

## **UNIT-5**

**Web Security:** Web Security Considerations, Secure Socket Layer (SSL) and Transport Layer Security (TLS), Secure Electronic Transaction (SET). **Intruders, Viruses and Firewalls:** Intruders, Intrusion Detection, Password Management, Virus and related threats, Countermeasures, Firewall Design Principles, Types of Firewalls.

**Case Studies on Cryptography and Security:** Secure Inter Branch Transactions, Cross Site Vulnerability, Virtual Elections.

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

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

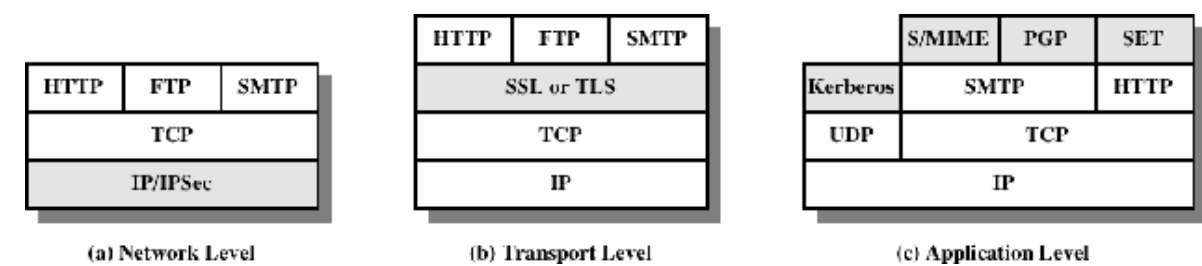
A summary of types of security threats faced in using the Web is given below:

	Threats	Consequences	Countermeasures
Integrity	<ul style="list-style-type: none"> <li>• Modification of user data</li> <li>• Trojan horse browser</li> <li>• Modification of memory</li> <li>• Modification of message traffic in transit</li> </ul>	<ul style="list-style-type: none"> <li>• Loss of information</li> <li>• Compromise of machine</li> <li>• Vulnerability to all other threats</li> </ul>	Cryptographic checksums
Confidentiality	<ul style="list-style-type: none"> <li>• Eavesdropping on the Net</li> <li>• Theft of info from server</li> <li>• Theft of data from client</li> <li>• Info about network configuration</li> <li>• Info about which client talks to server</li> </ul>	<ul style="list-style-type: none"> <li>• Loss of information</li> <li>• Loss of privacy</li> </ul>	Encryption, Web proxies
Denial of Service	<ul style="list-style-type: none"> <li>• Killing of user threads</li> <li>• Flooding machine with bogus threats</li> <li>• Filling up disk or memory</li> <li>• Isolating machine by DNS attacks</li> </ul>	<ul style="list-style-type: none"> <li>• Disruptive</li> <li>• Annoying</li> <li>• Prevent user from getting work done</li> </ul>	Difficult to prevent
Authentication	<ul style="list-style-type: none"> <li>• Impersonation of legitimate users</li> <li>• Data forgery</li> </ul>	<ul style="list-style-type: none"> <li>• Misrepresentation of user</li> <li>• Belief that false information is valid</li> </ul>	Cryptographic techniques

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

### Web Traffic Security Approaches

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



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

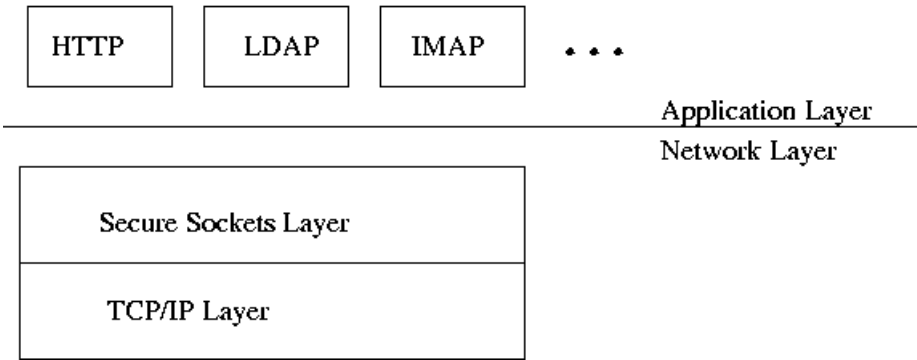
**IPSec** provides security at the network level and the main advantage is that it is transparent to end users and applications. In addition, IPSec includes a filtering capability so that only selected traffic can be processed. **Secure Socket Layer or Transport Layer Security (SSL/TLS)** provides security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

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

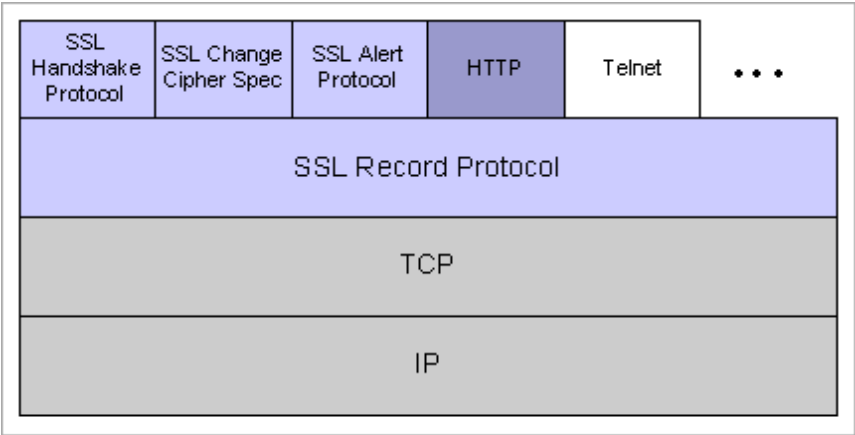
**SECURESOCKETLAYER/TRANSPORTLAYERSECURITY**

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

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



SSL provides for secure communication between client and server by allowing mutualauthentication, the use of digital signatures for integrity and encryption for privacy. SSLprotocolhasdifferentversionssuchasSSLv2.0,SSLv3.0,whereSSLv3.0hasanadvantagewith the addition of support for certificate chain loading. SSL 3.0 is the basis for theTransport Layer Security [TLS] protocol standard. SSL is designed to make use of TCP toprovide a reliable end-to-end secure service. SSL is not a single protocol, but rather twolayersofprotocols asshownbelow:



SSL Protocol Stack

The SSL Record Protocol provides basic security services to various higher-layer protocols. In particular, the Hypertext Transfer Protocol (HTTP), which provides the transfer service for Web client/server interaction, can operate on top of SSL. Three higher-layer protocols are defined as part of SSL: the Handshake Protocol, The Change Cipher Spec Protocol, and the Alert Protocol. Two important SSL concepts are the SSL session and the SSL connection, which are defined in the specification as follows:

- **Connection:** A connection is a transport (in the OSI layering model definition) that provides a suitable type of service. For SSL, such connections are peer-to-peer relationships. The connections are transient. Every connection is associated with one session.
- **Session:** An SSL session is an association between a client and a server. Sessions are created by the Handshake Protocol. Sessions define a set of cryptographic security parameters, which can be shared among multiple connections. Sessions are used to avoid the expensive negotiation of new security parameters for each connection.

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

A session state is defined by the following parameters:

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

☐☐**Peer certificate**: An X.509.v3 certificate of the peer. This element of the state may be null.

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

☐☐**Cipher spec**: Specifies the bulk data encryption algorithm (such as null, AES, etc.) and a hash algorithm (such as MD5 or SHA-1) used for MAC calculation. It also defines cryptographic attributes such as the hash\_size.

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

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

A connection state is defined by the following parameters:

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

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

☐☐**Client write MAC secret**: The secret key used in MAC operations on data sent by the client.

