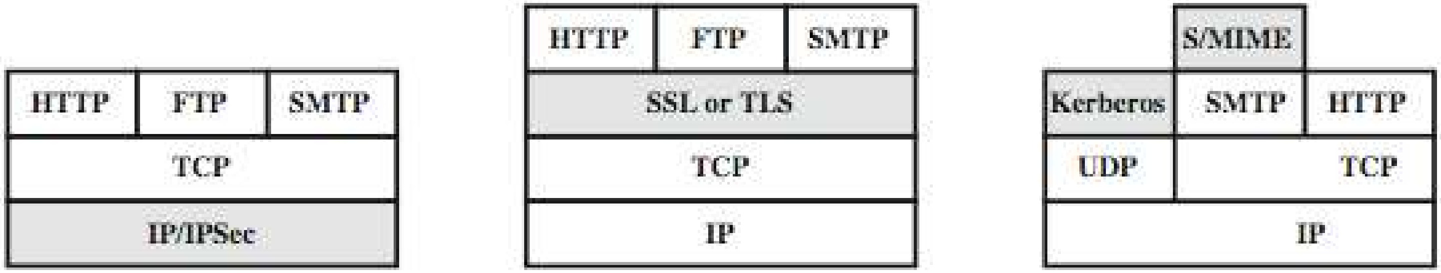
# MODULE VI

Web Security: Web Security considerations- secure Socket Layer and Transport layer Security - Secure electronic transaction. Firewalls - Packet filters- Application Level Gateway- Encrypted tunnels.

# Web Security

- Web now widely used by business, government, individuals
- but Internet & Web are vulnerable
- have a variety of threats
  - integrity
  - confidentiality
  - denial of service
  - authentication
- need added security mechanisms

# Web Traffic Security Approaches



(a) Network Level

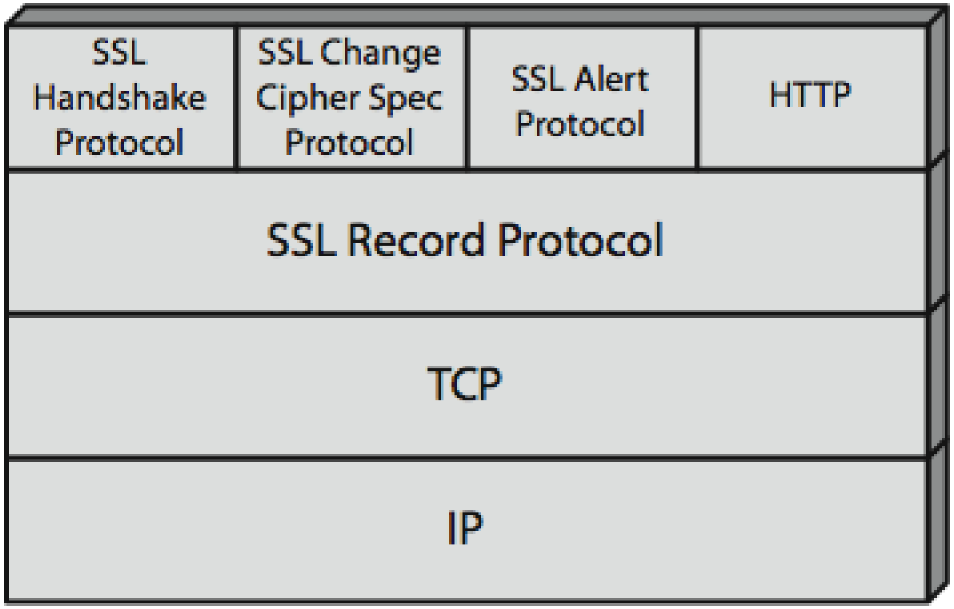
(b) Transport Level

(c) Application Level

# SSL (Secure Socket Layer)

- transport layer security service
- originally developed by Netscape
- version 3 designed with public input
- subsequently became Internet standard known as TLS (Transport Layer Security)
- uses TCP to provide a reliable end-to-end service
- SSL has two layers of protocols

# SSL Architecture



# SSL Architecture

- **SSL connection**
  - a transient, peer-to-peer, communications link
  - associated with 1 SSL session
- **SSL session**
  - an association between client & server
  - created by the Handshake Protocol
  - define a set of cryptographic parameters
  - may be shared by multiple SSL connections

# A session state is defined by the following parameters:

| Session identifier | Peer certificate | Compression method | Cipher spec | Master secret | Is resumable |
|---|---|---|---|---|---|
| An arbitrary byte sequence chosen by the server to identify an active or resumable session state | An X509.v3 certificate of the peer; this element of the state may be null | The algorithm used to compress data prior to encryption | Specifies the bulk data encryption algorithm and a hash algorithm used for MAC calculation; also defines cryptographic attributes such as the hash_size | 48-byte secret shared between the client and the server | 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 secret encryption key for data encrypted by the server and decrypted by the client

**Client write key**

- The symmetric 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
- 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 $2^{64}$ - 1

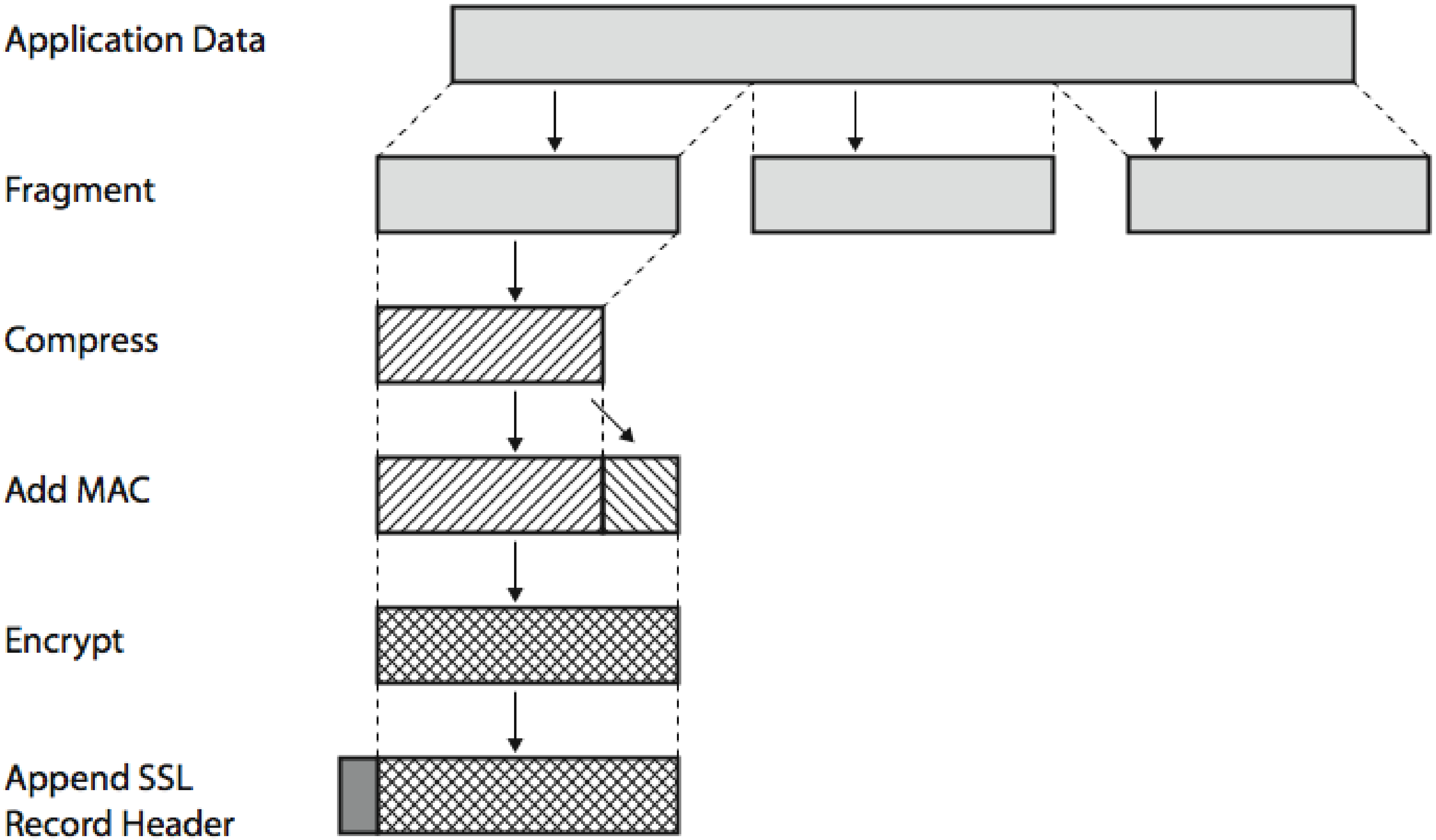
# Cipher Suite

- For public-key, symmetric encryption and certificate verification we need
  - public-key algorithm
  - symmetric encryption algorithm
  - message digest (hash) algorithm
- This collection is called a cipher suite
- SSL supports many different suites
- Client and server must decide on which one to use
- The client offers a choice; the server picks one

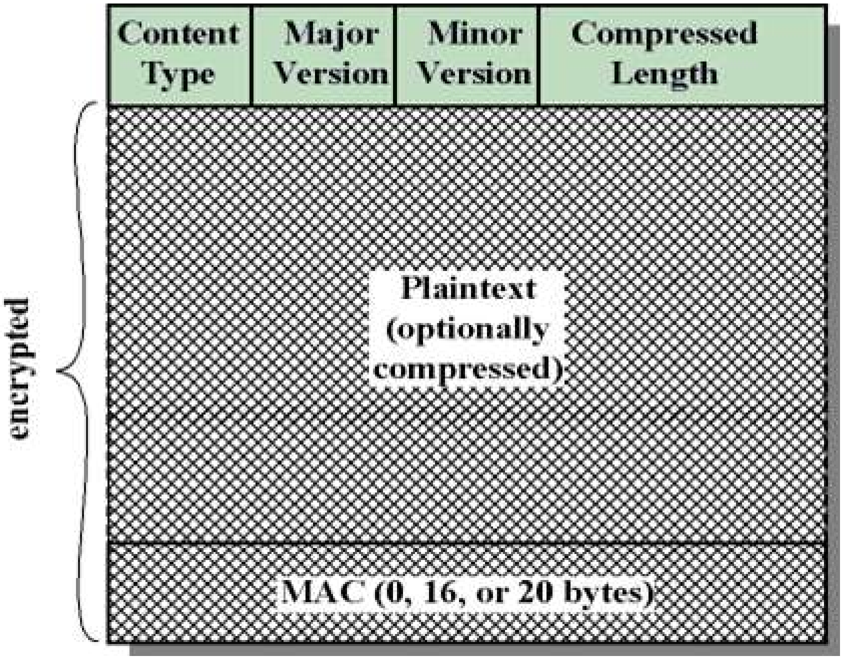
# SSL Record Protocol Services

- **confidentiality**
  - using symmetric encryption with a shared secret key defined by Handshake Protocol
  - AES, IDEA, RC2-40, DES-40, DES, 3DES, Fortezza, RC4-40, RC4-128
  - message is compressed before encryption
- **message integrity**
  - using a MAC with shared secret key
  - similar to HMAC but with different padding

