Network security

A network vulnerability is an inherent weakness in the design, implementation, or use of a hardware component or a software routine. A vulnerability invites attacks and makes the network susceptible to threats.

A threat is anything that can disrupt the operation of the network. A threat can even be accidental or an act of nature, but threats are mostly intentional. A threat can damage the network, slow it down, or make it unavailable. Any type of rogue software represents a threat.

An attack is a specific approach employed to exploit a known vulnerability. A passive attack is designed to monitor and record network activity in an attempt to collect information to be used later in an active attack. Examples of passive attacks are packet sniffing and traffic analysis. Passive attacks are difficult to detect.

An active attack tries to damage a network or its operation. Such attacks are easier to detect, but are also more damaging.

Model for network security

A message is to be transferred from one party to another across some sort of Internet service. The two parties, who are the *principals* in this transaction, must cooperate for the exchange to take place. A logical information channel is established by defining a route through the Internet from source to destination and by the cooperative use of communication protocols (e.g., TCP/IP) by the two principals.

Security aspects come into play when it is necessary or desirable to protect the information transmission from an opponent who may present a threat to confidentiality, authenticity, and so on. All of the techniques for providing security have two components:

1. A security-related transformation on the information to be sent. Examples include the encryption of the message, which scrambles the message so that it is unreadable by the opponent, and the addition of a code based on the contents of the message, which can be used to verify the identity of the sender.

2. Some secret information shared by the two principals and, it is hoped, unknown to the opponent. An example is an encryption key used in conjunction with the transformation

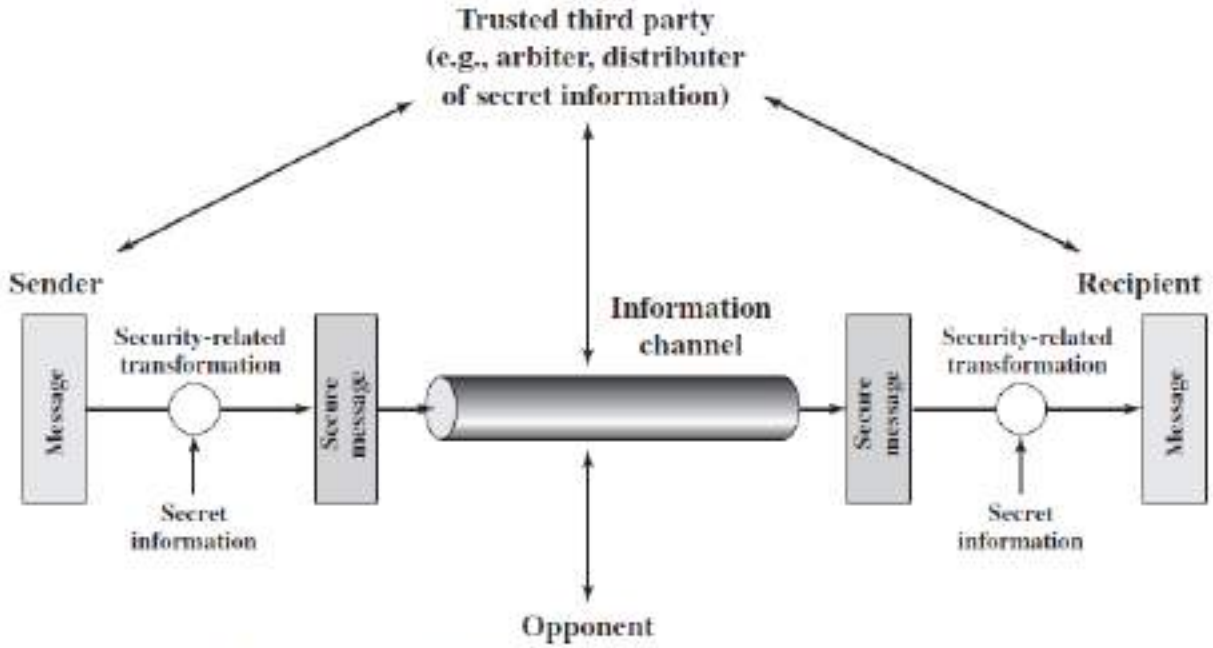


Figure Model for Network Security

Digital Certificates

Digital Certificates provide a means of proving your identity in electronic transactions, much like a driver license or a passport does in face-to-face interactions. The only difference is that a digital certificate is used in conjunction with a public key encryption system. Digital certificates are electronic files that simply work as an online passport. Digital certificates are issued by a third party known as a Certification Authority such as VeriSign or Thawte. These third party certificate authorities have the responsibility to confirm the identity of the certificate holder as well as provide assurance to the website visitors that the website is one that is trustworthy and capable of serving them in a trustworthy manner.

Digital certificates have two basic functions. The **first** is to certify that the people, the website, and the network resources such as servers and routers are reliable sources, in other words, who or what they claim to be. The **second** function is to provide protection for the data exchanged from the visitor and the website from tampering or even theft, such as credit card information.

Two parties are involved in the use of certificates. One party uses a certificate to identify itself, the other party must validate it. This process is referred to as a *handshake*. The protocol that is

used is Secure Sockets Layer/Transport Level Security (SSL/TLS). For the handshake process to work, both parties must store the certificates in their own certificate store. The certificate store is also referred as a *keystore* or a *key database*.