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

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

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

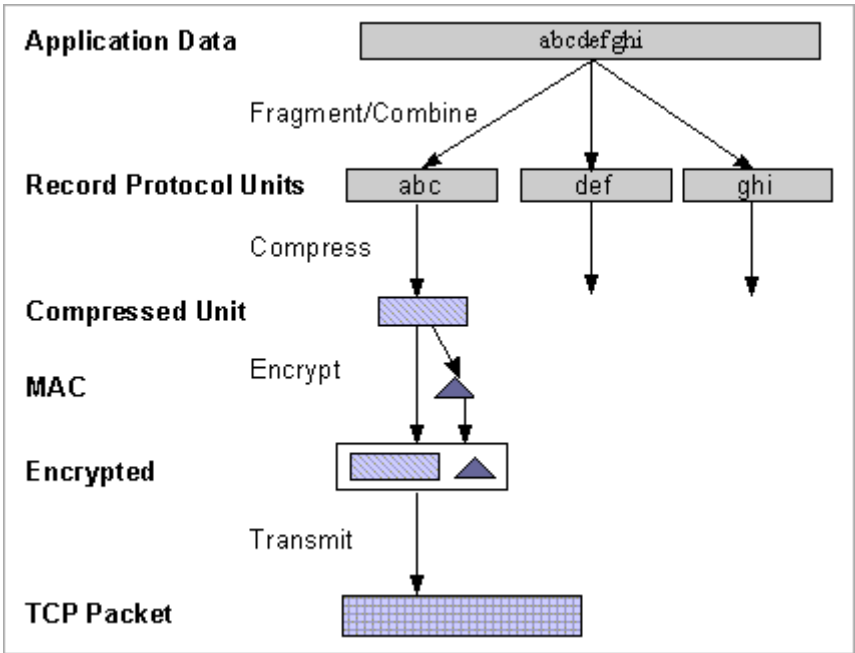
☐☐**Sequence numbers**: Each party maintains separate sequence numbers for transmitted and received messages for each connection. When a party sends or receives a change cipher spec message, the appropriate sequence number is set to zero. Sequence numbers may not exceed 2<sup>64</sup>-1.

## **SSL Record Protocol**

The SSL Record Protocol provides two services for SSL connections:

- Confidentiality: The Handshake Protocol defines a shared secret key that is used for conventional encryption of SSL payloads.
- Message Integrity: The Handshake Protocol also defines a shared secret key that is used to form a message authentication code (MAC).

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



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

$$\text{hash}(\text{MAC\_write\_secret} || \text{pad\_2} || \text{hash}(\text{MAC\_write\_secret} || \text{pad\_1} || \text{seq\_num} || \text{SSLCompressed.type} || \text{SSLCompressed.length} || \text{SSLCompressed.fragment}))$$

Where,

MAC\_write\_secret

=

the byte 0x36

=Secretsharedkeypad\_1

(00110110)repeated48times

(384bits)forMD5and40timesfor

pad\_2

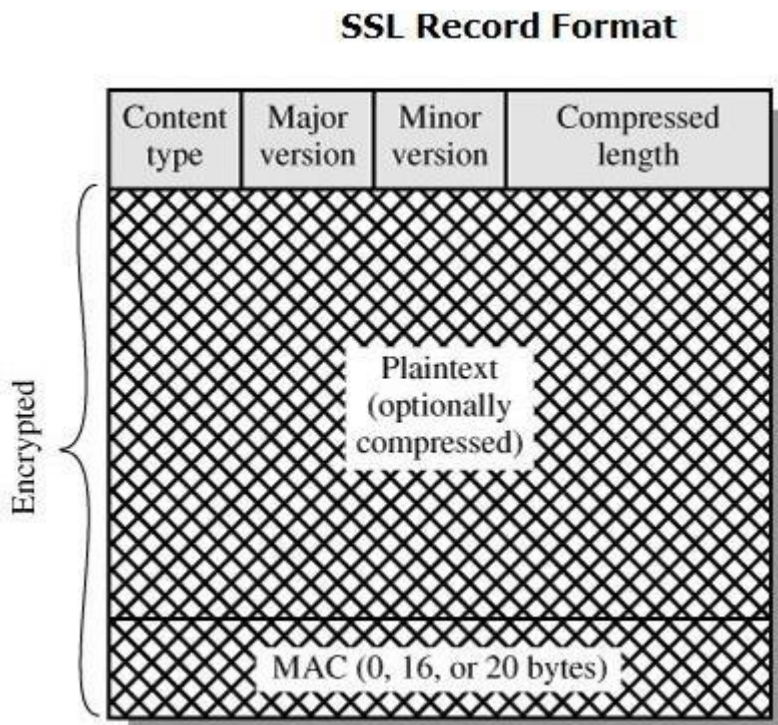
=

the byte 0x5C

(01011100) repeated 48timesforMD5and40timesfor

SHA-1

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



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

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

The content types that have been defined are change\_cipher\_spec, alert, handshake, and application\_data.

## **SSLChangeCipherSpecProtocol**

The Change Cipher Spec Protocol is one of the three SSL-specific protocols that use the SSL Record Protocol, and it is the simplest. This protocol consists of a single message, which consists of a single byte with the value 1.

The sole purpose of this message is to cause the pending state to be copied into the current state, which updates the cipher suite to be used on this connection.

## **SSLAlertProtocol**

The Alert Protocol is used to convey SSL-related alerts to the peer entity. As with other applications that use SSL, alert messages are compressed and encrypted, as specified by the current state. Each message in this protocol consists of two bytes.

The first byte takes the value warning(1) or fatal(2) to convey the severity of the message. If the level is fatal, SSL immediately terminates the connection. Other connections on the same session may continue, but no new connections on this session may be established. The second byte contains a code that indicates the specific alert. The fatal alerts are listed below

- **unexpected\_message:** An inappropriate message was received.
- **bad\_record\_mac:** An incorrect MAC was received.
- **decompression\_failure:** The decompression function received improper input (e.g., unable to decompress or decompress to greater than maximum allowable length).
- **handshake\_failure:** Sender was unable to negotiate an acceptable set of security parameters given the options available.
- **illegal\_parameter:** A field in a handshake message was out of range or inconsistent with other fields.

The remainder of the alerts are given below:

- `close_notify`: Notifies the recipient that the sender will not send any more messages on this connection. Each party is required to send a `close_notify` alert before closing the write side of a connection.
- `no_certificate`: May be sent in response to a certificate request if no appropriate certificate is available.
- `bad_certificate`: A received certificate was corrupt (e.g., contained a signature that did not verify).
- `unsupported_certificate`: The type of the received certificate is not supported.
- `certificate_revoked`: A certificate has been revoked by its signer.
- `certificate_expired`: A certificate has expired.
- `certificate_unknown`: Some other unspecified issue arose in processing the certificate, rendering it unacceptable.

### SSL Handshake Protocol

SSL Handshake protocol ensures establishment of reliable and secure session between client and server and also allows server & client to:

- authenticate each other
- to negotiate encryption & MAC algorithms
- to negotiate cryptographic keys to be used

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



(c) Handshake Protocol

- **Type (1 byte)**: Indicates one of 10 messages.
- **Length (3 bytes)**: The length of the message in bytes.
- **Content (≥ 0 bytes)**: The parameters associated with this message

The following figures show the initial exchange needed to establish a logical connection between client and server. The exchange can be viewed as having four phases.

