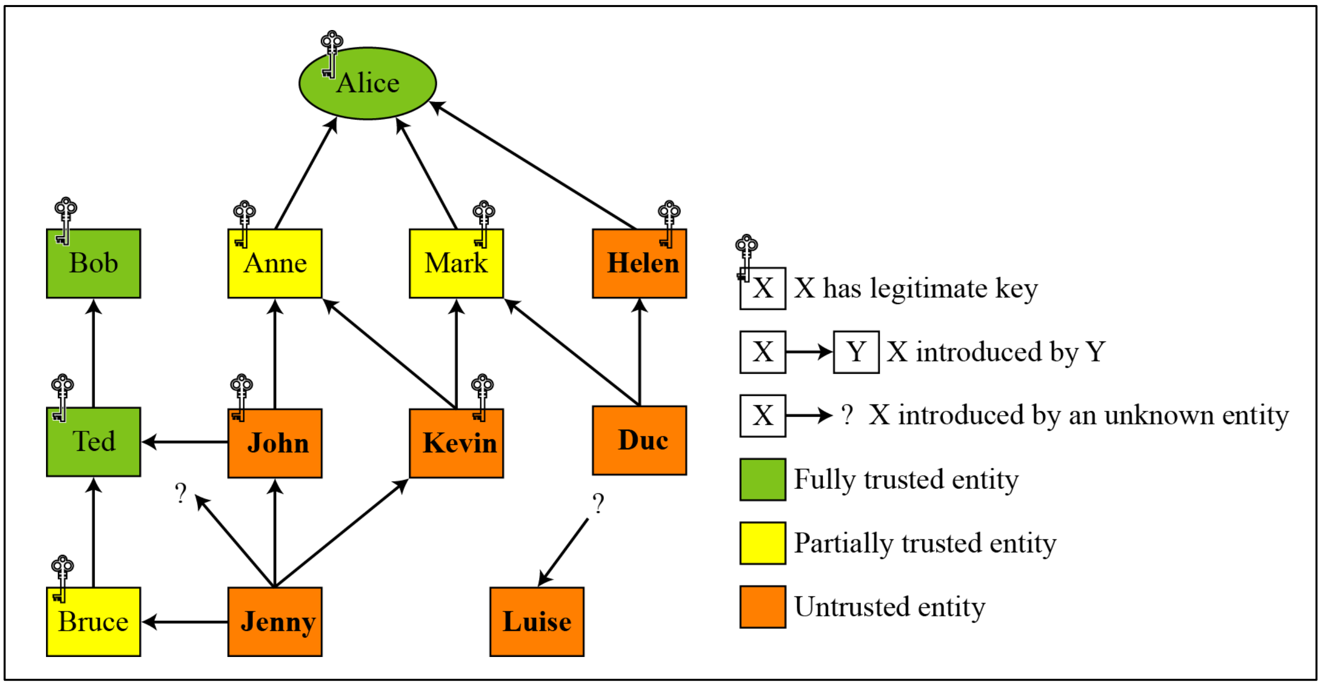


Figure 16.9 Trust model

- This is the trust model at Alice site.
- Each square represents one user. And each user maintains their own trust model based on their public-key table.
- The trust model of Alice is generated based on the public-key ring. It consists of
    - 3 entities in Alice's ring with **full trust - Alice herself, Bob, and Ted**
    - 3 entities with **partial trust** - Anne, Mark and Bruce and 6 entities with **no trust**.
    - Nine entities have a **legitimate key**.
    - three entities do not have any legitimate keys with Alice
- Alice can **encrypt** a message to any one of these entities or **verify** a signature received from one of these entities.
- Bob, Anne, and Mark have made their keys legitimate by sending their keys by e-mail and verifying their fingerprints by phone.
- Helen, has sent a certificate from a CA because she is not trusted by Alice and verification on the phone is not possible.
- Ted is fully trusted, he has given Alice a certificate signed by Bob.
- John has sent Alice two certificates, one signed by Ted and one by Anne.
- Kevin has sent two certificates to Alice, one signed by Anne and one by Mark. Each of these certificates gives Kavin half a point of legitimacy; therefore, Kavin's key is legitimate.
- Duc has sent two certificates to Alice, one signed by Mark and other by Helen. Since Mark is half-trusted and Helen is not trusted, Duc does not have a legitimate key.
- Jenny has sent four certificates, one signed by half-trusted entity, two by un-trusted entities, and one by an unknown entity. Jenny does not have enough points to make her key Legitimate.
- Luise has sent one certificate signed by an unknown entity.

Note: Alice may Luise's name in the table in case future certificates for Luise arrive.

4. Explain how email messages are protected using S/MIME signing and encryption?

S/MIME. adds some new Content types to include security services to the MIME. All of these new types include the parameter "applicationlpkcs7-mime", in which "pkcs" defines "Public Key Cryptography Specification."

Cryptographic Message Syntax (CMS): -
To define how security services. such as confidentiality or integrity, can be added toMIME content types. S/MIME has defined **Cryptographic Message Syntax (CMS)**. The syntax in each case defines the exact encoding scheme for each content The following describe the type of message and different subtypes that are created from these messages. For details, the reader is referred to RFC3369 and 3370.

a) Data Content Type: -This an arbitrary string. object created is called Data.

b) Signed-Data Content Type: -

- It provides only integrity of data.
- It contains any type and zero or more signature values.
- The encoded result is an object called Signed Data

The following are the steps to create **Signed-Data:**
Step 1: For each signer, a message digest is created from the content using the specific hash algorithm chosen by that signer.
Step 2: Each message digest is signed with the private key of the signer.
Step 3: The content, signature values, certificates, and algorithms are then collected to create the Signed Data Object.

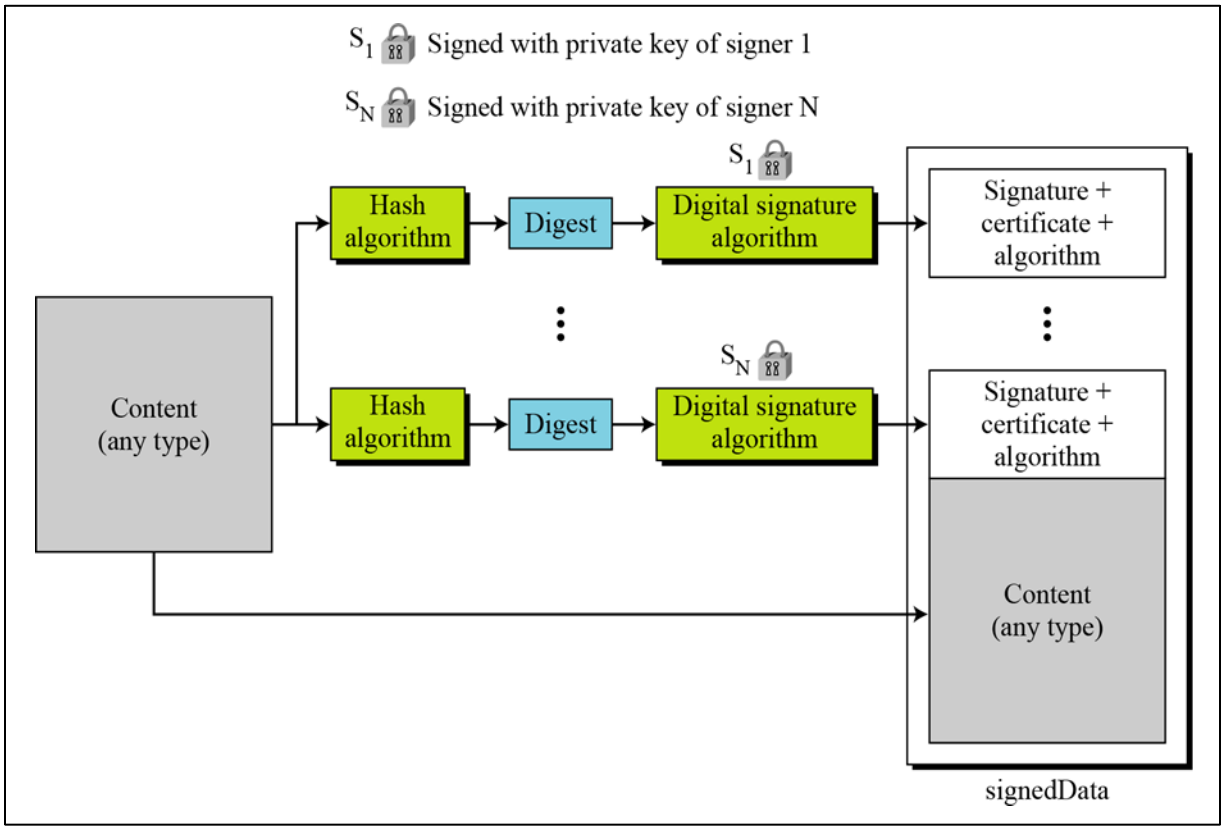

S_1 Signed with private key of signer 1
S_N Signed with private key of signer N
Content (any type)
Hash algorithm
Digest
S_1
Digital signature algorithm
Signature + certificate + algorithm
Hash algorithm
Digest
S_N
Digital signature algorithm
Signature + certificate + algorithm
Content (any type)
signedData