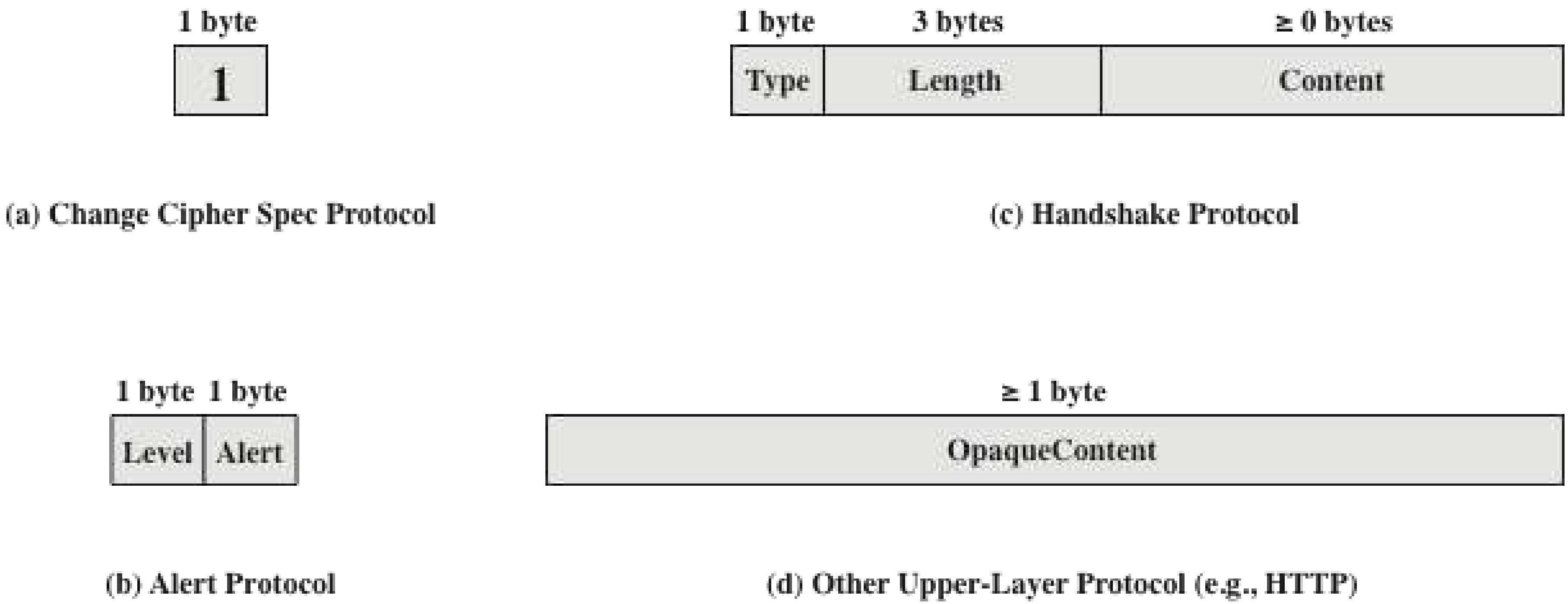
# SSL Record Protocol Operation



# SSL Record Protocol Format



- The content types that have been defined are:
- change_cipher_spec
- alert
- handshake
- application_data



(a) Change Cipher Spec Protocol

(c) Handshake Protocol

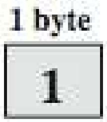
(b) Alert Protocol

(d) Other Upper-Layer Protocol (e.g., HTTP)

Figure 17.5 SSL Record Protocol Payload

# SSL Change Cipher Spec Protocol

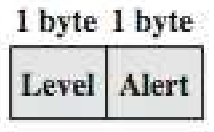
- one of 3 SSL specific protocols which use the SSL Record protocol
- a single message
- causes pending state to become current
- hence updating the cipher suite in use



(a) Change Cipher Spec Protocol

# SSL Alert Protocol

- conveys SSL-related alerts to peer entity
- severity
  - warning or fatal
- specific alert
  - fatal: unexpected message, bad record mac, decompression failure, handshake failure, illegal parameter
  - warning: close notify, no certificate, bad certificate, unsupported certificate, certificate revoked, certificate expired, certificate unknown
- compressed & encrypted like all SSL data

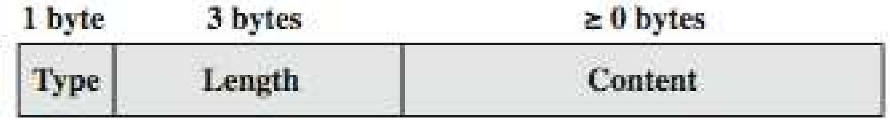


| 1 byte | 1 byte |
|---|---|
| Level | Alert |

(b) Alert Protocol

# SSL Handshake Protocol

- allows server & client to:
  - authenticate each other
  - to negotiate encryption & MAC algorithms
  - to negotiate cryptographic keys to be used
- comprises a series of messages in phases
  1. Establish Security Capabilities
  2. Server Authentication and Key Exchange
  3. Client Authentication and Key Exchange
  4. Finish



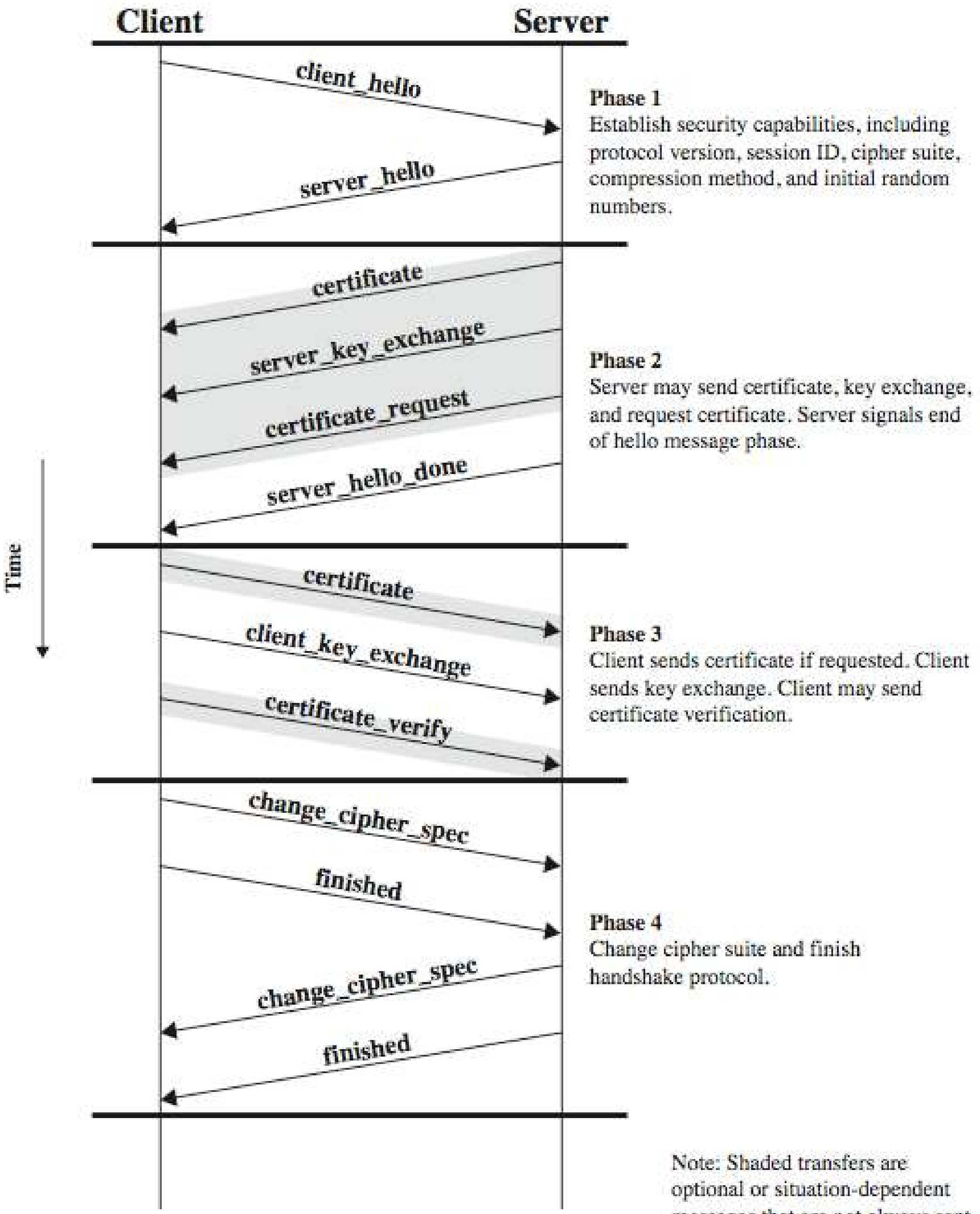
| 1 byte | 3 bytes | ≥ 0 bytes |
|---|---|---|
| Type | Length | Content |

(c) Handshake Protocol

# SSL Handshake Protocol Message Types

- hello_request
- client_hello
- server_hello
- Certificate
- server_key_exchange
- certificate_request
- server_done null
- certificate_verify
- client_key_exchange
- finished

# SSL Handshake Protocol



**Phase 1**
Establish security capabilities, including protocol version, session ID, cipher suite, compression method, and initial random numbers.

**Phase 2**
Server may send certificate, key exchange, and request certificate. Server signals end of hello message phase.