- 1. Establish Security Capabilities
- 2. Server Authentication and Key Exchange

o ClientAuthenticationandKeyExchange

o Finish

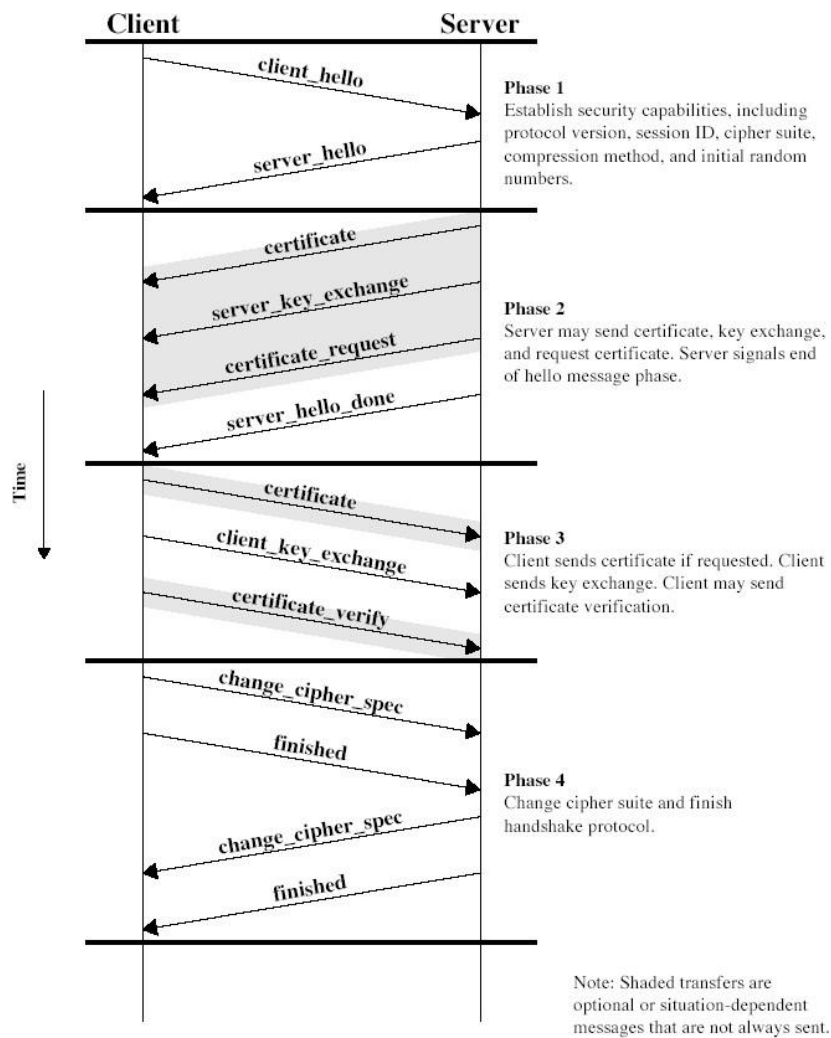
### **Phase1.EstablishSecurityCapabilities**

Thisphaseisusedtoinitiateallogicalconnectionandtoestablishthesecuritycapabilities that will be associated with it. The exchange is initiated by the client, which sends aclient\_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.

- Cipher Suite: This is a list that contains the combinations of cryptographic algorithms supported by the client, in decreasing order of preference. Each element of the list (each cipher suite) defines both a key exchange algorithm and a CipherSpec.
- Compression Method: This is a list of the compression methods the client supports.



## Phase 2. Server Authentication and Key Exchange

The server begins this phase by sending its certificate via a certificate message, which contains one or a chain of X.509 certificates. The **certificate message** is required for any agreed-on key exchange method except anonymous Diffie-Hellman. Next, a **server\_key\_exchange** message may be sent if it is required. It is not required in two instances: (1) The server has sent a certificate with fixed Diffie-Hellman parameters, or (2) RSA key exchange is to be used.

## Phase 3. Client Authentication and Key Exchange

Once the `server_done` message is received by client, it should verify whether a valid certificate is provided and check that the `server_hello` parameters are acceptable. If all is satisfactory, the client sends one or more messages back to the server. If the server has requested a certificate, the client begins this phase by sending a **certificate message**. If no suitable certificate is available, the client sends a `no_certificate` alert instead. Next is the **client\_key\_exchange** message, for which the content of the message depends on the type of key exchange.

#### **Phase 4. Finish**

This phase completes the setting up of a secure connection. The client sends a **change\_cipher\_spec** message and copies the pending `CipherSpec` into the current `CipherSpec`. The client then immediately sends the finished message under the new algorithms, keys, and secrets. The finished message verifies that the key exchange and authentication processes were successful.

## **TRANSPORT LAYER SECURITY**

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

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

MessageAuthenticationCode: Two differences arise one being the actual algorithm and the other being scope of MAC calculation. TLS makes use of the HMAC algorithm defined in RFC 2104. SSLv3 uses the same algorithm, except that the padding bytes are concatenated with the secret key rather than being XORed with the secret key padded to the block length. For TLS, the MAC calculation encompasses the fields indicated in the following expression:

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

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

PseudorandomFunction: TLS makes use of a pseudorandom function referred to as PRF to expand secrets into blocks of data for purposes of key generation or validation. The PRF is based on the following data expansion function:

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

where A() is defined

as A(0) = seed

A(i) = HMAC\_hash(secret, A(i-1))

The data expansion function makes use of the HMAC algorithm, with either MD5 or SHA-1 as the underlying hash function. As can be seen, P\_hash can be iterated as many times as necessary to produce the required quantity of data. Each iteration involves two executions of HMAC, each of which in turn involves two executions of the underlying hash algorithm.

## **SET (SECURE ELECTRONIC TRANSACTION)**

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

- Provides a secure communications channel among all parties involved in a transaction

- Provide trust by the use of X.509v3 digital certificates
- Ensure privacy because the information is only available to parties in a transaction when and where necessary

### **SET Requirements**

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

### **SET Key Features**

To meet the requirements, SET incorporates the following features:

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

### **SET Participants**

- ☐☐ Cardholder: purchasers interact with merchants from personal computers over the Internet
- ☐☐ Merchant: a person or organization that has goods or services to sell to the cardholder
- ☐☐ Issuer: a financial institution, such as a bank, that provides the cardholder with the payment card.
- ☐☐ Acquirer: a financial institution that establishes an account with a merchant and processes payment card authorizations and payments
- ☐☐ Payment gateway: a function operated by the acquirer or a designated third party that processes merchant payment messages
- ☐☐ Certification authority (CA): an entity that is trusted to issue X.509v3 public-key certificates for cardholders, merchants, and payment gateways