Figure 16.27 Signed-data content type

c) Enveloped-Data Content Type: -

- This type is used to provide privacy for the message.
- It contains any type and zero or more encrypted keys and certificates. The encoded result is an object called enveloped Data.
- The process of creating an object of this type:
  1. A pseudorandom session key is created for the symmetric-key algorithms to used
  2. For each recipient, a copy of the session key is encrypted with the public key of each recipient
  3. The content is encrypted using the defined algorithm and created session key.
  4. The encrypted contents, encrypted session keys, algorithm used, and certificates are encoded using Radix-64.

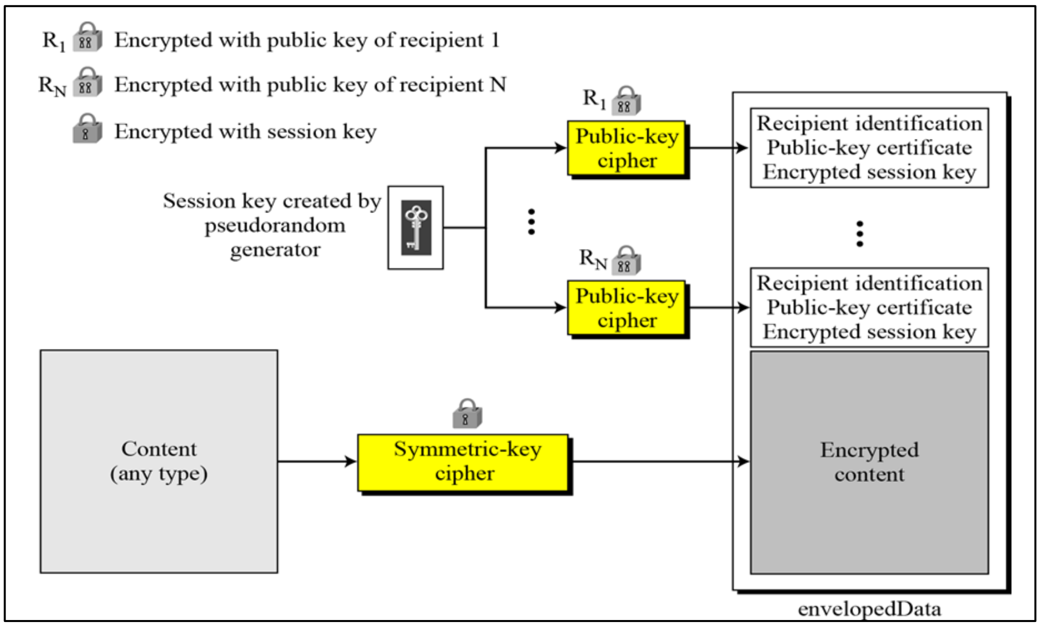


Figure 16.28 Enveloped-data content type

d) Digested data content type: -

It is used to provide Integrity for the message. The result is used as the enveloped-data content type. The encoded result is **digesteData.**

Process for creating this object:

6. A message digest is calculated from the content

7. The message digest the algorithm and the content are added together to create the **digesteData** object.

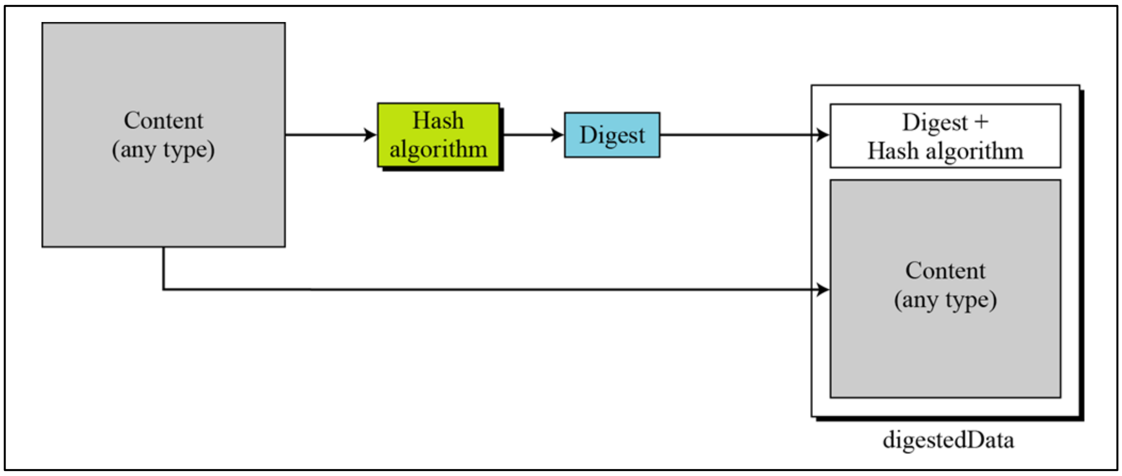
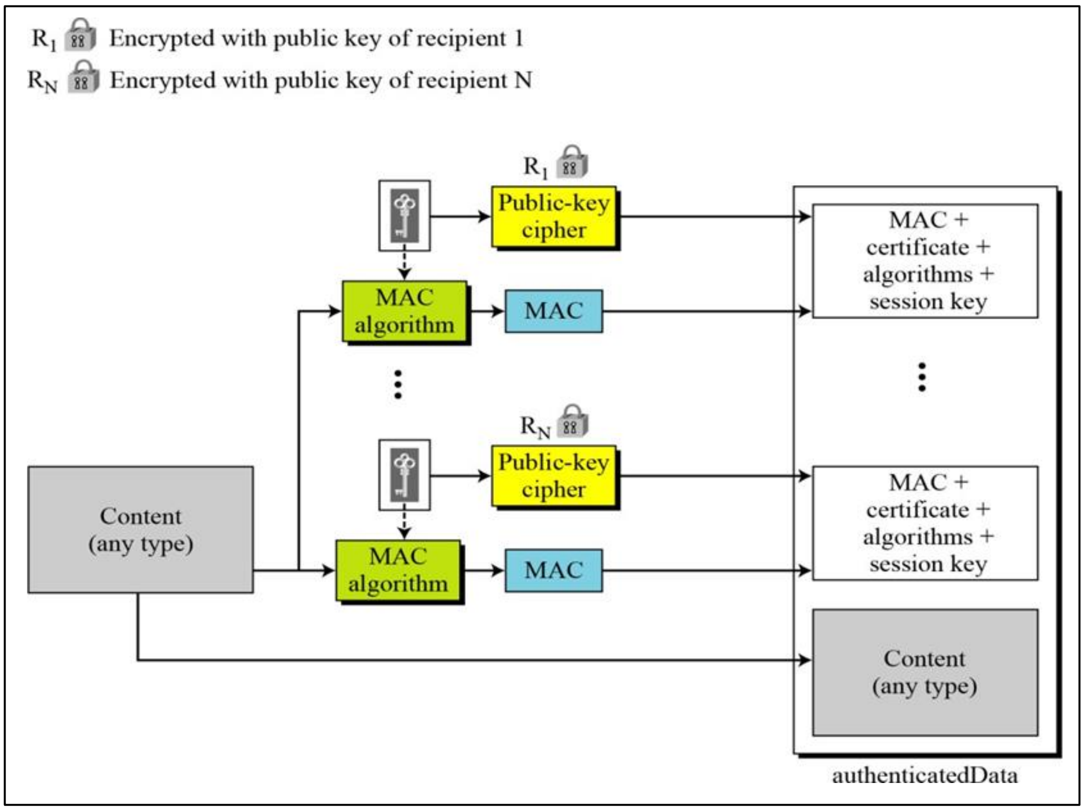


Figure 16.29 Digest-data content type

e) Encrypted-Data content type: -
It used to create an encrypted version of any content type. It has no recipient. It is used to store the encrypted data instead of transmitting it.
f) Authenticated-data content type: -
It used to provide **Authentication of the data**. The object is called authenticatedData.
The process:

1. Using pseudorandom generator, a MAC key is generated for each recipient.
2. The MAC key is encrypted with public key of the recipient
3. A MAC is created for the content
4. The content, MAC, algorithms and other information are collected together to form the autheticatedData object.



