SSL:- Secure Socket Layer

Secure Socket Layer is a security protocol that provides privacy, authentication and integrity to internet communications. SSL eventually evolved into Transport Layer Security.

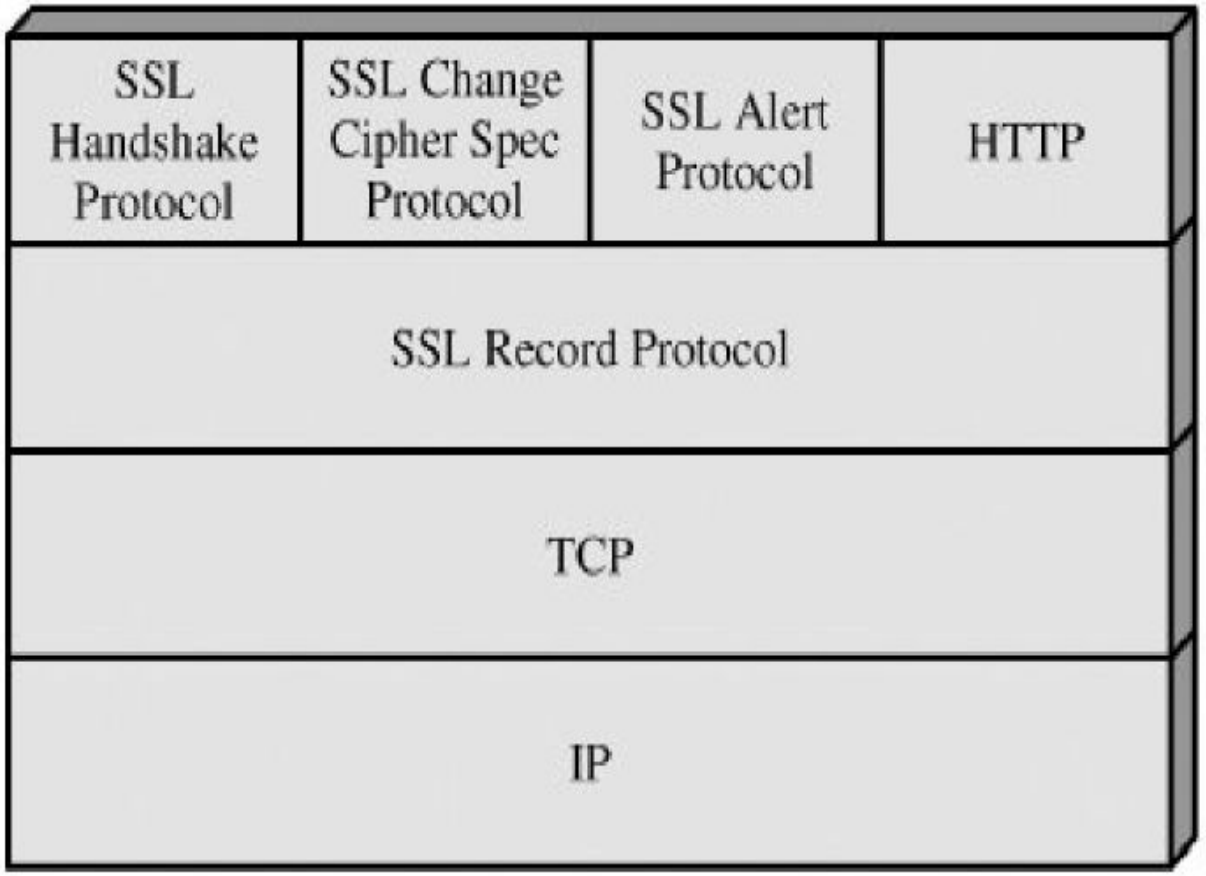


Fig : SSL Protocol Stack

- 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.

- These SSL-specific protocols are used in the management of SSL exchanges

There are 2 important concepts are SSL Connection and SSL Session

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.

A session state is defined by the following parameters (definitions taken from the SSL specification):

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.

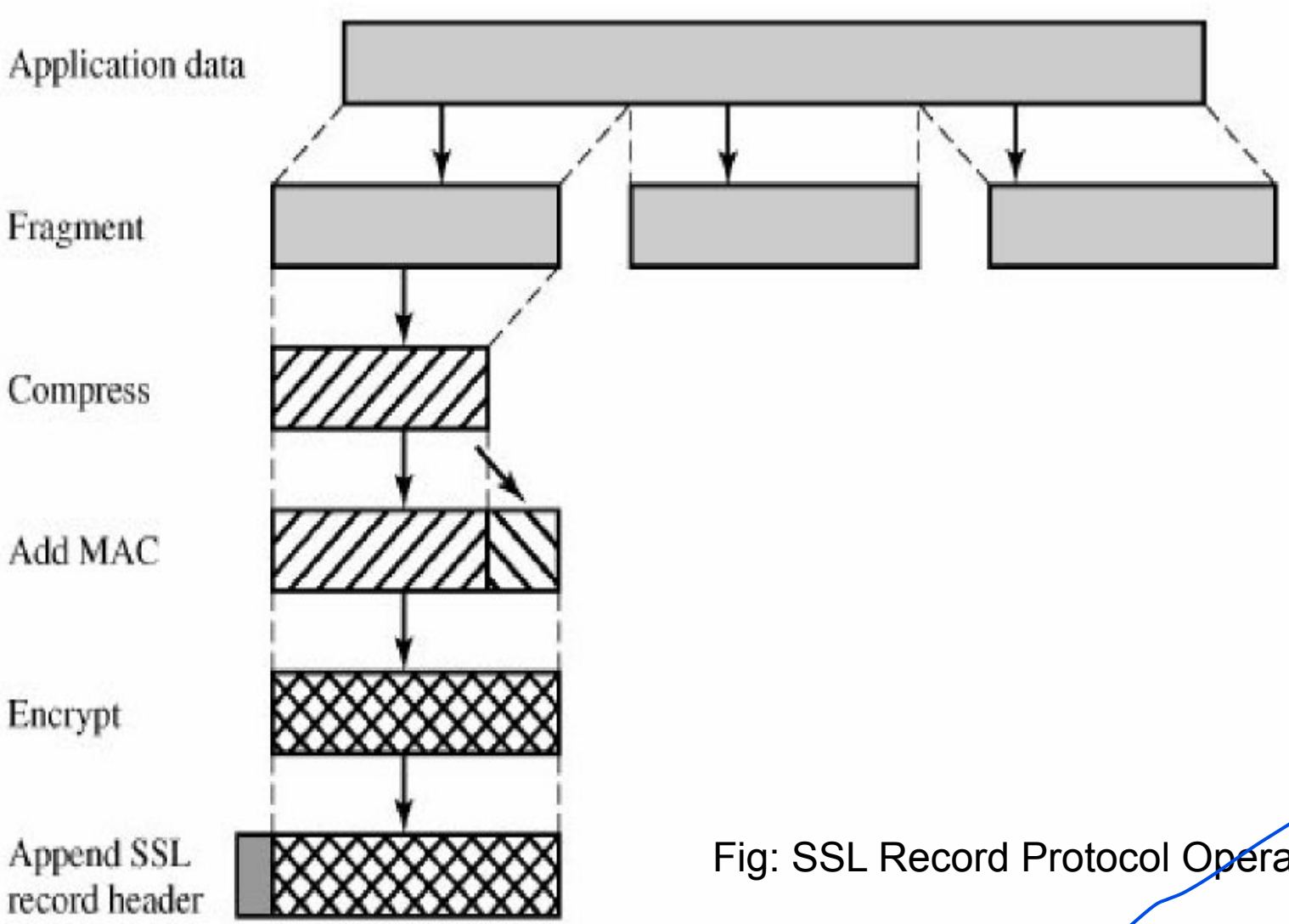


Fig: SSL Record Protocol Operation

SSL record protocol provides 2 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).

SSL record protocol operation:-- Adding Message Authentication Code

MAC:
hash(MAC_write_secret || pad_2 ||hash(MAC_write_secret || pad_1 || seq_num ||
SSLCompressed.type ||SSLCompressed.length || SSLCompressed.fragment))

Where,

|| = concatenation
MAC_write_secret = shared secret key
hash = cryptographic hash algorithm; either MD5 or SHA-1
pad_1 = the byte 0x36 (0011 0110) repeated 48 times (384 bits) for MD5 and 40 times (320 bits) for SHA-1
pad_2 = the byte 0x5C (0101 1100) repeated 48 times for MD5 and 40 times for SHA-1
seq_num = the sequence number for this message
SSLCompressed.type = the higher-level protocol used to process this fragment
SSLCompressed.length = the length of the compressed fragment
SSLCompressed.fragment = the compressed fragment (if compression is not used, the plaintext fragment)

Compressed msg plus mac are encrypted using following algorithms.

Block Cipher		Stream Cipher	
Algorithm	Key Size	Algorithm	Key Size
AES	128,256	RC4-40	40
IDEA	128	RC4-128	128
RC2-40	40		
DES-40	40		
DES	56		
3DES	168		
Fortezza	80		

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

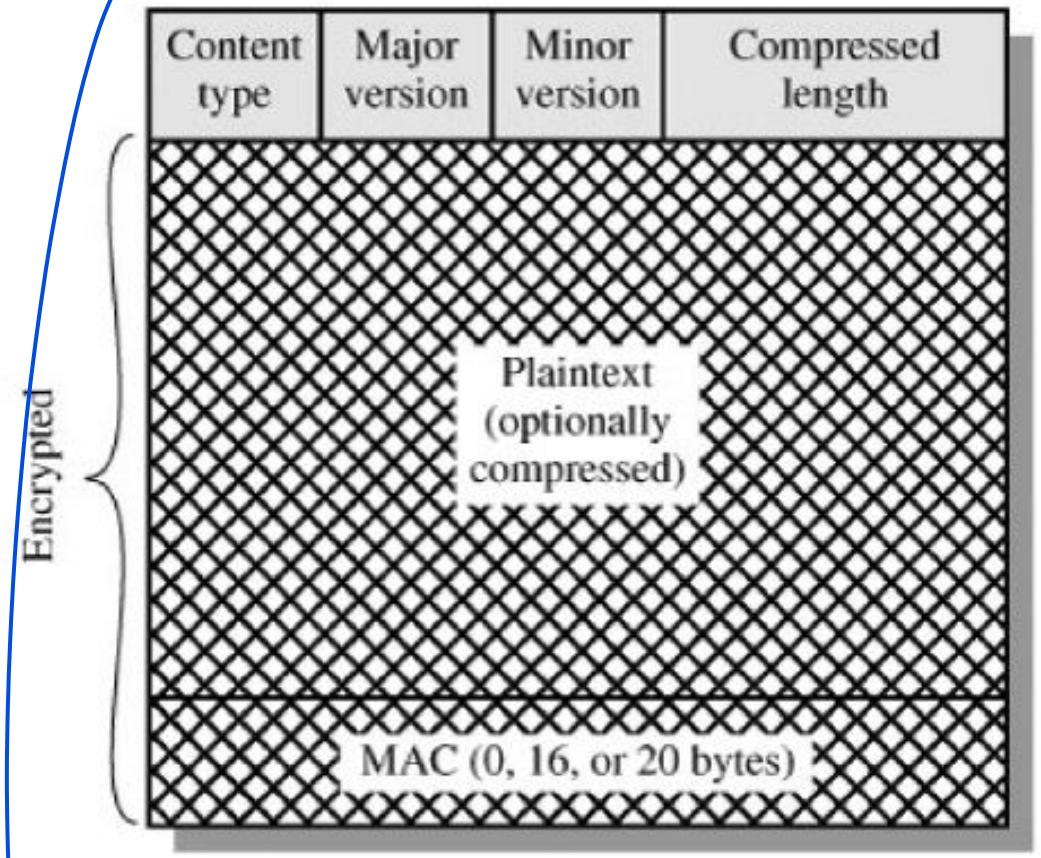
Content Type (8 bits): The higher layer protocol used to process the enclosed fragment.

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 $2^{14} + 2048$.

Figure 17.4. SSL Record Format

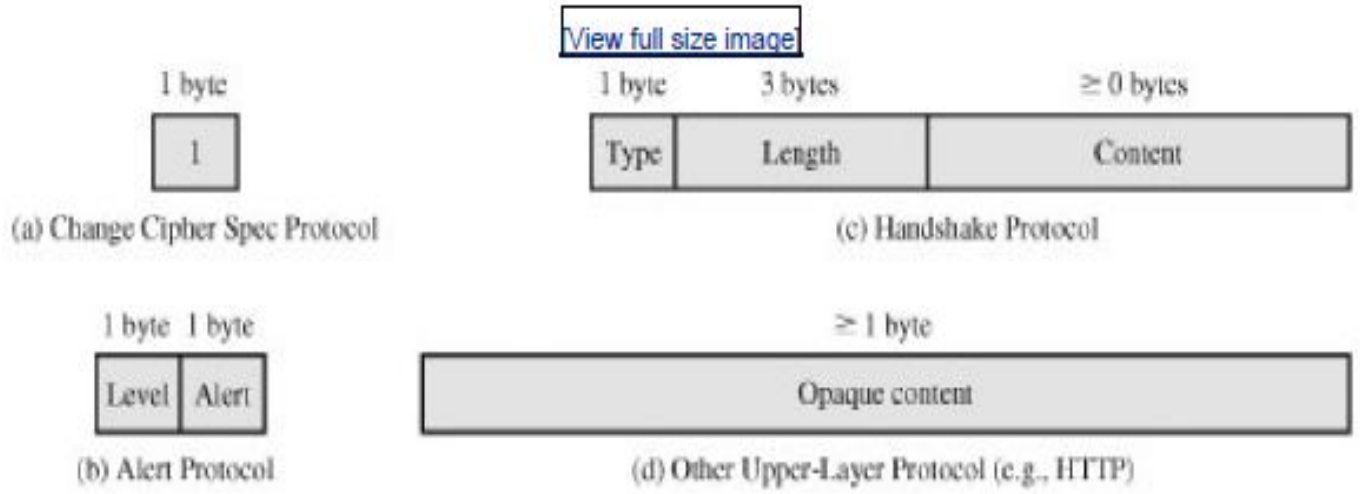


Change Cipher Spec Protocol

Figure 17.5. SSL Record Protocol Payload

(This item is displayed on page 536 in the print version)

View full size image



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 the following:

close_notify: Notifies the recipient that the sender will not send any more messages on this connection. Each party is required to send a close_notify alert before closing the write side of a connection.

no_certificate: May be sent in response to a certificate request if no appropriate certificate is available.