## Events in a transaction

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

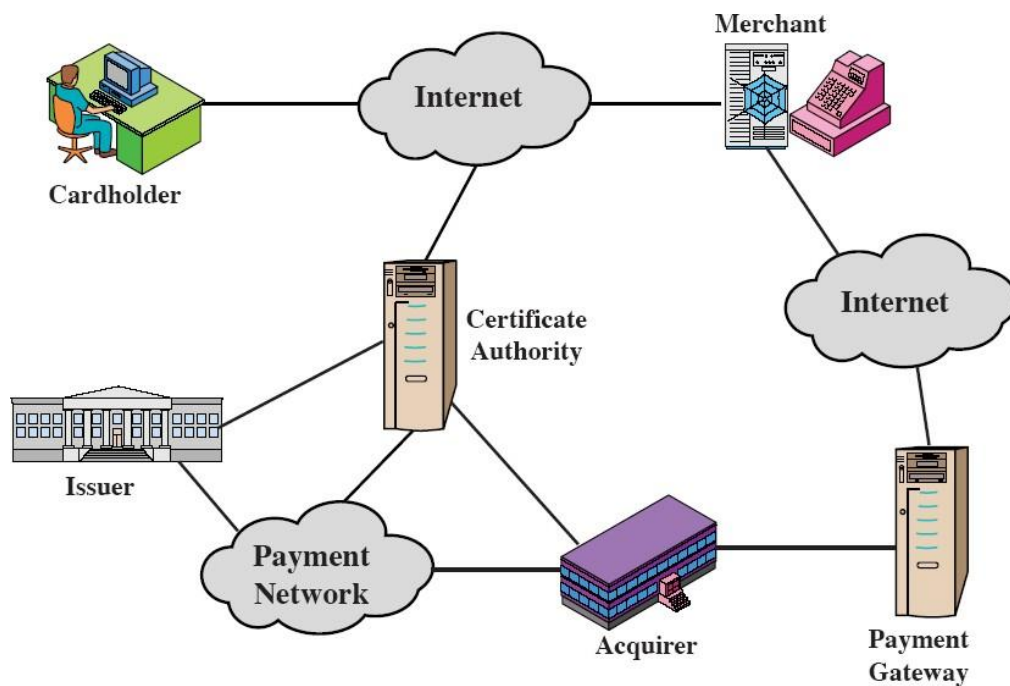
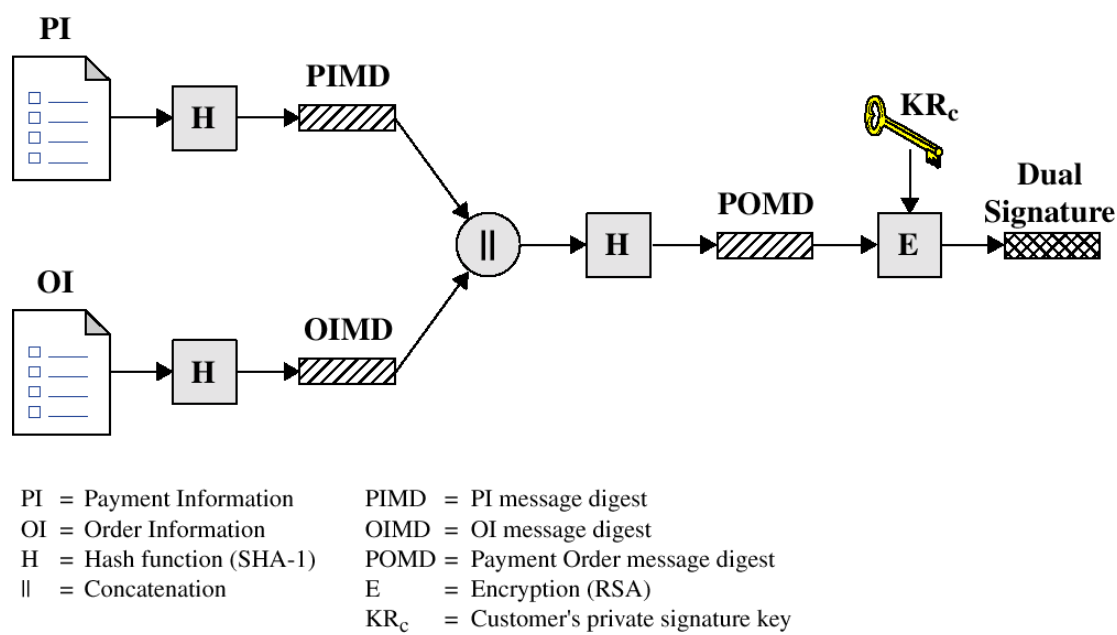


Figure 17.8 Secure Electronic Commerce Components

## DUAL SIGNATURE

The purpose of the dual signature is to link two messages that are intended for two different recipients. The customer wants to send the order information (OI) to the merchant and the payment information (PI) to the bank. The merchant does not need to

know the customer's credit card number, and the bank does not need to know the details of the customer's order. The customer is afforded extra protection in terms of privacy by keeping these two items separate. The two items must be linked and the link is needed so that the customer can prove that this payment is intended for this order and not for some other goods or service.



The customer takes the hash (using SHA-1) of the PI and the hash of the OI. These two hashes are then concatenated and the hash of the result is taken. Finally, the customer encrypts the final hash with his or her private signature key, creating the dual signature.

The operation can be summarized as

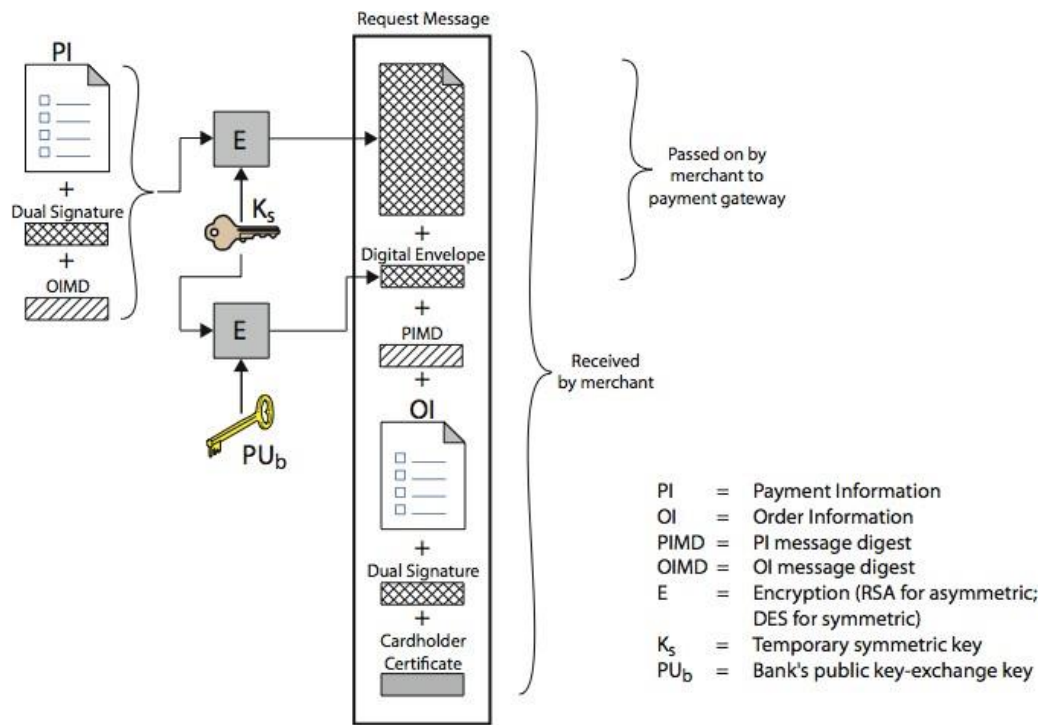
$$DS = E_{KR_c} [H(H(PI) || H(OI))]$$

where KR<sub>c</sub> is the customer's private signature key. Now suppose that the merchant is in possession of the dual signature (DS), the OI, and the message digest for the PI (PIMD). The merchant also has the public key of the customer, taken from the customer's certificate. Then the merchant can compute the quantities H(PIMS||H[OI]) and D<sub>KU<sub>c</sub></sub>(DS) where KU<sub>c</sub> is the customer's public signature key. If these two quantities are equal, then the merchant has verified the signature. Similarly, if the bank is in possession of DS, PI, the message digest for OI (OIMD), and the customer's public key, then the bank can compute H(H[OI]||OIMD) and D<sub>KU<sub>c</sub></sub>(DS). Again, if these two quantities are equal, then the bank has verified the signature. To summarize:

- ❑❑The merchant has received OI and verified the
- ❑❑signature.The bank has received PI and verified the signature.
- ❑❑The customer has linked the OI and PI and can prove the linkage.