R_1 Encrypted with public key of recipient 1
R_N Encrypted with public key of recipient N
R_1
Public-key cipher
MAC algorithm
MAC
R_N
Public-key cipher
MAC algorithm
MAC
Content (any type)
MAC + certificate + algorithms + session key
MAC + certificate + algorithms + session key
Content (any type)
authenticatedData

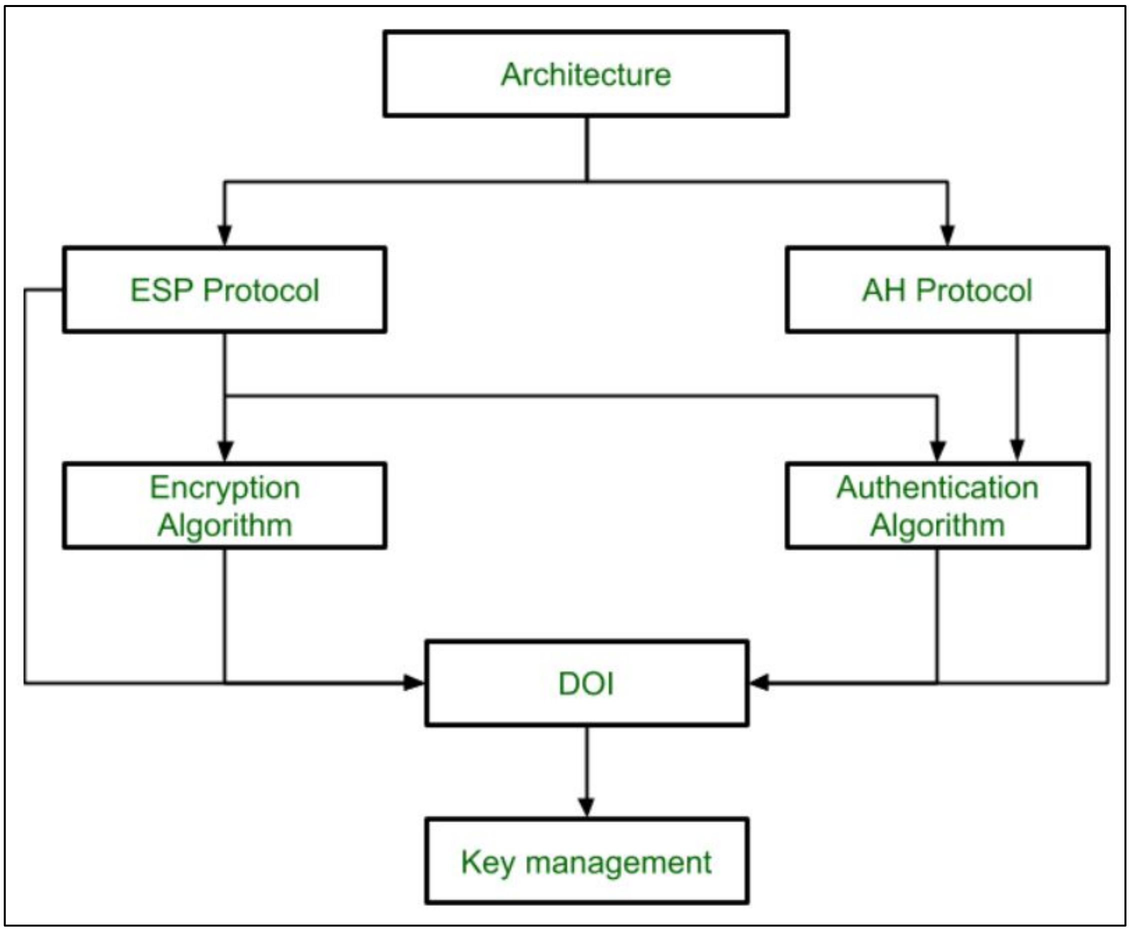
Figure 16.30 Authenticated-data content type

5. Draw and discuss the Architecture of IPSec

IPSec (IP Security) architecture uses two protocols to secure the traffic or data flow. These protocols are ESP (Encapsulation Security Payload) and AH (Authentication Header). IPSec Architecture includes protocols, algorithms, DOI, and Key Management. All these components are very important in order to provide the three main services:

- Confidentiality
- Authentication
- Integrity

IP Security Architecture:

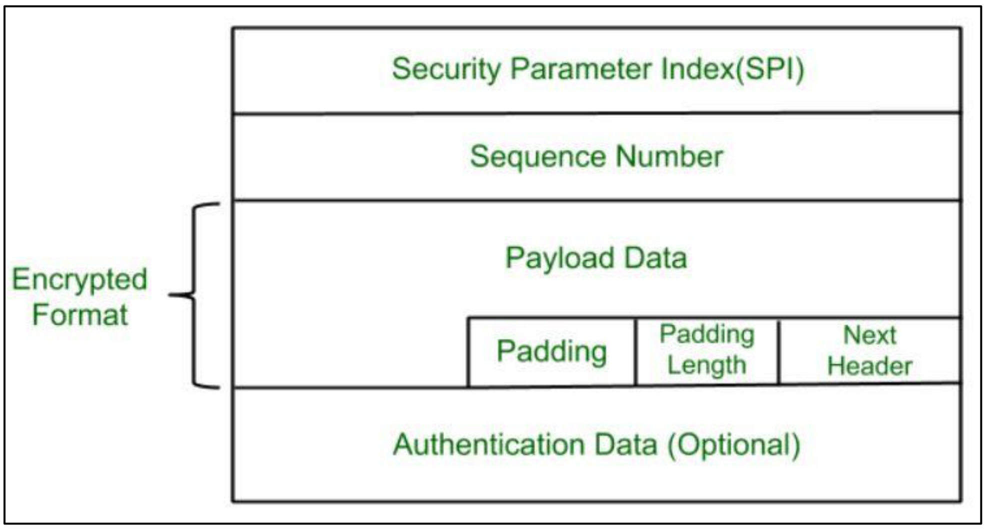


1. Architecture: Architecture or IP Security Architecture covers the general concepts, definitions, protocols, algorithms, and security requirements of IP Security technology.

2. ESP Protocol: ESP(Encapsulation Security Payload) provides a confidentiality service. Encapsulation Security Payload is implemented in either two ways:

- ESP with optional Authentication.
- ESP with Authentication.

Packet Format:

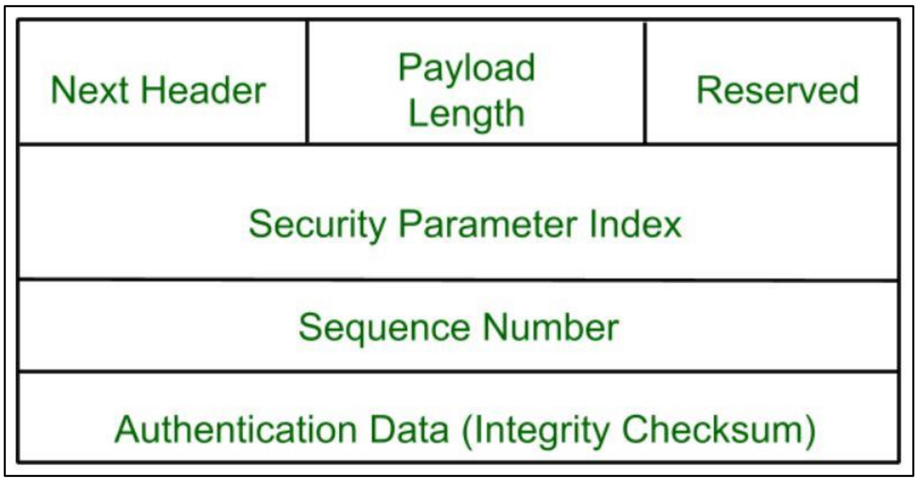


- **Security Parameter Index(SPI):** This parameter is used by Security Association. It is used to give a unique number to the connection built between the Client and Server.

