

IT352-Class-1

General Framework

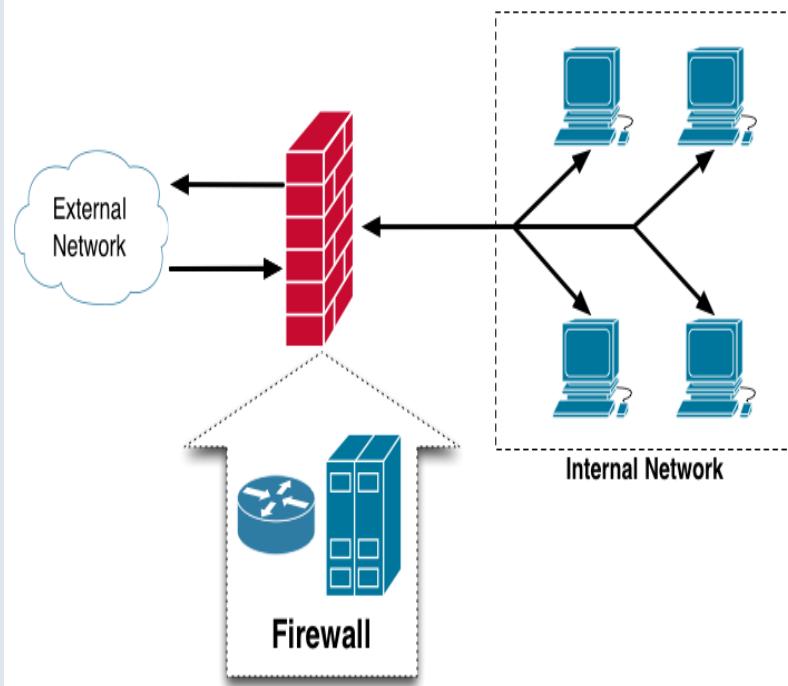
- What is a firewall?
 - A hardware device, a software package, or a combination of both
 - A barrier between the Internet and an edge network (internal network)
 - A mechanism to filter Incoming (ingress) and outgoing (egress) packets.
 - May be hardware and/or software
 - Hardware is faster but can be difficult to update
 - Software is slower but easier to update

A firewall is any device that prevents a specific types of information from moving between the outside world (untrusted network) and the inside world (trusted network).

The firewall may be

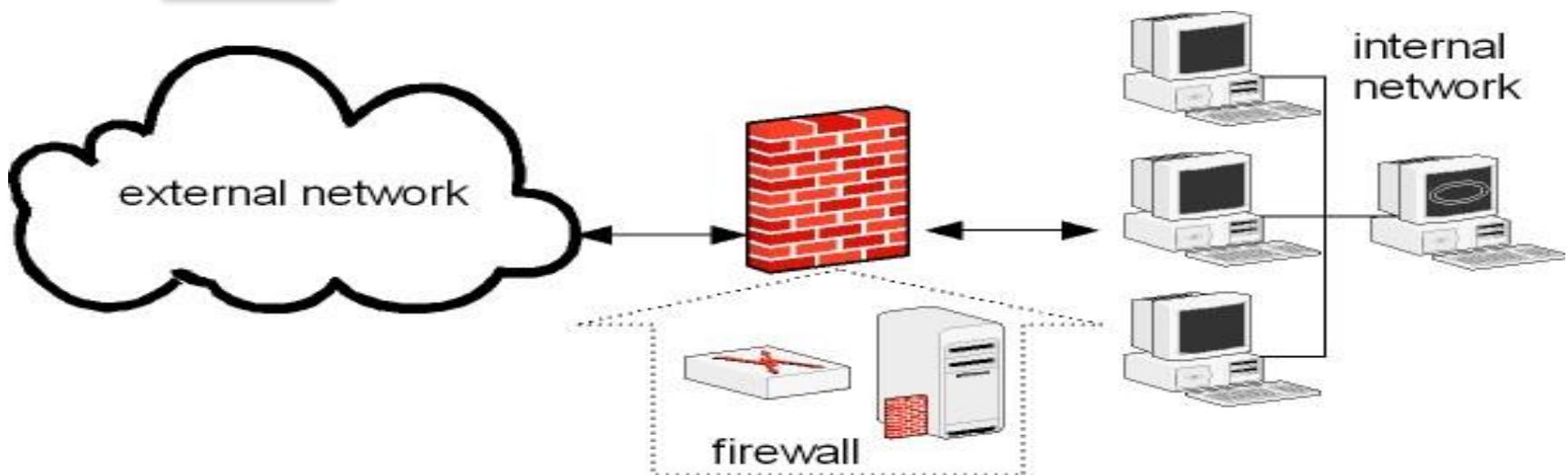
- a separate computer system
- a service running on an existing router or server
- a separate network containing a number of supporting devices

General Framework



Firewall can be categorized by processing mode, development era or structure.

There are five major processing mode categories of firewall: **Packet Filtering Firewall**, **Application Gateways**, **Circuit Gateway**, **MAC layer Gateways** and **Hybrid Firewalls**.



Firewalls Categorized by Processing Modes

- Packet filtering
- Application gateways
- Circuit gateways
- MAC layer firewalls
- Hybrids

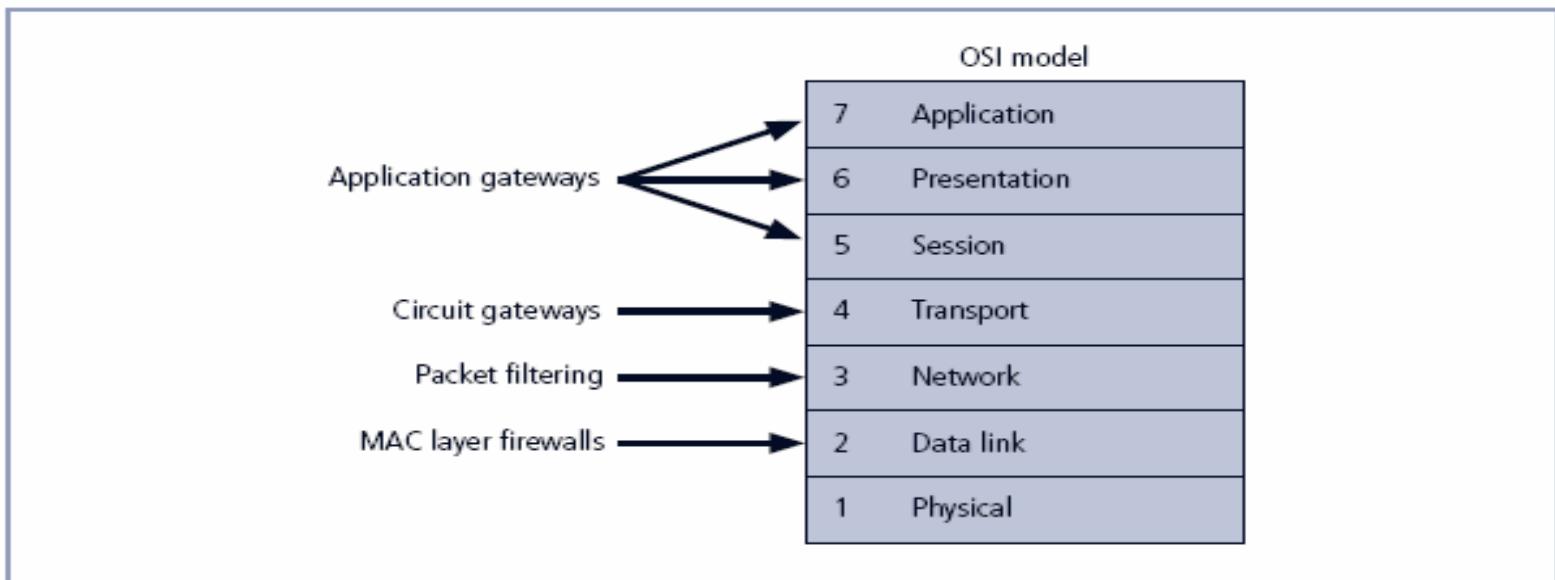
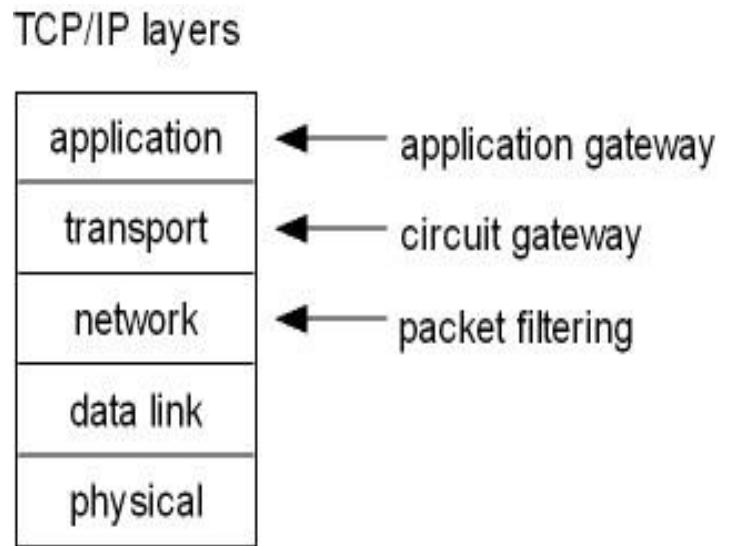


FIGURE 6-5 Firewall Types and the OSI Model

Packet Filtering

- Packet filtering firewalls examine header information of data packets
- Most often based on combination of:
 - Internet Protocol (IP) source and destination address
 - Direction (inbound or outbound)
 - Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) source and destination port requests
- Simple firewall models enforce rules designed to prohibit packets with certain addresses or partial addresses
- Three subsets of packet filtering firewalls:
 - **Static filtering:** requires that filtering rules governing how the firewall decides which packets are allowed and which are denied are developed and installed
 - **Dynamic filtering:** allows firewall to react to emergent event and update or create rules to deal with event
 - **Stateful inspection:** firewalls that keep track of each network connection between internal and external systems using a state table

Packet Filters Contd.