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

**Purchase Request**

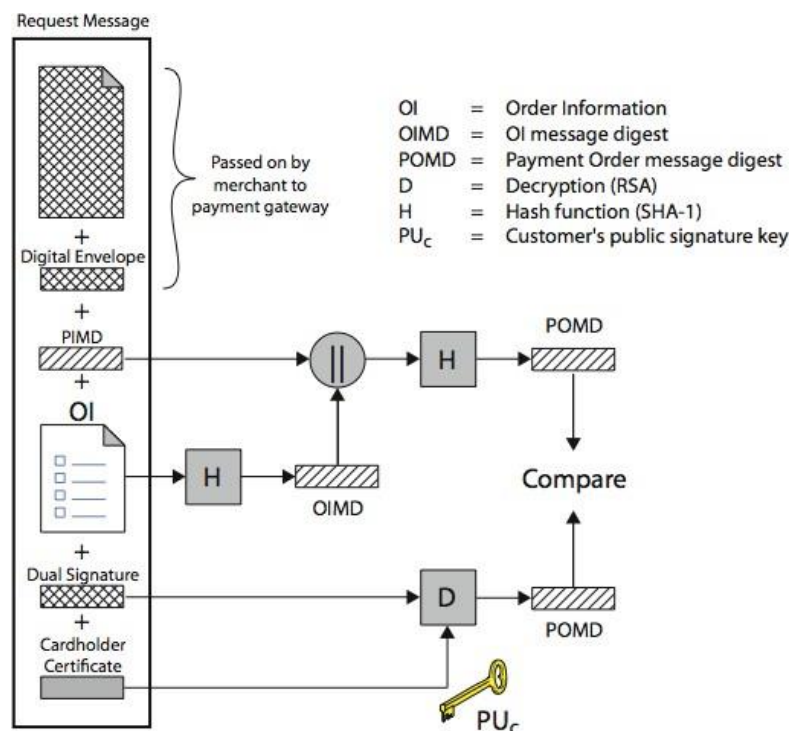


The message includes the following:

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

Merchant receives the Purchase Request message, the following actions are done:

1. verifies cardholder certificates using CA signs
2. verifies dual signature using customer's public signature key to ensure order has not been tampered with in transit & that it was signed using cardholder's private signature key
3. processes order and forwards the payment information to the payment gateway for authorization
4. sends a purchase response to cardholder



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

## INTRUDERS

One of the most publicized attacks to security is the intruder, generally referred to as a hacker or cracker. Three classes of intruders are as follows

- **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.

The masquerader is likely to be an outsider; the misfeasor generally is an insider; and the clandestine user can be either an outsider or an insider. Intruder attacks range from the benign to the serious. At the benign end of the scale, there are many people who simply wish to explore internets and see what is out there. At the serious end are individuals who are attempting to read privileged data, perform unauthorized modifications to data, or disrupt the system. Benign intruders might be tolerable, although they do consume resources and may slow performance for legitimate users. However, there is no way in advance to know whether an intruder will be benign or malign.

**Intrusion techniques** The objective of the intruders is to gain access to a system or to increase the range of privileges accessible on a system. Generally, this requires the intruder to acquire information that should be protected. In most cases, the information is in the form of a user password. Typically, a system must maintain a file that associates a password with each authorized user. If such a file is stored with no protection, then it is an easy matter to gain access to it. The password files can be protected in one of the two ways:

- **One-way encryption**– the system stores only an encrypted form of user's password. In practice, the system usually performs a one-way transformation (not reversible) in which the password is used to generate a key for the encryption function and in which a fixed length output is produced.
- **Access control** – access to the password file is limited to one or a very few accounts.

### **The following techniques are used for learning 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.

• Try words in the system's online dictionary or a list of likely passwords.

- Collect information about users such as their full names, the name of their spouse and children, pictures in their office and books in their office that are related to hobbies.
- Try user's phone number, social security numbers and room numbers.
- Try all legitimate license plate numbers.
- Use a torjan horse to bypass restriction on access.

- Tap the line between a remote user and the host

system. Two principle countermeasures:

- Detection—concerned with learning of an attack, either before or after it is successful.
- Prevention—challenging security goal and an uphill battle at all times.

## **INTRUSION DETECTION**

Inevitably, the best intrusion prevention system will fail. As a system's second line of defense is intrusion detection, and this has been the focus of much research in recent years. This interest is motivated by a number of considerations, including the following:

1. If an intrusion is detected quickly enough, the intruder can be identified and ejected from the system before any damage is done or any data are compromised.
2. An effective intrusion detection system can serve as a deterrent, so acting to prevent intrusions.
3. Intrusion detection enables the collection of information about intrusion techniques that can be used to strengthen the intrusion prevention facility.

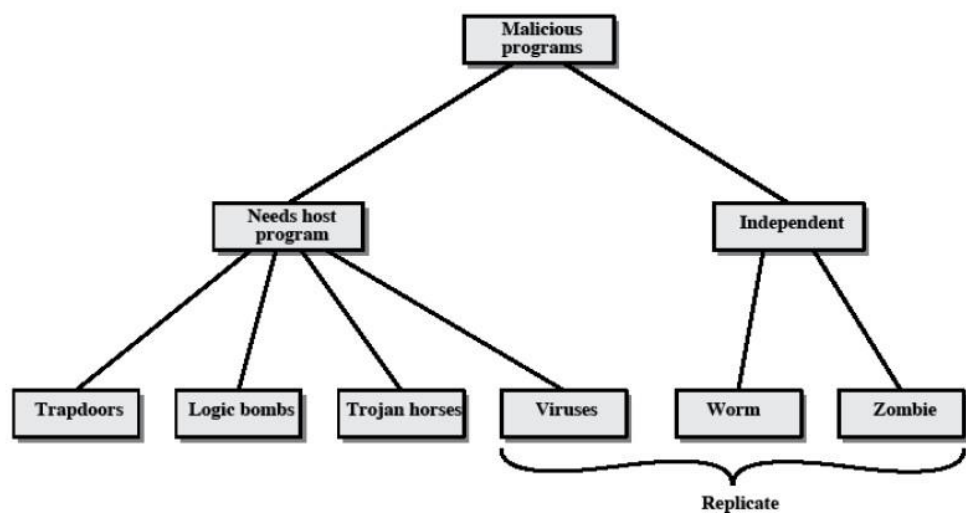
Intrusion detection is based on the assumption that the behavior of the intruder differs from that of a legitimate user in ways that can be quantified.

Figure 18.1 suggests, in very abstract terms, the nature of the task confronting the designer of an intrusion detection system. 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. On the other hand, an attempt to limit false positives by a tight interpretation of intruder behavior will lead to an increase in false negatives, or intruders not identified as intruders. Thus, there is an element of compromise and art in the practice of intrusion detection.

**VIRUSES AND RELATED THREATS**

Perhaps the most sophisticated types of threats to computer systems are represented by programs that exploit vulnerabilities in computing systems.

**Malicious Programs**



Name	Description
Virus	Attaches itself to a program and propagates copies of itself to other programs
Worm	Program that propagates copies of itself to other computers
Logic bomb	Triggers action when condition occurs
Trojan horse	Program that contains unexpected additional functionality

Backdoor(trapdoor)	Program modification that allows unauthorized access to functionality
Exploits	Code specific to a single vulnerability or set of vulnerabilities
Downloaders	Program that installs other items on a machine that is under attack. Usually, a downloader is sent in an e-mail.
Auto-rooter	Malicious hacker tools used to break into new machines remotely
Kit(virus generator)	Set of tools for generating new viruses automatically
Spammer programs	Used to send large volumes of unwanted e-mail
Flooders	Used to attack networked computer systems with a large volume of traffic to carry out a denial of service (DoS) attack
Keyloggers	Captures keystrokes on a compromised system
Rootkit	Set of hacker tools used after attacker has broken into a computer system and gained root-level access
Zombie	Program activated on an infected machine that is activated to launch attacks on other machines

Malicious software can be divided into two categories: those that need a host program, and those that are independent.