**Phase 3**
Client sends certificate if requested. Client sends key exchange. Client may send certificate verification.

**Phase 4**
Change cipher suite and finish handshake protocol.

Note: Shaded transfers are optional or situation-dependent messages that are not always sent.

# Transport Layer Security (TLS)

- The TLS Record Format is the same as that of the SSL Record Format
- 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.

# Transport Layer Security

- The same record format as the SSL record format.
- Defined in RFC 2246.
- Similar to SSLv3.
- Differences in the:
  - version number (For the current version of TLS, the Major Version is 3 and the Minor Version is 1.)
  - message authentication code
  - pseudorandom function
  - alert codes
  - cipher suites
  - client certificate types
  - certificate_verify and finished message
  - cryptographic computations
  - padding

# TLS

- TLS provides transport layer security for Internet applications
- It provides confidentiality and data integrity over a connection between two end points
- TLS operates on a reliable transport, such as TCP, and is itself layered into
  - TLS Record Protocol
  - TLS Handshake Protocol

# Advantage of TLS

- applications can use it transparently to securely communicate with each other
- TLS is visible to applications, making them aware of the cipher suites and authentication certificates negotiated during the set-up phases of a TLS session

# TLS Record Protocol

- TLS Record Protocol layers on top of a reliable connection-oriented transport, such as TCP
- TLS Record Protocol
  - provides data confidentiality using symmetric key cryptography
  - provides data integrity using a keyed message authentication checksum (MAC)
- The keys are generated uniquely for each session based on the security parameters agreed during the TLS handshake

# Basic operation of the TLS Record Protocol

1. read messages for transmit
2. fragment messages into manageable chunks of data
3. compress the data, if compression is required and enabled
4. calculate a MAC
5. encrypt the data
6. transmit the resulting data to the peer

# At the opposite end of the TLS connection, the basic operation of the sender is replicated, but in the reverse order

1. read received data from the peer
2. decrypt the data
3. verify the MAC
4. decompress the data, if compression is required and enabled
5. reassemble the message fragments
6. deliver the message to upper protocol layers

# TLS Handshake Protocol

- TLS Handshake Protocol is layered on top of the TLS Record Protocol
- TLS Handshake Protocol is used to
  - authenticate the client and the server
  - exchange cryptographic keys
  - negotiate the used encryption and data integrity algorithms before the applications start to communicate with each other

- Figure 14.1 illustrates the actual handshake message flow
  - [Step1]
    - the client and server exchange Hello messages
    - the client sends a ClientHello message, which is followed by the server sending a ServerHello message

- these two messages establish the TLS protocol version, the compression mechanism used, the cipher suite used, and possibly the TLS session ID
- additionally, both a random client nonce and a random server nonce are exchanged that are used in the handshake later on

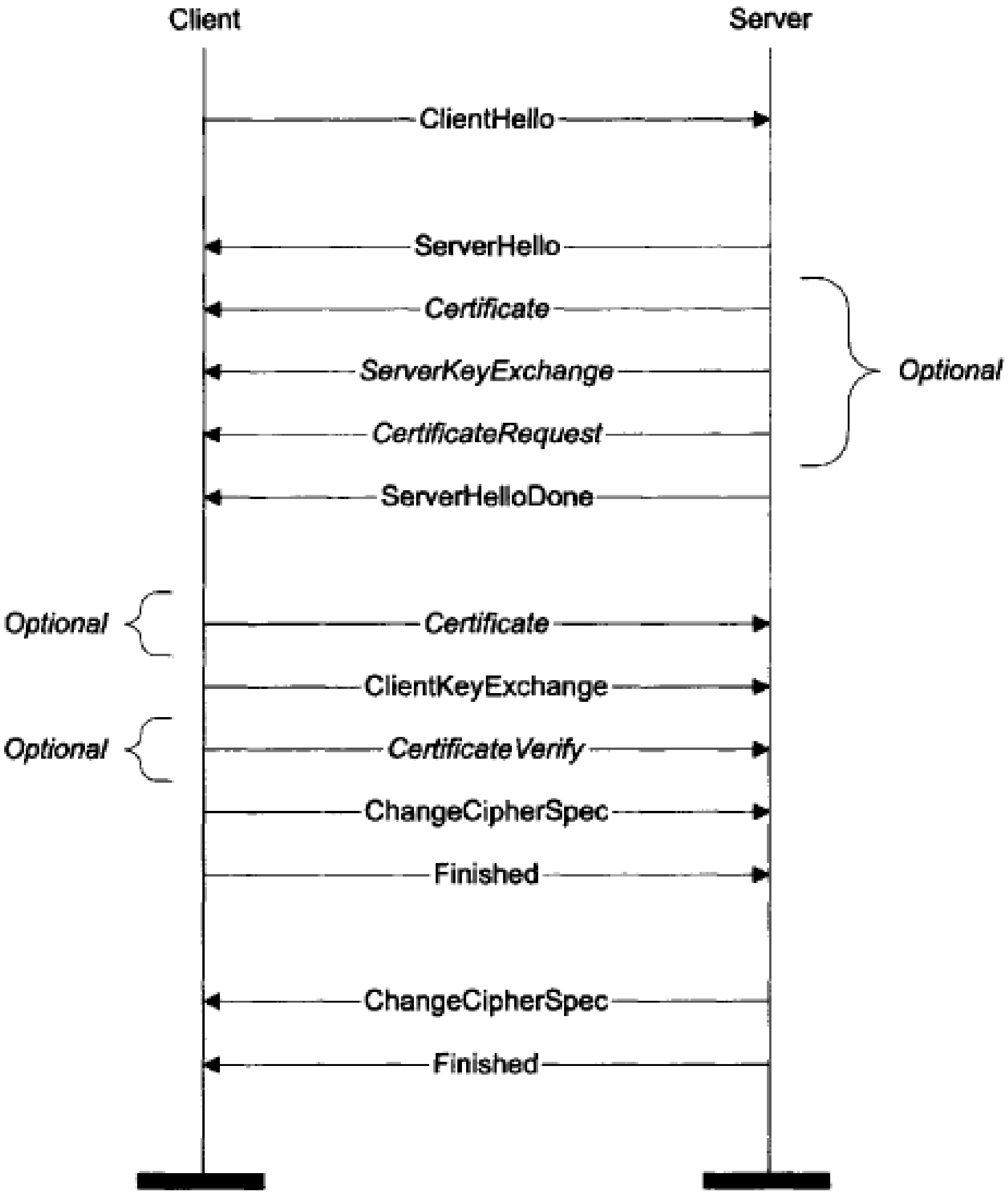


**Figure 14.1** The TLS handshake.

- [Step2]
  - the server may send any messages associated with the ServerHello
  - depending on the selected cipher suite, it will send its certificate for authentication
  - the server may also send a key exchange message and a certificate request message to the client, depending on the selected cipher suite

- to mark the end of the ServerHello and the Hello message exchange, the server sends a ServerHelloDone message

- [Step3]
  - next, if requested, the client will send its certificate to the server
  - in any case, the client will then send a key exchange message that sets the pre-master secret between the client and the server
  - optionally, the client may also send a Certificate Verify message to explicitly verify the certificate that the server requested

- [Step4]
  - then, both the client and the server send the ChangeCipherSpec messages and enable the newly negotiated cipher spec
  - the first message passed in each direction using the new algorithms, keys and secrets is the Finished message, which includes a digest of all the handshake messages
  - each end inspects the Finished message to verify that the handshake was not tampered with

- Digest of all the handshake messages
  - means the results of applying a one-way hash function to the handshake messages

# Cryptographic Computations

- Two further items are of interest:
  - The creation of a shared master secret by means of the key exchange
    - The shared master secret is a one-time 48-byte value generated for this session by means of secure key exchange
  - The generation of cryptographic parameters from the master secret
    - CipherSpecs require a client write MAC secret, a server write MAC secret, a client write key, a server write key, a client write IV, and a server write IV which are generated from the master secret in that order
      - These parameters are generated from the master secret by hashing the master secret into a sequence of secure bytes of sufficient length for all needed parameters

# Secure Electronic Transactions

- An open encryption and security specification.
- Protect credit card transaction on the Internet.
- Companies involved:
  - MasterCard, Visa, IBM, Microsoft, Netscape, RSA, Terisa and Verisign
- Not a payment system.
- Set of security protocols and formats.