- Perform ingress (incoming) and egress (outgoing) filtering on packets
- Only inspect IP and TCP/UDP headers, **not the payloads**
- Can perform either stateless or stateful filtering
 - **Stateless filtering:** easy to implement but very simple
 - **Stateful filtering:** harder to implement but more powerful

Packet Filters Contd.

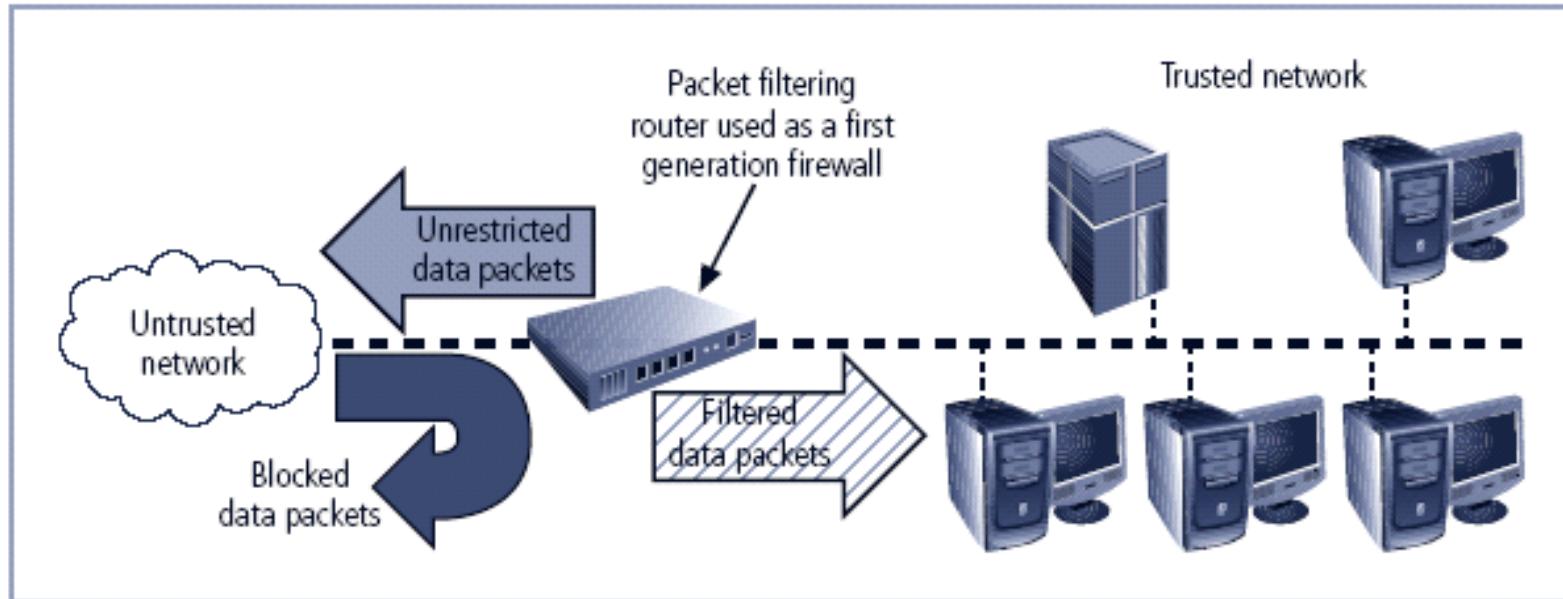


FIGURE 6-4 Packet Filtering Router

TABLE 6-1 Sample Firewall Rule and Format

Source Address	Destination Address	Service (HTTP, SMTP, FTP, Telnet)	Action (Allow or Deny)
172.16.x.x	10.10.x.x	Any	Deny
192.168.x.x	10.10.10.25	HTTP	Allow
192.168.0.1	10.10.10.10	FTP	Allow

Packet Filters Contd.

- There are three subsets of packet filtering firewalls:
 - Static Filtering
 - Dynamic Filtering
 - Stateful Inspection

Static Filtering

- Static Filtering requires that the filtering rules be developed and installed with the firewall.
- The rules are created and sequenced either by a person directly editing the rule set or by a person using a programmable interface to specify the rules and the sequence.
- This type of filtering is common in network routers and gateways

Packet Filters Contd.

Dynamic Filtering

Dynamic filtering firewall can react to an emergent event and update or create rules to deals with that event.

Dynamic packet filtering firewall allows only a particular packet with a particular source, destination and port address to enter through firewall.

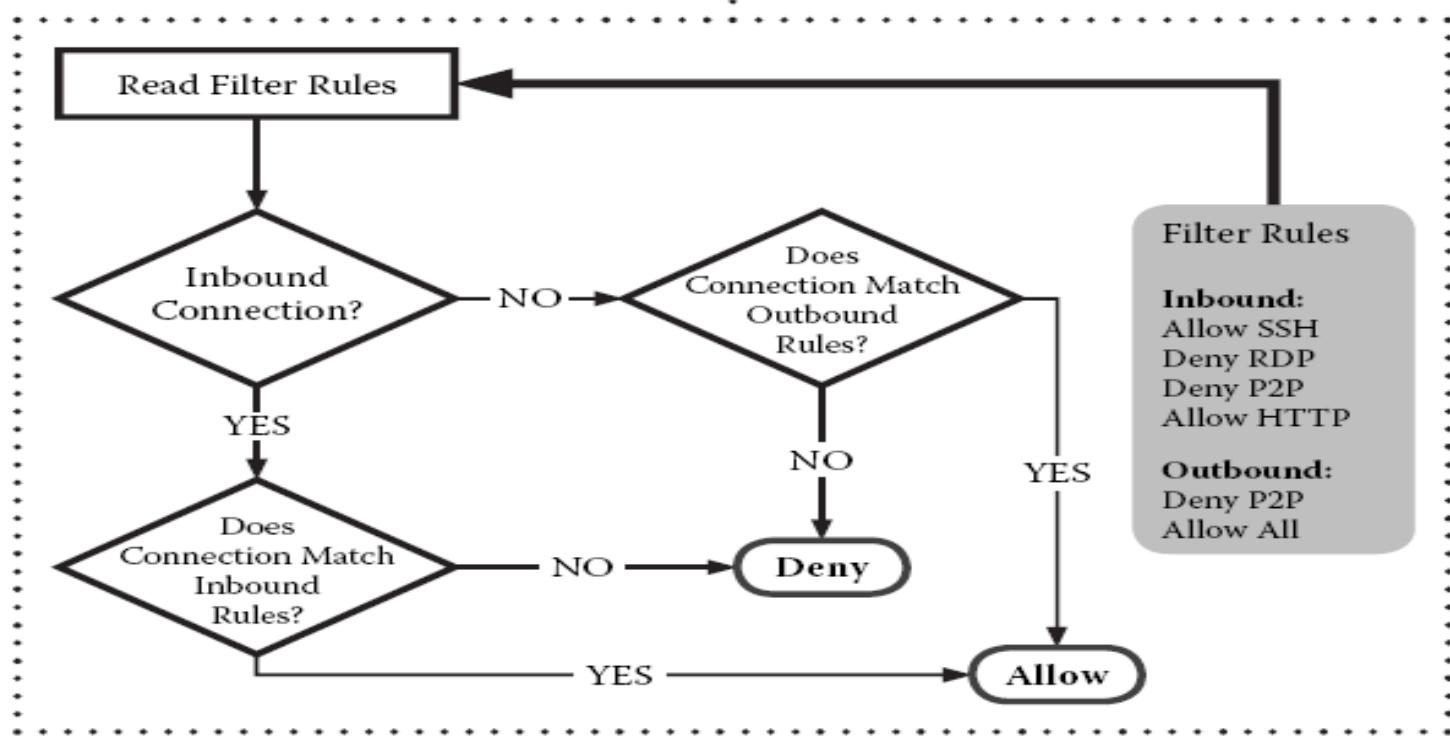
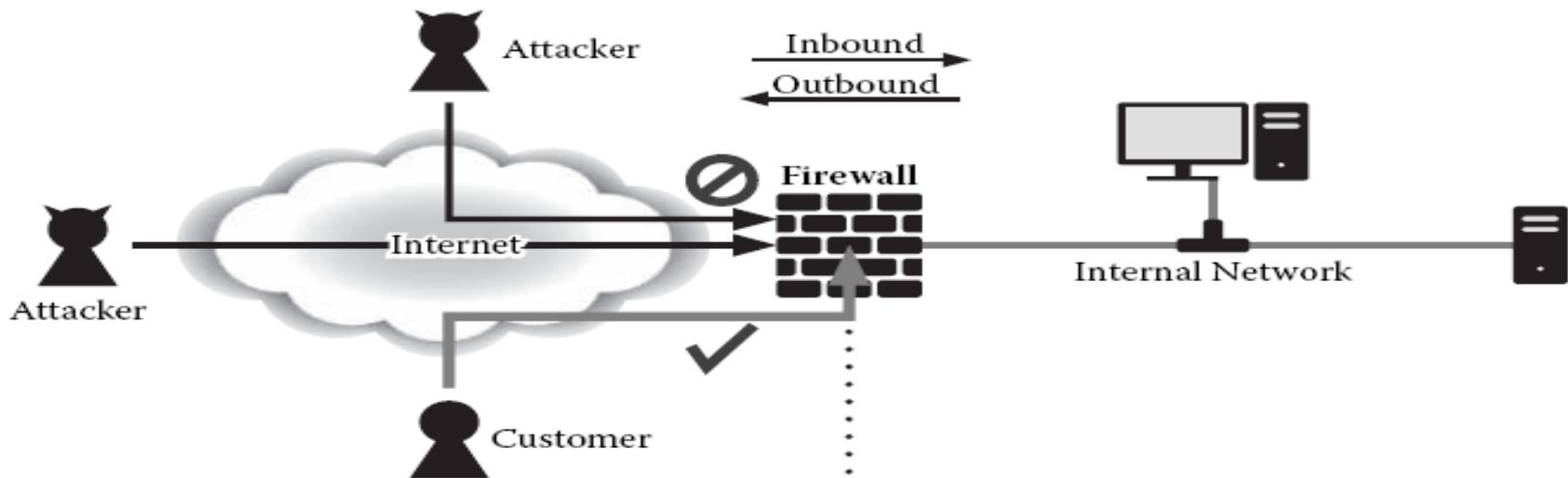
It does this by opening and closing doors in the firewall based on the information contained in the packet header which makes dynamic packet filters an intermediate form between traditional static packet filters and application proxies

Packet Filters Contd.