- **Sequence Number:** Unique Sequence numbers are allotted to every packet so that on the receiver side packets can be arranged properly.
- **Payload Data:** Payload data means the actual data or the actual message. The Payload data is in an encrypted format to achieve confidentiality.
- **Padding:** Extra bits of space are added to the original message in order to ensure confidentiality. Padding length is the size of the added bits of space in the original message.
- **Next Header:** Next header means the next payload or next actual data.
- **Authentication Data** This field is optional in ESP protocol packet format.

3. Encryption algorithm: The encryption algorithm is the document that describes various encryption algorithms used for Encapsulation Security Payload.

4. AH Protocol: AH (Authentication Header) Protocol provides both Authentication and Integrity service. Authentication Header is implemented in one way only: Authentication along with Integrity.



Authentication Header covers the packet format and general issues related to the use of AH for packet authentication and integrity.

5. Authentication Algorithm: The authentication Algorithm contains the set of documents that describe the authentication algorithm used for AH and for the authentication option of ESP.

6. DOI (Domain of Interpretation): DOI is the identifier that supports both AH and ESP protocols. It contains values needed for documentation related to each other.

7. Key Management: Key Management contains the document that describes how the keys are exchanged between sender and receiver.

6. Differentiate the packet structure of ESP and AH.

Feature	ESP	AH
Purpose	Provides confidentiality, integrity, and authentication for IP packets	Provides integrity and authentication for IP packets
Location in IP packet	Usually encapsulates the entire original IP packet	Placed between the IP header and the upper-layer protocol header (such as TCP or UDP)
Encapsulation	Encrypts and optionally authenticates the entire inner IP packet, including the payload	Only authenticates selected portions of the IP header and payload
Authentication	Provides authentication through integrity checks and optional encryption	Provides authentication by including a hash of the IP header and payload
Payload Protection	Encrypts the payload and optionally authenticates it	Doesn't encrypt the payload; only authenticates it
Sequence Number	Includes a sequence number to prevent replay attacks	Includes a sequence number to prevent replay attacks
Anti-Replay Protection	Offers anti-replay protection by using sequence numbers and anti-replay window	Offers anti-replay protection using sequence numbers and a sliding window mechanism
Additional Overhead	Adds additional overhead due to encryption and potentially padding	Adds less overhead compared to ESP, as it doesn't encrypt the payload
Compatibility	May encounter issues with Network Address Translation (NAT) due to encryption	Generally compatible with NAT as it doesn't encrypt the entire packet
Common Algorithms	Supports various encryption and authentication algorithms like AES, HMAC, etc.	Typically uses HMAC for authentication
Extension Headers Support	Compatible with IPv6 and can be used with IPv4 using tunnel mode	Compatible with both IPv4 and IPv6

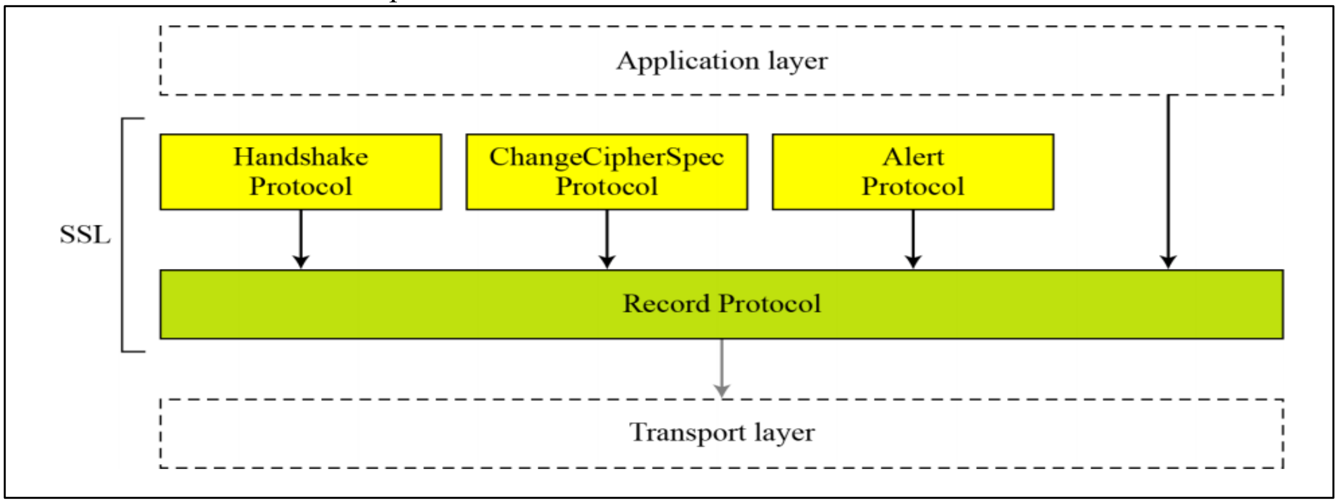
7. Explain about SSL protocol in detail

SSL - Secure Socket Layer

- SSL was developed by Netscape.
- It provides server authentication, client authentication and encrypted communication b/w two machines.
- TSL is derived from SSLv3. Therefore TSL is SSLv3.1
- SSL and TSL cryptographic protocols are referred to as "SSL".
- SSL uses a combination of public-key and symmetric-key encryption to secure a connection between two machines.
- These are used in applications such as web browser, e-mail, voice-over-IP(VoIP), File transfer etc

<u>SSL Architecture</u>

- SSL uses TCP for reliable end-to-end secure communication b/w two machines.
- SSL is two layers of protocols.
- Two layers of protocols
  - Higher layer
    - Handshake protocol
    - change cipher spec protocol
    - Alert protocol
  - Lower layer
    - Record protocol



- **SSL Record Protocol:**
  Handles data security and integrity; encapsulates data sent by higher level protocols, such as HTTP.
- **Handshake, Cipher change, Alert:**
  Establish a connection; session management, crypto management, SSL message transfer

Handshake Protocol

This protocol allows the server and client to

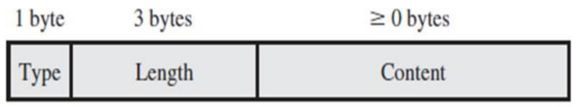
- authenticate each other
- negotiate an encryption and MAC algorithm
- agree cryptographic keys to be used to protect payload data

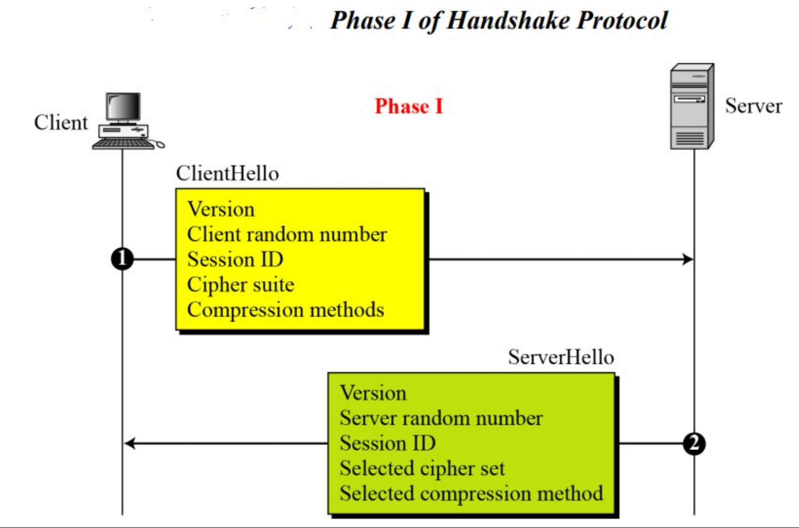
The handshake protocol is used before any application data is transmitted. It establishes a **session**.