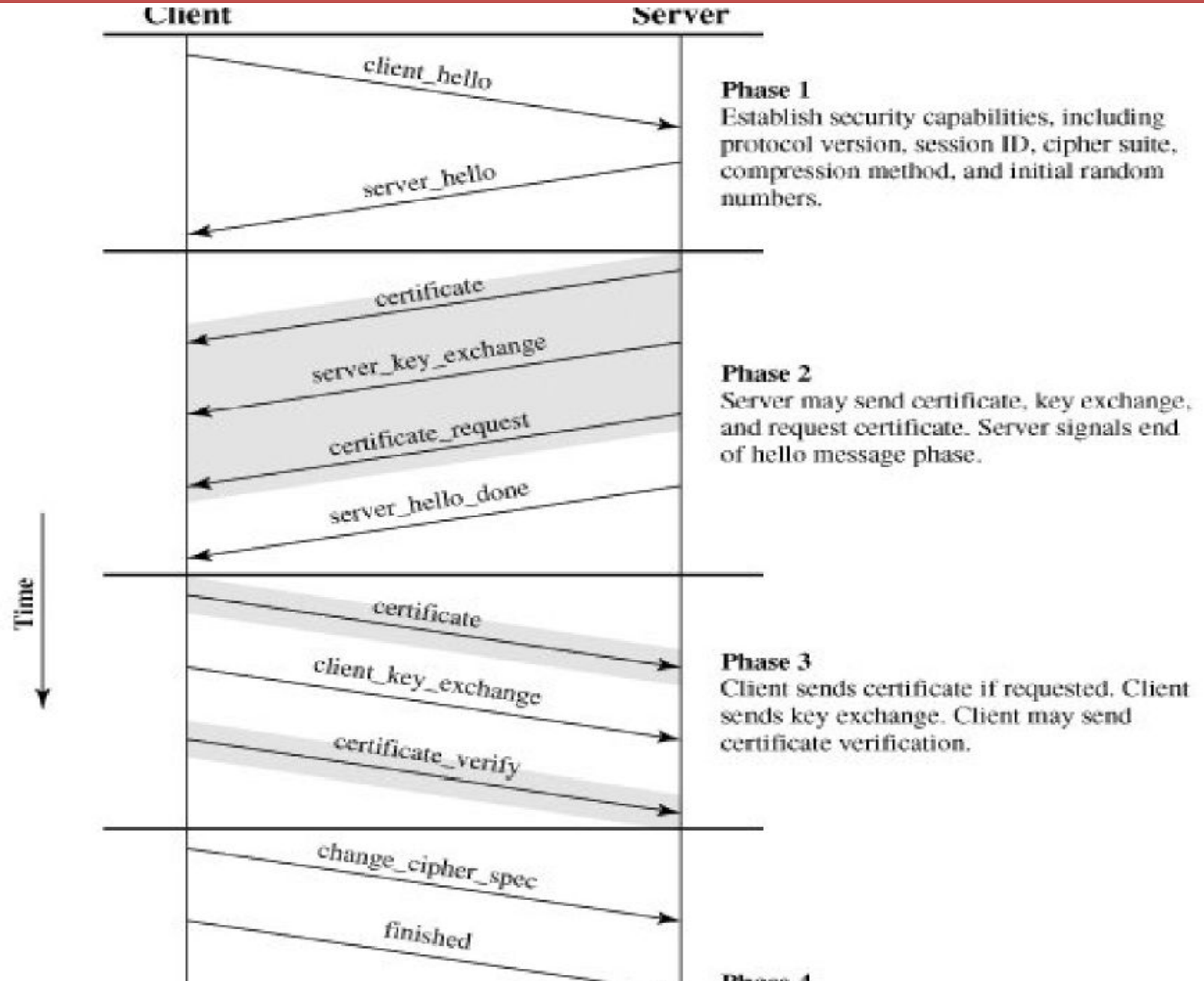
bad_certificate: A received certificate was corrupt (e.g., contained a signature that did not verify).

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.


Client
Server
client_hello
server_hello
Phase 1
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
Phase 2
Server may send certificate, key exchange, and request certificate. Server signals end of hello message phase.
Time
certificate
client_key_exchange
certificate_verify
Phase 3
Client sends certificate if requested. Client sends key exchange. Client may send certificate verification.
change_cipher_spec
finished

Unit-05 System Security—Handshake Protocol

The Handshake Protocol consists of a series of messages exchanged by client and server. All of these have the format shown in Figure 17.5c. Each message has three fields:

- **Type (1 byte):** Indicates one of 10 messages. Table 17.2 lists the defined message types.
- **Length (3 bytes):** The length of the message in bytes.
- **Content ($\geq$ 0 bytes):** The parameters associated with this message; these are listed in Table 17.2.

Table 17.2. SSL Handshake Protocol Message Types

Message Type	Parameters
hello_request	null
client_hello	version, random, session id, cipher suite, compression method
server_hello	version, random, session id, cipher suite, compression method
certificate	chain of X.509v3 certificates
server_key_exchange	parameters, signature
certificate_request	type, authorities
server_done	null
certificate_verify	signature
client_key_exchange	parameters, signature
finished	hash value

Handshake Protocol

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.

Cipher uite (Cryptographic algorithms) are:

RSA: The secret key is encrypted with the receiver's RSA public key. A public-key certificate for the receiver's key must be made available.
Fixed Diffie-Hellman: This is a Diffie-Hellman key exchange in which the server's certificate contains the Diffie-Hellman public parameters signed by the certificate authority (CA). That is, the public-key certificate contains the Diffie-Hellman public-key parameters. The client provides its Diffie-Hellman public key parameters either in a certificate, if client authentication is required, or in a key exchange message. This method results in a fixed secret key between two peers, based on the Diffie-Hellman calculation using the fixed public keys.
Ephemeral Diffie-Hellman: This technique is used to create ephemeral (temporary, one-time) secret keys. In this case, the Diffie-Hellman public keys are exchanged, signed using the sender's private RSA or DSS key. The receiver can use the corresponding public key to verify the signature. Certificates are used to authenticate the public keys. This would appear to be the most secure of the three Diffie-Hellman options because it results in a temporary, authenticated key.
Anonymous Diffie-Hellman: The base Diffie-Hellman algorithm is used, with no authentication. That is, each side sends its public Diffie-Hellman parameters to the other, with no authentication. This approach is vulnerable to man-in-the-middle attacks, in which the attacker conducts anonymous Diffie-Hellman with both parties.
Fortezza: The technique defined for the Fortezza scheme.

key exchange method is the CipherSpec, which includes the following fields:

CipherAlgorithm: Any of the algorithms mentioned earlier: RC4, RC2, DES, 3DES, DES40, IDEA, Fortezza
MACAlgorithm: MD5 or SHA-1

CipherType: Stream or Block

IsExportable: True or False

HashSize: 0, 16 (for MD5), or 20 (for SHA-1) bytes

Key Material: A sequence of bytes that contain data used in generating the write keys

IV Size: The size of the Initialization Value for Cipher Block Chaining (CBC) encryption

Phase 2 : **Server Authentication and Key Exchange**

The server begins this phase by sending its certificate if it needs to be authenticated; the message contains one or a chain of x.509 certificates.

Certificate Message is required for any agreed on key exchange method except anonymous Diffie –hellman.

Server Key Exchange message is needed for the following:

-- Anonymous Diffie Hellman– The message consists of two global Diffie Hellman values plus server's public Diffie- Hellman key.
-- Ephermal Diffie Hellman
-- RSA key Exchange
--Fortezza

Hash can be used as:

hash(ClientHello.random || ServerHello.random || ServerParams)

The certificate request message contains following certificate types:

- RSA, signature only
- DSS, signature only
- RSA for fixed Diffie-Hellman; in this case the signature is used only for authentication, by sending a certificate signed with RSA
- DSS for fixed Diffie-Hellman; again, used only for authentication
- RSA for ephermeral Diffe Hellman
- DSS for ephermeral Diffie Hellman

Phase 3: Client Authentication and Key Exchange

Upon receiving the request of the server_done message, the client should verify that the server provided a valid certificate and check server_hello parameters are acceptable, if satisfactory client sends one or more messages back to server.
The Client_key_exchange same algorithms as server_key _exchange mentioned in previous phase.

However certificate_verify message contains following hash code message:

CertificateVerify.signature.md5_hash
MD5(master_secret || pad_2 || MD5(handshake_messages ||
master_secret || pad_1));
Certificate.signature.sha_hash
SHA(master_secret || pad_2 || SHA(handshake_messages ||
master_secret || pad_1));

Phase 4: Finish

This phase completes the setting up of a secure connection. Finished message contains 2 hash values shown below:

MD5(master_secret || pad2 || MD5(handshake_messages ||
Sender || master_secret || pad1))
SHA(master_secret || pad2 || SHA(handshake_messages ||
Sender || master_secret || pad1))

Cryptographic Computations:

Master Secret Creation:

The shared master secret is a one-time 48 byte value generates for this session by means of secure key exchange. The creation is in 2 stages.

a) pre_master_secret is exchanged.
b) Master_secret is calculated by both parties

For pre_master_secret exchange ,there are 2 possibilities:

->RSA: A 48 byte pre_master _secret is generated by the client, encrypted with the server's public RSA key and sent to the server. The server decrypts the ciphertext using its private key to recover the pre_master_secret.

->Diffie-Hellman: Both client and server generate a Diffie hellman public key

Now both sides master_secret as

master_secret = MD5(pre_master_secret || SHA('A' || pre_master_secret ||ClientHello.random || ServerHello.random)) || MD5(pre_master_secret || SHA('BB' || pre_master_secret || ClientHello.random || ServerHello.random)) || MD5(pre_master_secret || SHA('CCC' || pre_master_secret || ClientHello.random || ServerHello.random))

Generation of Cryptographic Parameters:

key_block = MD5(master_secret || SHA('A' || master_secret || ServerHello.random || ClientHello.random)) || MD5(master_secret || SHA('BB' || master_secret || ServerHello.random || ClientHello.random)) || MD5(master_secret || SHA('CCC' || master_ secret || ServerHello.random || ClientHello.random)) || . . .

TLS: Transport Layer Security

Transport Layer Security, or TLS, is a widely adopted security protocol designed to facilitate privacy and data security for communications over the Internet. A primary use case of TLS is encrypting the communication between web applications and servers, such as web browsers loading a website.

Version Number:
The TLS Record Format is the same as that of the SSL Record Format(as incase of SSL) and fields are also same, only the difference is version of TLS.
Major version is 3 and minor version is 3.