Stateful Inspection Firewall

- Also called as stateful firewall.
- It keeps track of each network connection between internal and external system using a state table.
- A state table tracks the state and context of each packet in the conversation by recording which station sent what packet and when.
- Whereas simple packet filtering firewalls only allow or deny certain packet based on their address, a stateful firewall can expedite informing packets that are responses to internal requests.
- If the stateful firewall receives an incoming packet that it cannot match in its state table, it refer to its ACL to determine whether to allow the packet to pass.
- The primary disadvantage of this type of firewall is the additional processing required to manage and verify packets against the statetable

Basic Firewall Network



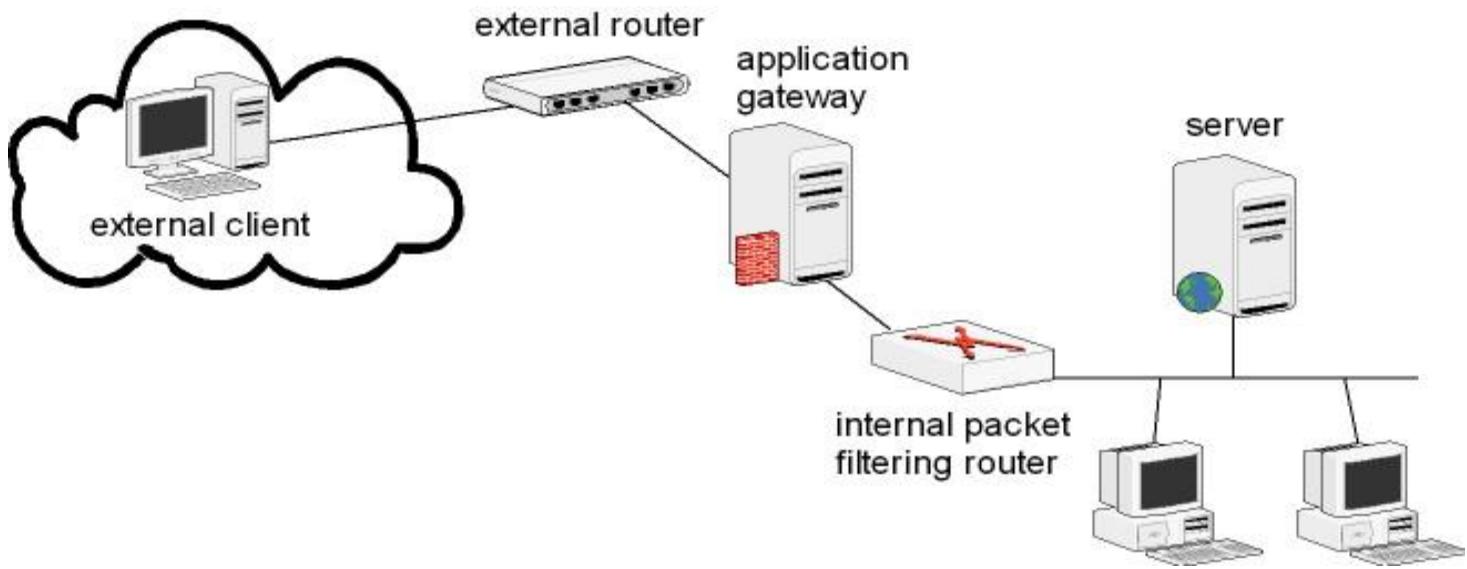
Application Gateways

- Also called application-level gateway or proxy server
- Act like a proxy for internal hosts, processing service request from external clients.
- Perform deep packet inspection on all packet
 - Inspect application program formats
 - Apply rules based on the payload
 - Have the ability to detect malicious and suspicious packets
- Extremely resource intensive

Application Gateways

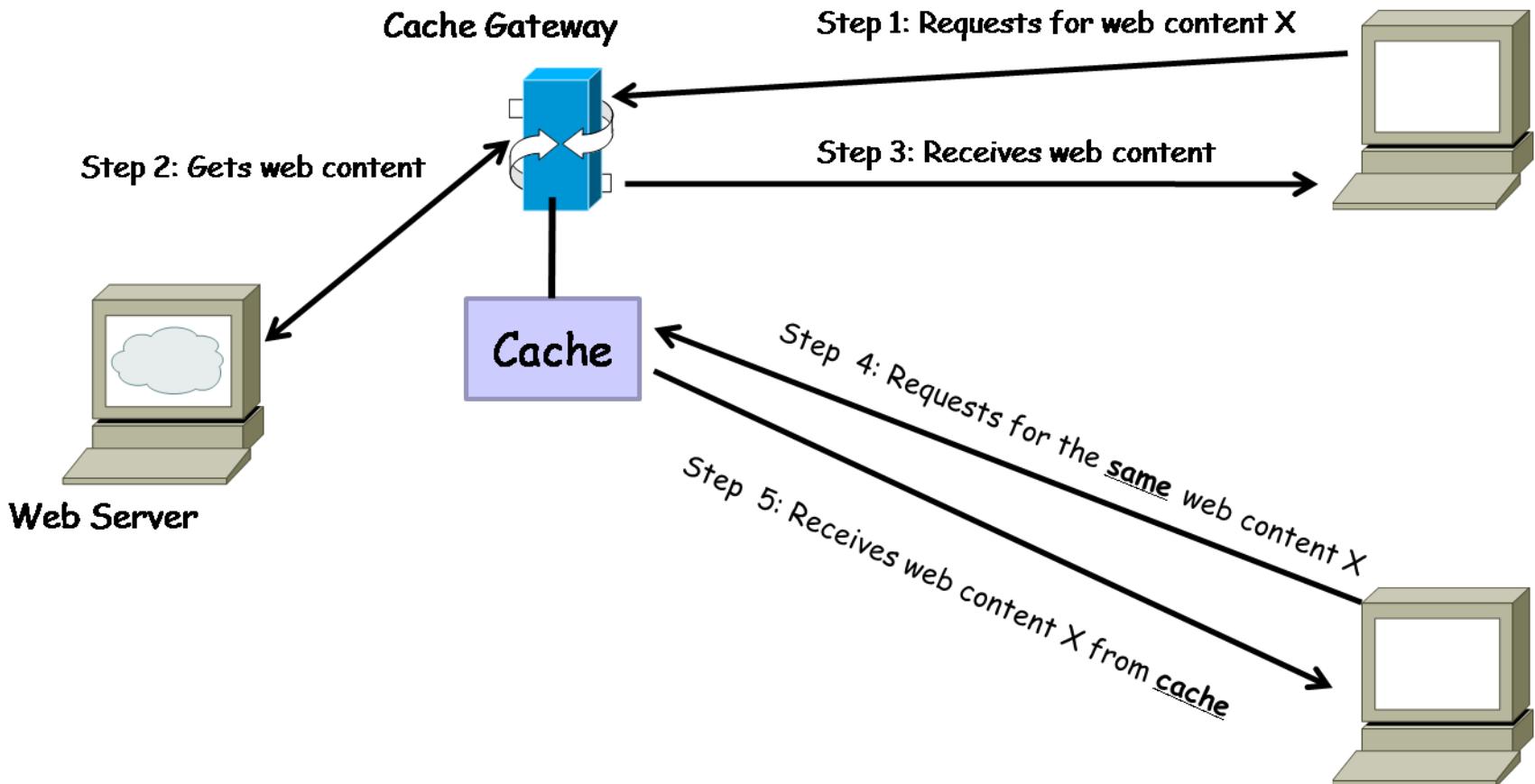
- Application Gateway also know as **Application Level Gateway** or **Application Firewall**.
- Frequently installed on a dedicated computer, separate from the filtering router, but it is commonly used in conjunction with a filtering router.
- **It is also known as a proxy server**, since it runs special software that acts as a proxy for a service request.
- Proxy server is placed in an unsecured area of the netwrok or in the demilitarized zone (DMZ) – an intermediate area between a trusted network and untrusted netwok.
- Additional filtering routers can be implemented behind the proxy server, further protecting internal systems
- The primary disadvantage of application-level firewall is that **they are designed for one or a few specific protocol** and **cannot easily be reconfigured to protect against attack on other protocols**.