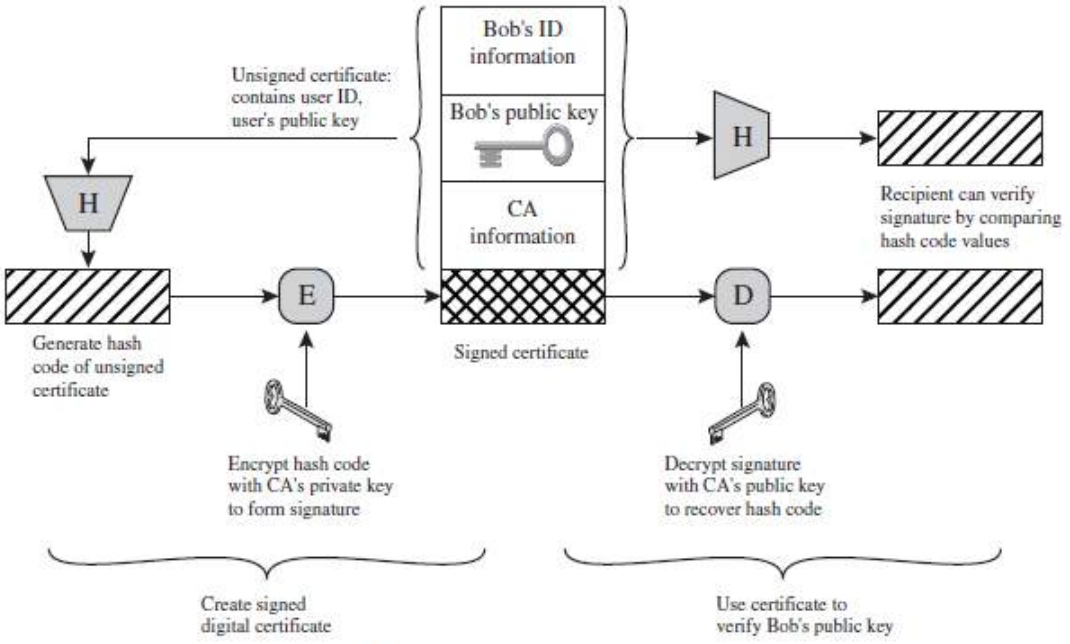


Figure Public-Key Certificate Use

But where do you obtain the certificates to put in the certificate store for the SSL/TLS protocol?

A certificate authority (CA) issues certificates. There are well-known CAs that sell certificates. There are internal CAs that issue certificates for their own enterprise. All certificates are created by using common standards. That is, no matter which CA sells you the certificate, no matter what CA and on what platform the certificate is created, it can be used by any application on any platform. Therefore, choosing the CA is an independent consideration of the platform on which the certificate is being used.

X.509 Certificates

ITU-T recommendation X.509 is part of the X.500 series of recommendations that define a directory service. The directory is, in effect, a server or distributed set of servers that maintains a database of information about users. The information includes a mapping from user name to network address, as well as other attributes and information about the users.

X.509 defines a framework for the provision of authentication services by the X.500 directory to its users. The directory may serve as a repository of public-key certificates. Each certificate contains the public key of a user and is signed with the private key of a trusted certification authority. In addition, X.509 defines alternative authentication protocols based on the use of public-key certificates.

X.509 is an important standard because the certificate structure and authentication protocols defined in X.509 are used in a variety of contexts. For example, the X.509 certificate format is used in S/MIME, IP Security, and SSL/TLS.

X.509 was initially issued in 1988. The standard was subsequently revised to address some of the security concerns. a revised recommendation was issued in 1993. A third version was issued in 1995 and revised in 2000.

X.509 is based on the use of public-key cryptography and digital signatures. The standard does not dictate the use of a specific algorithm but recommends RSA. The digital signature scheme is assumed to require the use of a hash function. Again, the standard does not dictate a specific hash algorithm. The 1988 recommendation included the description of a recommended hash algorithm; this algorithm has since been shown to be insecure and was dropped from the 1993 recommendation.

Certificates

The heart of the X.509 scheme is the public-key certificate associated with each user. These user certificates are assumed to be created by some trusted certification authority (CA) and placed in the directory by the CA or by the user. The directory server itself is not responsible for the creation of public keys or for the certification function; it merely provides an easily accessible location for users to obtain certificates.

Following figure shows the general format of a certificate, which includes the following elements.

•**Version:** Differentiates among successive versions of the certificate format; the default is version 1. If the *issuer unique identifier* or *subject unique identifier* are present, the value must be version 2. If one or more extensions are present, the version must be version 3.

• **Serial number:** An integer value unique within the issuing CA that is unambiguously associated with this certificate.

• **Signature algorithm identifier:** The algorithm used to sign the certificate together with any associated parameters. Because this information is repeated in the signature field at the end of the certificate, this field has little, if any, utility.

• **Issuer name:** X.500 is the name of the CA that created and signed this certificate.

• **Period of validity:** Consists of two dates: the first and last on which the certificate is valid.

• **Subject name:** The name of the user to whom this certificate refers. That is, this certificate certifies the public key of the subject who holds the corresponding private key.

• **Subject's public-key information:** The public key of the subject, plus an identifier of the algorithm for which this key is to be used, together with any associated parameters.

• **Issuer unique identifier:** An optional-bit string field used to identify uniquely the issuing CA in the event the X.500 name has been reused for different entities.

Subject unique identifier: An optional-bit string field used to identify uniquely the subject in the event the X.500 name has been reused for different entities.

• **Extensions:** A set of one or more extension fields. Extensions were added in version 3 and are discussed later in this section.

• **Signature:** Covers all of the other fields of the certificate; it contains the hash code of the other fields encrypted with the CA's private key. This field includes the signature algorithm identifier.