MAC(Message Authentication Code)-
TLS MAC scheme is called HMAC.

$$\text{HMAC}_K(M) = H[(K^+ \oplus \text{opad}) || H[(K^+ \oplus \text{ipad}) || M]]$$

where

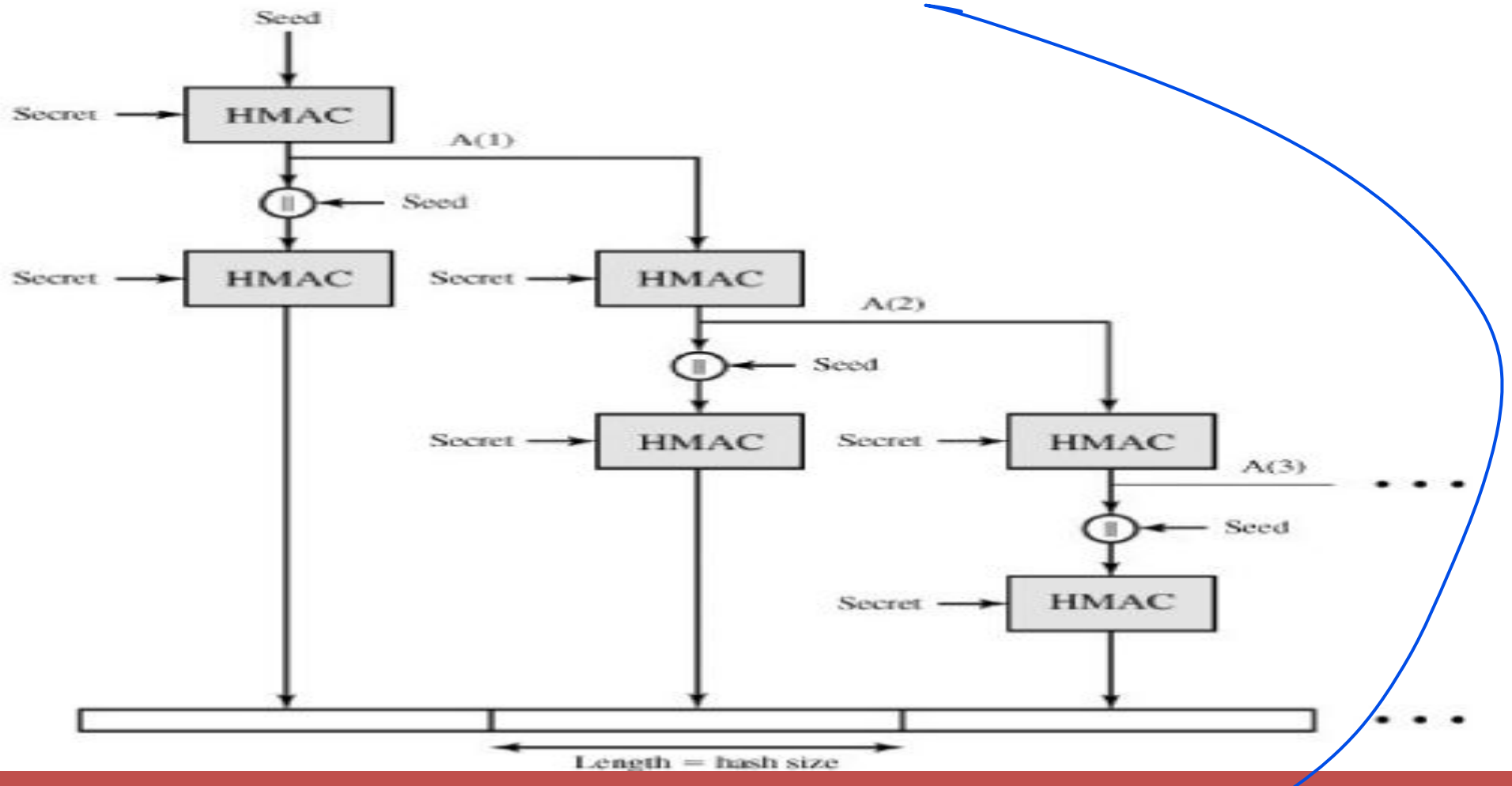
H	=	embedded hash function (for TLS, either MD5 or SHA-1)
M	=	message input to HMAC
K^+	=	secret key padded with zeros on the left so that the result is equal to the block length of the hash code(for MD5 and SHA-1, block length = 512 bits)
ipad	=	00110110 (36 in hexadecimal) repeated 64 times (512 bits)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

For TLS, the MAC calculation encompasses the fields indicated in the following expression:

MAC(MAC_write_secret, seq_num || TLScompressed.type || TLSCompressed.version ||TLSCompressed.length || TLSCompressed.fragment)

TLS: Transport Layer Security

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



Pseudo Random 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 defined as
A(0) = seed
A(*i*) = HMAC_hash (secret, A(*i* - 1))

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

PRF(secret, label, seed) = P_MD5(S1, label || seed) ⊕
P_SHA-1(S2, label || seed)

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

TLS: Transport Layer Security– Alert Codes

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

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

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

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

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

TLS: Transport Layer Security– Alert Codes

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

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

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

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

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

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

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

no_renegotiation: Sent by a client in response to a hello request or by the server in response to a client hello after initial handshaking.

TLS: Transport Layer Security– Cipher Suites

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

Key Exchange: TLS supports all of the key exchange techniques of SSLv3 with the exception of Fortezza.

Symmetric Encryption Algorithms: TLS includes all of the symmetric encryption algorithms found in SSLv3, with the exception of Fortezza.

Cryptographic Computations:

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

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

The algorithm is performed until 48 bytes of pseudorandom output are produced. The calculation of the key block material (MAC secret keys, session encryption keys, and IVs) is defined as follows:

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

Secure Electronic Transactions(SET)

- 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.

Secure Electronic Transactions(SET) Requirements

SET specification lists the following business requirements for secure payment processing with credit cards over the Internet and other networks:

- **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**

Secure Electronic Transactions(SET) Features

To meet the requirements SET incorporates following features:

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

Integrity of data: Payment information sent from cardholders to merchants includes order information, personal data, and payment instructions. SET guarantees that these message contents are not altered in transit. RSA digital signatures, using SHA-1 hash codes, provide message integrity. Certain messages are also protected by HMAC using SHA-1.

Cardholder account authentication: SET enables merchants to verify that a cardholder is a legitimate user of a valid card account number. SET uses X.509v3 digital certificates with RSA signatures for this purpose.

Merchant authentication: SET enables cardholders to verify that a merchant has a relationship with a financial institution allowing it to accept payment cards. SET uses X.509v3 digital certificates with RSA signatures for this purpose.

Secure Electronic Transactions(SET) Participants

- Card Holder
- Merchant
- Issuer
- Acquirer
- Payment
- Certificate Authority

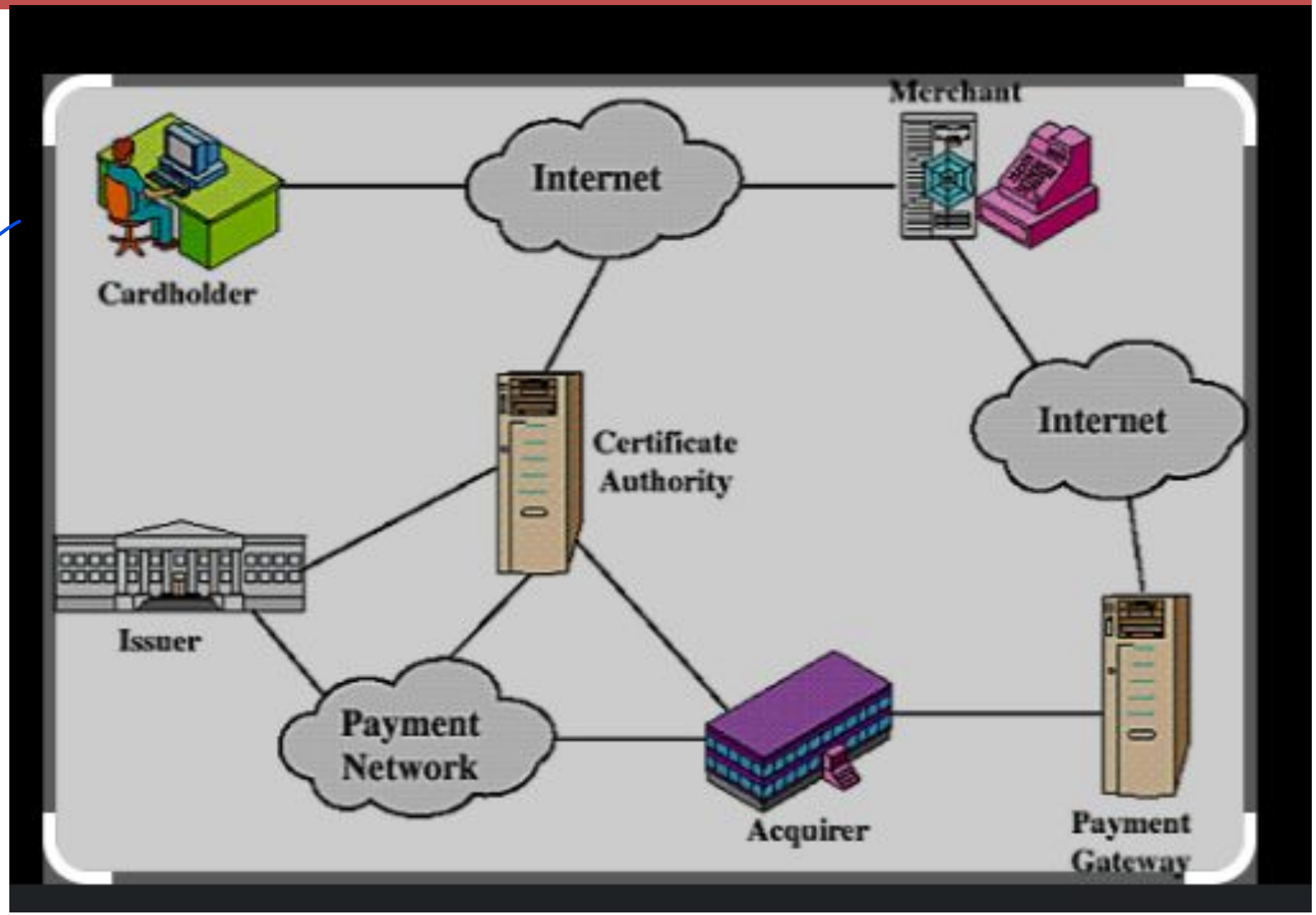


Fig: Secure Electronic Commerce Components

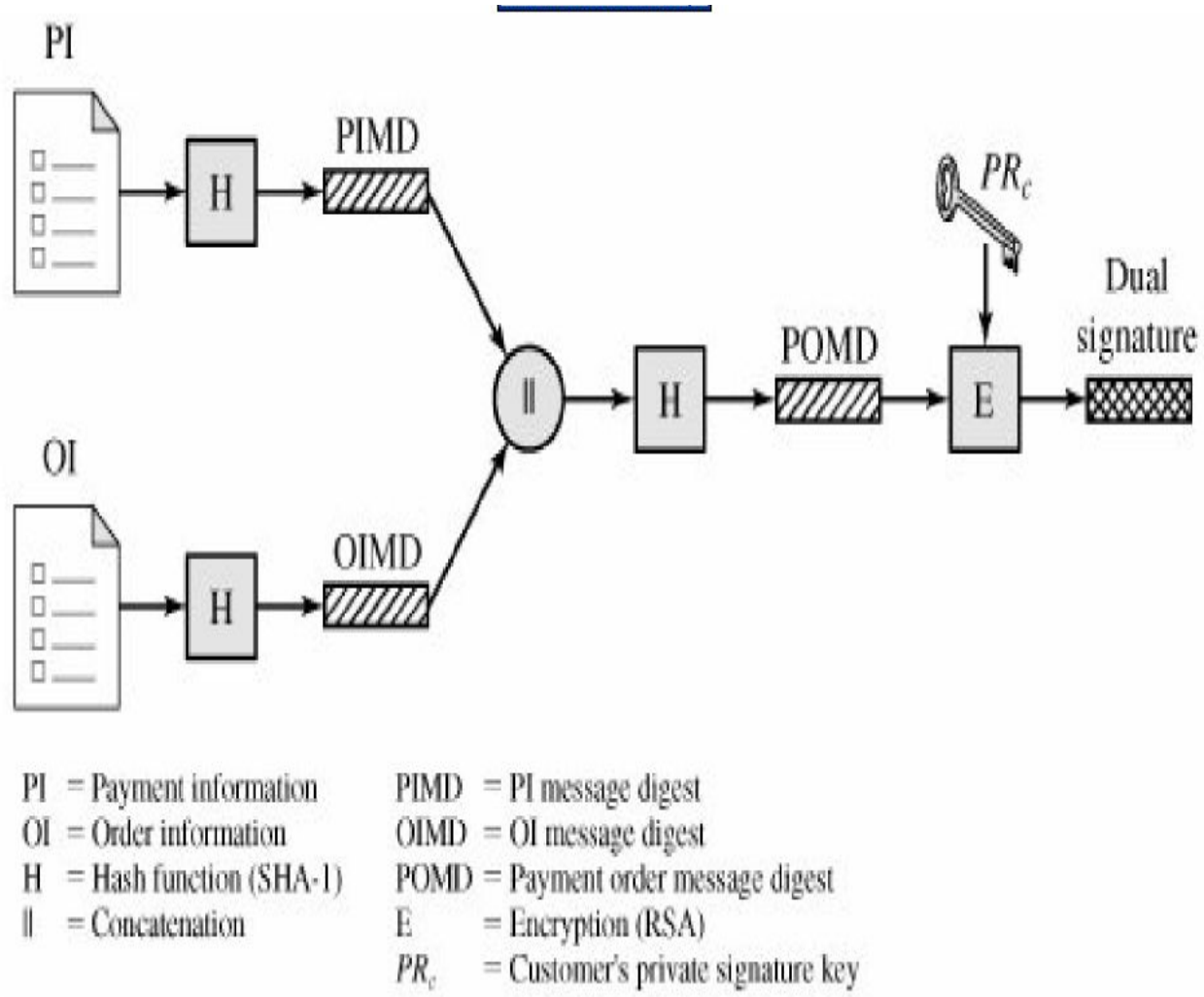
Secure Electronic Transactions(SET) Participants

Sequence of events required for transaction:

- **The customer receives a certificate.**
- **Merchants have their own certificates.**
- **The customer places an order.**
- **The merchant is verified.**
- **The order and payment are sent.**
- **The merchant requests payment authorization**
- **The merchant confirms the order**.
- **The merchant provides the goods or service.**
- **The merchant requests payment.**

Secure Electronic Transactions(SET) –Dual Signature

SET uses Dual Signature.The purpose of the dual signature is to link two messages that are intended for two different recipients. In this case, the customer wants to send the order information (OI) to the merchant and the payment information (PI) to the bank. The merchant does not need to know the customer's credit card number, and the bank does not need to know the details of the customer's order.