The former are essentially fragments of programs that cannot exist independently of some actual application program, utility, or system program. Viruses, logic bombs, and backdoors are examples. The latter are self-contained programs that can be scheduled and run by the operating system. Worms and zombie programs are examples.

**The 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 many functions, 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. Each infected program will now contain a clone of the virus, which will itself enter a propagation phase.
- **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.

### ***Virus Structure***

A virus can be prepended or postpended to an executable program, or it can be embedded in some other fashion. The key to its operation is that the infected program, when invoked, will first execute the virus code and then execute the original code of the program. **An infected program begins with the virus code and works as follows.**

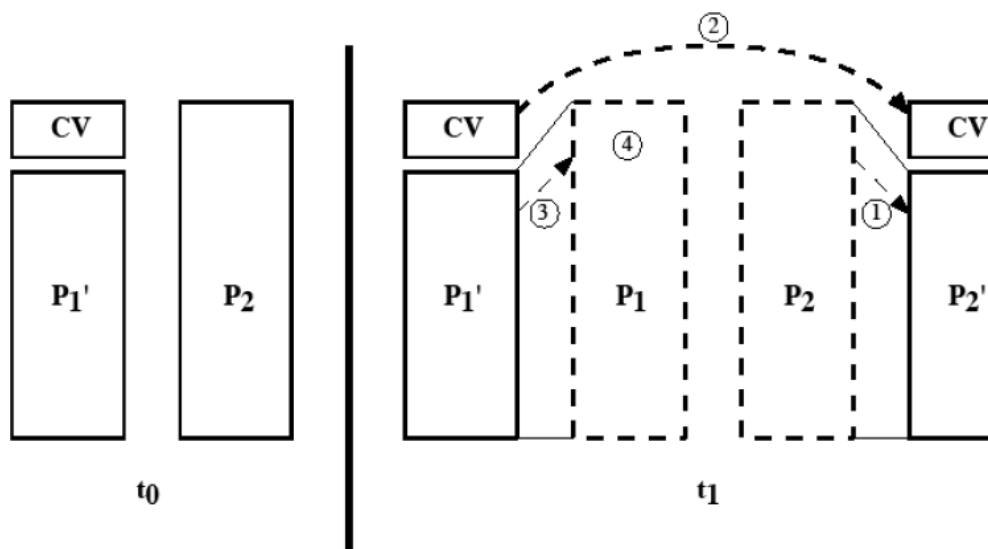
The first line of code is a jump to the main virus program. The second line is a special marker that is used by the virus to determine whether or not a potential victim program has already

been infected with this virus. When the program is invoked, control is immediately transferred to the main virus program. The virus program first seeks out uninfected executable files and infects them. Next, the virus may perform some action, usually detrimental to the system. This action could be performed every time the program is invoked, or it could be a logic bomb that triggers only under certain conditions. Finally, the virus transfers control to the original program. If the infection phase of the program is reasonably rapid, a user is unlikely to notice any difference between the execution of an infected and uninfected program.

A virus such as the one just described is easily detected because an infected version of a program is longer than the corresponding uninfected one. A way to thwart such a simple means of detecting a virus is to compress the executable files so that both the infected and uninfected versions are of identical length. The key lines in this virus are numbered, and

Figure 19.3 [COHE94] illustrates the operation. We assume that program P1 is infected with the virus CV. When this program is invoked, control passes to its virus, which performs the following steps:

- For each uninfected file P2 that is found, the virus first compresses that file to produce P'2, which is shorter than the original program by the size of the virus.
- A copy of the virus is prepended to the compressed program.
- The compressed version of the original infected program, P'1, is uncompressed.
- The uncompressed original program is executed.



In this example, the virus does nothing other than propagate. As in the previous example, the virus may include a logic bomb.

### ***Initial Infection***

Once a virus has gained entry to a system by infecting a single program, it is in a position to infect some or all other executable files on that system when the infected program executes. Thus, viral infection can be completely prevented by preventing the virus from gaining entry in the first place. Unfortunately, prevention is extraordinarily difficult because a virus can be part of any program outside a system. Thus, unless one is content to take an absolutely bare piece of iron and write all one's own system and application programs, one is vulnerable.

## Types of Viruses

Following categories are being among the most significant 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.

One example of a **stealth virus** was discussed earlier: a virus that uses compression so that the infected program is exactly the same length as an uninfected version. Far more sophisticated techniques are possible. For example, a virus can place intercept logic in disk I/O routines, so that when there is an attempt to read suspected portions of the disk using these routines, the virus will present back the original, uninfected program.

A **polymorphic virus** creates copies during replication that are functionally equivalent but have distinctly different bit patterns.

## Macro Viruses

In the mid-1990s, macro viruses became by far the most prevalent type of virus. Macro viruses are particularly threatening for a number of reasons:

1. A macro virus is platform independent. Virtually all of the macro viruses infect Microsoft Word documents. Any hardware platform and operating system that supports Word can be infected.

2. Macroviruses infect documents, not executable portions of code. Most of the information introduced onto a computer system is in the form of a document rather than a program.
3. Macroviruses are easily spread. A very common method is by electronic mail.

Macroviruses take advantage of a feature found in Word and other office applications such as Microsoft Excel, namely the macro. In essence, a macro is an executable program embedded in a word processing document or other type of file. Typically, users employ macros to automate repetitive tasks and thereby save keystrokes. The macro language is usually some form of the Basic programming language. A user might define a sequence of keystrokes in a macro and set it up so that the macro is invoked when a function key or special short combination of keys is input. Successive releases of Word provide increased protection against macroviruses. For example, Microsoft offers an optional Macro Virus Protection tool that detects suspicious Word files and alerts the customer to the potential risk of opening a file with macros. Various antivirus product vendors have also developed tools to detect and correct macroviruses.

### **E-mail Viruses**

A more recent development in malicious software is the e-mail virus. The first rapidly spreading e-

mail viruses, such as Melissa, made use of a Microsoft Word macro embedded in an attachment. If the recipient opens the e-mail attachment, the Word macro is activated. Then

1. The e-mail virus sends itself to everyone on the mailing list in the user's e-mail package.
2. The virus does local damage.

### **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. Network worm programs use network connections to spread from system to system. Once active within a system, a network worm can behave as a computer virus or bacteria, or it could implant Trojan horse programs or perform any number of disruptive or destructive actions. 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.

The new copy of the worm program is then run on the remote system where, in addition to any functions that it performs at that system, it continues to spread in the same fashion. A network worm exhibits the same characteristics as a computer virus: a dormant phase, a propagation phase, a triggering phase, and an execution phase.

### ***The Morris Worm***

The Morris worm was designed to spread on UNIX systems and used a number of different techniques for propagation.

1. It attempted to log on to a remote host as a legitimate user. In this method, the worm first attempted to crack the local password file, and then used the discovered passwords and corresponding user IDs. The assumption was that many users would use the same password on different systems. To obtain the passwords, the worm ran a password-cracking program that tried

- a. Each user's account name and simple permutations of it
- b. A list of 432 built-in passwords that Morris thought to be likely candidates
- c. All the words in the local system directory

2. It exploited a bug in the finger protocol, which reports the whereabouts of a remote user.

3. It exploited a trapdoor in the debug option of the remote process that receives and sends mail.

If any of these attacks succeeded, the worm

achieved communication with the operating system command interpreter.