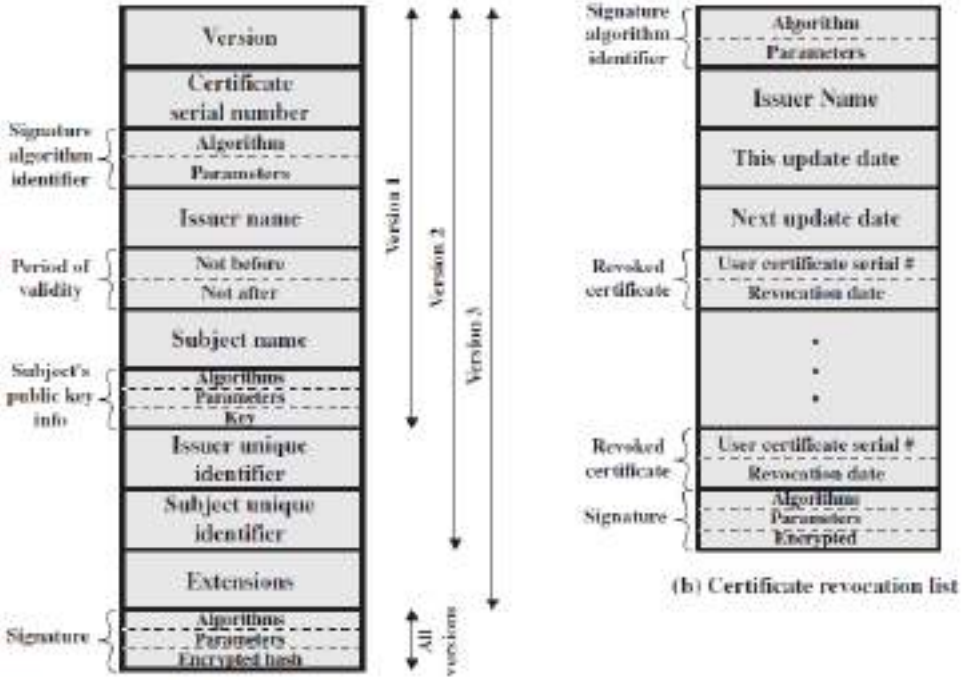


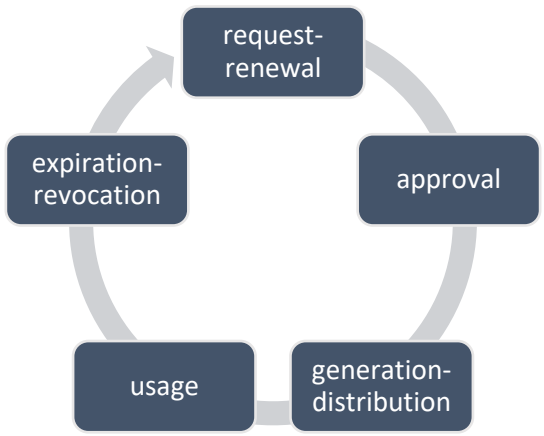
Figure X.509 Formats

Digital certificate lifecycle

Digital certificates go through phases. There are key activity points within these phases, which are referred to as the *digital certificate lifecycle*.

The following points are typical activities within the certificate's lifecycle:

- Request or Renewal: The initial request for the digital certificate or a renewal when the certificate is to expire.
- Approval: The request is approved or rejected based on the request information.
- Generation and Distribution: The digital certificate is generated with all the required elements and is distributed.
- Usage: The usage period occurs.
- Expiration or revocation: The digital certificate expires and the end of the requested usage period or it might be revoked by an administrator for any valid reason, such as the keys being compromised.



Digital certificates have a lifetime during which they are considered valid. When this lifetime expires, the certificate can no longer be used for authentication and must be updated to restore its validity. A certificate can also become invalid from being revoked by a CA. Common reasons for which a CA might revoke a digital certificate include a change in job status or suspicion of a compromised private key.

The CA typically provides a means to revoke digital certificates (*certificate revocation*). This process depends on the mechanism that the CA for each certificate makes available to communicate certificate revocation. Typically, a client that utilizes a PKI can check with the CA to update its list of revoked digital certificates so that it can fail the authentication of any revoked identities. This process can be manual or automated, depending on how the PKI is able to respond to each CAs revocation process. At a minimum, manually removing a revoked identity from a server key store or a revoked root CA certificate from a client certificate store is sufficient to handle the revocation once it is known.

Public key infrastructure

A **public key** infrastructure (PKI) consists of the components necessary to securely distribute public keys. Ideally, it consists of certificates, a repository for retrieving certificates, a method of revoking certificates, and a method of evaluating a chain of certificates from public keys that are known and trusted in advance (trust anchors) to the target name. There have been some public-key-based systems deployed that leave out components such as revocation, or even certificates. Whether such systems are worthy of being called PKIs is a matter for debate. In practice, many people do use public key technology for protecting communication and don't use a PKI. Instead, they exchange public keys in email, or download the public key from the IP address at which they assume their target is located. In theory, if an active attacker were watching when the initial exchange occurred, the attacker could change the public key in the message and act as a man-in-the-middle. But in practice, this mechanism is reasonably secure.

If Alice signs a certificate vouching for Bob's name and key, then Alice is the issuer and Bob is the subject. If Alice wants to find a path to Bob's key, then Bob's name is the target. If Alice is evaluating a chain of certificates, she is the verifier, sometimes called the relying party. Anything that has a public key is known as a principal. A trust anchor is a public key that the verifier has decided through some means is trusted to sign certificates. In a verifiable chain of certificates, the first certificate will have been signed by a trust anchor.

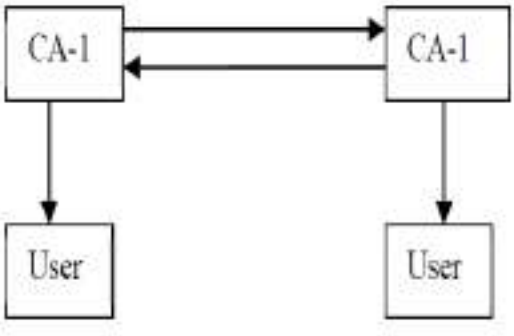
PKI Trust Models

A trust Model is collection of rules that informs application on how to decide the legitimacy of a Digital Certificate.

Architecture of a PKI is composed of operations and security policies, security services and protocols that support interoperability using public key encryption and key management certificates. In PKI a digital certificate issued by

CA and applications are usually processed by the Registration Authorities (RA). The responsibility of an RA is to analyze individual user who examines each application and notifies the CA, which is closer to the level of confidence of the applicant by checking the level of confidence, CA issue the certificate. The architecture of a PKI system describes the organization of its CAs and the trust relationship among them

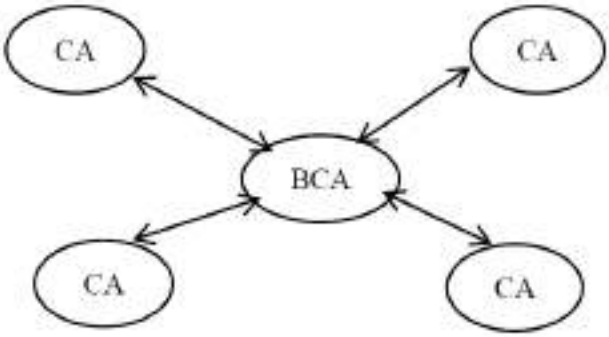
a. Peer to Peer trust model



In this model, there is no starting point as a trusted root CA. Certificate users typically rely on their own local CA, and as a starting point of trust. The two CA are now isolated. They are different trust domains; domain users can verify the domain user. This trust model is the most prominent feature is its flexibility, making the trust domain extension is very convenient. With the increasing number of CA, the certification path building is a very difficult task, may appear multiple certification paths, there may be an infinite loop, thus making it difficult to verify the certificate, thereby increasing the burden on users. Therefore, such model is applicable to small, small number of groups of coequal status of the implementation of the organizational PKI.