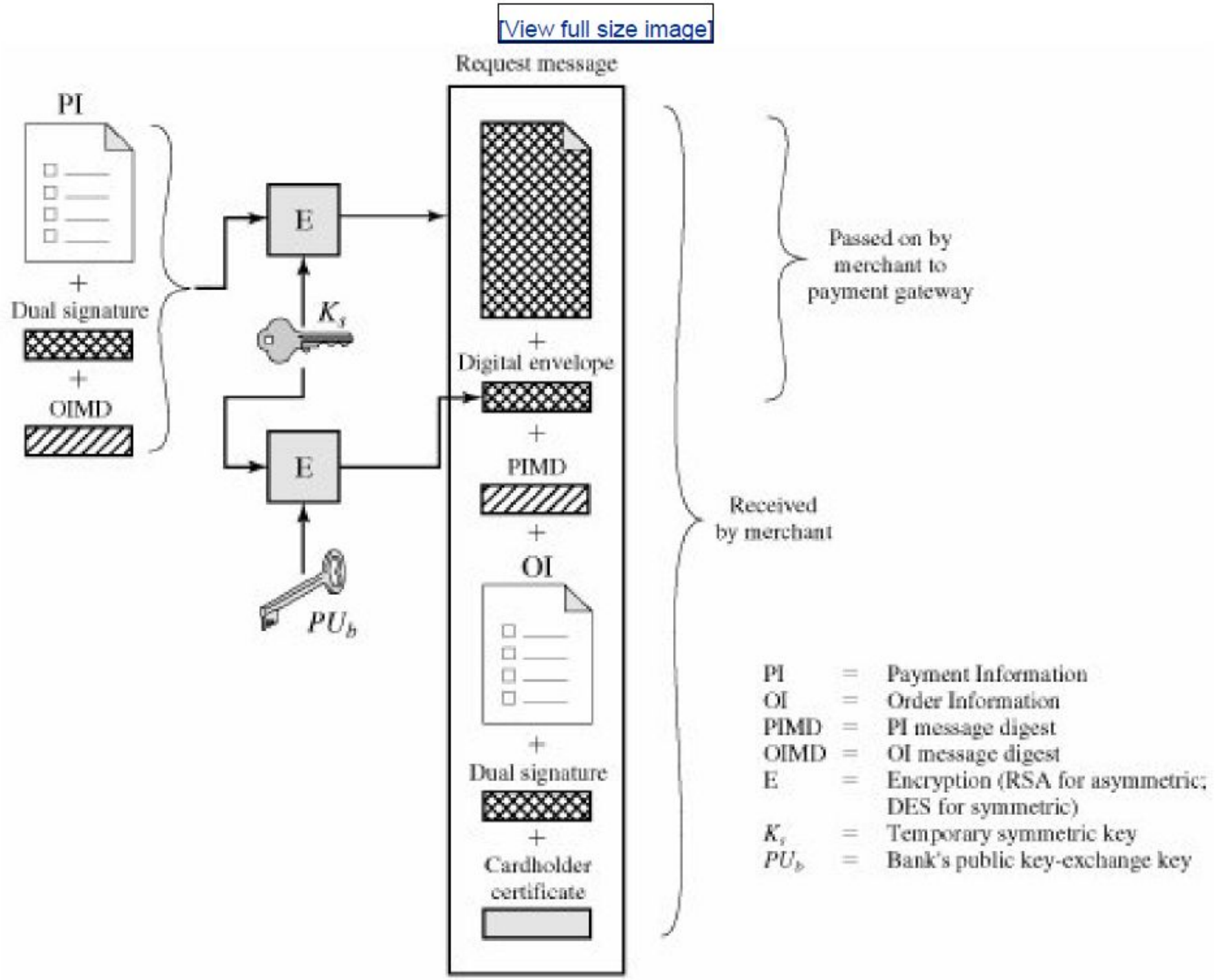


SET Transaction Types:

Cardholder registration	Cardholders must register with a CA before they can send SET messages to merchants.
Merchant registration	Merchants must register with a CA before they can exchange SET messages with customers and payment gateways.
Purchase request	Message from customer to merchant containing OI for merchant and PI for bank.
Payment authorization	Exchange between merchant and payment gateway to authorize a given amount for a purchase on a given credit card account.
Payment capture	Allows the merchant to request payment from the payment gateway.
Certificate inquiry and status	If the CA is unable to complete the processing of a certificate request quickly, it will send a reply to the cardholder or merchant indicating that the requester should check back later. The cardholder or merchant sends the *Certificate Inquiry* message to determine the status of the certificate request and to receive the certificate if the request has been approved.
Purchase inquiry	Allows the cardholder to check the status of the processing of an order after the purchase response has been received. Note that this message does not include information such as the status of back ordered goods, but does indicate the status of authorization, capture and credit processing.
Authorization reversal	Allows a merchant to correct previous authorization requests. If the order will not be completed, the merchant reverses the entire authorization. If part of the order will not be completed (such as when goods are back ordered), the merchant reverses part of the amount of the authorization.
Capture reversal	Allows a merchant to correct errors in capture requests such as transaction amounts that were entered incorrectly by a clerk.
Credit	Allows a merchant to issue a credit to a cardholder's account such as when goods are returned or were damaged during shipping. Note that the SET *Credit* message is always initiated by the merchant, not the cardholder. All communications between the cardholder and merchant that result in a credit being processed happen outside of SET.

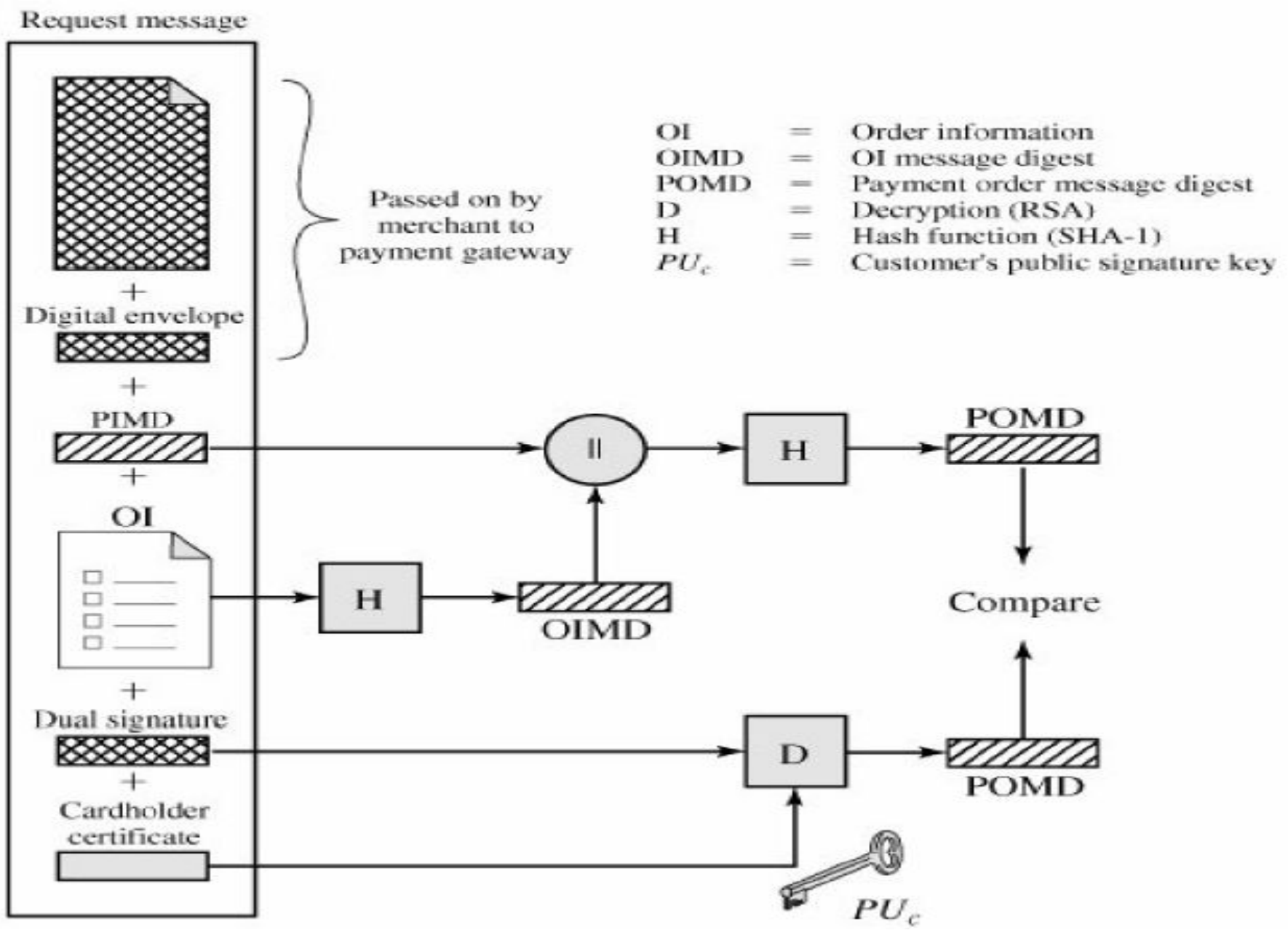
SET Payment Processing:

Card holder sends the purchase request shown in this figure.



SET Payment Processing:

Merchant verifies the Customer Purchase Request shown in this figure.



Intruders

Intruders:-

There are 3 classes of intruders

Masquerader: An individual who is not authorized to use the computer and who penetrates a system's access controls to exploit a legitimate user's account.

Misfeasor: A legitimate user who accesses data, programs, or resources for which such access is not authorized, or who is authorized for such access but misuses his or her privileges.

Clandestine user: An individual who seizes supervisory control of the system and uses this control to evade auditing and access controls or to suppress audit collection.

Intruders

Intrusion Techniques:

The objective of the intruder is to gain access to a system or to increase the range of privileges accessible on a system. The attacker does this by accessing the user passwords.

However User passwords can be protected in 2 ways,

One-way function: The system stores only the value of a function based on the user's password. When the user presents a password, the system transforms that password and compares it with the stored value.

Access control: Access to the password file is limited to one or a very few accounts.

Intruders

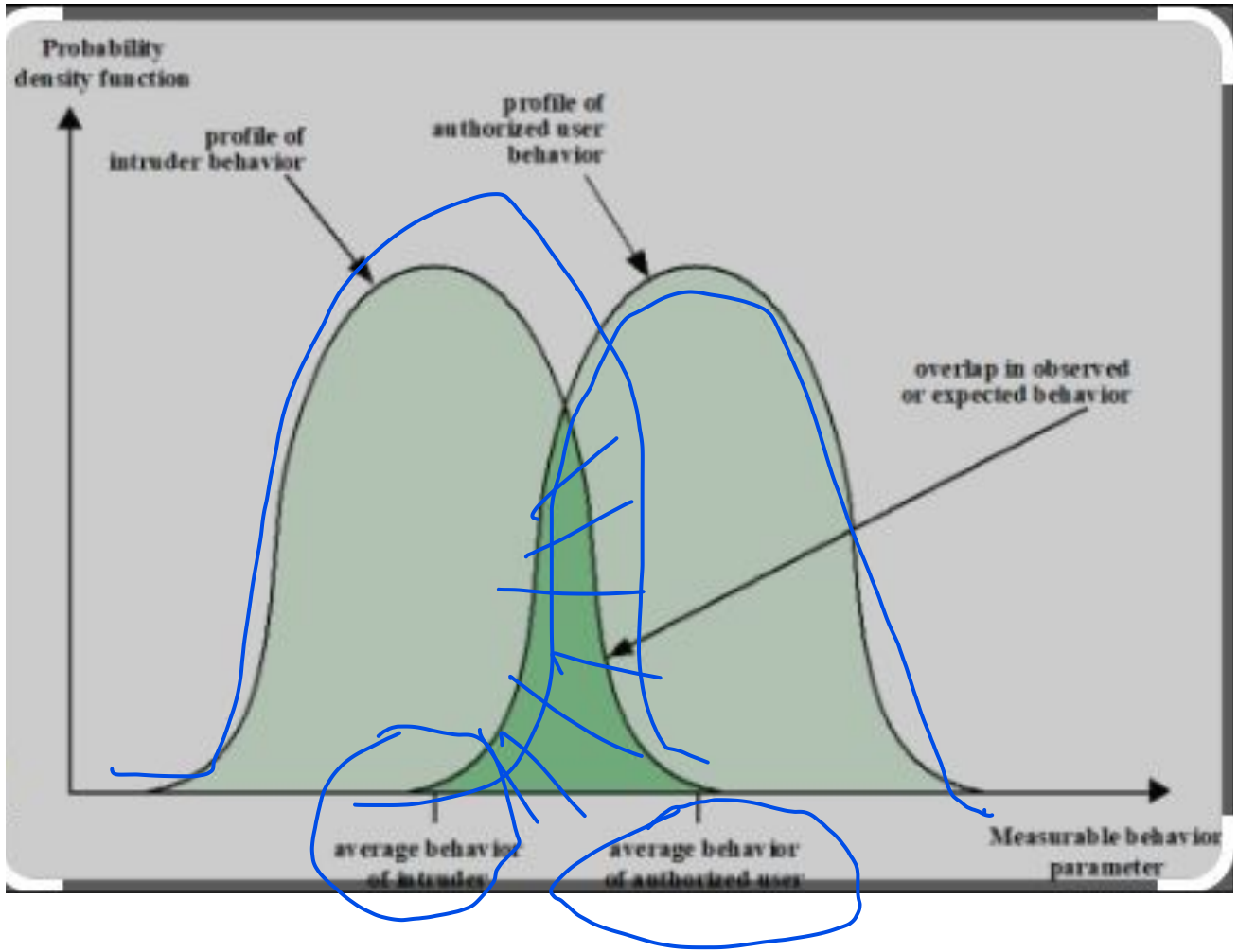
However attacker uses the following techniques to learn the passwords.