**Recent Worm Attacks** In late 2001, a more versatile worm appeared, known as Nimda. Nimda spreads by multiple mechanisms:

- from client to client via e-mail
- from client to client via open network shares
- from Web server to client via browsing of compromised Websites
- from client to Web server via active scanning for and exploitation of various Microsoft

# FIREWALLS

A firewall is inserted between the premises network and the Internet to establish a controlled link and to erect an outer security wall or perimeter, forming a single chokepoint where security and audit can be imposed. A firewall:

1. Defines a single chokepoint that keeps unauthorized users out of the protected network, prohibits potentially vulnerable services from entering or leaving the network, and provides protection from various kinds of IP spoofing and routing attacks.
2. provides a location for monitoring security-related events
3. is a convenient platform for several Internet functions that are not security related, such as NAT and Internet usage audits or logs
4. A firewall can serve as the platform for IPsec to implement virtual private networks.

## Design Goals of Firewalls

All traffic from inside to outside must pass through the firewall (physically blocking all access to the local network except via the firewall)

Only authorized traffic (defined by the local security policy) will be allowed to pass

The firewall itself is immune to penetration (use of trusted system with a secure operating system)

The four general techniques that firewalls use to control access and enforce the site's security policies are:

☐☐ Service control: Determines the types of Internet services that can be accessed, inbound or outbound

☐☐ Direction control: Determines the direction in which particular service requests are allowed to flow

☐☐ User control: Controls access to a service according to which user is attempting to access it

☐☐ Behavior control: Controls how particular services are used (e.g. filter e-mail)

## The limitations of firewalls are:

1. Cannot protect against attacks that bypass the firewall, eg PCs with dial-out capability to an ISP, or dial-in modem pool use.

2. do not protect against internal threats, eg disgruntled employee or one who cooperates with an attacker

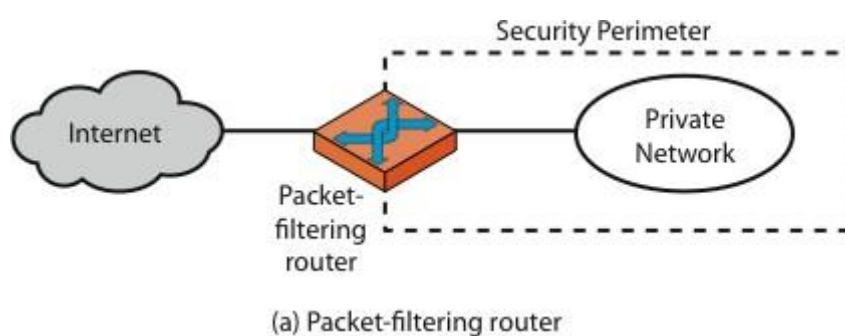
3. cannot protect against the transfer of virus-infected programs or files, given wide variety of O/S & applications supported

### Types of Firewalls

Firewalls are generally classified as three types: packet filters, application-level gateways, & circuit-level gateways.

#### Packet-filtering Router

A packet-filtering router applies a set of rules to each incoming and outgoing IP packet to forward or discard the packet. Filtering rules are based on information contained in a network packet such as source & destination IP addresses, ports, transport protocol & interface.



If there is no match to any rule, then one of two default policies are applied:

❑❑ that which is not expressly permitted is prohibited (default action is discard packet), conservative policy

❑❑ that which is not expressly prohibited is permitted (default action is forward packet), permissive policy

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

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

The table gives some examples of packet-filtering rule sets. In each set, the rules are applied top to bottom.

Table 20.1 Packet-Filtering Examples

A	action	ourhost	port	theirhost	port	comment	
	block	*	*	SPIGOT	*	we don't trust these people	
	allow	OUR-GW	25	*	*	connection to our SMTP port	
B	action	ourhost	port	theirhost	port	comment	
	block	*	*	*	*	default	
C	action	ourhost	port	theirhost	port	comment	
	allow	*	*	*	25	connection to their SMTP port	
D	action	src	port	dest	port	flags	comment
	allow	{ our hosts }	*	*	25		our packets to their SMTP port
	allow	*	25	*	*	ACK	their replies
E	action	src	port	dest	port	flags	comment
	allow	{ our hosts }	*	*	*		our outgoing calls
	allow	*	*	*	*	ACK	replies to our calls
	allow	*	*	*	>1024		traffic to non servers

A. Inbound mail is allowed to a gateway host only (port 25 is for SMTP incoming)

B. explicit statement of the default policy

C. tries to specify that any inside host can send mail to the outside, but has problem that an outside machine could be configured to have some other application linked to port 25

D. properly implements mail sending rule, by checking ACK flag of a TCP segment is set

E. this rule set is one approach to handling FTP connections

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

❏❏ **IP address spoofing**: where intruder transmits packets from the outside with internal host source IP addresses, need to filter & discard such packets

❏❏ **Source routing attacks**: where source specifies the route that a packet should take to bypass security measures, should discard all source routed packets

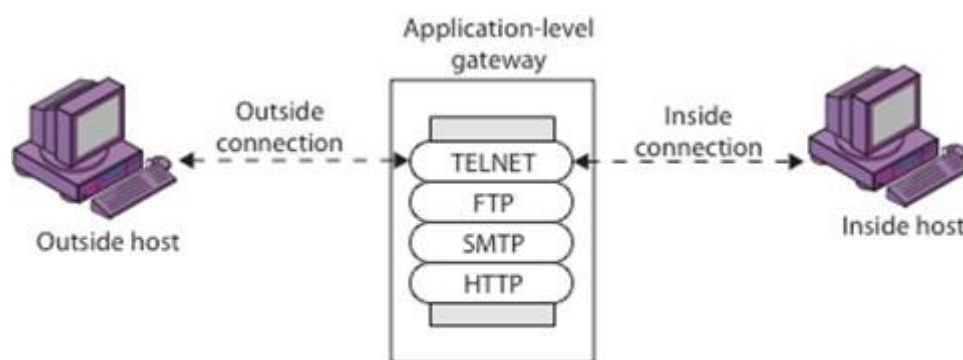
❏❏ **Tiny fragment attacks**: intruder uses the IP fragmentation option to create extremely small fragments and force the TCP header information into separate fragments to circumvent filtering rules needing full header info, can enforce minimum fragment size to include full header.

### ***StatefulPacketFilters***

A traditional packet filter makes filtering decisions on an individual packet basis and does not take into consideration any higher layer context. A stateful inspection packet filter tightens up the rules for TCP traffic by creating a directory of outbound TCP connections, and will allow incoming traffic to high-numbered ports only for those packets that fit the profile of one of the entries in this directory. Hence they are better able to detect bogus packets sent out of context.

### **APPLICATION LEVEL GATEWAY**

An application-level gateway (or proxy server), acts as a relay of application-level traffic. 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.



(b) Application-level gateway

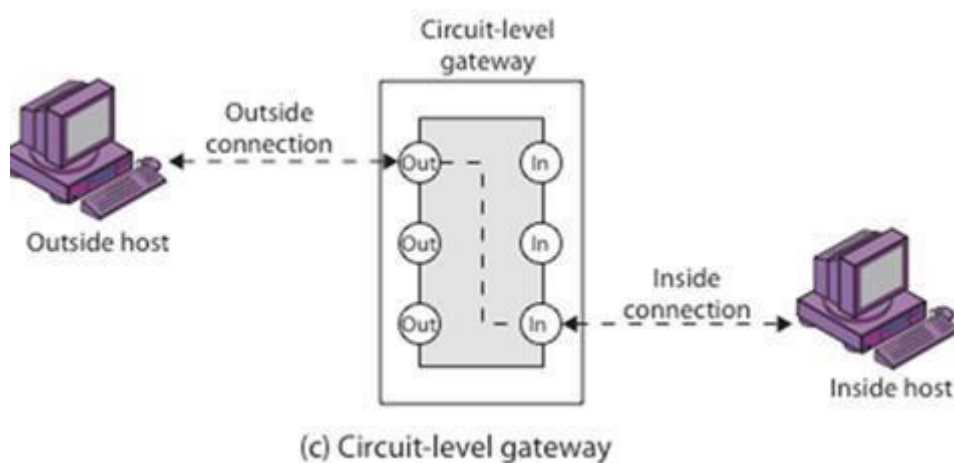
### **Application-**

level gateway tends to be more secure than packet filters. Rather than trying to deal with the numerous possible combinations that are to be allowed and forbidden at the TCP and IP level, the application-level gateway need only scrutinize a few allowable applications. In addition, it is easy to log and audit all incoming traffic at the application level. A prime disadvantage of this type of gateway is the additional processing overhead on each connection. In effect, there are two spliced connections between the end users, with the gateway at the splice point, and the gateway must examine and forward all traffic in both directions.