Application Gateways



Place a router behind the gateway to protect connections between the gateway and the internal hosts

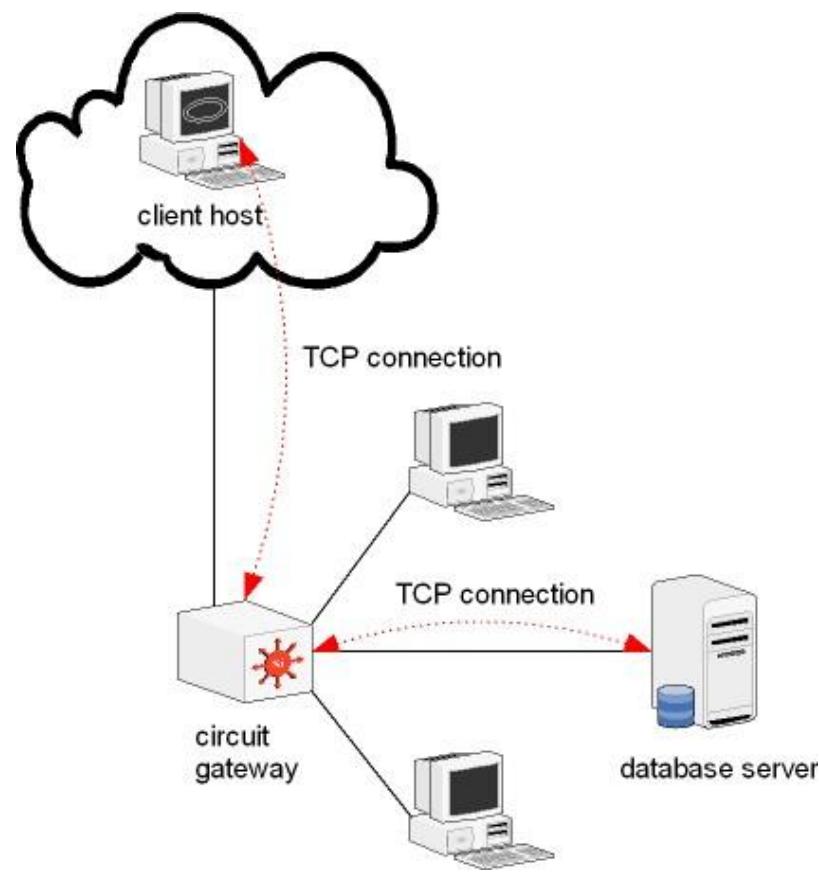
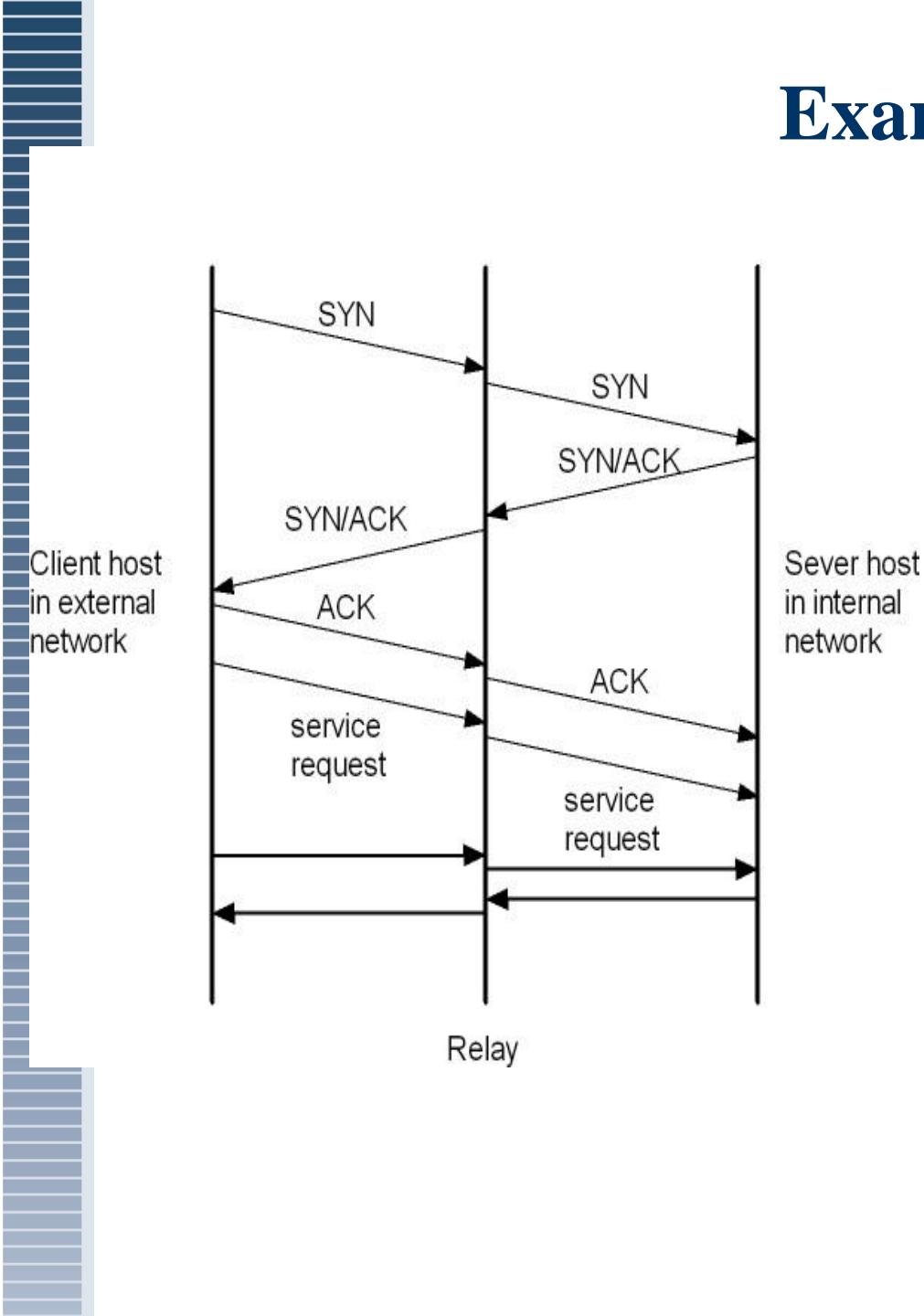
Cache Gateway



Circuit Gateways

- Circuit gateway firewall operates at transport layer
- Like filtering firewalls, do not usually look at data traffic flowing between two networks, but prevent direct connections between one network and another
- Accomplished by creating tunnels connecting specific processes or systems on each side of the firewall, and allow only authorized traffic in the tunnels
- Does not permit end-to-end connections
 - Sets up two TCP connection (inner host to gateway, gateway to outer host)
 - Gateway relays segments from one connection to the other
 - Does not examine contents of segments
- Security function is to determine what connections are allowed
- Could be a standalone system or function performed by application-level gateway for some applications

Examples



MAC Layer Firewalls

- Designed to operate at the Media Access Control (MAC) layer of OSI network model.
- Able to consider specific host computer's identity in its filtering decisions.
- MAC addresses of specific host computers are linked to Access Control List (ACL) entries that identify specific types of packets that can be sent to each host; all other traffic is blocked

Hybrid Firewalls

- Combine elements of other types of firewalls; i.e., elements of **packet filtering and proxy services**, or of **packet filtering and circuit gateways**
- Actually consists of two separate firewall devices; each is a separate firewall system, but they are connected so that they work in tandem.
- An added advantage to the hybrid firewall approach is that it enables an organization to make a security improvement without completely replacing its existing firewalls

IT352-Class-2

Firewalls Categorized by Architectural Implementation

- Firewall devices can be configured in a number of network connection architectures
- Four common architectural implementations of firewalls:
 - Packet filtering routers
 - Screened host firewalls
 - Dual-homed firewalls
 - Screened subnet firewalls

Bastion Hosts

- Bastion hosts are computers with strong defensive mechanisms.
- They often serve as host computers for implementing Application Gateway, Circuit Gateway, and other types of firewall.
- Bastion hosts is operated on a trusted operating system that must not contain unnecessary functionalities or program. This measure helps to reduce error probabilities and makes its easier to conduct security checks.
- Only application program that are absolutely necessary e.g., SSH, DNS, SMTP and authentication program are installed on the bastion host.

Requirements

- Gateway software should be written using only small modules
- May provide user authentication at the network level
- Should be connected to the smallest possible number of internal hosts
- Extensive logs should be kept of all activity passing through the system
- If they are running on a single host, multiple gateways must operate independently
- Hosts should avoid writing data to their hard disks
- Gateways running on bastion hosts should not be given administration rights

Firewall Configuration

- Gateway running on a bastion host are often used with packet filters. Without loss of generality, assume that router has build-in packet filter.
- For the sake of convenience, use bastion host to represent a proxy server where bastion host may run several proxy servers simultaneously and independently.
- Several common firewall configuration
 - Single-Homed Bastion Host System (SHBH)
 - Dual-Homed Bastion Host System (DHBH)
 - Screened Subnets

Packet Filtering Routers

- Most organizations with Internet connection have a router serving as interface to Internet
- Many of these routers can be configured to reject packets that organization does not allow into network
- Drawbacks include a lack of auditing and strong authentication