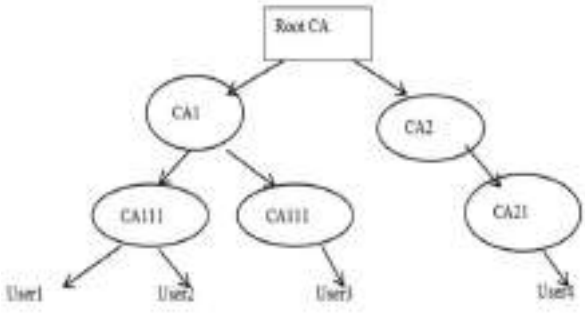
b. Bridge CA

The "Bridge PKI" model is designed to support PKI applications across enterprises and avoid the situation where the user has to maintain information of a large number of trust points or an organization needs to establish crosslink to a large number of other organizations. Using BCAs (in place of bilateral arrangements between separate PKIs) can decrease the total number of cross certificates required to join the PKIs. The BCA does not become a trust anchor for any of the PKIs as it is not directly trusted by any of the PKI entities. Rather trust is referenced from internal PKI trust anchors.



c. Hierarchical trust model

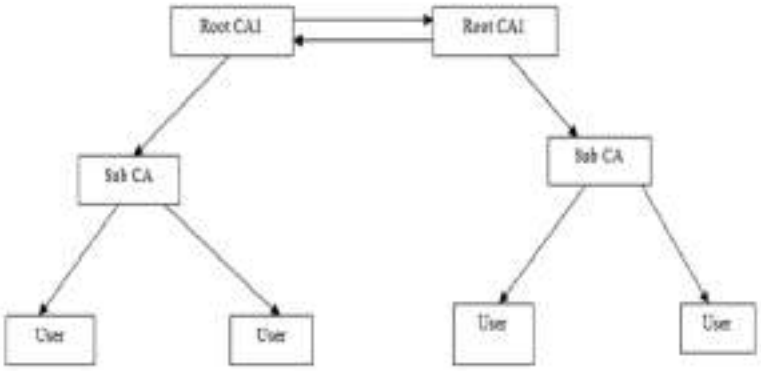
Hierarchical trust model is like an inverted tree structure, in which root is the starting point of trust, that we all trusted root CA, the top-down parts of the branches have a CA, the leaf node is the user. Root CA for the issuance of its certificate of direct



descendants of the node; intermediate nodes as its direct descendant CA. CA issued certificates node; intermediate nodes that the end-user CA can issue certificates, but for the end-user certificates issued under the CA cannot have a CA. All nodes of the model have to trust the root CA, and keep a root CA's public key certificate. Communication between any two users, in order to validate each other's public key certificate, must be achieved through the root CA.

d. Hybrid Trust Model

In practice, the general use of hybrid model is a combination of several models. As shown in the Root CA1 and CA to form a trust domain, the root CA, CA2 and its children constitute another trusted domain. They are hierarchical trust model. Then, the root of the root CA1 and CA2 for cross-certification, which constitute the two other models. Thus, the two domain users can verify each other. This model has the advantage of easy to operate, widely used.



e. Web-of-trust model

Web of trust is a term used in cryptography to describe decentralized security models in which participants authenticate the identities of other users. Web of trust is a concept used in PGP, GnuPG, and other OpenPGP-compatible systems to establish the authenticity of the binding between a public key and its owner. Its decentralized trust model is an alternative to the centralized trust model of a public key infrastructure (PKI), which relies exclusively on a certificate authority (or a hierarchy of such). In this type of system each user creates and signs certificates for the people he or she knows. Therefore, no central infrastructure needs to be developed.

The web-of-trust model differs greatly from the hierarchical model. The hierarchical model is easily represented with computers as an inverted tree, but the web-of-trust more closely relates to how people determine trust in their own lives. The system allows users to specify how much trust to place in a signature by indicating how many independent signatures must be placed on a certificate for it to be considered valid.

This model works very well for small groups who have preexisting relationships, but it doesn't scale well for large groups or where consistency of assurance (e.g. level of authentication required before a certificate is issued) is important.

PKIX

The Internet Engineering Task Force (IETF) Public Key Infrastructure X.509 (PKIX) working group has been the driving force behind setting up a formal (and generic) model based on X.509 that is suitable for deploying a certificate-based architecture on the Internet.

Figure alongside shows the interrelationship among the key elements of the PKIX model. These elements are

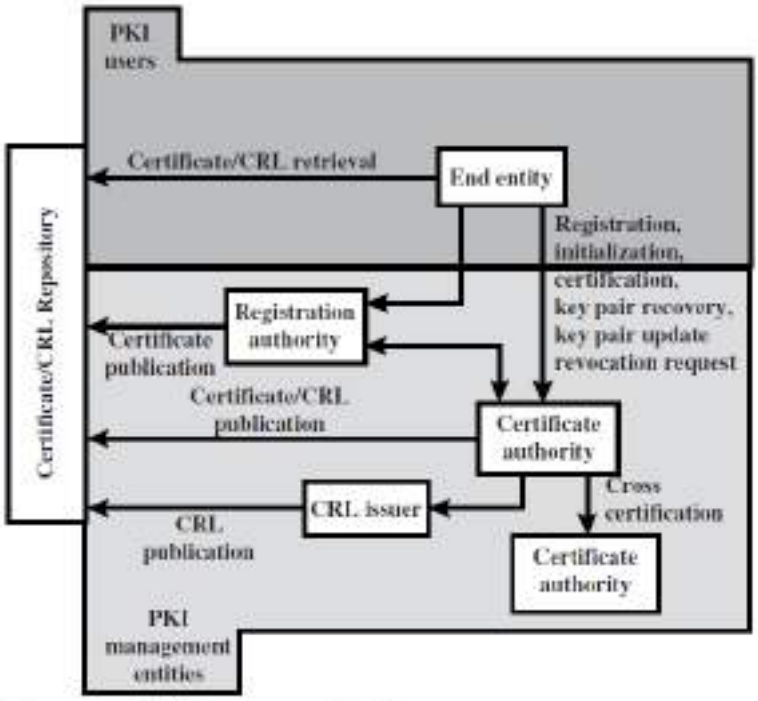


Figure PKIX Architectural Model

• **End entity:** A generic term used to denote end users, devices (e.g., servers, routers), or any other entity that can be identified in the subject field of a public key certificate. End entities typically consume and/or support PKI-related services.

• **Certification authority (CA):** The issuer of certificates and (usually) certificate revocation lists (CRLs). It may also support a variety of administrative functions, although these are often delegated to one or more Registration Authorities.

• **Registration authority (RA):** An optional component that can assume a number of administrative functions from the CA. The RA is often associated with the end entity registration process but can assist in a number of other areas as well.