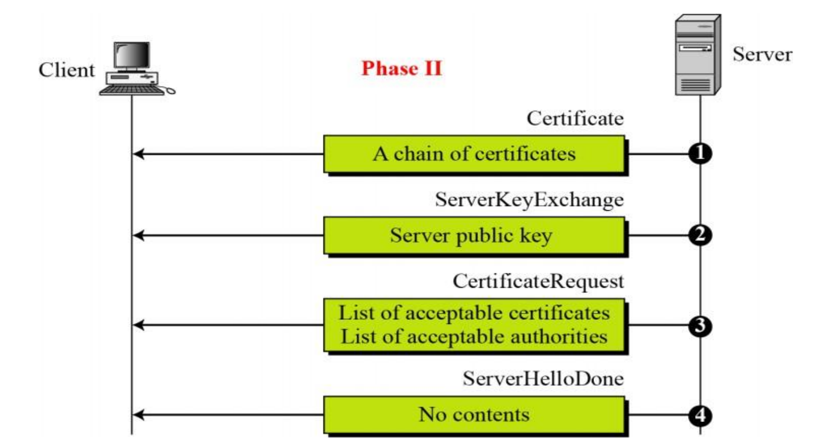
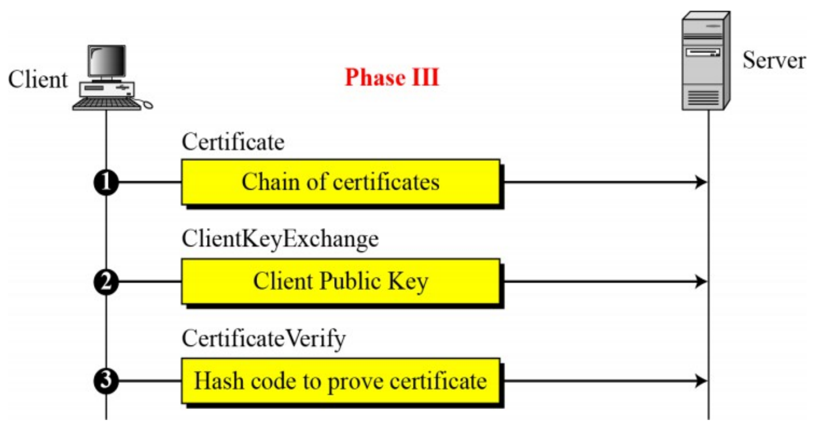
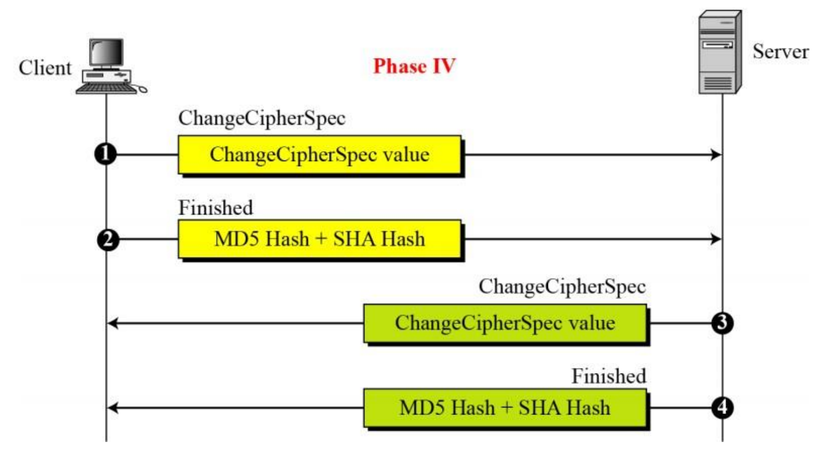
- A session defines the set of cryptographic security parameters to be used
- Multiple secure connections between client and server can share the same session
    - reduces computation cost

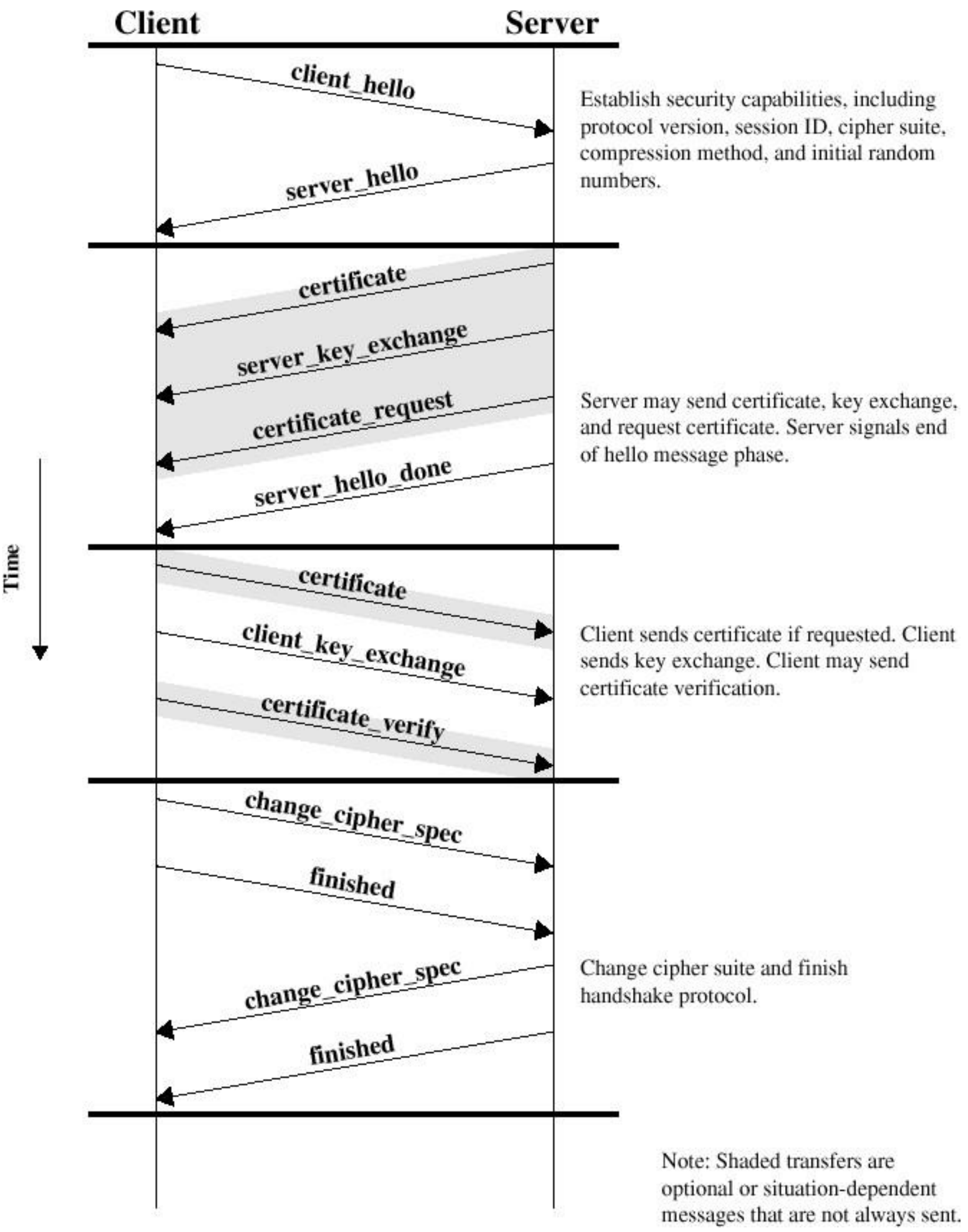
The Handshake Protocol consists of a series of messages exchanged by client and server, in **four phases.**

Phase 1. Establish Security Capabilities
Phase 2. Server authentication and key exchange
Phase 3. Client authentication and key exchange
Phase 4. Finalizing the Handshake protocol.

Handshake protocol Message Format  (c) Handshake Protocol	**Type (1 byte):** Indicates one of 10 messages.(ex: Hello, Certificate, key exchange) **Length (3 bytes):** The length of the message. **Content (bytes):** The parameters associated with this message.

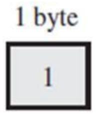
 Phase I of Handshake Protocol Phase I Client Server ClientHello Version Client random number Session ID Cipher suite Compression methods ServerHello Version Server random number Session ID Selected cipher set Selected compression method	**After Phase I, the client and server know the following:** ❑ The version of SSL ❑ The algorithms for key exchange, message authentication, and encryption ❑ The compression method ❑ The two random numbers for key generation

Phase II of Handshake Protocol  Client Phase II Server Certificate A chain of certificates 1 ServerKeyExchange Server public key 2 CertificateRequest List of acceptable certificates List of acceptable authorities 3 ServerHelloDone No contents 4	**After Phase II,** ❑ The server is authenticated to the client. ❑ The client knows the public key of the server if required.
Phase III of Handshake Protocol  Client Phase III Server Certificate 1 Chain of certificates ClientKeyExchange 2 Client Public Key CertificateVerify 3 Hash code to prove certificate	**After Phase III,** ❑ The client is authenticated for the server. ❑ Both the client and the server know the pre-master secret.
Phase IV of Handshake Protocol  Client Phase IV Server ChangeCipherSpec 1 ChangeCipherSpec value Finished 2 MD5 Hash + SHA Hash ChangeCipherSpec ChangeCipherSpec value 3 Finished MD5 Hash + SHA Hash 4	**After Phase IV**, the client and server are ready to exchange data