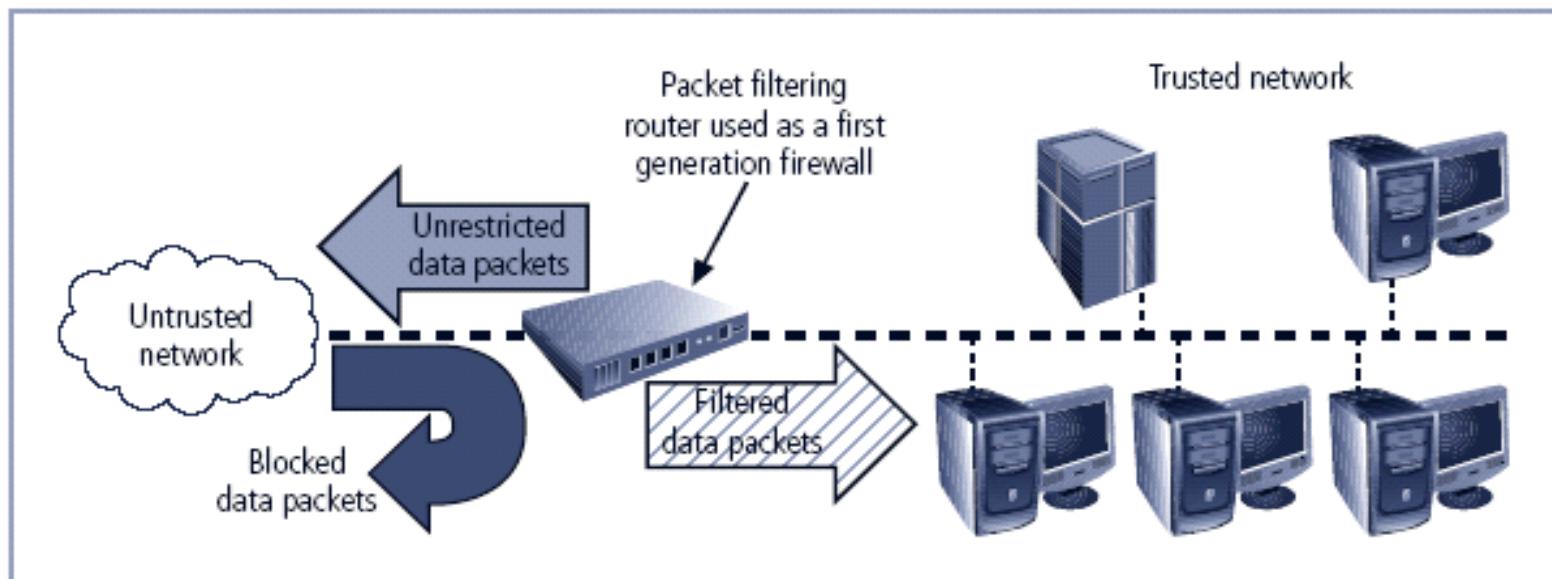
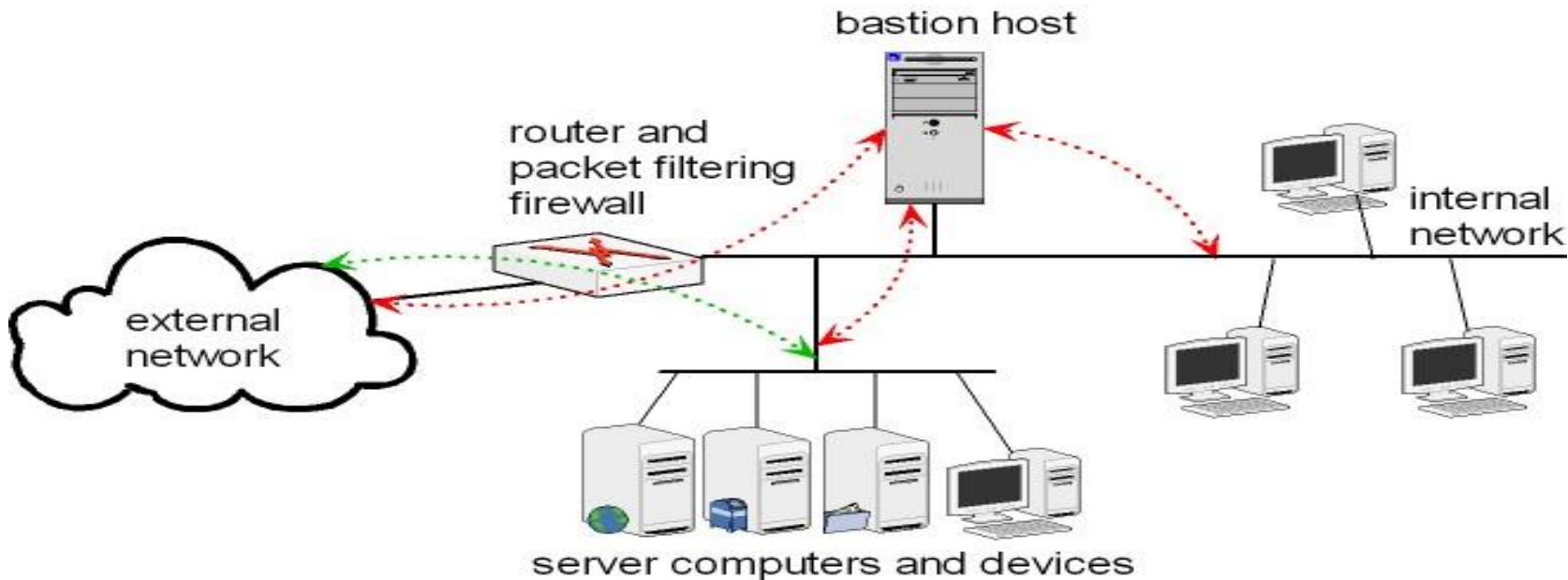


FIGURE 6-4 Packet Filtering Router

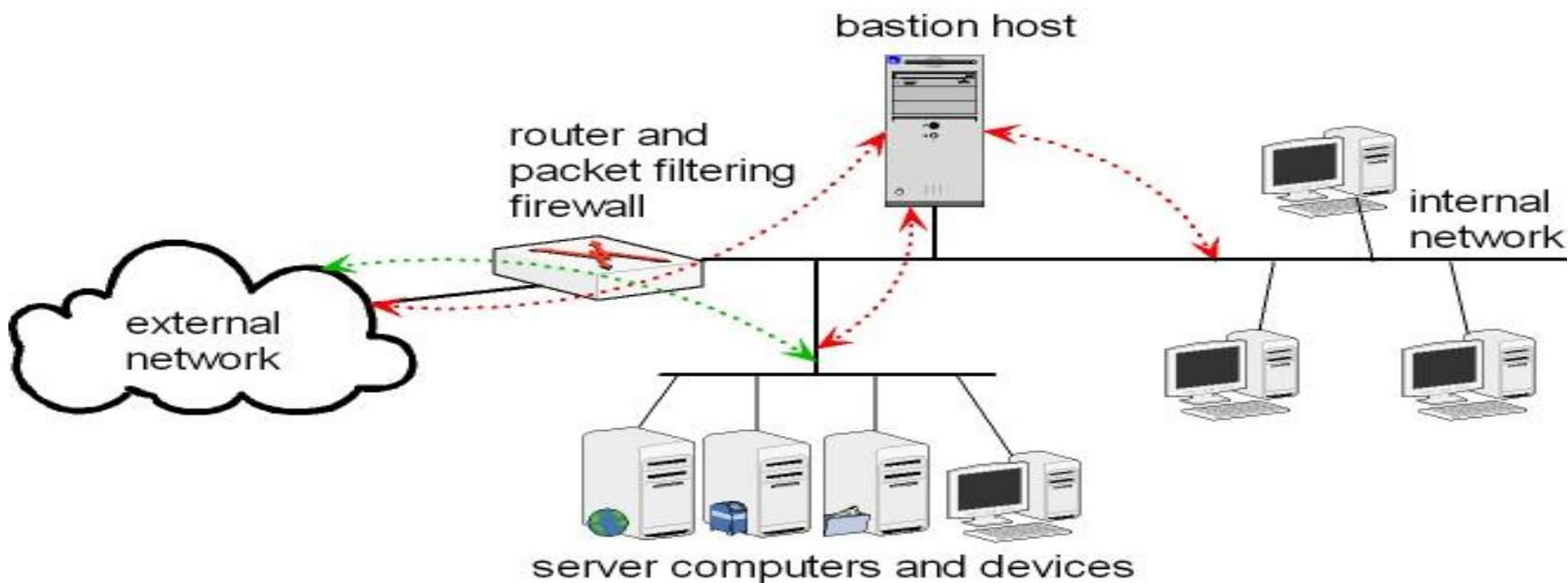
Single-Homed Bastion Host System

- A single-homed bastion host system consists of a packet filtering router and bastion host, where the router connects the internal network to external network and the bastion host is inside the internal network.
- The router announces to the public the IP address and the port numbers of internal server computers.
- Router does not forward the ingress packets directly to server computers. Instead, the router inspects an ingress packet. If the packet passes the inspection, the router passes it to the bastion host.



Single-Homed Bastion Host System

- In an SHBH system, if an attacker compromises the packet filtering router, attacker can modify ACL rules to bypass the bastion host. That is, malice could modify the ACL rules to forward packets directly to internal hosts, making the bastion host becomes nothing but an ornament. This problem can be solved using a dual-homed bastion host.

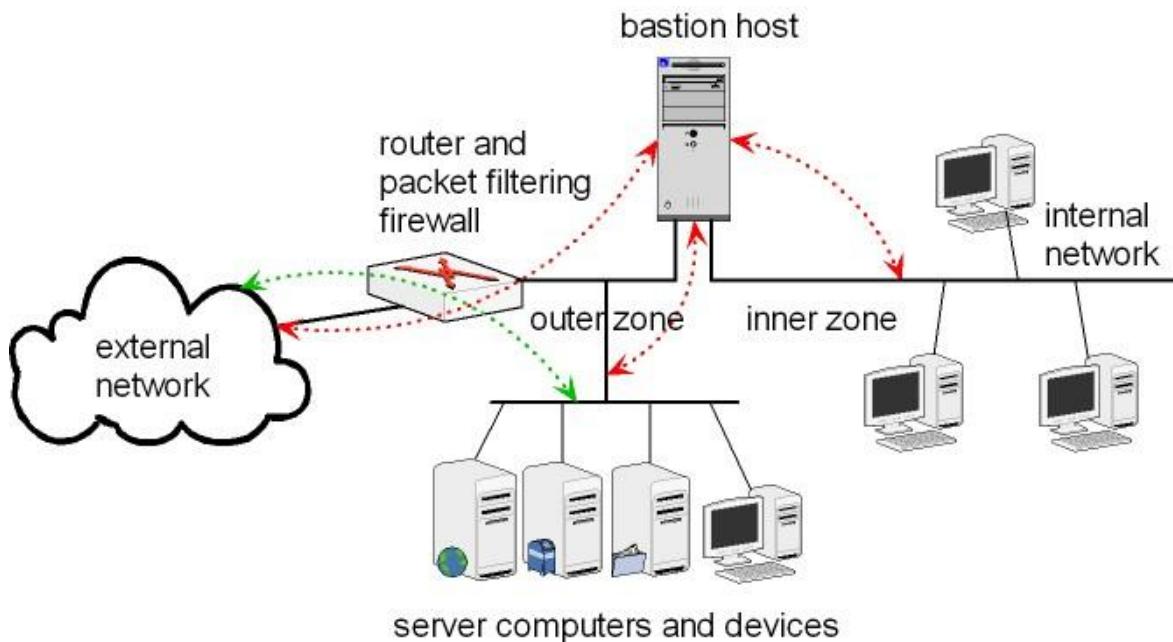


Dual-Homed Bastion Host System

- Bastion host contains two network interface cards (NICs): one connected to external network, one connected to internal network
- Implementation of this architecture often makes use of network address translation (NAT), creating another barrier to intrusion from external attackers