• **CRL issuer:** An optional component that a CA can delegate to publish CRLs.

• **Repository:** A generic term used to denote any method for storing certificates and CRLs so that they can be retrieved by end entities.

Email Security

Email security refers to the collective measures used to secure the access and content of an email account or service. It allows an individual or organization to protect the overall access to one or more email addresses/accounts. An email service provider implements email security to secure subscriber email accounts and data from hackers - at rest and in transit.

Two schemes stand out as approaches that enjoy widespread use: Pretty Good Privacy (PGP) and S/MIME (Secure/Multipurpose Internet Mail Extensions). **Secure/Multipurpose Internet Mail Extension** (S/MIME) is a security enhancement to the MIME Internet e-mail format standard based on technology from RSA Data Security.

Pretty Good Privacy:

Pretty Good Privacy (PGP) was invented by Phil Zimmermann. PGP provides a confidentiality and authentication service that can be used for electronic mail and file storage applications. In essence, Zimmermann has done the following:

1. Selected the best available cryptographic algorithms as building blocks.

2. Integrated these algorithms into a general-purpose application that is independent of operating system and processor and that is based on a small set of easy-to-use commands.

3. Made the package and its documentation, including the source code, freely available via the Internet, bulletin boards, and commercial networks such as AOL (America On Line).

4. Entered into an agreement with a company (Viacrypt, now Network Associates) to provide a fully compatible, low-cost commercial version of PGP.

Table Summary of PGP Services

Function	Algorithms Used	Description
Digital signature	DSS/SHA or RSA/SHA	A hash code of a message is created using SHA-1. This message digest is encrypted using DSS or RSA with the sender's private key and included with the message.
Message encryption	CAST or IDEA or Three-key Triple DES with Diffie-Hellman or RSA	A message is encrypted using CAST-128 or IDEA or 3DES with a one-time session key generated by the sender. The session key is encrypted using Diffie-Hellman or RSA with the recipient's public key and included with the message.
Compression	ZIP	A message may be compressed for storage or transmission using ZIP.
E-mail compatibility	Radix-64 conversion	To provide transparency for e-mail applications, an encrypted message may be converted to an ASCII string using radix-64 conversion.

Secure Socket Layer Protocol

The Secure Socket Layer (SSL) is a standard developed by Netscape Corporation to provide security in WWW browsers and servers. The current version, SSLv3, is the basis for an Internet standard protocol under development. The newer protocol, the Transport Layer Security (TLS) protocol, is compatible with SSLv3 and has only minor changes. It has not yet been adopted formally.

SSL Main Goals

1. **Cryptography security:** One of the SSL protocol's primary goals is to establish a secure connection between two parties. A symmetric Encryption is used after an initial handshake to define a secret key.

2. **Reliability:** The connection is reliable. Message transport includes a message integrity check using a keyed MAC computed using secure hash functions.

3. **Interoperability:** Different applications should be able to successfully exchange cryptographic parameters without knowledge of one another's code.

4. **Extensibility:** Provide a framework that allows new public-key and bulk encryption methods to be incorporated as necessary. This will also achieve the goal of avoiding the need to implement an entire new security library.

5. **Relative efficiency:** Cryptographic operations tend to be highly CPU intensive. For this reason the SSL protocol has some options (such as caching and compression), which allow a reduction in the number of connections that need to be established from scratch and a reduction in network activity.

SSL Architecture

SSL is designed to make use of TCP to provide a reliable end-to-end secure service. SSL is not a single protocol but rather two layers of protocols, as illustrated in Figure.

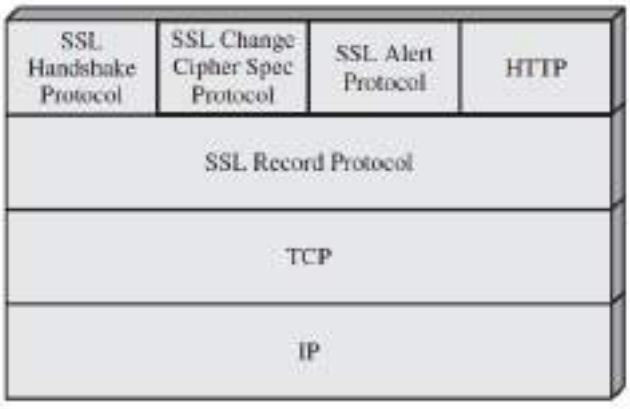


Figure SSL Protocol Stack

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.

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.

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

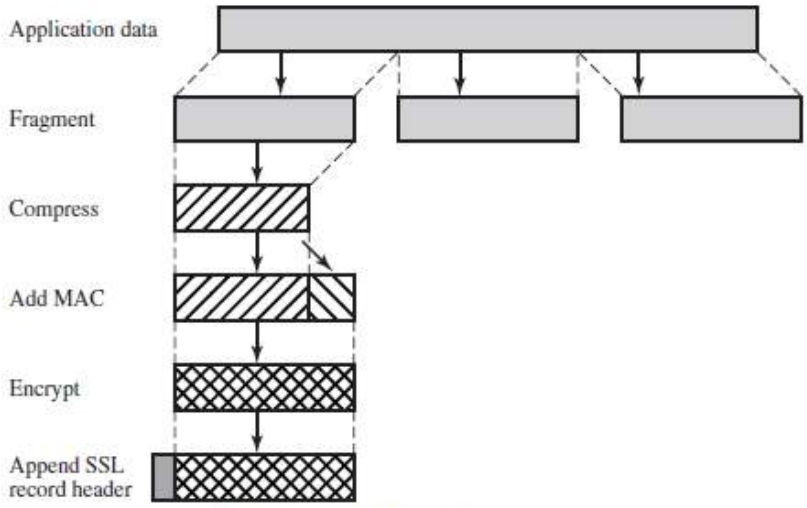


Figure SSL Record Protocol Operation

The first step is **fragmentation**. Each upper-layer message is fragmented into blocks of 2^{14} 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. In SSLv3 (as well as the current version of TLS), no compression algorithm is specified, so the default compression algorithm is null.

The next step in processing is to compute a **message authentication code** over the compressed data. For this purpose, a shared secret key is used. The calculation is defined as

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

Where,

||	=	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, this is the plaintext fragment)

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 final step of SSL Record Protocol processing is to prepare 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 0.

The content types that have been defined are change_cipher_spec, alert, handshake, and application_data. The first three are the SSL-specific protocols, discussed next. Note that no

distinction is made among the various applications (e.g., HTTP) that might use SSL; the content of the data created by such applications is opaque to SSL.