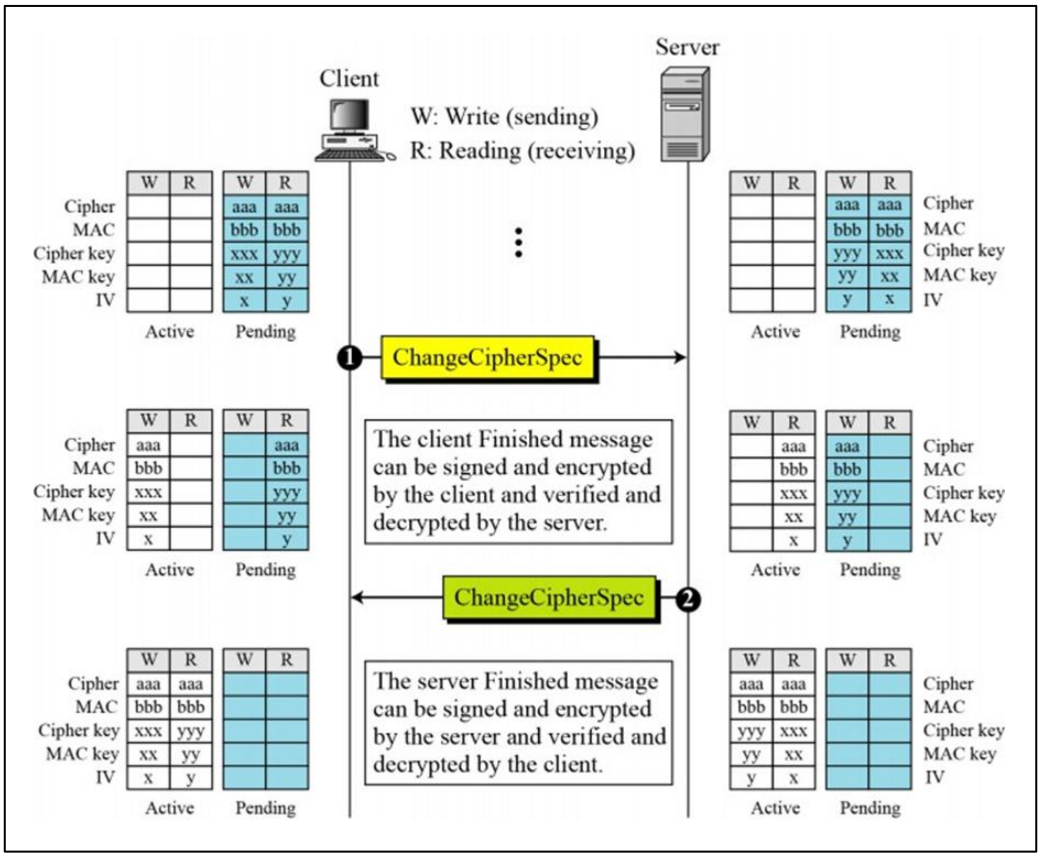

Client
Server
client_hello
server_hello
Establish security capabilities, including protocol version, session ID, cipher suite, compression method, and initial random numbers.
certificate
server_key_exchange
certificate_request
server_hello_done
Server may send certificate, key exchange, and request certificate. Server signals end of hello message phase.
Time
certificate
client_key_exchange
certificate_verify
Client sends certificate if requested. Client sends key exchange. Client may send certificate verification.
change_cipher_spec
finished
change_cipher_spec
finished
Change cipher suite and finish handshake protocol.
Note: Shaded transfers are optional or situation-dependent messages that are not always sent.

ChangeCipherSpec Protocol

- It uses the SSL Record Protocol.
- This protocol consists of a single message, which consists of a single byte with the value 1.
- It causes the pending state to be copied into the current state, which updates the cipher suite to be used on this connection.

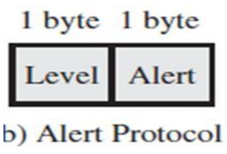


Change Cipher Spec Protocol

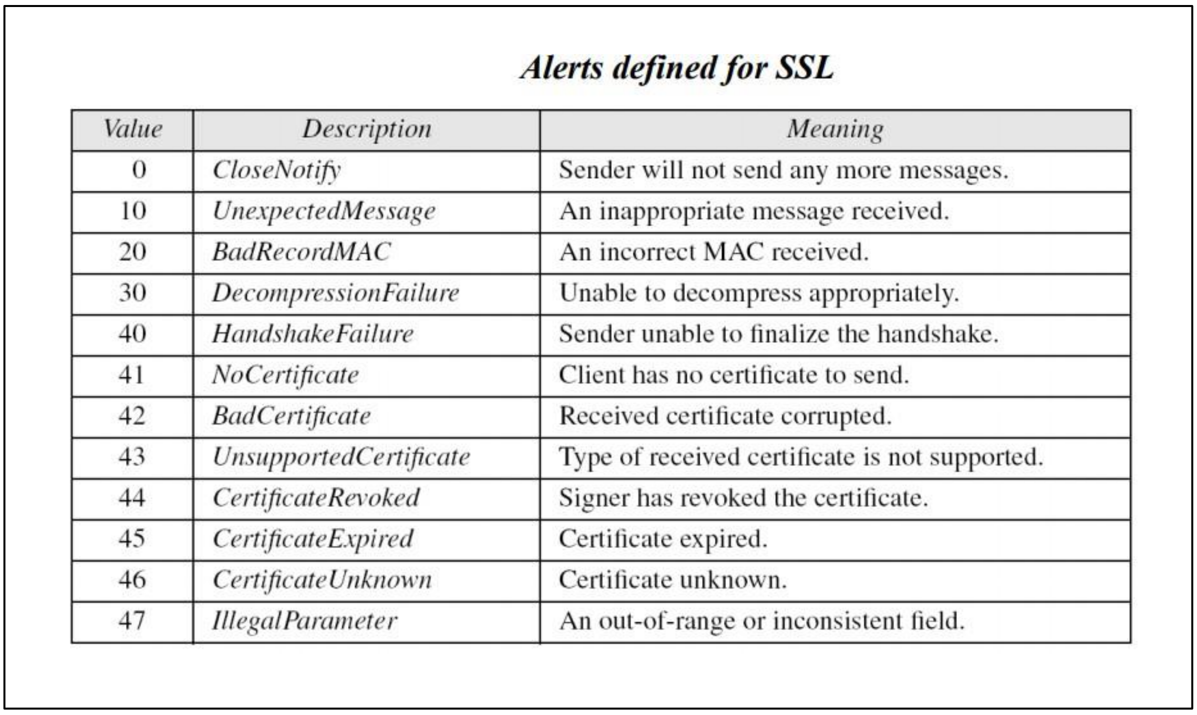

Client
Server
W: Write (sending)
R: Reading (receiving)
W R
Cipher
MAC
Cipher key
MAC key
IV
aaa
bbb
xxx
yyy
xx
yy
x
y
Active
Pending
ChangeCipherSpec
The client Finished message can be signed and encrypted by the client and verified and decrypted by the server.
ChangeCipherSpec
The server Finished message can be signed and encrypted by the server and verified and decrypted by the client.

5.8 Alert protocol

- It conveys SSL-related alerts to peer.
- Alert messages are compressed and encrypted.
- Each message consists of two bytes



- 1st byte indicates alert level
  - Warning or fatal
  - SSL will immediately terminates the connection
  - Ex: UnexpectedMessage, BadRecordMAC, DecompressionFailure etc.
- 2nd byte indicates specific alerts
  - Warning alerts
  - Ex: CloseNotify, NoCertificate, BadCertificate etc.



Alerts defined for SSL

Value	Description	Meaning
0	*CloseNotify*	Sender will not send any more messages.
10	*UnexpectedMessage*	An inappropriate message received.
20	*BadRecordMAC*	An incorrect MAC received.
30	*DecompressionFailure*	Unable to decompress appropriately.
40	*HandshakeFailure*	Sender unable to finalize the handshake.
41	*NoCertificate*	Client has no certificate to send.
42	*BadCertificate*	Received certificate corrupted.
43	*UnsupportedCertificate*	Type of received certificate is not supported.
44	*CertificateRevoked*	Signer has revoked the certificate.
45	*CertificateExpired*	Certificate expired.
46	*CertificateUnknown*	Certificate unknown.
47	*IllegalParameter*	An out-of-range or inconsistent field.

5.9 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.
- **Message Integrity**:
  - The Handshake Protocol also defines a shared secret key that is used to form a message authentication code (MAC).