Dual-Homed Bastion Host System

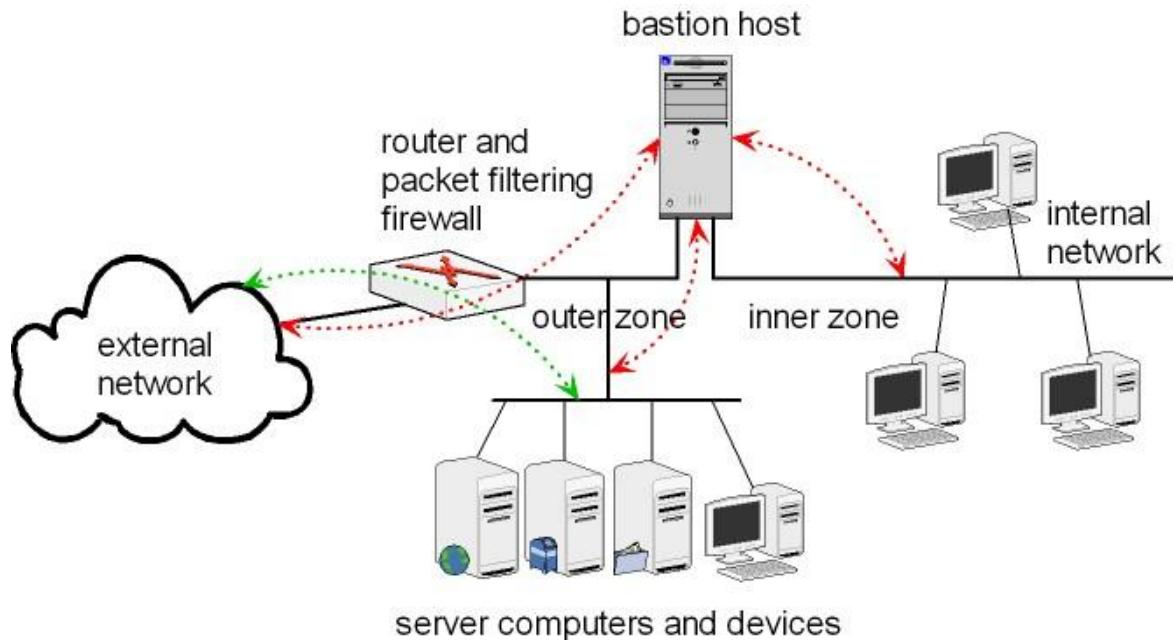
- A dual-homed bastion host network divides the internal network into two zones. These two zones are referred to as the inner zone and the outer zone. The inner zone is also referred to as the private zone.
- The IP address of the host computer in the private zone are not reachable from the external network. The IP addresses of the host computers in the outer zone may be directly reachable from the Internet.
- In particular, the router is placed between the external network and the outer zone, and between the external network and the bastion host.



Dual-Homed Bastion Host System Contd.

Unlike, the SHBH, the inner zone in DHBH is connected to the bastion host only. Thus, host computers inner zone are protected by both the bastion host and the packet filtering router.

Similar to SHBH system, a DHBH allows the server computers in the outer zone to communicate to the internet without going through the bastion host. In other words, the ACL in the router allows each inbound packet to pass through its source address is allowed and destination IP address and port number match with the IP address of a server computer and an open port of the server.



Dual-Homed Bastion Host System Contd.

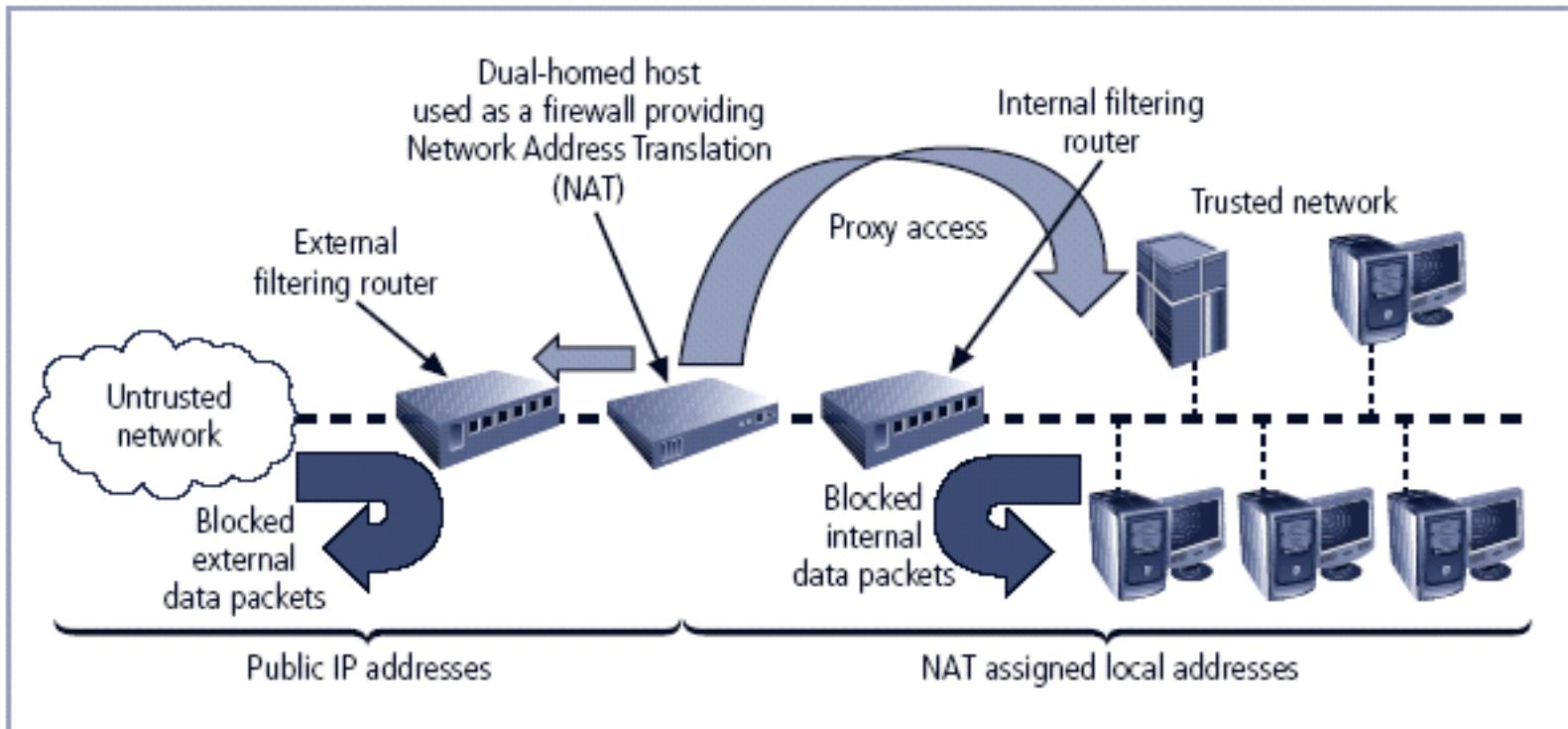
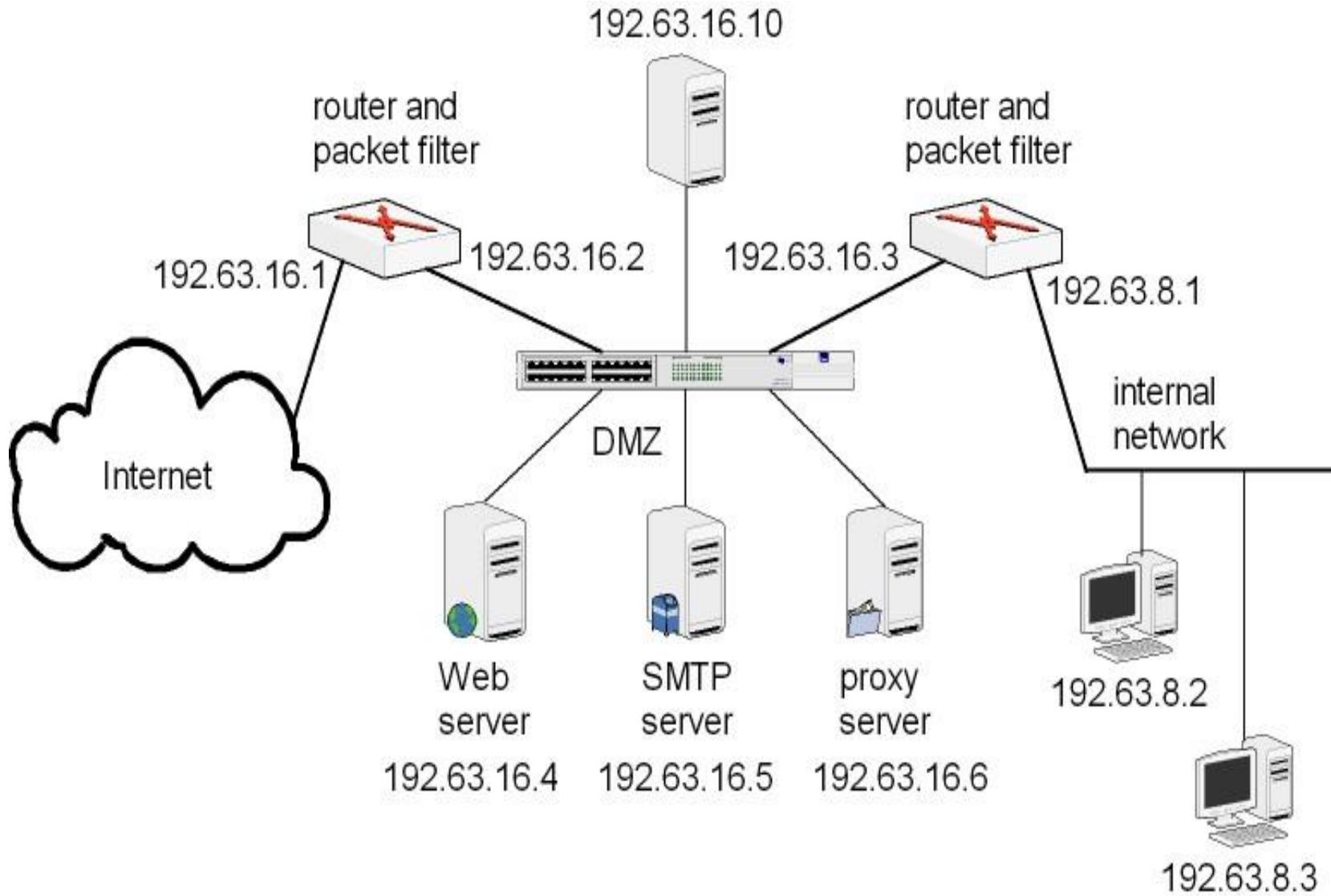