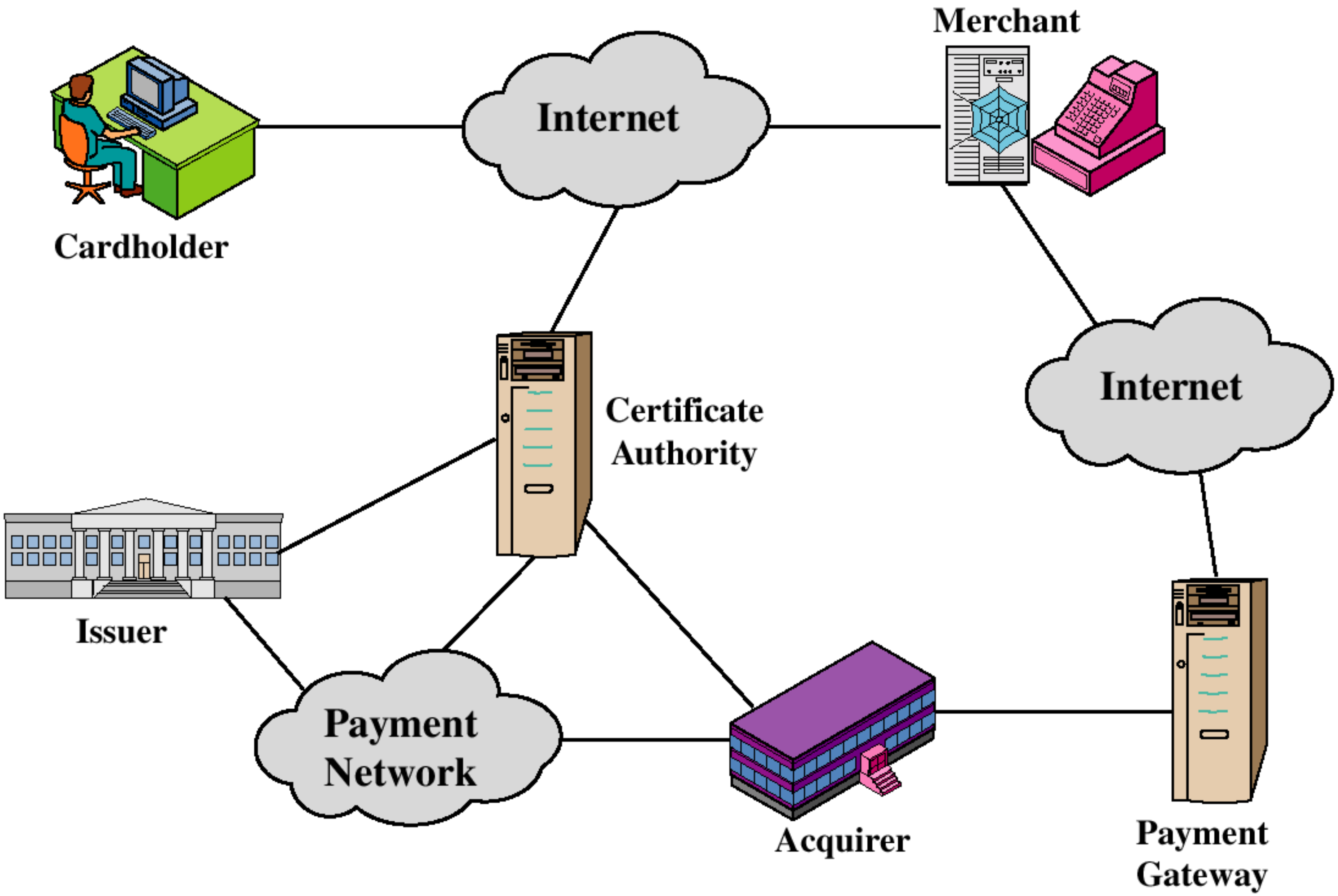
# SET Services

- Provides a secure communication channel in a transaction.
- Provides tust by the use of X.509v3 digital certificates.
- Ensures privacy.

# SET Overview

- Key Features of SET:
  - Confidentiality of information
  - Integrity of data
  - Cardholder account authentication
  - Merchant authentication

# SET Participants

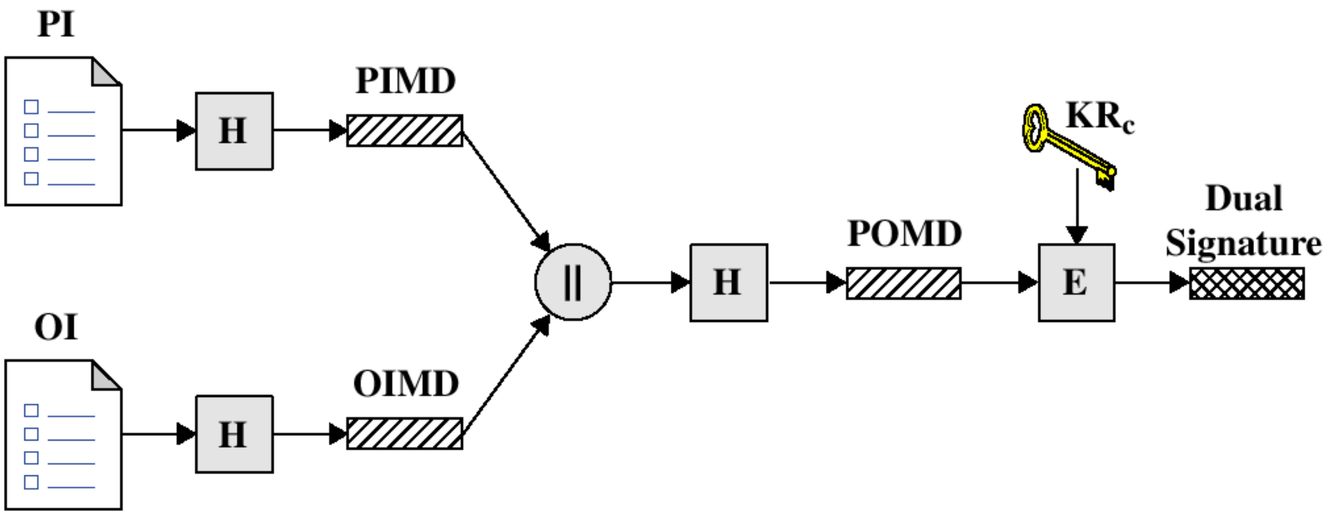


# Sequence of events for transactions

1. The customer opens an account.
2. The customer receives a certificate.
3. Merchants have their own certificates.
4. The customer places an order.
5. The merchant is verified.
6. The order and payment are sent.
7. The merchant request payment authorization.
8. The merchant confirm the order.
9. The merchant provides the goods or service.
10. The merchant requests payments.

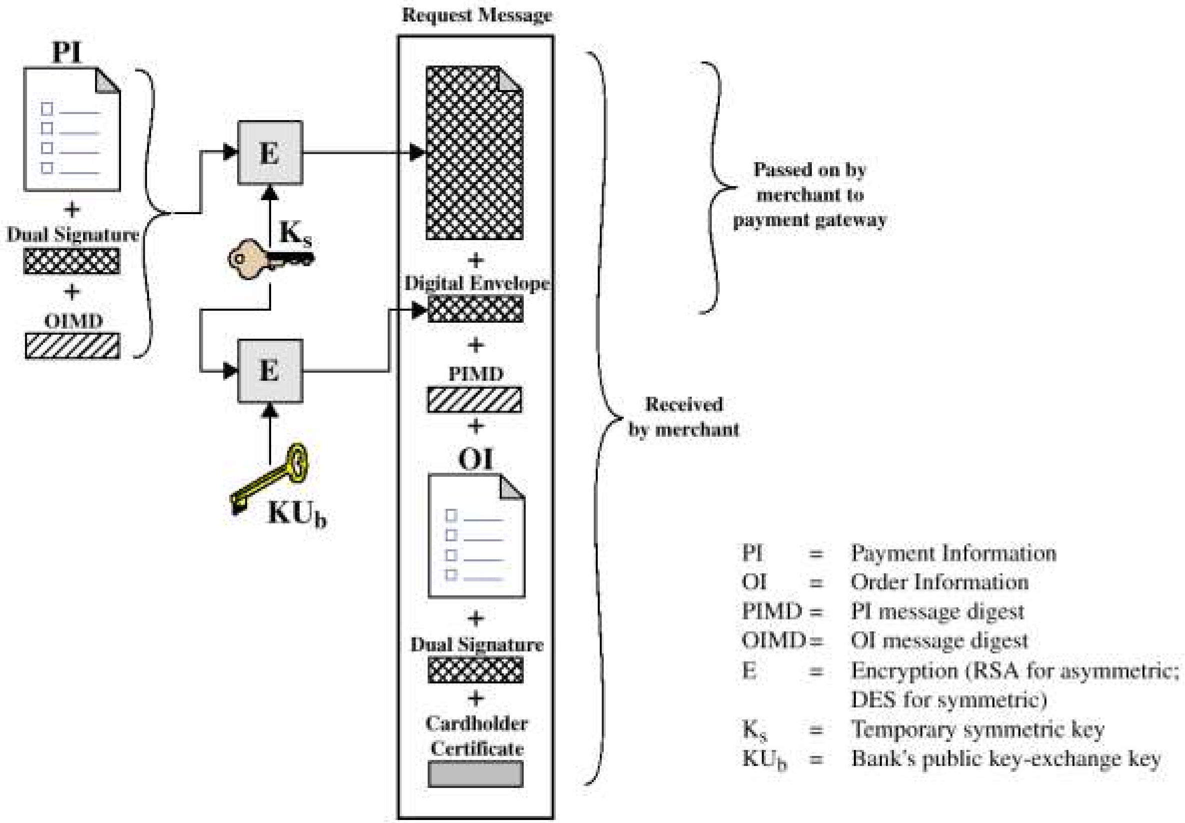
# Dual Signature

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



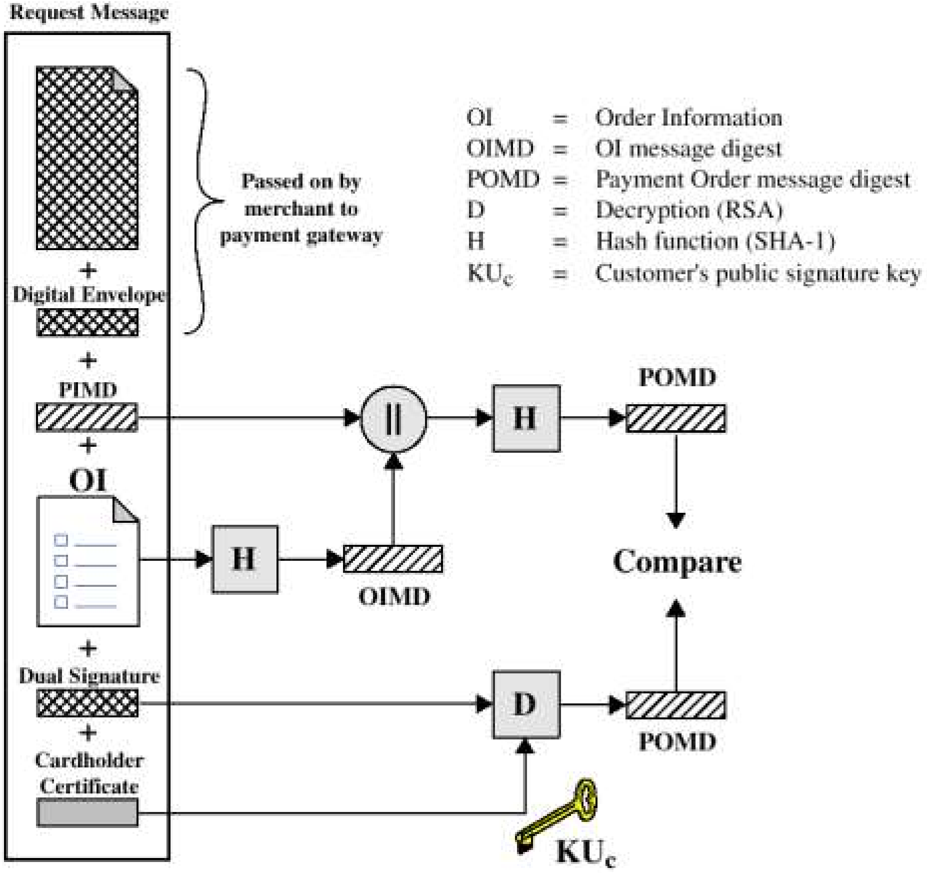
| | | | |
|---|---|---|---|
| PI | = Payment Information | PIMD | = PI message digest |
| OI | = Order Information | OIMD | = OI message digest |
| H | = Hash function (SHA-1) | POMD | = Payment Order message digest |
| \|\| | = Concatenation | E | = Encryption (RSA) |
| | | $KR_c$ | = Customer's private signature key |