- Try default passwords used with standard accounts that are shipped with the system. Many administrators do not bother to change these defaults.
- Exhaustively try all short passwords (those of one to three characters). Try words in the system's online dictionary or a list of likely passwords. Examples of the latter are readily available on hacker bulletin boards.
- Collect information about users, such as their full names, the names of their spouse and children, pictures in their office, and books in their office that are related to hobbies.
- Try users' phone numbers, Social Security numbers, and room numbers.
- Try all legitimate license plate numbers for this state.
- Use a Trojan horse to bypass restrictions on access.
- Tap the line between a remote user and the host system.

Intrusion Detection

Profiled Behaviors of Intruders and Authorised Users:

Although the typical behavior of an intruder differs from the typical behavior of an authorized user, there is an overlap in these behaviors. Thus, a loose interpretation of intruder behavior, which will catch more intruders, will also lead to a number of "false positives," or authorized users identified as intruders



Intrusion Detection Techniques

1. Statistical anomaly detection: Involves the collection of data relating to the behavior of legitimate users over a period of time. Then statistical tests are applied to observed behavior to determine with a high level of confidence whether that behavior is not legitimate user behaviour.

a. **Threshold detection**: This approach involves defining thresholds, independent of user, for the frequency of occurrence of various events.

b. **Profile based**: A profile of the activity of each user is developed and used to detect changes in the behavior of individual accounts.

2. Rule-based detection: Involves an attempt to define a set of rules that can be used to decide that a given behavior is that of an intruder.

a. Anomaly detection: Rules are developed to detect deviation from previous usage patterns.
b. Penetration identification: An expert system approach that searches for suspicious behavior.

Intrusion Detection —Audit Records

A fundamental tool for intrusion detection is the audit record

- **Native audit records:** Virtually all multiuser operating systems include accounting software that collects information on user activity. The advantage of using this information is that no additional collection software is needed. The disadvantage is that the native audit records may not contain the needed information or may not contain it in a convenient form.

- **Detection-specific audit records:** A collection facility can be implemented that generates audit records containing only that information required by the intrusion detection system. One advantage of such an approach is that it could be made vendor independent and ported to a variety of systems. The disadvantage is the extra overhead involved in having, in effect, two accounting packages running on a machine

Intrusion Detection —Audit Records

Audit Records contains following fields:

- **Subject:** Initiators of actions. A subject is typically a terminal user but might also be a process acting on behalf of users or groups of users.
- A**ction:** Operation performed by the subject on or with an object; for example, login, read, perform I/O, execute.
- **Object:** Receptors of actions. Examples include files, programs, messages, records, terminals, printers, and user- or program-created structures.
- **Exception-Condition:** Denotes which, if any, exception condition is raised on return.
- **Resource-Usage:** A list of quantitative elements in which each element gives the amount used of some resource
- **Time-Stamp:** Unique time-and-date stamp identifying when the action took place.

Intrusion Detection —Statistical Anomaly Detection

Statistical anomaly detection: Involves the collection of data relating to the behavior of legitimate users over a period of time. Then statistical tests are applied to observed behavior to determine with a high level of confidence whether that behavior is not legitimate user behaviour.

a. **Threshold detection**:

- This approach involves defining thresholds, independent of user, for the frequency of occurrence of various events.
- Threshold detection involves counting the number of occurrences of a specific event type over an interval of time.
- If the count surpasses what is considered a reasonable number that one might expect to occur, then intrusion is assumed.

b. **Profile based**: A profile of the activity of each user is developed and used to detect changes in the behavior of individual accounts.

Intrusion Detection —Statistical Anomaly Detection

b. **Profile based**:

- A profile of the activity of each user is developed and used to detect changes in the behavior of individual accounts.
- Profile-based anomaly detection focuses on characterizing the past behavior of individual users or related groups of users and then detecting significant deviations.
- A profile may consist of a set of parameters, so that deviation on just a single parameter may not be sufficient in itself to signal an alert.
- Metrics useful for Profile based Detection:
  - **Counter**, **Gauge**, **Interval Timer** , **Resource Utilization**

Based on the above metrics various tests such as

- **Mean and Standard Deviation, Multivariate, Markov Process, Time Series and Operational**

Statistical Anomaly Detection

Various Statistical Tests:

- **mean and standard deviation** of a parameter over some historical period. This gives a reflection of the average behavior and its variability.
- A **multivariate** model is based on correlations between two or more variables. Intruder behavior may be characterized with greater confidence by considering such correlations.
- A **Markov process** model is used to establish transition probabilities among various states. As an example, this model might be used to look at transitions between certain commands.
- A **time series** model focuses on time intervals, looking for sequences of events that happen too rapidly or too slowly. A variety of statistical tests can be applied to characterize abnormal timing.
- **operational model** is based on a judgment of what is considered abnormal, rather than an automated analysis of past audit records. Typically, fixed limits are defined and intrusion is suspected for an observation that is outside the limits

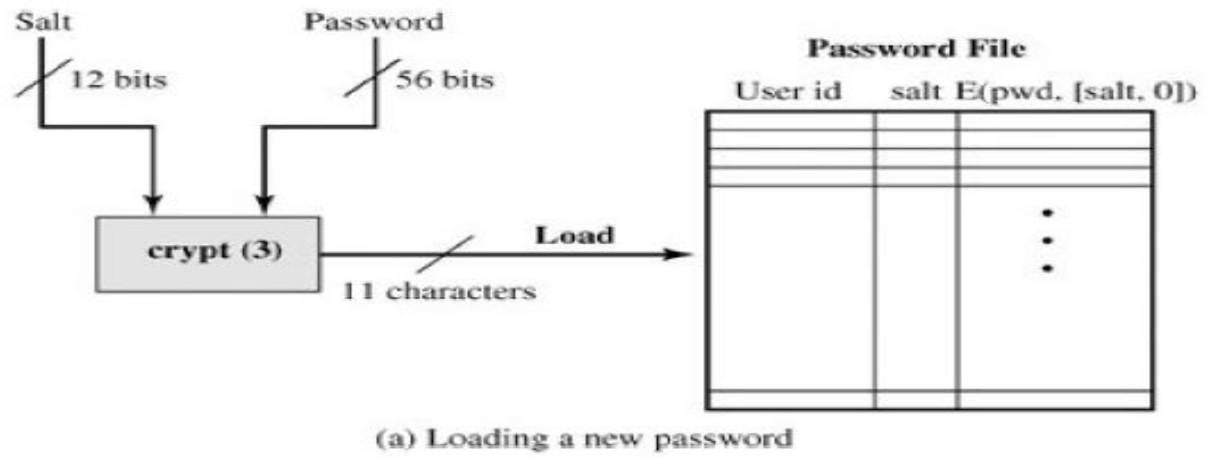
Password Management

Password protection:

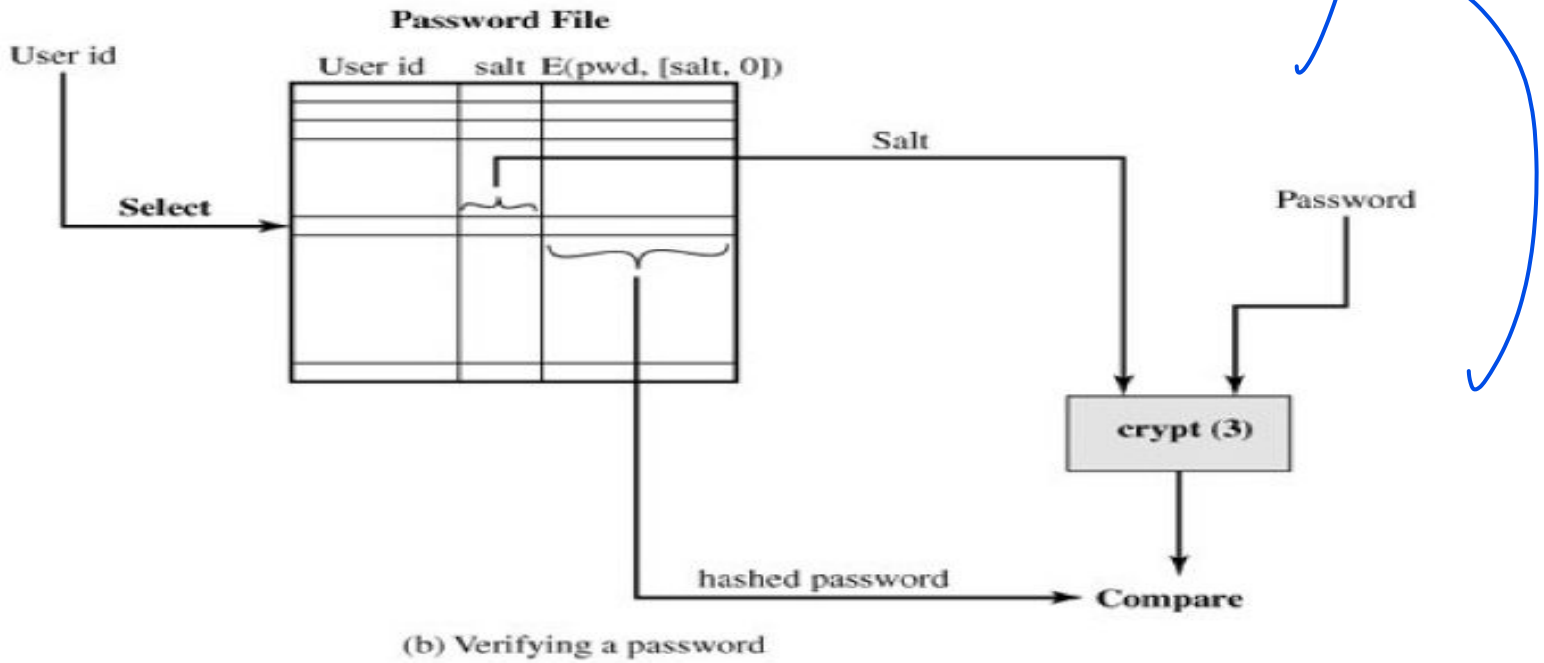
All the multiuser systems not only require user ID also needs strong password. The password servers to authenticate user ID. The user ID provides security in the following ways.

- The ID determines whether the user is authorized to gain access to a system. In some systems, only those who already have an ID filed on the system are allowed to gain access.
- The ID determines the privileges accorded to the user. A few users may have supervisory or "superuser" status that enables them to read files and perform functions that are especially protected by the operating system. Some systems have guest or anonymous accounts, and users of these accounts have more limited privileges than others.
- The ID is used in what is referred to as discretionary access control. For example, by listing the IDs of the other users, a user may grant permission to them to read files owned by that user.

Vulnerability of Passwords



(a) Loading a new password



(b) Verifying a password

Password Protection

- The Salt servers 3 purposes:

- It prevents duplicate passwords from being visible in the password file. Even if two users choose the same password, those passwords will be assigned at different times. Hence, the "extended" passwords of the two users will differ.

- It effectively increases the length of the password without requiring the user to remember two additional characters. Hence, the number of possible passwords is increased by a factor of 4096, increasing the difficulty of guessing a password.

- It prevents the use of a hardware implementation of DES, which would ease the difficulty of a brute-force guessing attack.

Password Protection

Access Control:

- One way to thwart a password attack is to deny the opponent access to the password file.
- If the encrypted password portion of the file is accessible only by a privileged user, then the opponent cannot read it without already knowing the password of a privileged user.

However several flaws in this strategy:

- Many systems, including most UNIX systems, are susceptible to unanticipated break-ins. Once an attacker has gained access by some means, he or she may wish to obtain a collection of passwords in order to use different accounts for different logon sessions to decrease the risk of detection. Or a user with an account may desire another user's account to access privileged data or to sabotage the system.
- An accident of protection might render the password file readable, thus compromising all the accounts.
- Some of the users have accounts on other machines in other protection domains, and they use the same password. Thus, if the passwords could be read by anyone on one machine, a machine in another location might be compromised

Password Selection Strategies

Four Techniques are used:

user education strategy is unlikely to succeed at most installations, particularly where there is a large user population or a lot of turnover.

Computer-generated passwords also have problems. If the passwords are quite random in nature, users will not be able to remember them. Even if the password is pronounceable, the user may have difficulty remembering it and so be tempted to write it down.

A **reactive password checking** strategy is one in which the system periodically runs its own password cracker to find guessable passwords. The system cancels any passwords that are guessed and notifies the user. This tactic has a number of drawbacks. First, it is resource intensive if the job is done right. Furthermore, any existing passwords remain vulnerable until the reactive password checker finds them.