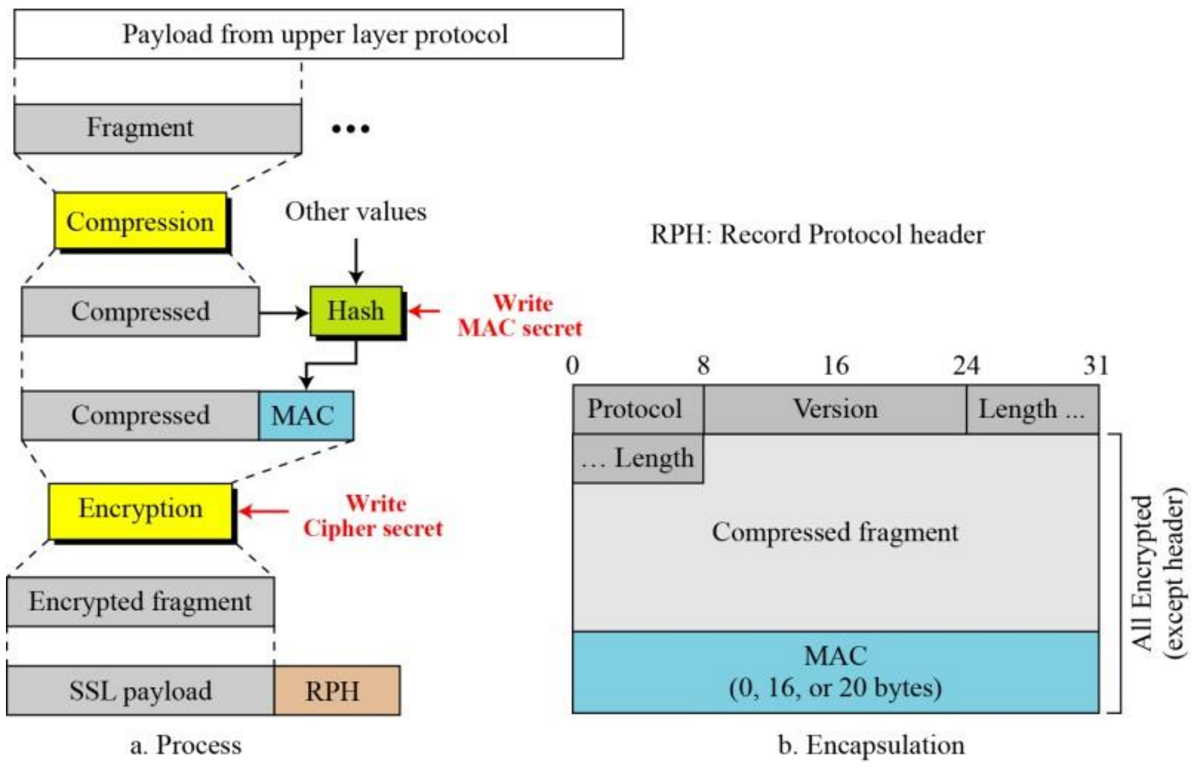
Record Protocol operation:

1. **Fragmentation**: Each upper-layer message is fragmented into blocks of 2^{14} bytes or less.
2. **Compression**: Optional step and may not increase the content length by more than 1024 bytes
3. **Message Authentication Code**: A shared secret key is used to compute MAC.
4. **Encryption**: compressed message and MAC are encrypted using symmetric encryption.
5. **RPH :** Record Protocol Header preparation:

<u>Header consists of the following:</u>

- Protocol (1 byte): The higher-layer protocol used to process the enclosed fragment.
- Version (2 bytes): holds Major and Minor version of SSL.
- Length (2 bytes): The length in bytes of the plaintext fragment (or compressed fragment).

Processing done by the Record Protocol



8. What is the use of SSL protocol? Explain SSL record protocol operation with SSL record format.

SSL ensures the data that is transferred between a client and a server remains private. This protocol enables the client to authenticate the identity of the server. When your server has a digital certificate, SSL-enabled browsers can communicate securely with your server, using SSL.

Record Protocol

- The record protocol carries messages from the upper layers.
- The message is fragmented and optionally compressed; a MAC is added to the compressed message using the negotiated hash algorithm.
- The compressed fragment and the MA using the negotiated encryption algorithm.
- Finally the SSL header encrypted message. Figure 17.21 shows this process at the sender.
- 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.
- **Message Integrity:** The Handshake Protocol also defines a shared secret key that is used to form a message authentication code (MAC).

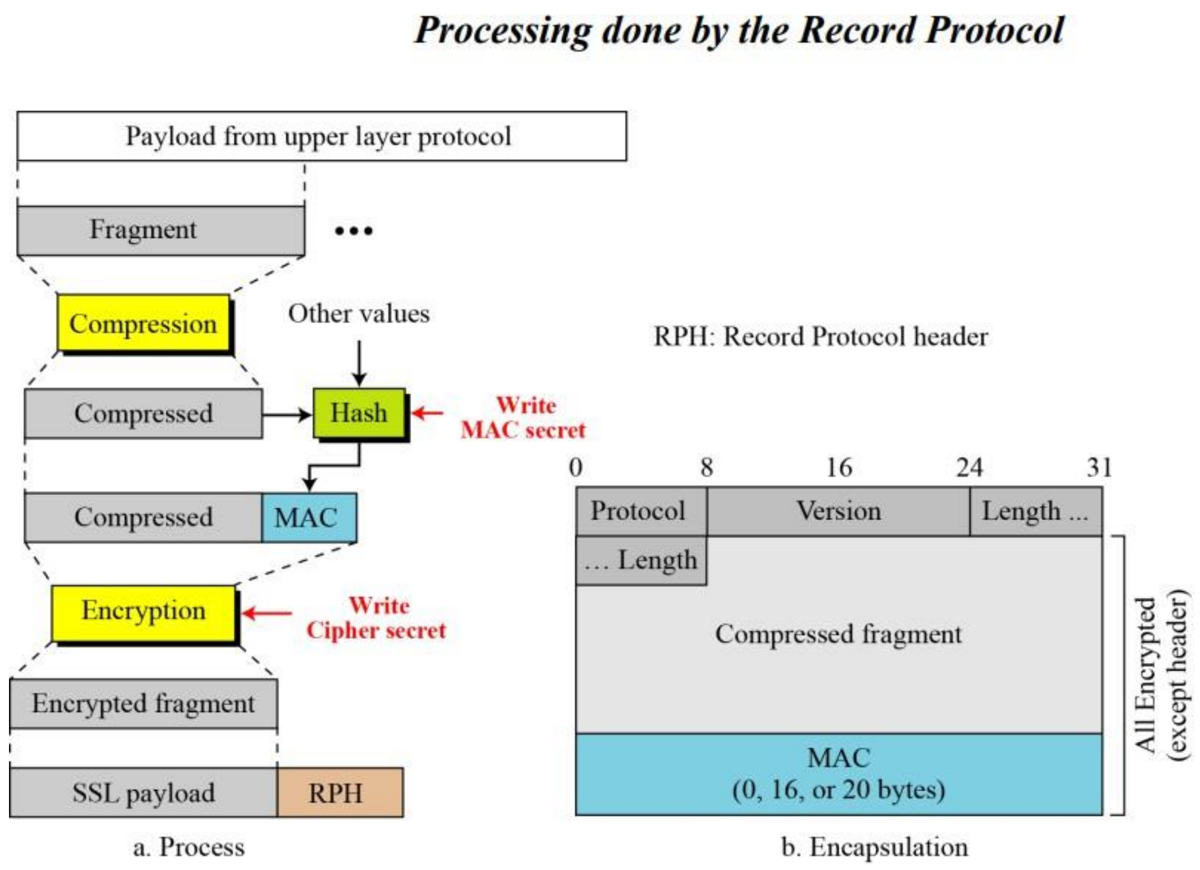


Figure 17.21 Processing done by the Record Protocol

Record Protocol operation:

Fragmentation/Combination

- At the sender, a message from the application layer is fragmented bytes, with the last block possibly less than the size.
- At the receiver combined together to make a replica of the original message.