# Payment processing



Cardholder sends Purchase Request

# Payment processing



Merchant Verifies Customer Purchase Request

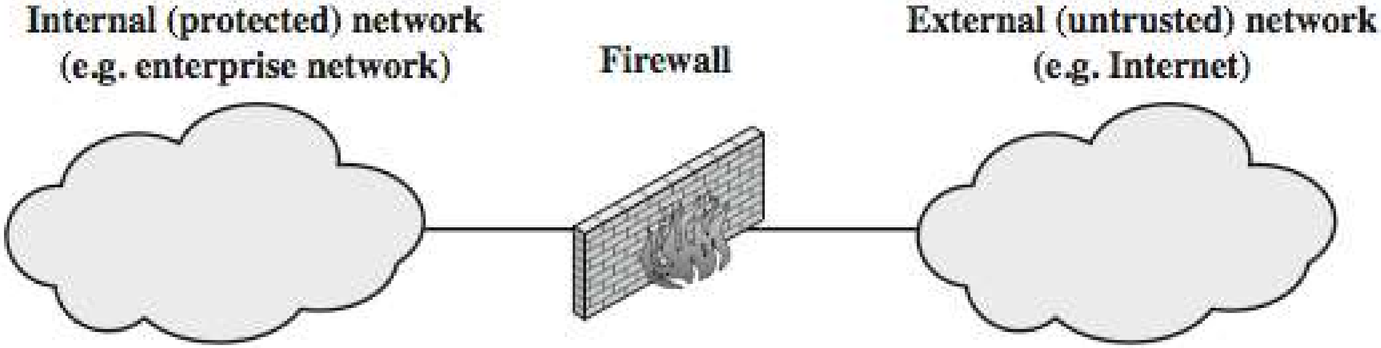
# Payment processing

- Payment Authorization:
  - Authorization Request
  - Authorization Response
- Payment Capture:
  - Capture Request
  - Capture Response

# What is a Firewall?

- a **choke point** of control and monitoring
- interconnects networks with differing trust
- imposes restrictions on network services
  - only authorized traffic is allowed
- auditing and controlling access
  - can implement alarms for abnormal behavior
- provide NAT & usage monitoring
- implement VPNs using IPSec
- must be immune to penetration

# What is a Firewall?


Internal (protected) network
(e.g. enterprise network)
Firewall
External (untrusted) network
(e.g. Internet)

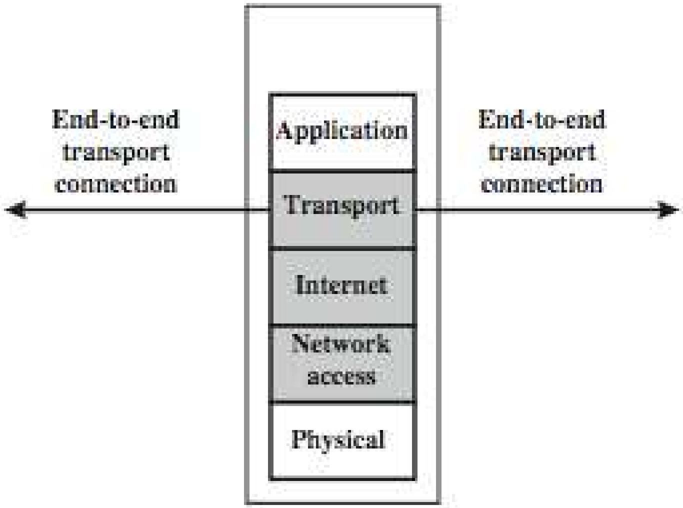
# Firewall Limitations

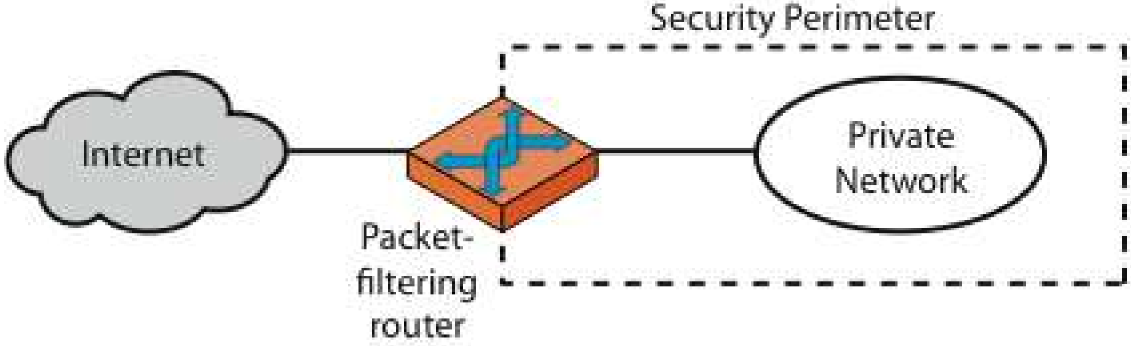
- cannot protect from attacks bypassing it
  - eg sneaker net, utility modems, trusted organisations, trusted services (eg SSL/SSH)
- cannot protect against internal threats
  - eg disgruntled or colluding employees
- cannot protect against access via WLAN
  - if improperly secured against external use
- cannot protect against malware imported via laptop, PDA, storage infected outside

# Firewalls – Packet Filters

- simplest, fastest firewall component
- foundation of any firewall system
- examine each IP packet (no context) and permit or deny according to rules
- hence restrict access to services (ports)
- possible default policies

# Firewalls – Packet Filters


End-to-end transport connection
Application
Transport
Internet
Network access
Physical
End-to-end transport connection



(a) Packet-filtering router

# Firewalls – Packet Filters

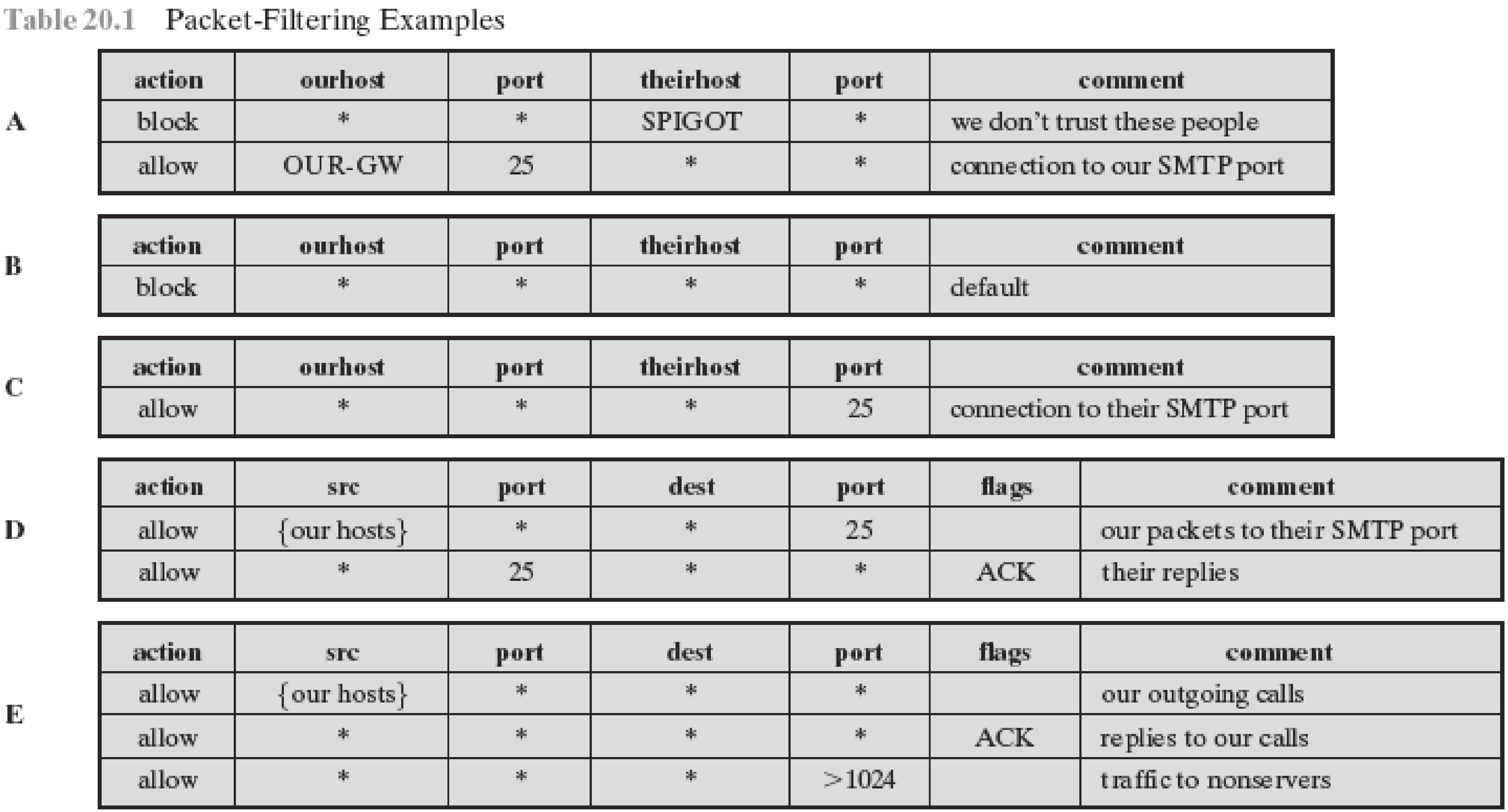


Table 20.1 Packet-Filtering Examples

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 nonservers |

# Attacks on Packet Filters

- IP address spoofing
  - fake source address to be trusted
  - add filters on router to block
- source routing attacks
  - attacker sets a route other than default
  - block source routed packets
- tiny fragment attacks
  - split header info over several tiny packets
  - either discard or reassemble before check