A **proactive password checker**. In this scheme, a user is allowed to select his or her own password. However, at the time of selection, the system checks to see if the password is allowable and, if not, rejects it.

Malicious Software

Terminologies of Malicious Programs:

Virus: Attaches itself to a program and propagates copies of itself to other programs.

Worm: Program that propagates copies of itself to other computers.

Logical bomb: Triggers action when condition occurs. The logic bomb is code embedded in some legitimate program that is set to "explode" when certain conditions are met. Examples of conditions that can be used as triggers for a logic bomb are the presence or absence of certain files, a particular day of the week or date, or a particular user running the application.

Backdoor: A backdoor, also known as a trapdoor, is a secret entry point into a program that allows someone that is aware of the backdoor to gain access without going through the usual security access procedures.

Trojan Horses: Trojan horse programs can be used to accomplish functions indirectly that an unauthorized user could not accomplish directly. For example, to gain access to the files of another user on a shared system, a user could create a Trojan horse program that, when executed, changed the invoking user's file permissions so that the files are readable by any user.

Zombie: A zombie is a program that secretly takes over another Internet-attached computer and then uses that computer to launch attacks that are difficult to trace to the zombie's creator. Zombies are used in denial-of-service attacks, typically against targeted Web sites.

Malicious Software– Nature of Viruses

- A virus is a piece of software that can "infect" other programs by modifying them; the modification includes a copy of the virus program, which can then go on to infect other programs.
- A virus can do anything that other programs do. The only difference is that it attaches itself to another program and executes secretly when the host program is run. Once a virus is executing, it can perform any function, such as erasing files and programs.

During its lifetime, a typical virus goes through the following four phases:

Dormant phase: The virus is idle. The virus will eventually be activated by some event, such as a date, the presence of another program or file, or the capacity of the disk exceeding some limit. Not all viruses have this stage.

Propagation phase: The virus places an identical copy of itself into other programs or into certain system areas on the disk.

Triggering phase: The virus is activated to perform the function for which it was intended. As with the dormant phase, the triggering phase can be caused by a variety of system events, including a count of the number of times that this copy of the virus has made copies of itself.

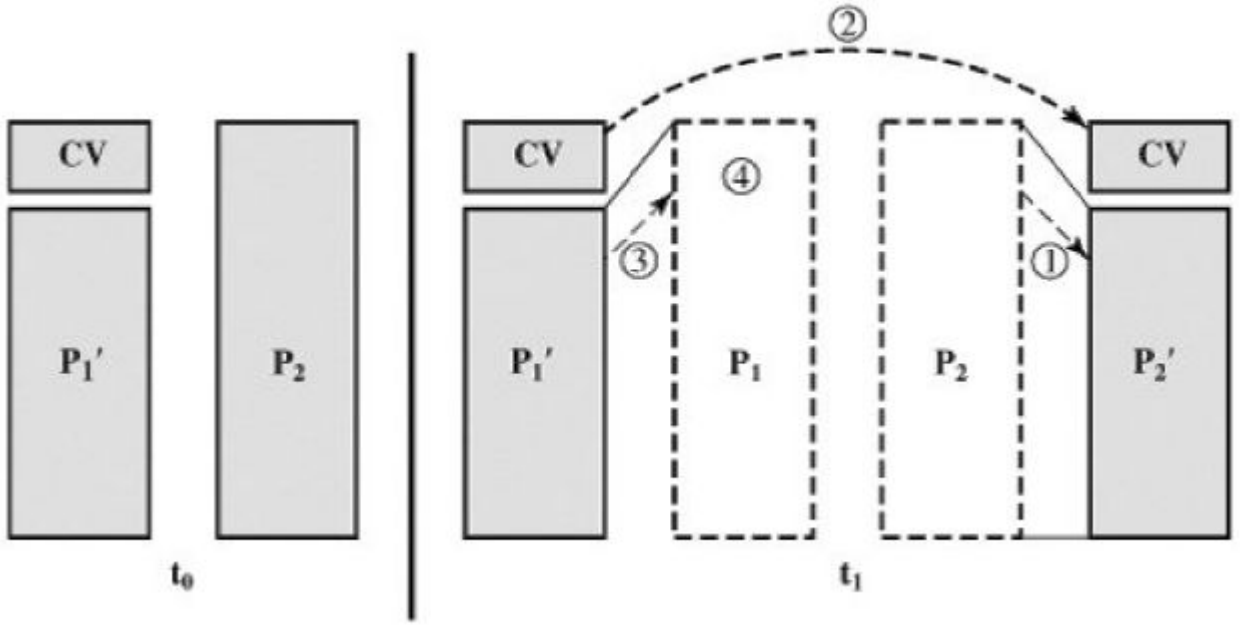
Execution phase: The function is performed. The function may be harmless, such as a message on the screen, or damaging, such as the destruction of programs and data files.

Malicious Software– Virus Structure

```
      program V :=

{goto main;
      1234567;

      subroutine infect-executable :=
            {loop:
            file := get-random-executable-file;
            if (first-line-of-file = 1234567)
               then goto loop
               else prepend V to file; }

      subroutine do-damage :=
            {whatever damage is to be done}

      subroutine trigger-pulled :=
            {return true if some condition holds}

main:     main-program :=
            {infect-executable;
            if trigger-pulled then do-damage;
            goto next;}
next:

}
```

Fig: A Simple Virus

Figure 19.3. A Compression Virus



Types of Viruses

Parasitic virus: The traditional and still most common form of virus. A parasitic virus attaches itself to executable files and replicates, when the infected program is executed, by finding other executable files to infect.

Memory-resident virus: Lodges in main memory as part of a resident system program. From that point on, the virus infects every program that executes.

Boot sector virus: Infects a master boot record or boot record and spreads when a system is booted from the disk containing the virus.

Stealth virus: A form of virus explicitly designed to hide itself from detection by antivirus software.

Polymorphic virus: A virus that mutates with every infection, making detection by the "signature" of the virus impossible.

Metamorphic virus: As with a polymorphic virus, a metamorphic virus mutates with every infection. The difference is that a metamorphic virus rewrites itself completely at each iteration, increasing the difficulty of detection. Metamorphic viruses may change their behavior as well as their appearance.

Malicious Software--Worms

- A worm is a program that can replicate itself and send copies from computer to computer across network connections. Upon arrival, the worm may be activated to replicate and propagate again.

To replicate itself, a network worm uses some sort of network vehicle.

Examples include the following:

Electronic mail facility: A worm mails a copy of itself to other systems.

Remote execution capability: A worm executes a copy of itself on another system.

Remote login capability: A worm logs onto a remote system as a user and then uses commands to copy itself from one system to the other.

Morris Worm : The Morris worm was designed to spread on UNIX systems and used a number of different techniques for propagation. When a copy began execution, its first task was to discover other hosts known to this host that would allow entry from this host. The worm performed this task by examining a variety of lists and tables, including system tables that declared which other machines were trusted by this host, users' mail forwarding files:

Virus Countermeasures

The ideal solution to the threat of viruses is prevention: Do not allow a virus to get into the system in the first place. This goal is, in general, impossible to achieve, although prevention can reduce the number of successful viral attacks. The next best approach is to be able to do the following:

Detection: Once the infection has occurred, determine that it has occurred and locate the virus.

Identification: Once detection has been achieved, identify the specific virus that has infected a program.

Removal: Once the specific virus has been identified, remove all traces of the virus from the infected program and restore it to its original state.

Virus Countermeasures

There are 4 generation of AntiVirus Software:

- A **first-generation** scanner requires a virus signature to identify a virus. The virus may contain "wildcards" but has essentially the same structure and bit pattern in all copies. Such signature-specific scanners are limited to the detection of known viruses.
- A **second-generation** scanner does not rely on a specific signature. Rather, the scanner uses heuristic rules to search for probable virus infection. One class of such scanners looks for fragments of code that are often associated with viruses.
- **Third-generation** programs are memory-resident programs that identify a virus by its actions rather than its structure in an infected program.
- **Fourth-generation** products are packages consisting of a variety of antivirus techniques used in conjunction. These include scanning and activity trap components. In addition, such a package includes access control capability, which limits the ability of viruses to penetrate a system and then limits the ability of a virus to update files in order to pass on the infection.

Generic Decryption

Generic decryption (GD) technology enables the antivirus program to easily detect even the most complex polymorphic viruses, while maintaining fast scanning speeds.

In order to detect such a structure, executable files are run through a GD scanner, which contains the following elements:

CPU emulator: A software-based virtual computer. Instructions in an executable file are interpreted by the emulator rather than executed on the underlying processor. The emulator includes software versions of all registers and other processor hardware, so that the underlying processor is unaffected by programs interpreted on the emulator.

Virus signature scanner: A module that scans the target code looking for known virus signatures.

Emulation control module: Controls the execution of the target code.

Digital Immune:

There are 2 major trends have played very important role in rate of virus propagation:

Integrated mail systems: Systems such as Lotus Notes and Microsoft Outlook make it very simple to send anything to anyone and to work with objects that are received.

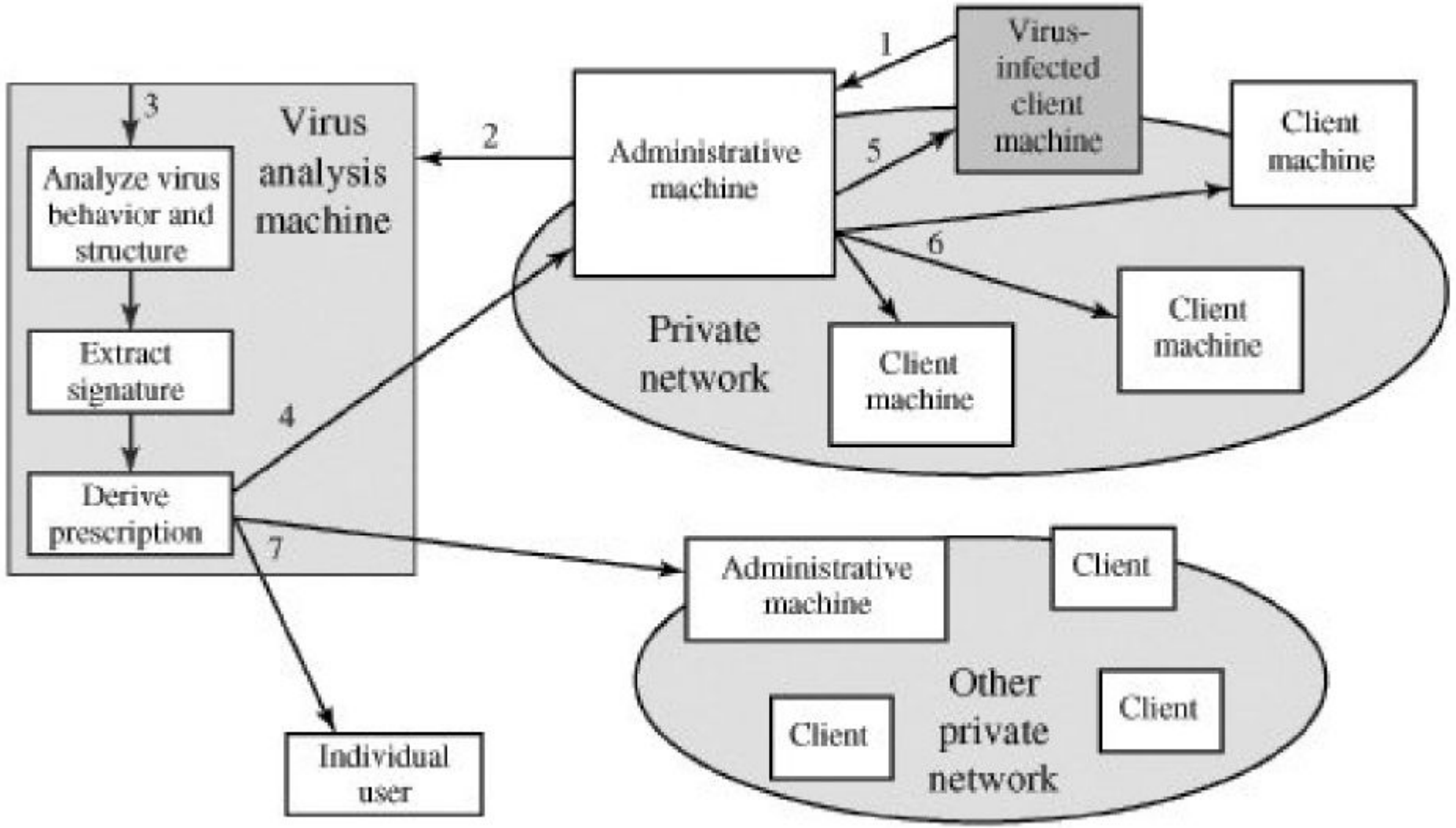
Mobile-program systems: Capabilities such as Java and ActiveX allow programs to move on their own from one system to another.

In response to the threat posed by these Internet-based capabilities, IBM has developed a prototype **digital immune system**.

When a new virus enters an organization, the immune system automatically captures it, analyzes it, adds detection and shielding for it, removes it, and passes information about that virus to systems running IBM AntiVirus so that it can be detected before it is allowed to run elsewhere.