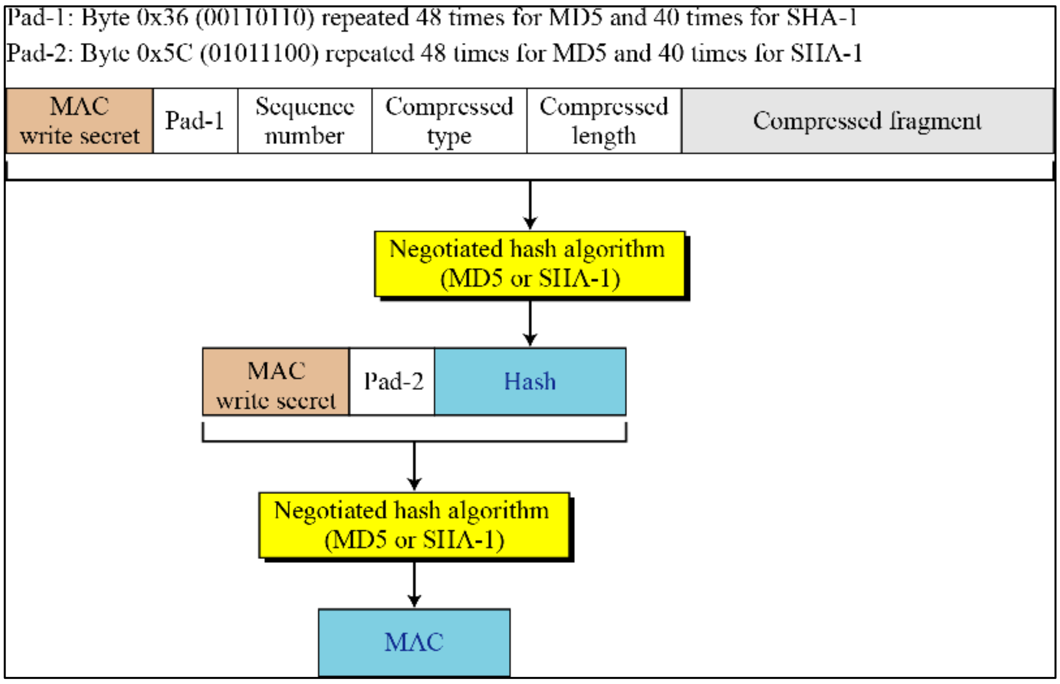
Compression/Decompression

- At the sender site, application layer fragments are compressed by the compression method negotiated during the handshaking.
- The compression method needs to be lossless.
- The size of the fragment must not exceed 1024 bytes.
- At the receiver, the compressed fragment is decompressed to create a replica of the original.
- If the decomposed fragment exceeds 214, a fatal decompression Alert message is issued.
- Note that compression/Decompression is optional in SSL.

Signing / Verifying

- At the sender, the authentication method defined during the handshake creates a signature (MAC) as shown in figure 17.22.Figure 17.22 Calculation of MAC
- The hash algorithm is applied twice. First, a hash is created from the concatenations of the following values.
    a. The MAC write secret (authentication key for the outbound message)
    b. Pad-1, which is the byte 0x36 repeated 48 times for MD5 and 40 times for SHA-1
    c. The sequence number for this message.
    d. The compressed type, which define the upper-layer protocol that provided the compressed fragment.

e. The compressed length, which is the length of the compressed format.
f. The compressed fragment itself.



Pad-1: Byte 0x36 (00110110) repeated 48 times for MD5 and 40 times for SHA-1
Pad-2: Byte 0x5C (01011100) repeated 48 times for MD5 and 40 times for SHA-1
MAC write secret
Pad-1
Sequence number
Compressed type
Compressed length
Compressed fragment
Negotiated hash algorithm (MD5 or SHA-1)
MAC write secret
Pad-2
Hash
Negotiated hash algorithm (MD5 or SHA-1)
MAC

Figure 17.22 Calculation of MAC

- Second, the final hash (MAC) is created from the concatenation of the following values:
  a. The MAC write secret.
  b. Pad 2, which is the byte 0x5C repeated 48 times for MD5 and 40 times for SHA-1.
  c. The hash created from the first step.
- At the receiver, the verifying is done by calculating a new hash and comparing it to the received hash.

Encryption/Decryption

- At the sender site, the compressed fragment and the hash are encrypted using the cipher write secret. At the receiver, the received message is decrypted using the cipher read secret. For block encryption, padding is added to make the size of the encryptable message a multiple of the block size.

Framing/Deframing

- After the encryption, the Record Protocol header is added at the sender. The header is removed at the receiver before decryption.

9. Explain Advantages and Disadvantages of Packet Filters, Circuit-Level Firewalls, and Application Layer Firewalls

1. PACKET FILTER

Advantages

1. Packet filters are faster than other techniques.

2. Less complicated, in the sense that a single rule controls deny or allow of packets.

3. They do not require client computers to be configured specially.

4. They shield the internal IP address from the external world,

Disadvantages

1. Packet filters do not understand application layer protocols and hence cannot restrict access to FTP services, such and PUT and GET commands.

2. They are stateless, and hence not suitable for application layer protocols.

3. Packet filters have almost no audit event generation and alerting mechanisms.

2. CIRCUIT-LEVEL FIREWALLS

Advantages

1. They are faster than application layer firewalls.

2. They are more secured than packet filter firewalls.

3. They maintain limited state information of the protocols.

4. They protect against spoofing of packets.

5. They shield internal IP addresses from external networks by network address translation.

Disadvantages

1. They cannot restrict access to protocol subsets other than TCP.

2. They have limited audit event generation capabilities.

3. They cannot perform security checks on higher level protocols.

3.APPLICATION LAYER FIREWALL

Advantages

1. They enforce and understand high level protocols, like HTTP and FTP.
2. They maintain information about the communication passing through the firewall server: partial communication derived state information, full application derived state information, partial session information.
3. They can be used to deny access to certain network services, while allowing others.
4. They are capable of processing and manipulating packet data.
5. They do not allow direct communication between external servers and internal systems, thus shielding internal IP addresses from the outside network.

6. They are transparent between the user and the external network.
7. They provide features like HTTP object caching, URL filtering, and user authentication.
8. They are good at generating auditing records, allowing administrators to monitor threats to the firewall.

Disadvantages

1. They require replacing the native network stack on the firewall server.
2. They do not allow network servers to run on the firewall servers, as the proxy servers use the same port to listen.
3. They are slow and thus lead to degradation in performance.
4. They are not scalable, as each new network service adds onto the number of proxy services required.
5. Proxy services require modifications to client procedures.
6. They rely on operating system support and thus are vulnerable to bugs in the system. Thus bugs in NDIS, TCP/IP, WinSock, Win32 or the standard C library can cause security concerns in the security provided by the application layer firewalls.

10.What is a firewall? What is the need for firewalls? What is the role of firewalls in protecting networks?

A firewall is like a security guard for your computer network. It's a barrier or a filter that monitors and controls incoming and outgoing network traffic based on predefined security rules.

The need for firewalls arises from the ever-present threat of unauthorized access, data breaches, and malicious attacks on computer networks. Here are some reasons why firewalls are essential:

1. **Protection from Unauthorized Access**: Firewalls prevent unauthorized users or malicious software from gaining access to your network. They act as a barrier between your internal network and external networks, such as the internet.
2. **Network Segmentation**: Firewalls can segment a network into different zones, each with its own security level. This helps in controlling and limiting access between different parts of the network.
3. **Traffic Filtering**: Firewalls inspect network traffic and filter out potentially harmful packets based on predefined rules. This helps in blocking malicious content, such as viruses, malware, and phishing attempts.
4. **Access Control**: Firewalls enforce access control policies, determining which users or systems are allowed to access specific resources or services on the network. This helps in preventing unauthorized access to sensitive data or services.
5. **Protection from DoS Attacks**: Firewalls can mitigate Denial of Service (DoS) attacks by detecting and blocking excessive traffic from malicious sources attempting to overwhelm the network.
6. **Logging and Monitoring**: Firewalls log network traffic and security events, allowing network administrators to monitor and analyze network activity for security breaches or policy violations.

The role of firewalls in protecting networks is multifaceted:

- **Perimeter Defence**: Firewalls create a secure perimeter around the network, controlling the flow of traffic between internal and external networks.
- **Intrusion Prevention**: They detect and block unauthorized access attempts, preventing intruders from exploiting vulnerabilities in network services.
- **Content Filtering**: Firewalls inspect network traffic for malicious content and filter out harmful packets, protecting against malware and other threats.
- **Policy Enforcement**: Firewalls enforce security policies by regulating access to network resources based on predefined rules, ensuring compliance with security guidelines and regulations.
- **Traffic Monitoring**: They monitor network traffic in real-time, providing visibility into network activity and helping identify and respond to security incidents promptly.