FIGURE 6-12 Dual-Homed Host Firewall

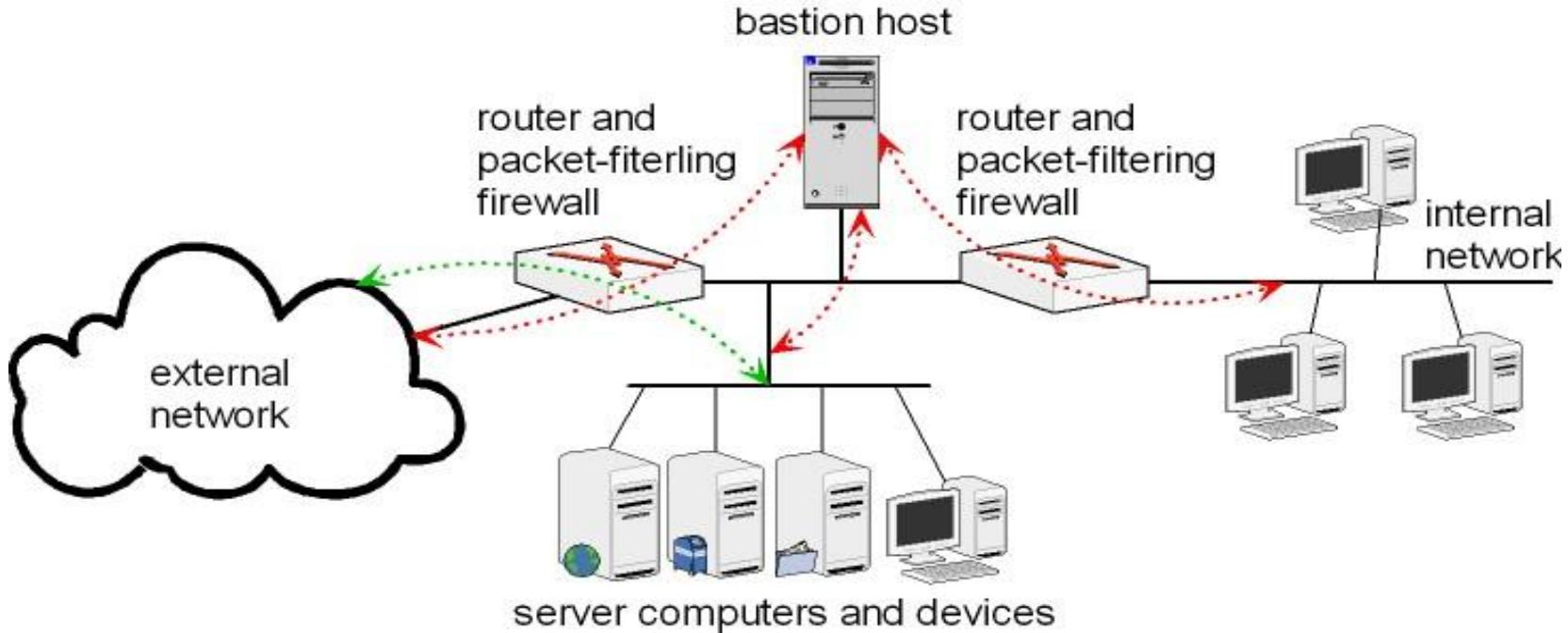
Demilitarized Zones (DMZ)

- A subnet between two firewalls in an internal network is often referred as a Demilitarized Zone (DMZ).
- The external firewall protects the DMZ subnetwork from external networks and the internal firewall protects the internal network from the DMZ.
- DMZ subnetwork may or not have a bastion host.
- Server computers and devices that do not require strong security protections are placed in the DMZ subnetwork so that they will not be exposed to the external networks without any security protection.
- The concept of single-layer DMZ can be generalized to a multiple-layer DMZ where a DMZ may also contain a sub-DMZ.
- Computers needing the most security protections are placed in a subnetwork that is connected to innermost firewall.
- Computers needing the least security protections are placed in the outermost DMZ, which is protected only by the outermost firewall.

Demilitarized Zones (DMZ) Contd.



Screened Subnets



- Screened subnets are the most secure firewall configuration. A screened subnet consists of a bastion host and two packet filtering routers.
- Screened subnet is a SHBH network with a second packet filtering router (i.e., inner router) inserted between the bastion host and the internal network . That is, in a screened subnet, one router is placed between the Internet and the bastion host, and the other router is placed between the bastion host and the internal network.

Screened Subnets Contd.

- The two packet-filtering firewalls create an isolated, screened sub-network in between.
- The outer routers announces to the public IP addresses and port numbers of the server computers and devices connected to the screened sub-network.
- The inner router announces to the internal network the IP addresses and port numbers of the server computers and devices connected to the screened sub-network. Thus, the structure of the internal network is hidden from the outside world.
- The internal hosts can only communicate with external hosts through server computers or devices in the screened sub-network.

Screened Subnets Contd.

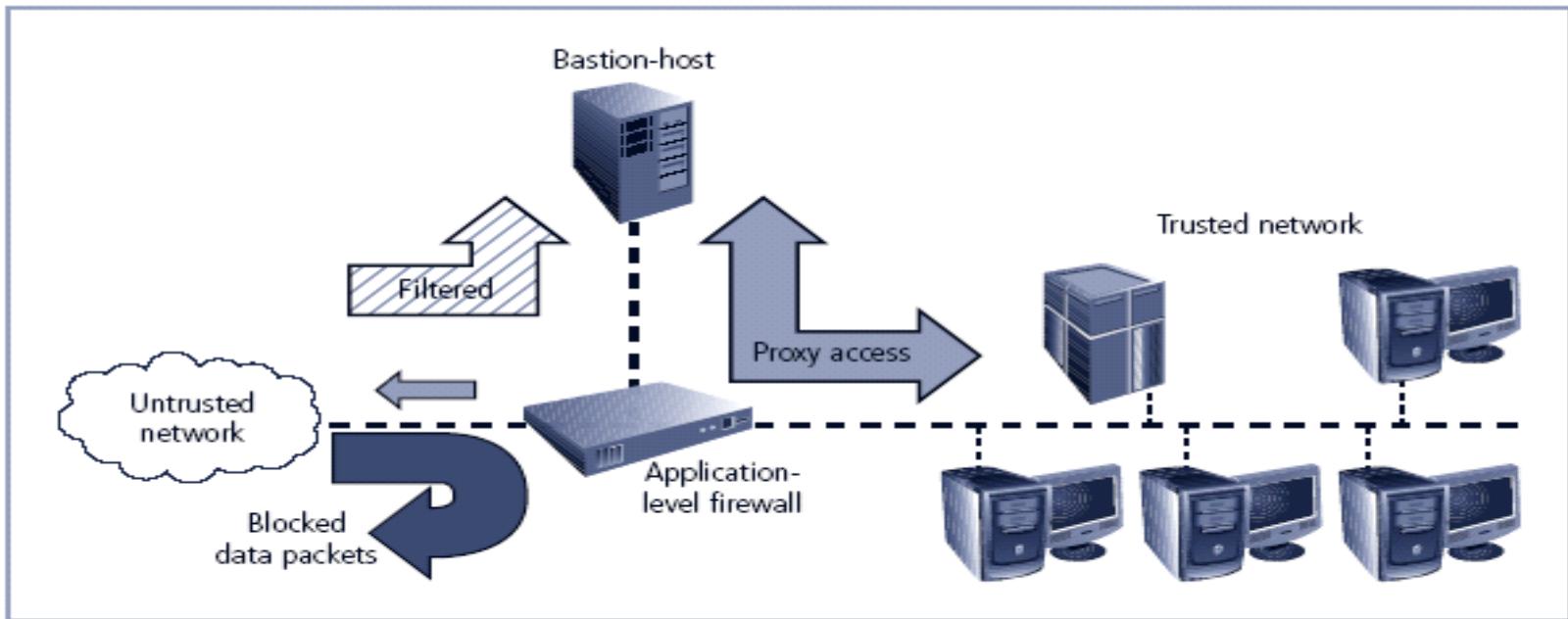


FIGURE 6-11 Screened Host Firewall

Table 6-4 Reserved Non-Routable Address Ranges

Class	From	To	CIDR Mask	Decimal Mask
Class "A" or 24 Bit	10.0.0.0	10.255.255.255	/8	255.0.0.0
Class "B" or 20 Bit	172.16.0.0	172.31.255.255	/12 or /16	255.240.0.0 or 255.255.0.0
Class "C" or 16 Bit	192.168.0.0	192.168.255.255	/16 or /24	255.255.0.0 or 255.255.255.0