Handshake Protocol

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

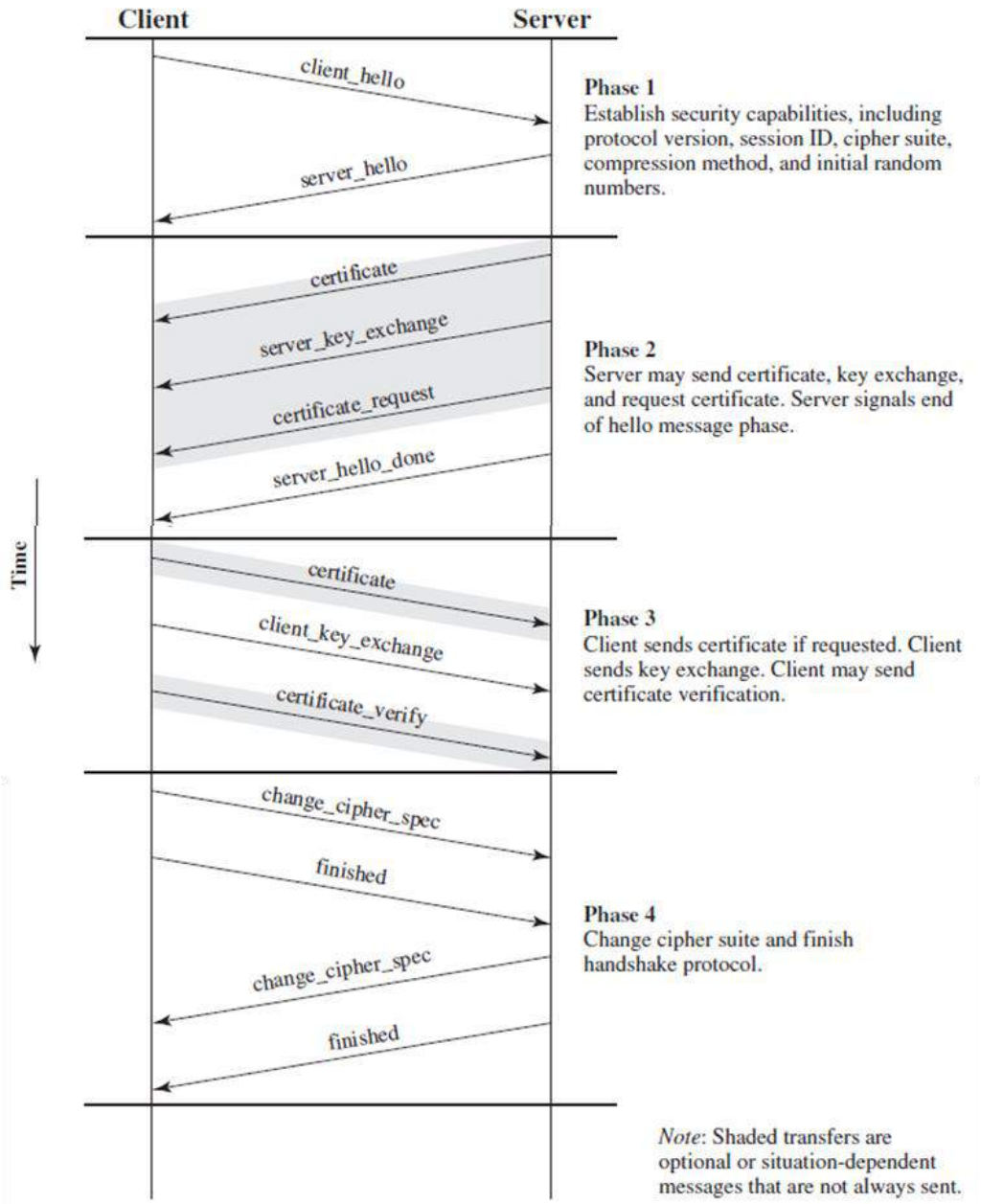


Figure 16.6 Handshake Protocol Action

IP Security

In 1994, the Internet Architecture Board (IAB) issued a report titled "Security in the Internet Architecture" (RFC 1636). The report identified key areas for security mechanisms. Among these were the need to secure the network infrastructure from unauthorized monitoring and control of network traffic and the need to secure end-user-to-end-user traffic using authentication and encryption mechanisms.

To provide security, the IAB included authentication and encryption as necessary security features in the next-generation IP, which has been issued as IPv6. Fortunately, these security capabilities were designed to be usable both with the current IPv4 and the future IPv6. This means that vendors can begin offering these features now, and many vendors now do have some IPsec capability in their products. The IPsec specification now exists as a set of Internet standards.

Applications of IPsec

IPsec provides the capability to secure communications across a LAN, across private and public WANs, and across the Internet. Examples of its use include:

• **Secure branch office connectivity over the Internet:** A company can build a secure virtual private network over the Internet or over a public WAN. This enables a business to rely heavily on the Internet and reduce its need for private networks, saving costs and network management overhead.

• **Secure remote access over the Internet:** An end user whose system is equipped with IP security protocols can make a local call to an Internet Service Provider (ISP) and gain secure access to a company network. This reduces the cost of toll charges for traveling employees and telecommuters.

• **Establishing extranet and intranet connectivity with partners:** IPsec can be used to secure communication with other organizations, ensuring authentication and confidentiality and providing a key exchange mechanism.

• **Enhancing electronic commerce security:** Even though some Web and electronic commerce applications have built-in security protocols, the use of IPsec enhances that security. IPsec guarantees that all traffic designated by the network administrator is both encrypted and authenticated, adding an additional layer of security to whatever is provided at the application layer.