## CIRCUIT LEVEL GATEWAY

A circuit-level gateway relays two TCP connections, one between itself and an inside TCP user, and the other between itself and a TCP user on an outside host. Once the two connections are established, it relays TCP data from one connection to the other without examining its contents. The security function consists of determining which connections will be allowed. It is typically used when internal users are trusted to decide whether external services to access.

One of the most common circuit-level gateways is SOCKS, defined in RFC 1928. It consists of a SOCKS server on the firewall, and a SOCKS library & SOCKS-aware applications on internal clients. 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.



## Bastion Host

A bastion host is a critical strong point in the network's security, serving as a platform for an application-level or circuit-level gateway, or for external services. It is thus potentially exposed to "hostile" elements and must be secured to withstand this. Common characteristics of a bastion host include that it:

- executes a secure version of its OS, making it a trusted system
- has only essential services installed on the bastion host

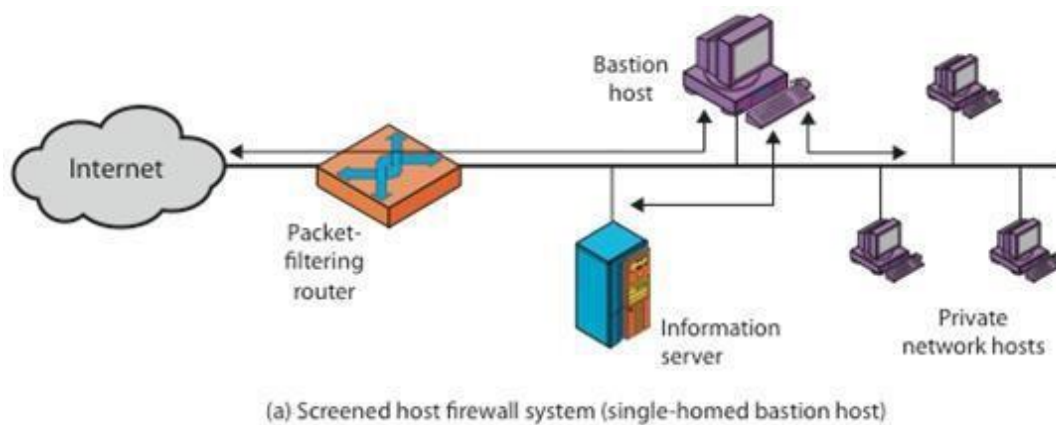
- may require additional authentication before a user is allowed access to the proxy services
- is configured to support only a subset of the standard application's command set, with access only to specific hosts
- maintains detailed audit information by logging all traffic
- has each proxy module a very small software package specifically designed for network security
- has each proxy independent of other proxies on the bastion host
- have a proxy perform no disk access other than to read its initial configuration file
- have each proxy run as a non-privileged user in a private and secured directory
- A bastion host may have two or more network interfaces (or ports), and must be trusted to enforce trusted separation between these network connections, relaying traffic only according to policy.

## Firewall Configurations

In addition to the use of a simple configuration consisting of a single system, more complex configurations are possible and indeed more common. There are three common firewall configurations.

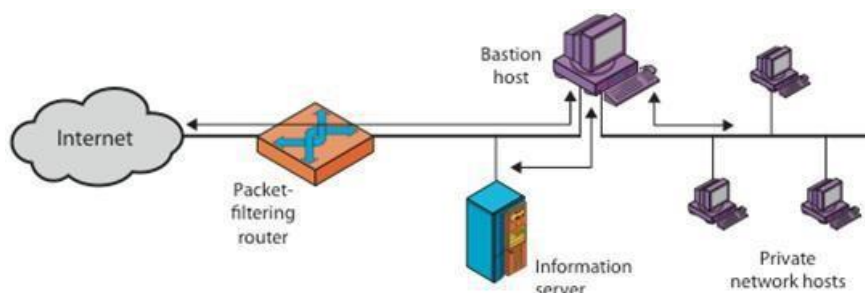
The following figure shows the “**screened host firewall, single-homed bastion configuration**”, where the firewall consists of two systems:

- a packet-filtering router - allows Internet packets to/from bastion only
- a bastion host - performs authentication and proxy functions



This configuration has greater security, as it implements both packet-level & application-level filtering, forces an intruder to generally penetrate two separate systems to compromise internal security, & also affords flexibility in providing direct Internet access to specific internal servers (eg web) if desired.

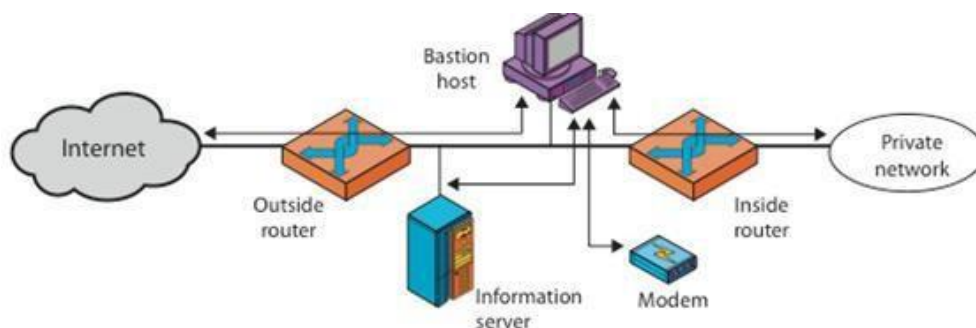
The next configuration illustrates the “**screened host firewall, dual-homed bastion configuration**” which physically separates the external and internal networks, ensuring two systems must be compromised to breach security. The advantages of dual layers of security are also present here.



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

Again, an information server or other hosts can be allowed direct communication with the router if this is in accord with the security policy, but are now separated from the internal network.

The third configuration illustrated below shows the “**screened subnet firewall configuration**”, being the most secure shown.



(c) Screened-subnet firewall system

It has two packet-filtering routers, one between the bastion host and the Internet and the other between the bastion host and the internal network, creating an isolated sub-network. This may consist of simply the bastion host but may also include one or more information servers and modems for dial-in capability. Typically, both the Internet and the internal network have access to hosts on the screened subnet, but traffic across the screened subnet is blocked.

This configuration offers several advantages:

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

## 16. ADDITIONAL TOPICS

### COMPUTER FORENSICS

Computer security and computer forensics are distinct but related disciplines due to the degree of overlap of raw material used by both fields. In general, computer security aims to preserve a system as it is meant to be (as per the security policies) whereas computer forensics (and especially network or intrusion forensics) sets out to explain how a policy became violated. Therefore, the main difference can be seen as one of system integrity versus culpability for an event or set of events.

Whereas the two fields may use similar data sources, they have different and sometimes opposing aims. For example, security countermeasures such as encryption or data wiping tools may work against the computer forensic investigation. The security measures will complicate the investigation as the data must be decrypted prior to analysis. In addition, security functions tend to only implement minimal logging by design. Therefore, not all the information required will be available to the forensic analyst.

Computer security is an established field of computer science, whilst computer forensics is an emergent area. Increasingly, computer security will involve forensic investigation techniques, and vice versa. Therefore, both fields have much to learn from each other.