Advanced Antivirus Techniques

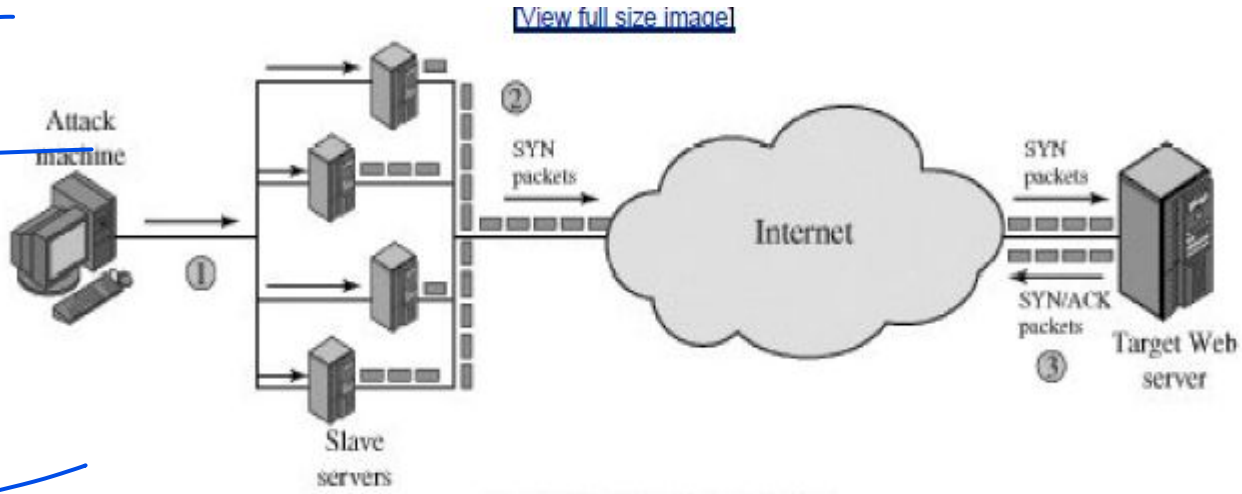
Fig: Digital Immune



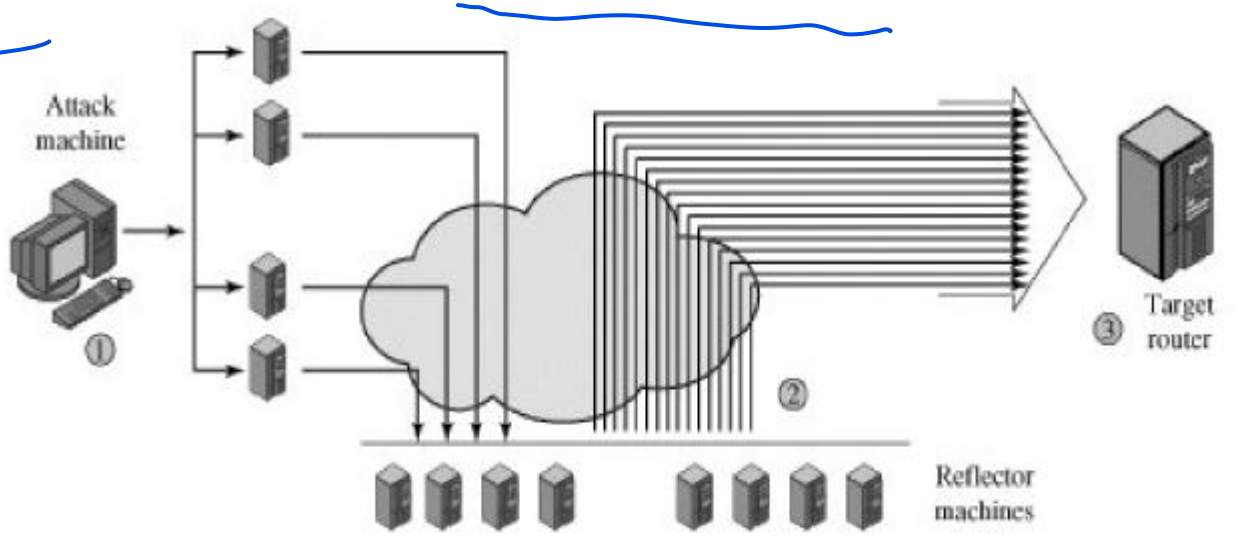
DDOS(Denial of Service Attack)

A denial of service (DoS) attack is an attempt to prevent legitimate users of a service from using that service. When this attack comes from a single host or network node, then it is simply referred to as a DoS attack.

A more serious threat is posed by a DDoS attack. In a DDoS attack, an attacker is able to recruit a number of hosts throughout the Internet to simultaneously or in a coordinated fashion launch an attack upon the target.



(a) Distributed SYN flood attack



(a) Distributed ICMP attack

DDOS(Denial of Service Attack)

SynFlood Attack

1. The attacker takes control of multiple hosts over the Internet, instructing them to contact the target Web server.

2. The slave hosts begin sending TCP/IP SYN (synchronize/initialization) packets, with erroneous return IP address information, to the target.

3. Each SYN packet is a request to open a TCP connection. For each such packet, the Web server responds with a SYN/ACK (synchronize/acknowledge) packet, trying to establish a TCP connection with a TCP entity at a spurious IP address.

ICMP Flood Attack

1. The attacker takes control of multiple hosts over the Internet, instructing them to send ICMP ECHO packets with the target's spoofed IP address to a group of hosts that act as reflectors, as described subsequently.

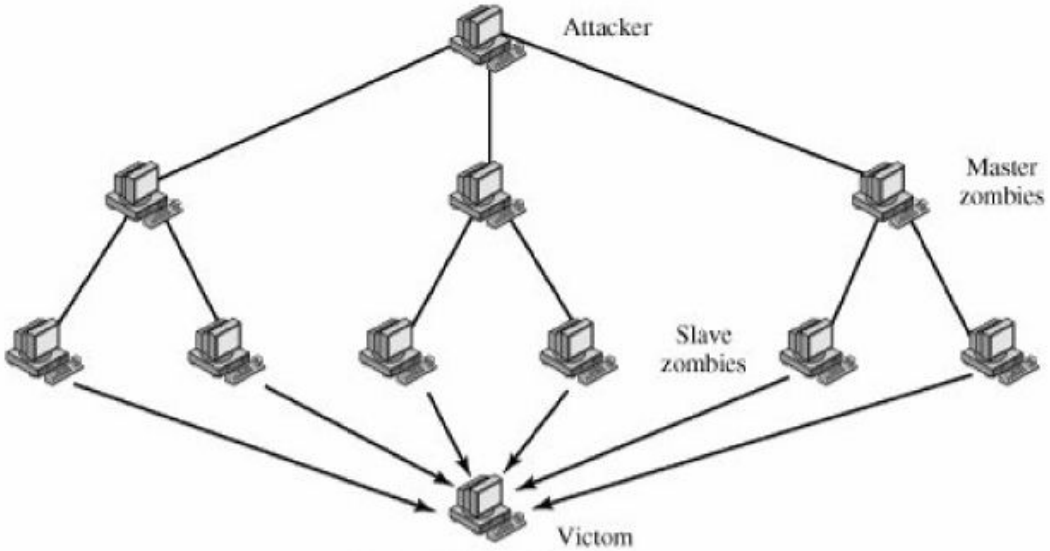
2. Nodes at the bounce site receive multiple spoofed requests and respond by sending echo reply packets to the target site.

3. The target's router is flooded with packets from the bounce site, leaving no data transmission capacity for legitimate traffic.

Types of DDOS(Denial of Service Attack)

Direct DDOS Attack

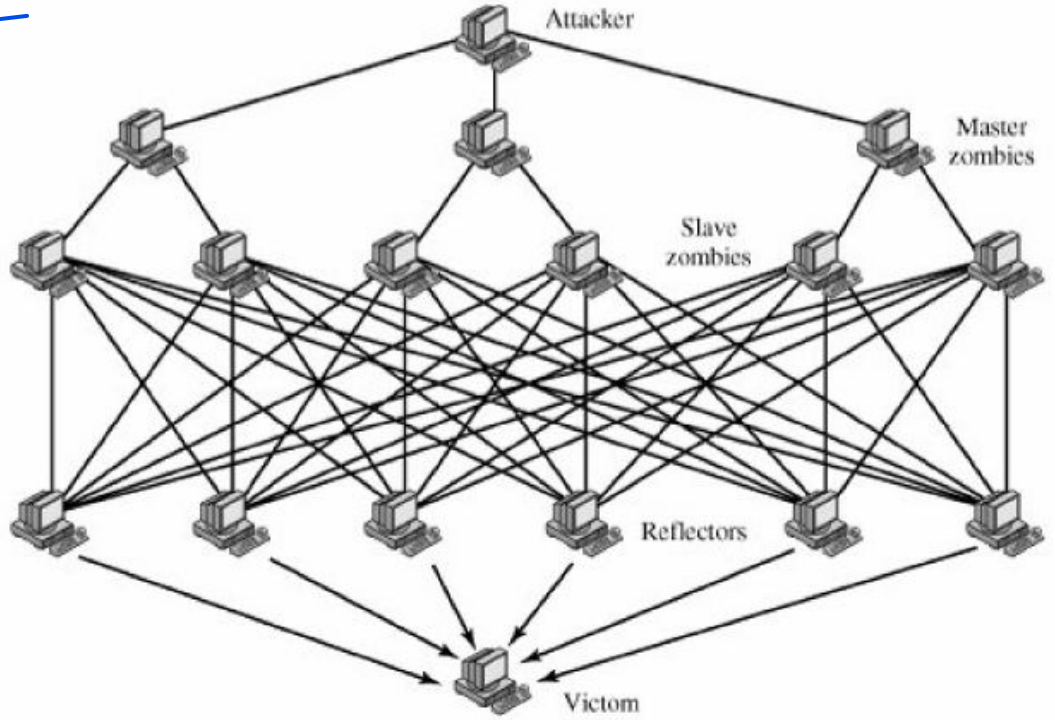
Here the attacker is able to implant zombie software on a number of sites distributed throughout the Internet. Often, the DDoS attack involves two levels of zombie machines: master zombies and slave zombies.



(a) Direct DDoS Attack

Reflector DDOS Attack

In this type of attack, the slave zombies construct packets requiring a response that contain the target's IP address as the source IP address in the packet's IP header. These packets are sent to uninfected machines known as reflectors. The uninfected machines respond with packets directed at the target machine. A reflector DDoS attack can easily involve more machines and more traffic than a direct DDoS attack and hence be more damaging.



(b) Reflector DDoS Attack

Firewalls

A firewall is a network security device, either hardware or software-based, which monitors all incoming and outgoing traffic and based on a defined set of security rules accepts, rejects, or drops that specific traffic.

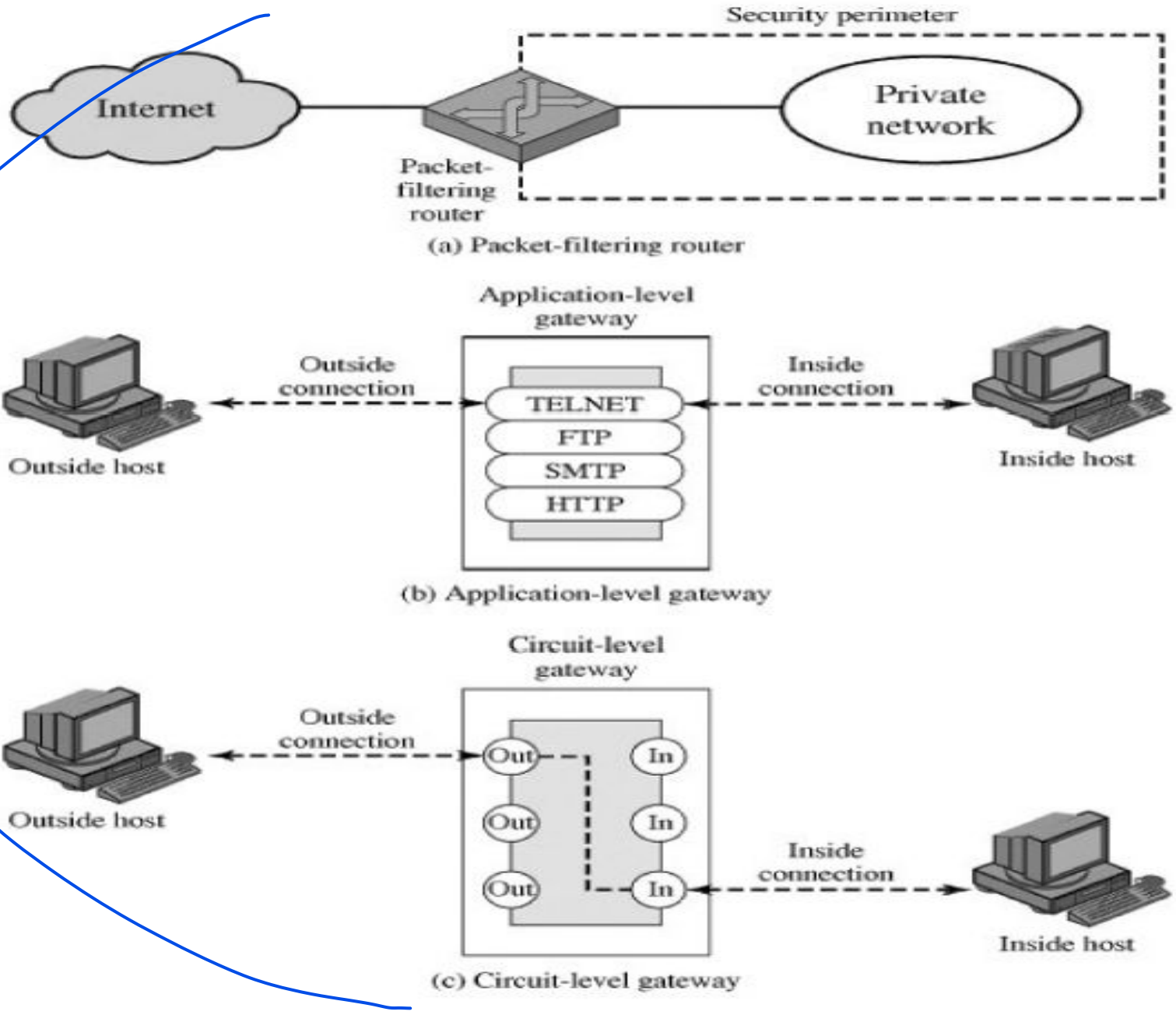
- **Accept:** allow the traffic
- **Reject:** block the traffic but reply with an “unreachable error”
- **Drop:** block the traffic with no reply

A firewall is a type of network security device that filters incoming and outgoing network traffic with security policies that have previously been set up inside an organization.