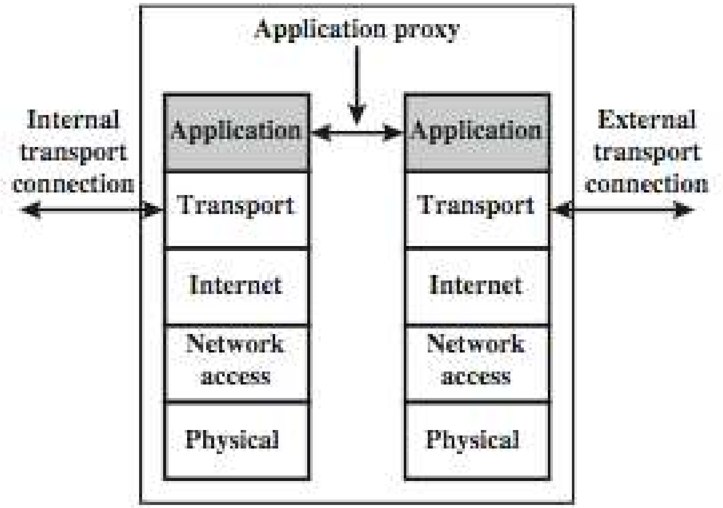
# Firewalls – Stateful Packet Filters

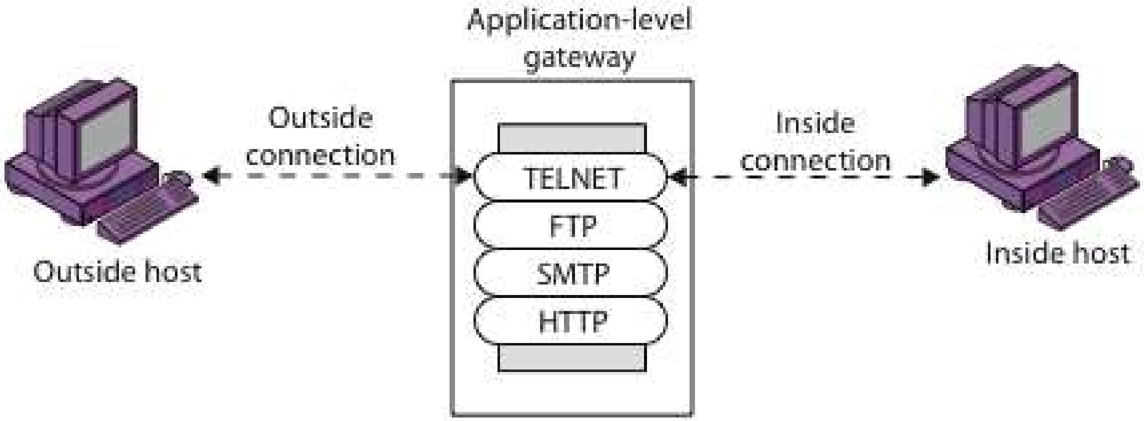
- traditional packet filters do not examine higher layer context
  - ie matching return packets with outgoing flow
- stateful packet filters address this need
- they examine each IP packet in context
  - keep track of client-server sessions
  - check each packet validly belongs to one
- hence are better able to detect bogus packets out of context
- may even inspect limited application data

# Firewalls - Application Level Gateway (or Proxy)

- have application specific gateway / proxy
- has full access to protocol
  - user requests service from proxy
  - proxy validates request as legal
  - then actions request and returns result to user
  - can log / audit traffic at application level
- need separate proxies for each service
  - some services naturally support proxying
  - others are more problematic

# Firewalls - Application Level Gateway (or Proxy)



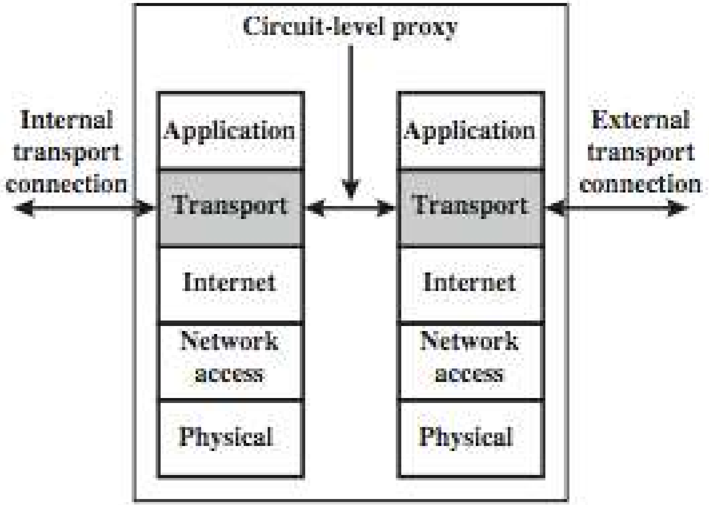


(b) Application-level gateway

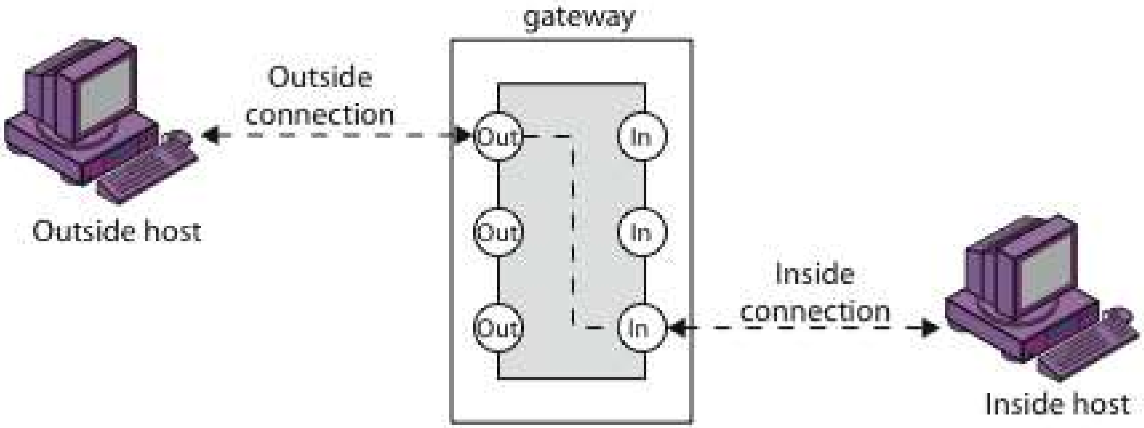
# Firewalls - Circuit Level Gateway

- relays two TCP connections
- imposes security by limiting which such connections are allowed
- once created usually relays traffic without examining contents
- typically used when trust internal users by allowing general outbound connections

# Firewalls - Circuit Level Gateway



(e) Circuit-level proxy firewall



(c) Circuit-level gateway

# Bastion Host

- highly secure host system
- runs circuit / application level gateways
- or provides externally accessible services
- potentially exposed to "hostile" elements
- hence is secured to withstand this
  - hardened O/S, essential services, extra auth
  - proxies small, secure, independent, non-privileged
- may support 2 or more net connections
- may be trusted to enforce policy of trusted separation between these net connections

# Encrypted Tunnels

- In computer networks, an encrypted tunneling protocol allows a network user to access or provide a network service that the underlying network does not support or provide directly
- One important use of a tunneling protocol is to allow a foreign protocol to run over a network that does not support that particular protocol

- A Secure Shell (SSH) tunnel consists of an encrypted tunnel created through an SSH protocol connection.
- Users may set up SSH tunnels to transfer unencrypted traffic over a network through an encrypted channel.

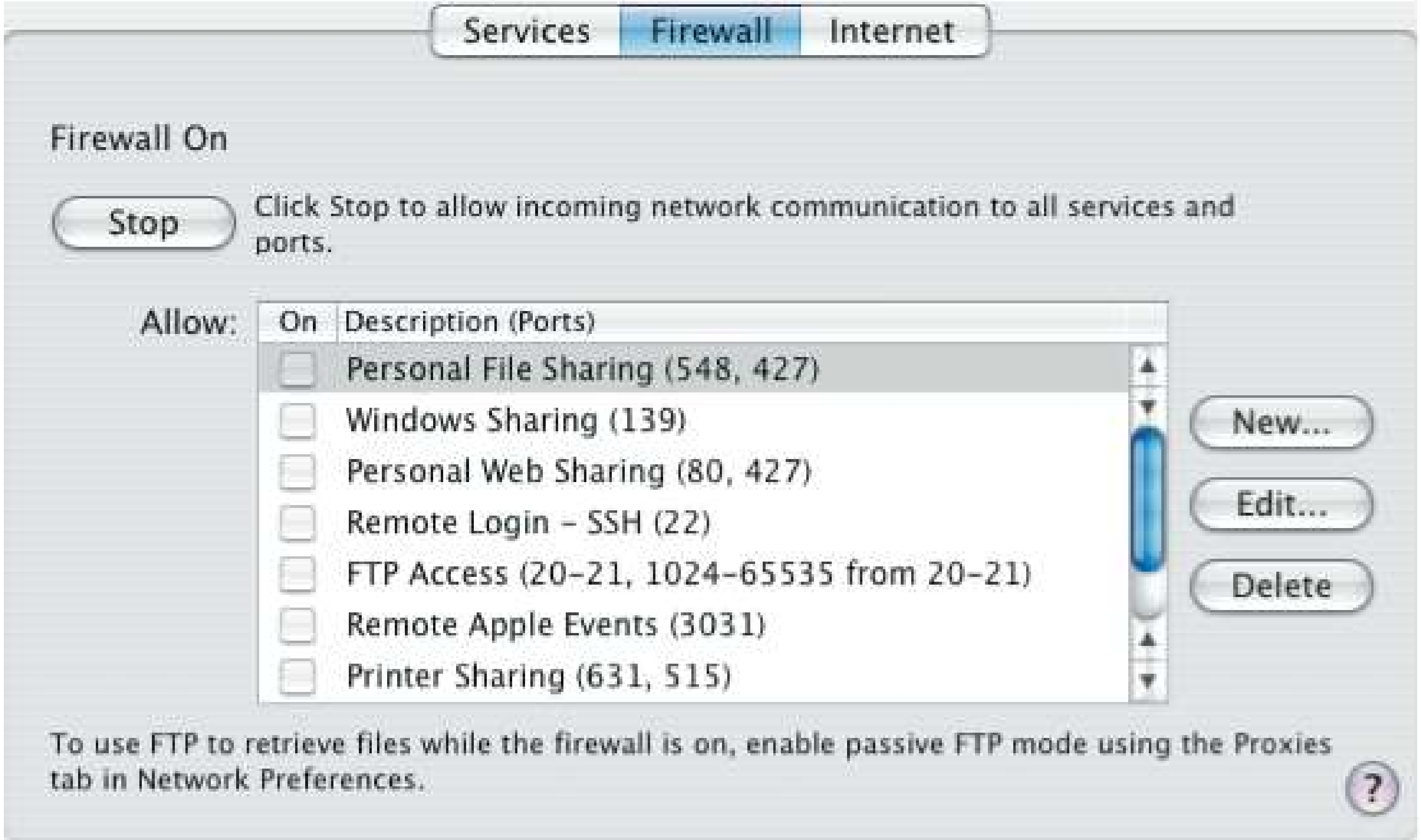
# Host-Based Firewalls

- s/w module used to secure individual host
  - available in many operating systems
  - or can be provided as an add-on package
- often used on servers
- advantages:
  - can tailor filtering rules to host environment
  - protection is provided independent of topology
  - provides an additional layer of protection