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

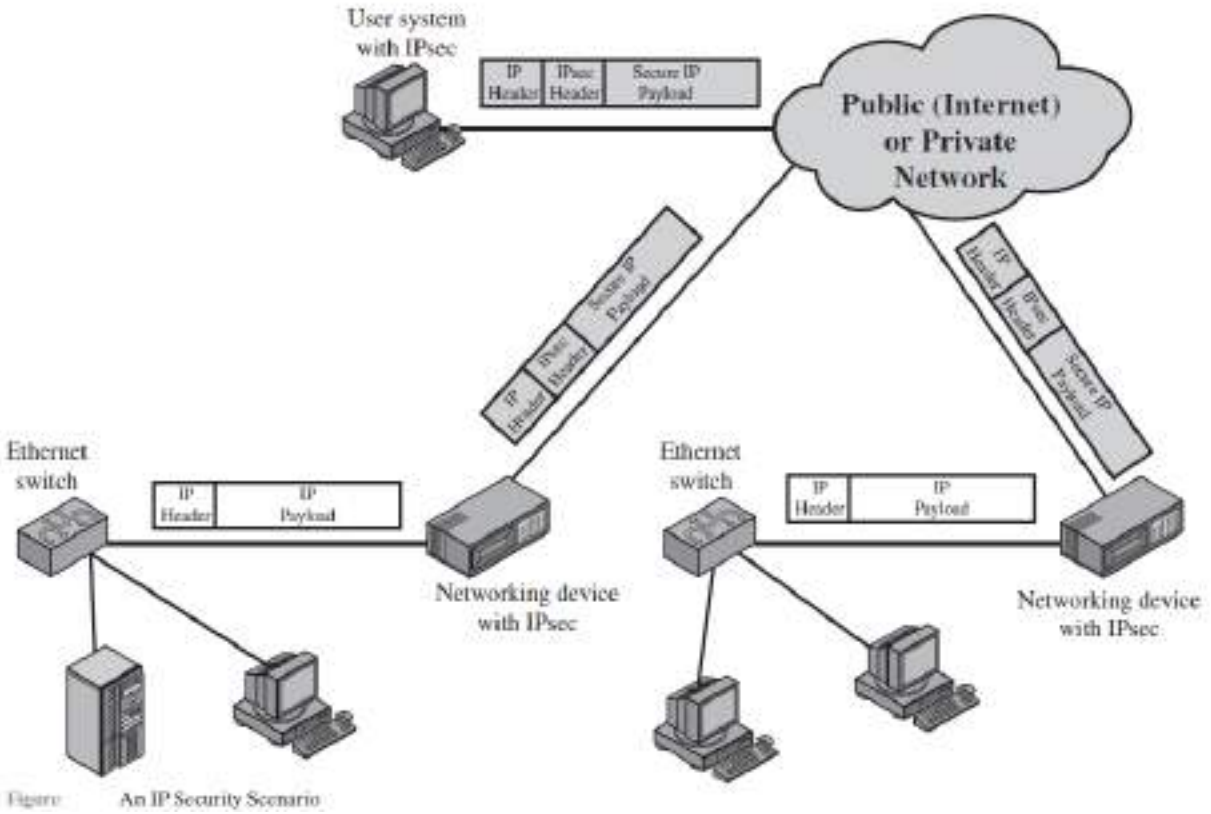


Figure An IP Security Scenario

Figure above is a typical scenario of IPsec usage. An organization maintains LANs at dispersed locations. Nonsecure IP traffic is conducted on each LAN. For traffic offsite, through some sort of private or public WAN, IPsec protocols are used. These protocols operate in networking devices, such as a router or firewall that connect each LAN to the outside world. The IPsec networking device will typically encrypt and compress all traffic going into the WAN and decrypt and decompress traffic coming from the WAN; these operations are transparent to workstations and servers on the LAN. Secure transmission is also possible with individual users who dial into the WAN. Such user workstations must implement the IPsec protocols to provide security.

Benefits of IPsec

Some of the benefits of IPsec:

• When IPsec is implemented in a firewall or router, it provides strong security that can be applied to all traffic crossing the perimeter. Traffic within a company or workgroup does not incur the overhead of security-related processing.

• IPsec in a firewall is resistant to bypass if all traffic from the outside must use IP and the firewall is the only means of entrance from the Internet into the organization.

• IPsec is below the transport layer (TCP, UDP) and so is transparent to applications. There is no need to change software on a user or server system when IPsec is implemented in the firewall or router. Even if IPsec is implemented in end systems, upper-layer software, including applications, is not affected.

• IPsec can be transparent to end users. There is no need to train users on security mechanisms, issue keying material on a per-user basis, or revoke keying material when users leave the organization.

• IPsec can provide security for individual users if needed. This is useful for offsite workers and for setting up a secure virtual subnetwork within an organization for sensitive applications.

Firewalls

A firewall forms a barrier through which the traffic going in each direction must pass. A firewall security policy dictates which traffic is authorized to pass in each direction. A firewall may be designed to operate as a filter at the level of IP packets, or may operate at a higher protocol layer. Firewalls can be an effective means of protecting a local system or network of systems from network-based security threats while at the same time affording access to the outside world via wide area networks and the Internet.

Firewall Characteristics

1. 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. Various configurations are possible, as explained later in this chapter.

2. 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, as explained later in this chapter.

3. The firewall itself is immune to penetration. This implies the use of a hardened system with a secured operating system. Trusted computer systems are suitable for hosting a firewall and often required in government applications.

Types of Firewalls:

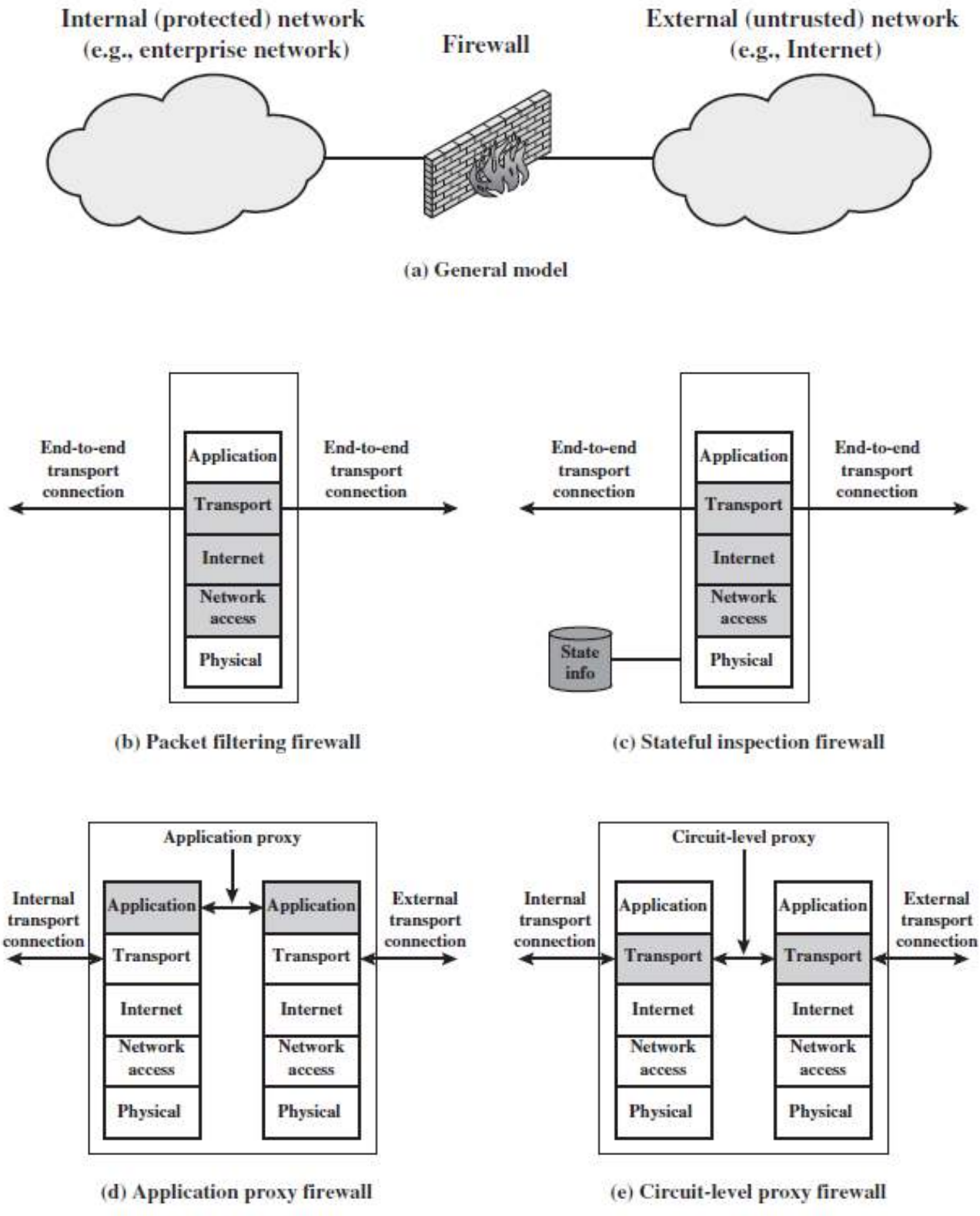


Figure Types of Firewalls

1. Packet Filtering Firewall

A packet filtering firewall applies a set of rules to each incoming and outgoing IP packet and then forwards or discards the packet (Figure b).The firewall 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 firewall with three or more ports, which interface of the firewall the packet came from or which interface of the firewall the packet is destined for

The packet filter is typically set up as a list of rules based on matches to fields in the IP or TCP header. If there is a match to one of the rules, that rule is invoked to determine whether to forward or discard the packet. If there is no match to any rule, then a default action is taken.Two default policies are possible:

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

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

2. Stateful Inspection Firewalls

A traditional packet filter makes filtering decisions on an individual packet basis and does not take into consideration any higher layer context. To understand what is meant by *context* and why a traditional packet filter is limited with regard to context, a little background is needed. Most standardized applications that run on top of TCP follow a client/server model. For example, for the Simple Mail Transfer Protocol (SMTP), e-mail is transmitted from a client system to a server system. The client system generates new e-mail messages, typically from user input. The server system accepts incoming e-mail messages and places them in the appropriate user mailboxes. SMTP operates by setting up a TCP connection between client and server, in which the TCP server port number, which identifies the SMTP server application, is 25. The TCP port number for the SMTP client is a number between 1024 and 65535 that is generated by the SMTP client.

In general, when an application that uses TCP creates a session with a remote host, it creates a TCP connection in which the TCP port number for the remote (server) application is a number less than 1024 and the TCP port number for the local (client) application is a number between 1024 and 65535.The numbers less than 1024 are the "well-known" port numbers and are assigned permanently to particular applications (e.g., 25 for server SMTP). The numbers between 1024 and 65535 are generated dynamically and have temporary significance only for the lifetime of a TCP connection.

A simple packet filtering firewall must permit inbound network traffic on all these high-numbered ports for TCP-based traffic to occur. This creates a vulnerability that can be exploited by unauthorized users.

A stateful inspection packet firewall tightens up the rules for TCP traffic by creating a directory of outbound TCP connections. There is an entry for each currently established connection. The packet filter will now allow incoming traffic to high-numbered ports only for those packets that fit the profile of one of the entries in this directory.

A stateful packet inspection firewall reviews the same packet information as a packet filtering firewall, but also records information about TCP connections (Figure c). Some stateful firewalls also keep track of TCP sequence numbers to prevent attacks that depend on the sequence number, such as session hijacking. Some even inspect limited amounts of application data for some well-known protocols like FTP, IM and SIPS commands, in order to identify and track related connections.

3. Circuit-Level Gateway

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

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

4. Application-Level Gateway

An application-level gateway, also called an **application proxy**, acts as a relay of application-level traffic (Figure d). 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. Further, the gateway can be configured to support only specific features of an application that the network administrator considers acceptable while denying all other features.

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

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

5. Next Generation Firewall (NGFW)

A next generation firewall (NGFW) is, as Gartner defines it, a "deep-packet inspection firewall that moves beyond port/protocol inspection and blocking to add application-level inspection, intrusion prevention, and bringing intelligence from outside the firewall."

Traditional firewalls vs. Next generation firewalls

As their name suggests, next generation firewalls are a more advanced version of the traditional firewall, and they offer the same benefits. Like regular firewalls, NGFW use both static and dynamic packet filtering and VPN support to ensure that all connections between the network, internet, and firewall are valid and secure. Both firewall types should also be able to translate network and port addresses in order to map IPs.

There are also fundamental differences between the traditional firewall and next generation firewalls. The most obvious difference between the two is an NGFW's ability to filter packets based on applications. These firewalls have extensive control and visibility of applications that it is able to identify using analysis and signature matching. They can use whitelists or a signature-

based IPS to distinguish between safe applications and unwanted ones, which are then identified using SSL decryption. Unlike most traditional firewalls, NGFWs also include a path through which future updates will be received.

The differentiating features of next generation firewalls create unique benefits for the companies using them. NGFWs are able to block malware from entering a network, something that traditional firewalls would never be able to achieve. They are better equipped to address Advanced Persistent Threats (APTs). NGFWs can be a low-cost option for companies looking to improve their basic security because they can incorporate the work of antiviruses, firewalls, and other security applications into one solution. The features of this include application awareness, inspection services, as well as a protection system and awareness tool that benefit the offering at all odds.