Firewall Characteristics

- All traffic from inside to outside, and vice versa, must pass through the firewall. This is achieved by physically blocking all access to the local network except via the firewall.
- Only authorized traffic, as defined by the local security policy, will be allowed to pass. Various types of firewalls are used, which implement various types of security policies.

Types of Firewalls



Types of Firewalls

A packet-filtering router applies a set of rules to each incoming and outgoing IP packet and then forwards or discards the packet. The router is typically configured to filter packets going in both directions (from and to the internal network). Filtering rules are based on information contained in a network packet:

Source IP address: The IP address of the system that originated the IP packet (e.g., 192.178.1.1)

Destination IP address: The IP address of the system the IP packet is trying to reach (e.g., 192.168.1.2)

Source and destination transport-level address: The transport level (e.g., TCP or UDP) port number, which defines applications such as SNMP or TELNET.

IP protocol field: Defines the transport protocol.

Interface: For a router with three or more ports, which interface of the router the packet came from or which interface of the router the packet is destined for.

Two default policies are possible:

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

Types of Firewalls

Table 20.1. Packet-Filtering Examples

(This item is displayed on page 627 in the print version)

	action	ourhost	port	theirhost	port	comment
A	block	*	*	SPIGOT	*	we don't trust these people
	allow	OUR-GW	25	*	*	connection to our SMTP port

	action	ourhost	port	theirhost	port	comment
B	block	*	*	*	*	default

	action	ourhost	port	theirhost	port	comment
C	allow	*	*	*	25	connection to their SMTP port

	action	src	port	dest	port	flags	comment
D	allow	{our hosts}	*	*	25		our packets to their SMTP port
	allow	*	25	*	*	ACK	their replies

	action	src	port	dest	port	flags	comment
E	allow	{our hosts}	*	*	*		our outgoing calls
	allow	*	*	*	*	ACK	replies to our calls
	allow	*	*	*	>1024		traffic to nonservers

Packet Filtering Weaknesses

- Because packet filter firewalls do not examine upper-layer data, they cannot prevent attacks that employ application-specific vulnerabilities or functions. For example, a packet filter firewall cannot block specific application commands; if a packet filter firewall allows a given application, all functions available within that application will be permitted.

- Because of the limited information available to the firewall, the logging functionality present in packet filter firewalls is limited. Packet filter logs normally contain the same information used to make access control decisions (source address, destination address, and traffic type).

- Most packet filter firewalls do not support advanced user authentication schemes. Once again, this limitation is mostly due to the lack of upper-layer functionality by the firewall.

- They are generally vulnerable to attacks and exploits that take advantage of problems within the TCP/IP specification and protocol stack, such as *network layer address spoofing*. Many packet filter firewalls cannot detect a network packet in which the OSI Layer 3 addressing information has been altered. Spoofing attacks are generally employed by intruders to bypass the security controls implemented in a firewall platform.

Application Level Gateway firewall

An application-level gateway, also called a proxy server, acts as a relay of application-level .The user contacts the gateway using a TCP/IP application, such as Telnet or FTP, and the gateway asks the user for the name of the remote host to be accessed. When the user responds and provides a valid user ID and authentication information, the gateway contacts the application on the remote host and relays TCP segments containing the application data between the two endpoints. If the gateway does not implement the proxy code for a specific application, the service is not supported and cannot be forwarded across the firewall.

Application-level gateways tend to be more secure than packet filters, the application-level gateway need only scrutinize a few allowable applications. In addition, it is easy to log and audit all incoming traffic at the application level.

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

Circuit Level Gateway Protocol

A circuit-level gateway does not permit an end-to-end TCP connection; rather, the gateway sets up two TCP connections, one between itself and a TCP user on an inner host and one between itself and a TCP user on an outside host. Once the two connections are established, the gateway typically relays TCP segments from one connection to the other without examining the contents. The security function consists of determining which connections will be allowed.

An example of a circuit-level gateway implementation is the SOCKS package ,version 5 of SOCKS is defined in RFC 1928. The RFC defines SOCKS in the following fashion:

The protocol described here is designed to provide a framework for client-server applications in both the TCP and UDP domains to conveniently and securely use the services of a network firewall.

The protocol is conceptually a "shim-layer" between the application layer and the transport layer, and as such does not provide network-layer gateway services, such as forwarding of ICMP messages.

Circuit Level Gateway Protocol

SOCKS consists of the following components:

The SOCKS server, which runs on a UNIX-based firewall.
The SOCKS client library, which runs on internal hosts protected by the firewall.
SOCKS-ified versions of several standard client programs such as FTP and TELNET. The implementation of the SOCKS protocol typically involves the recompilation or relinking of TCP-based client applications to use the appropriate encapsulation routines in the SOCKS library.

Circuit Level Gateway Protocol

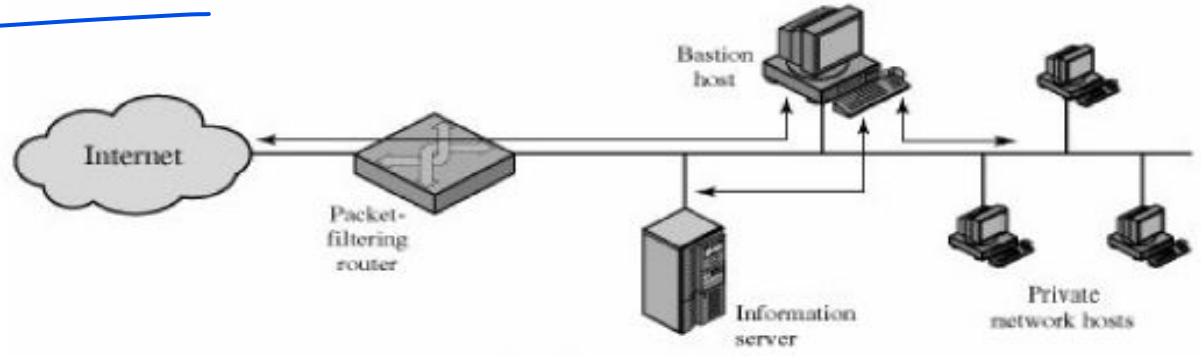
Bastion Host:

A bastion host is a system identified by the firewall administrator as a critical strong point in the network's security. Typically, the bastion host serves as a platform for an application-level or circuit-level gateway.

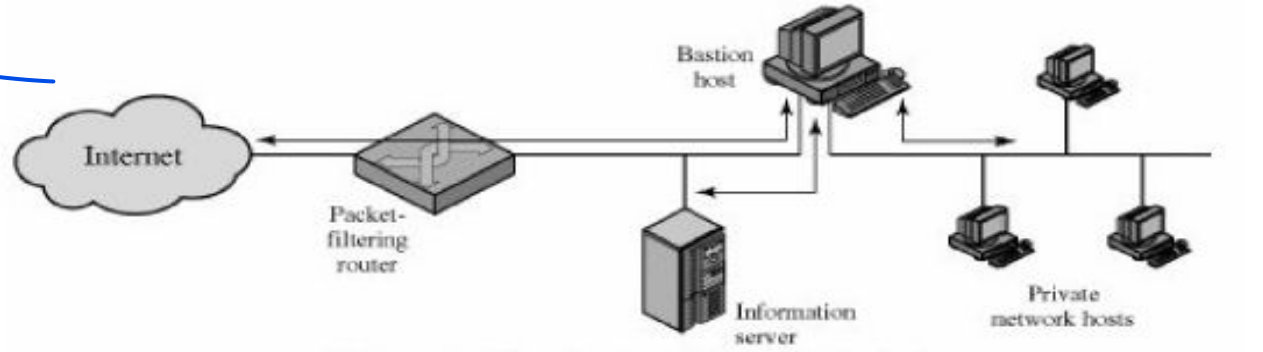
Common characteristics of a bastion host include the following:

- The bastion host hardware platform executes a secure version of its operating system, making it a trusted system.
- Only the services that the network administrator considers essential are installed on the bastion host. These include proxy applications such as Telnet, DNS, FTP, SMTP, and user authentication.
- The bastion host may require additional authentication before a user is allowed access to the proxy services.

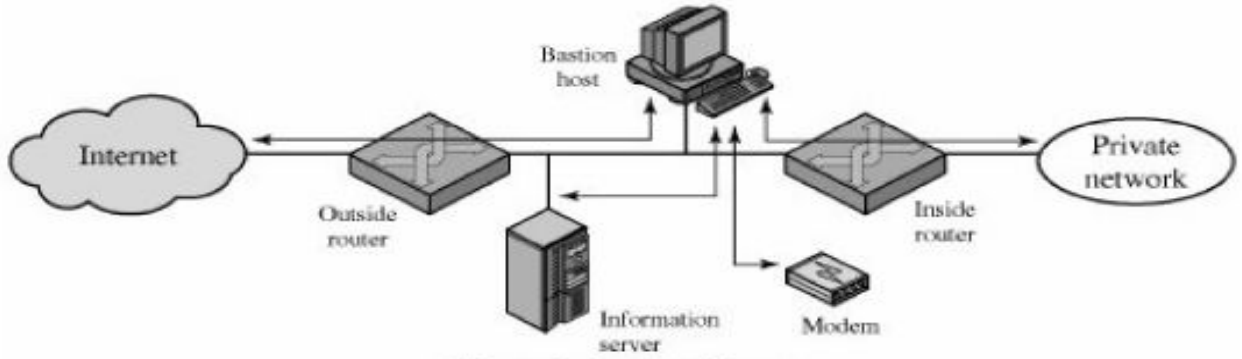
Firewalls Configurations



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



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



(c) Screened-subnet firewall system

Trusted Systems

A trusted system is typically designed with a set of security features, such as access controls, authentication mechanisms, and encryption algorithms, that are carefully integrated to provide a comprehensive security solution. These security features are often implemented using hardware, software, or a combination of both, and are rigorously tested to ensure they meet the security requirements of the system.

Trusted systems are often used in government, military, financial, and other high-security environments where the protection of sensitive information is critical. They are also used in commercial settings where the protection of intellectual property, trade secrets, and other confidential information is important.

Trusted Systems

Trusted systems are designed with a set of security principles and practices that are used to build a system that can be trusted to operate securely. These principles include the following:

1. **Least Privilege:** Trusted systems are designed to provide users with the minimum level of access necessary to perform their tasks. This principle ensures that users cannot accidentally or intentionally access information or resources they are not authorized to use.
2. **Defense in Depth:** Trusted systems implement multiple layers of security controls to protect against threats. This principle involves using a combination of physical, technical, and administrative controls to create a comprehensive security solution.
3. **Integrity:** Trusted systems ensure that data and systems are not modified or altered in an unauthorized manner. This principle ensures that data remains accurate and trustworthy over time.
4. **Confidentiality:** Trusted systems protect sensitive information from unauthorized access. This principle ensures that sensitive data remains private and confidential.
5. **Availability:** Trusted systems ensure that systems and data are available to authorized users when needed. This principle ensures that critical information and systems are accessible and operational at all times.

Trusted Systems are built to ensure the following security

1. **Hardware-based security:** Trusted systems often rely on specialized hardware, such as secure processors, to provide a secure environment for critical operations. These hardware-based solutions can provide a high level of security and are often used in environments where security is paramount.
2. **Virtualization:** Virtualization is a technique that is often used in trusted systems to create multiple virtual machines running on a single physical machine. Each virtual machine can be isolated from the others, providing an additional layer of security.
3. **Multi-factor authentication:** Trusted systems often use multi-factor authentication to verify the identity of users. This involves requiring users to provide more than one form of identification, such as a password and a smart card, before granting access.
4. **Encryption:** Trusted systems often use encryption to protect sensitive data. Encryption involves converting data into a coded format that can only be decoded using a specific key.
5. **Auditing:** Trusted systems often use auditing to track and monitor system activity. Auditing can help detect and prevent security breaches by identifying unusual or suspicious behavior.