# Personal Firewalls

- controls traffic between PC/workstation and Internet or enterprise network
- a software module on personal computer
- or in home/office DSL/cable/ISP router
- typically much less complex than other firewall types
- primary role to deny unauthorized remote access to the computer
- and monitor outgoing activity for malware

# Personal Firewalls



Firewall On

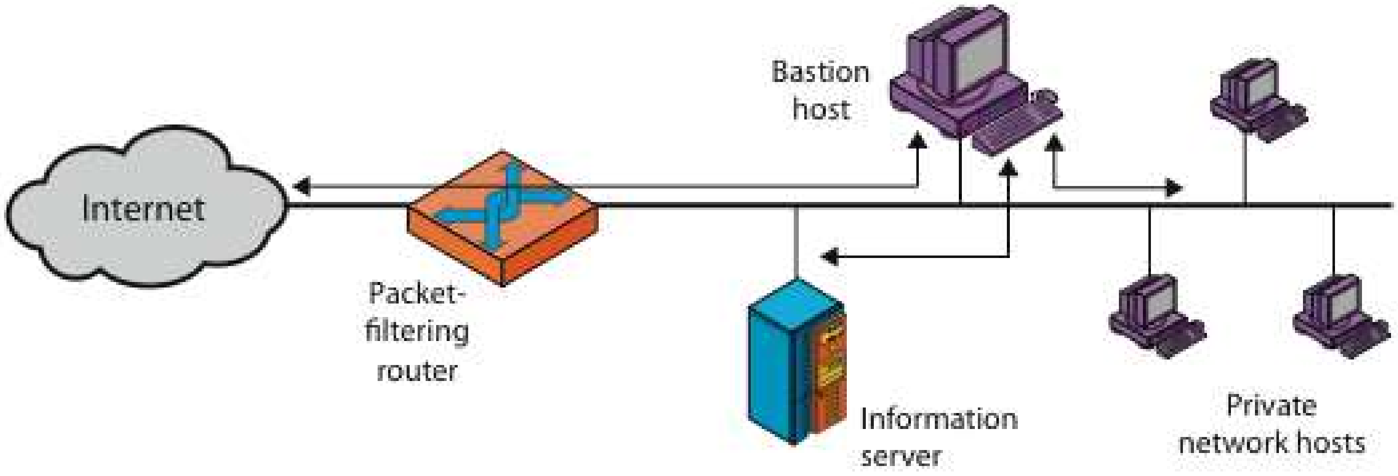
Click Stop to allow incoming network communication to all services and ports.

Allow:

| On | Description (Ports) |
|---|---|
| □ | Personal File Sharing (548, 427) |
| □ | Windows Sharing (139) |
| □ | Personal Web Sharing (80, 427) |
| □ | Remote Login – SSH (22) |
| □ | FTP Access (20-21, 1024-65535 from 20-21) |
| □ | Remote Apple Events (3031) |
| □ | Printer Sharing (631, 515) |

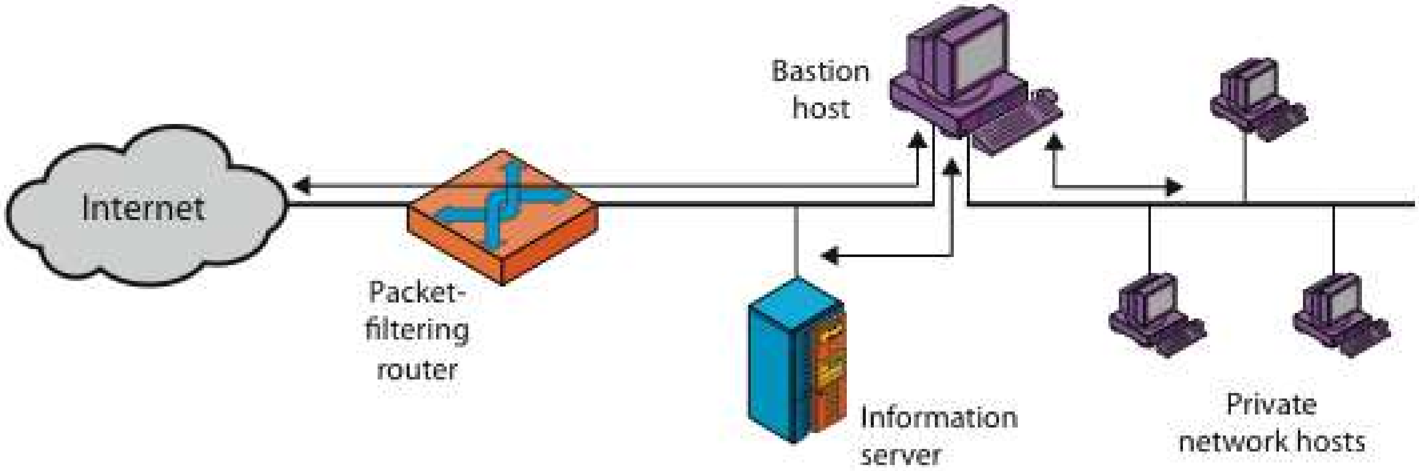
To use FTP to retrieve files while the firewall is on, enable passive FTP mode using the Proxies tab in Network Preferences.

# Firewall Configurations



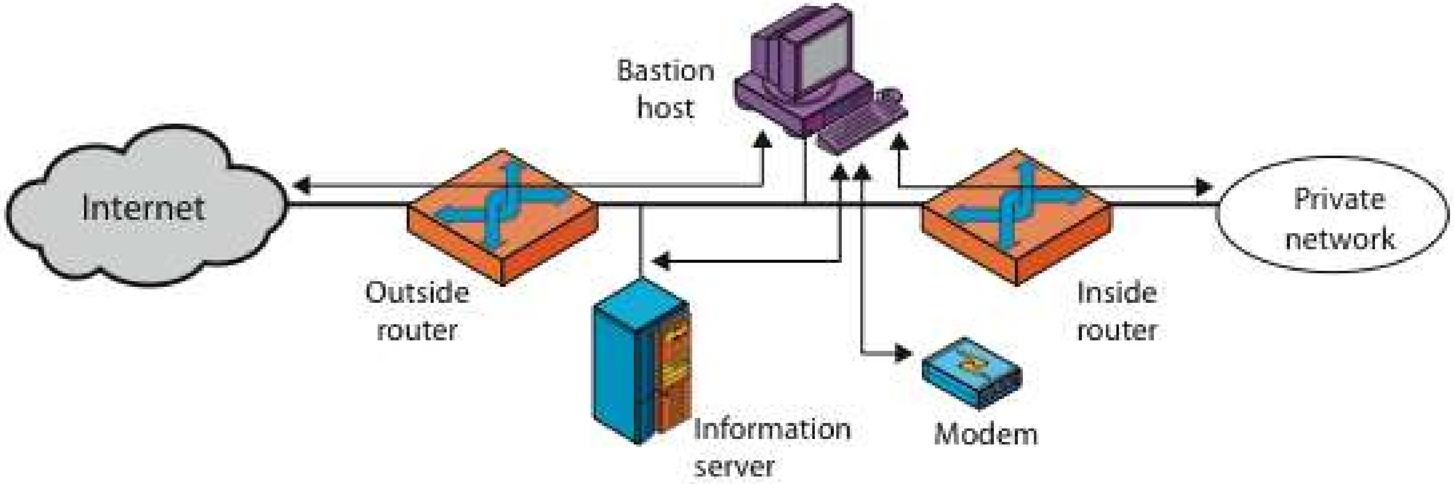
(a) Screened host firewall system (single-homed bastion host)

# Firewall Configurations



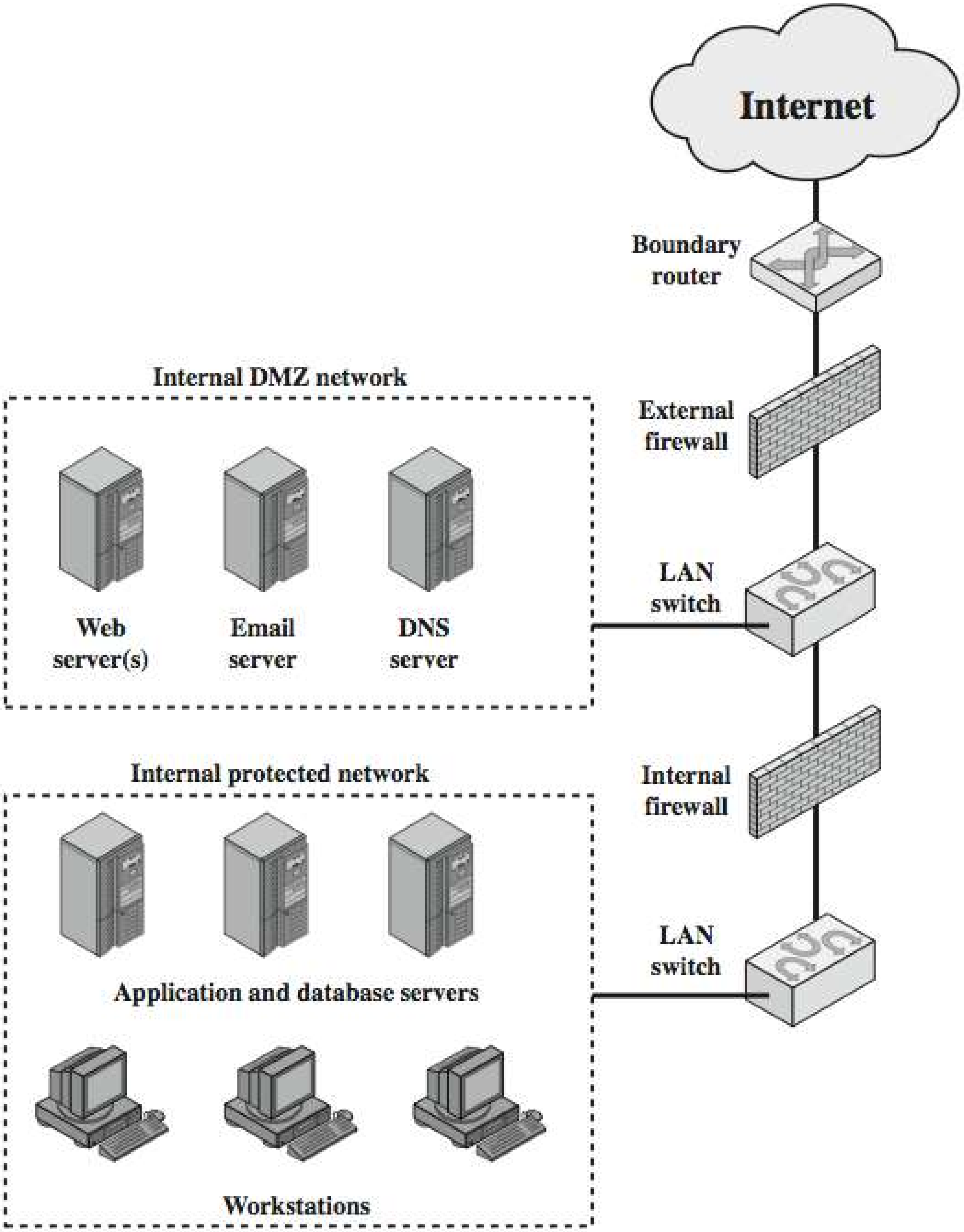
(b) Screened host firewall system (dual-homed bastion host)

# Firewall Configurations

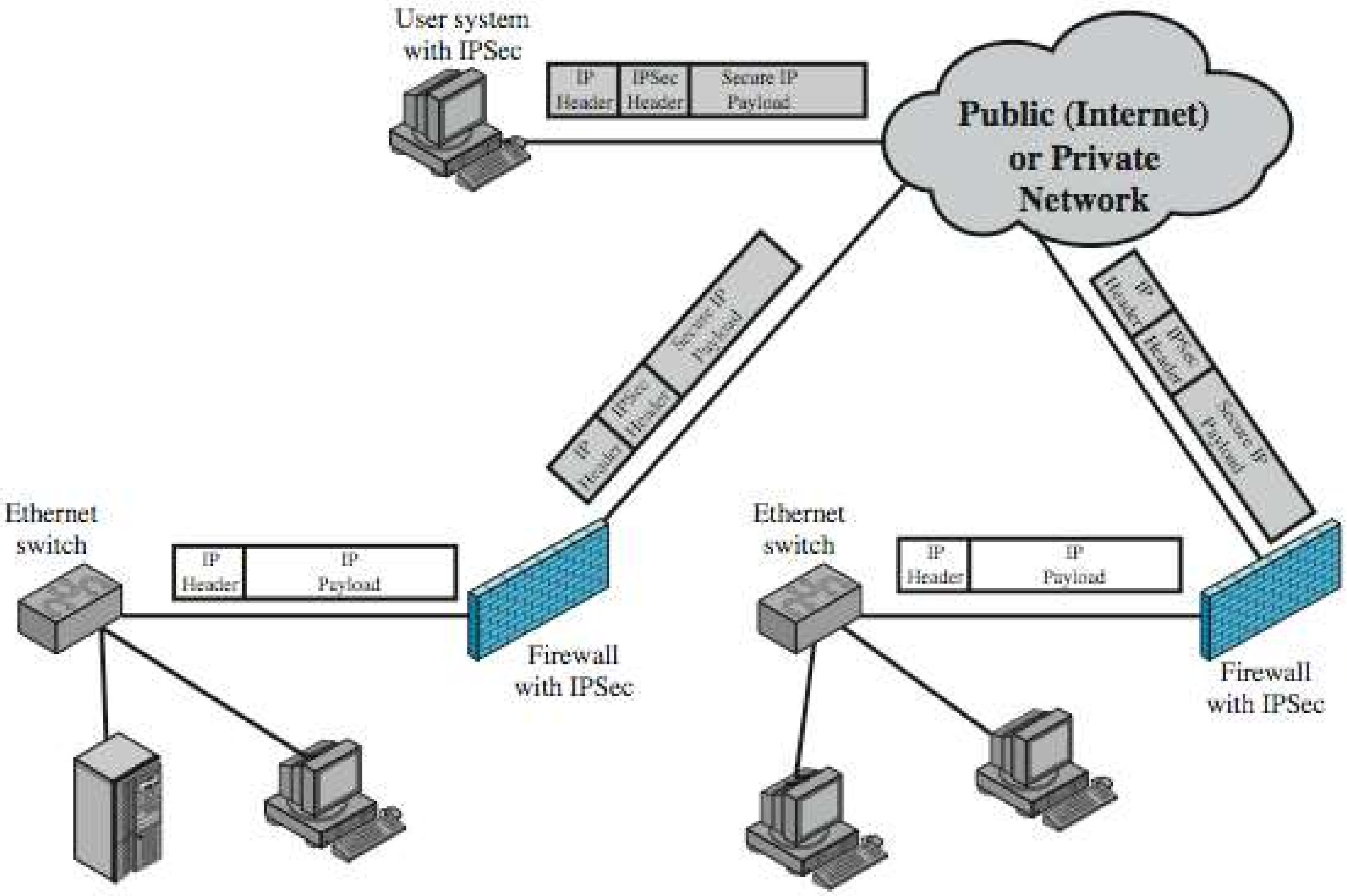


(c) Screened-subnet firewall system

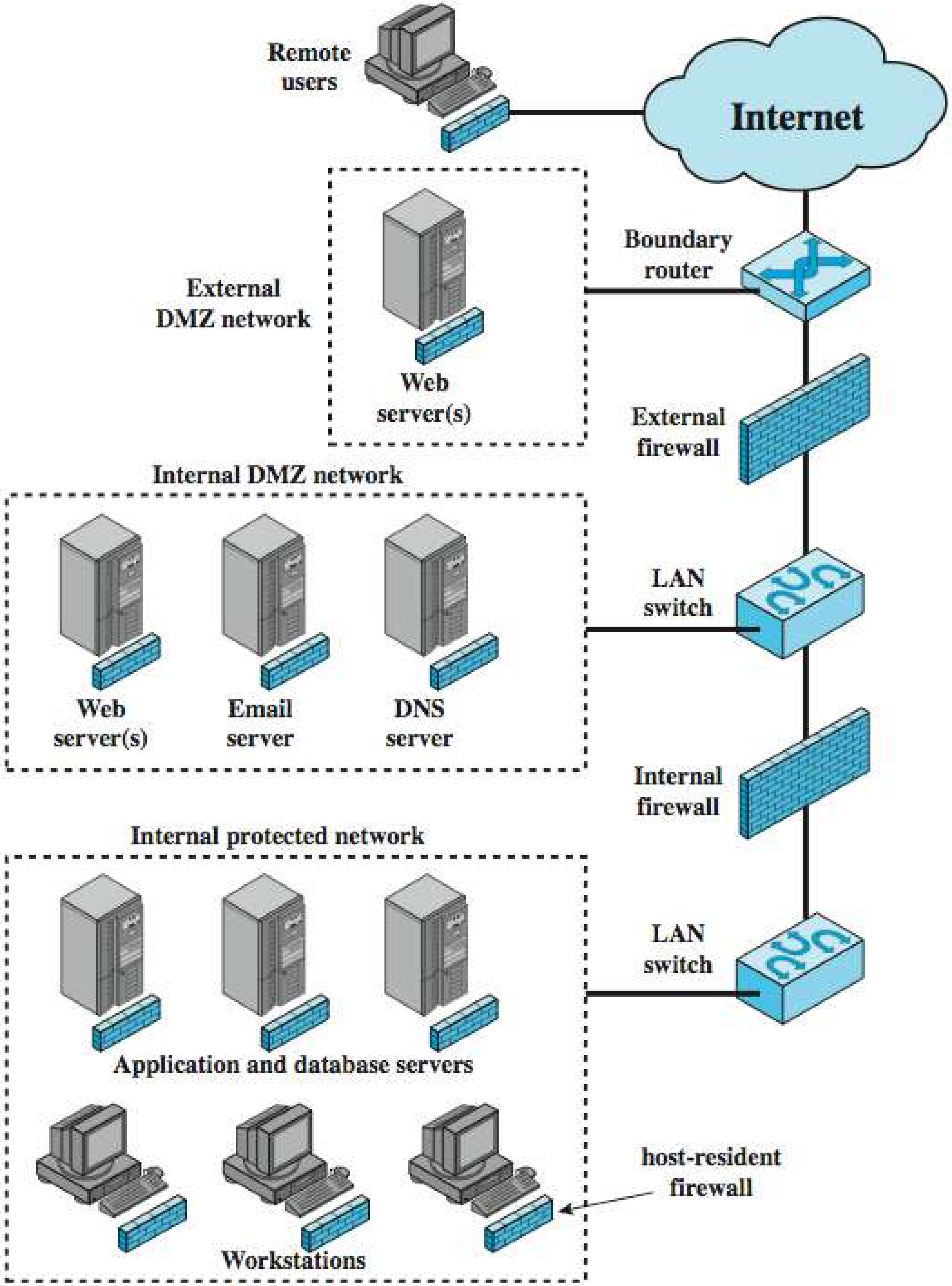
# DMZ Networks



# Virtual Private Networks



# Distributed Firewalls


Remote users
Internet
External DMZ network
Web server(s)
Boundary router
External firewall
Internal DMZ network
Web server(s)
Email server
DNS server
LAN switch
Internal firewall
Internal protected network
LAN switch
Application and database servers
host-resident firewall
Workstations