Screened Subnet Firewalls (with DMZ)

- Dominant architecture used today is the screened subnet firewall
- Commonly consists of two or more internal bastion hosts behind packet filtering router, with each host protecting trusted network:
 - Connections from outside (untrusted network) routed through external filtering router
 - Connections from outside (untrusted network) are routed into and out of routing firewall to separate network segment known as DMZ
 - Connections into trusted internal network allowed only from DMZ bastion host servers.
 - Screened subnet performs two functions:
 - Protects DMZ systems and information from outside threats
 - Protects the internal networks by limiting how external connections can gain access to internal systems
- Another facet of DMZs: extranets

Screened Subnet Firewalls (with DMZ)

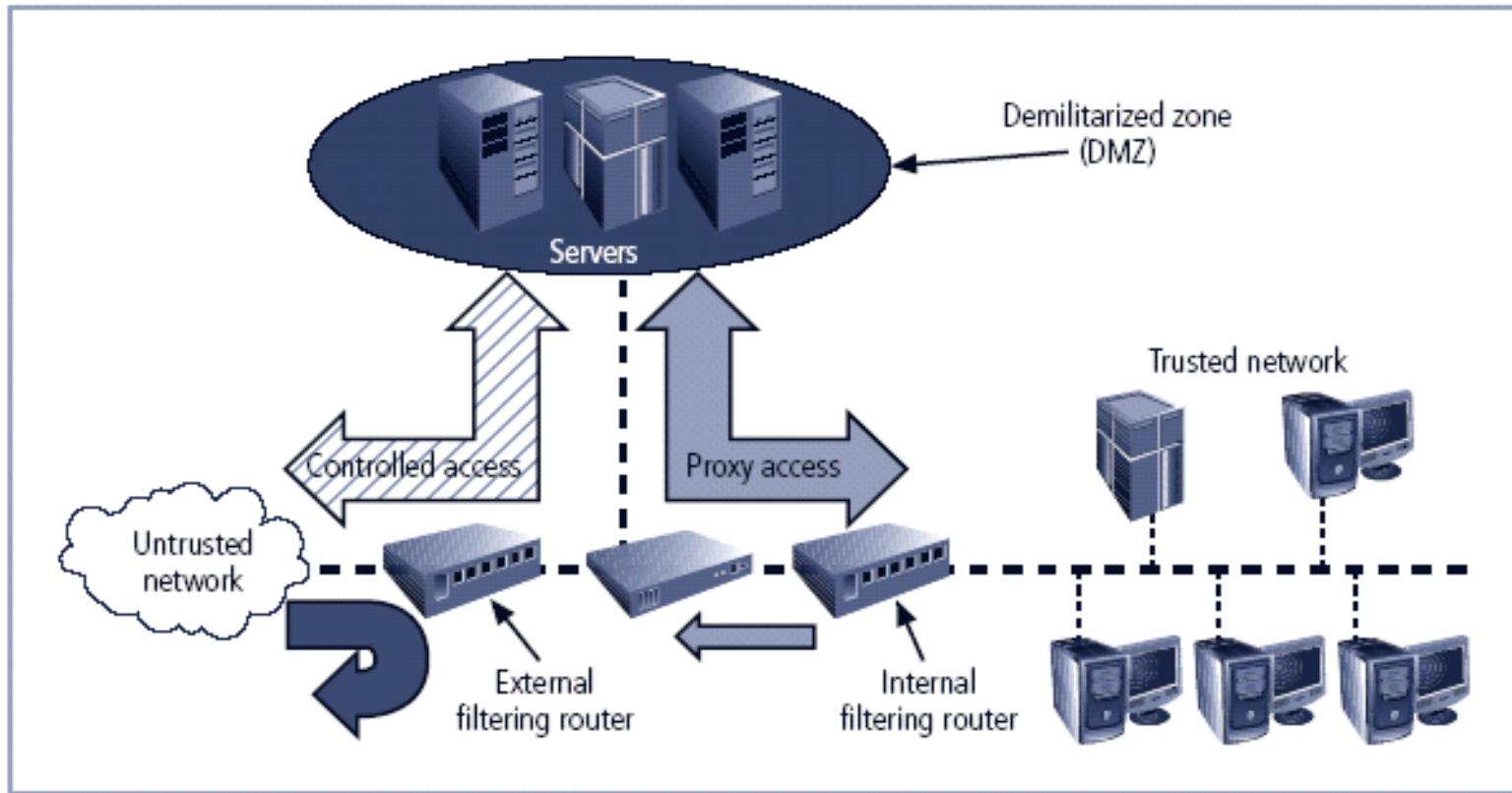


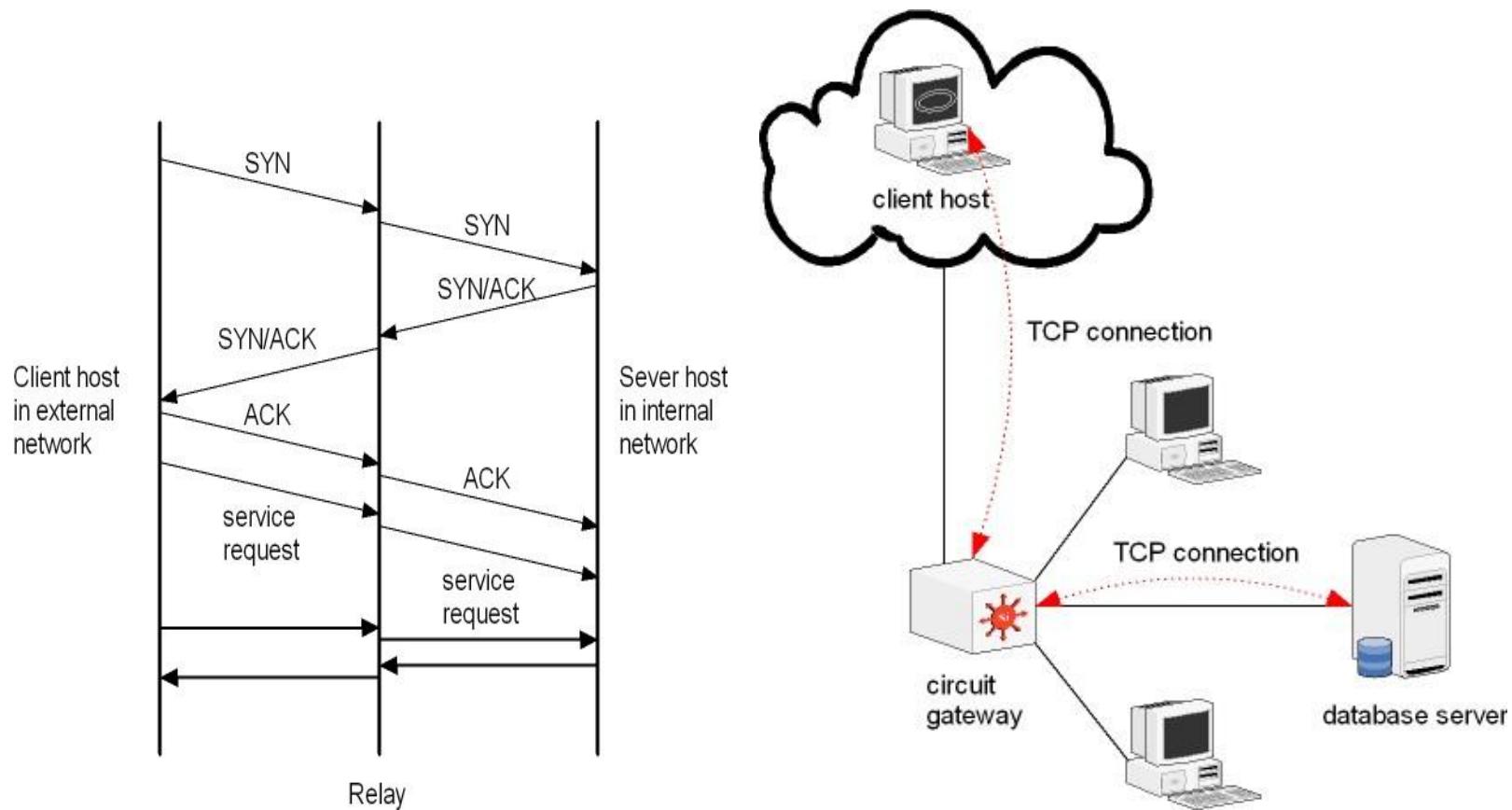
FIGURE 6-13 Screened Subnet (DMZ)

SOCKS

- SOCKS (shorts for sockets)
- SOCKS is a network protocol for implementing circuit gateway.
- SOCKS consists of three components: the SOCKS server, SOCKS client and SOCKS client library.
- SOCKS server runs on packet filtering firewall through port 1080. SOCKS client library runs on an internal hosts and SOCKS client runs on an external host.
- SOCKS client executes on the modified version of FTP and other standard TCP-based client application programs which are modified version for SOCKS.
- When an external client wants to obtain service from an internal server under protection of SOCKS, the client must first establish a TCP connection with the SOCKS server. It negotiates with the SOCKS server to select an authentication algorithm, provides information for authentication and submits a relay request .
- SOCKS server verifies the information submitted for authentication and determines whether to establish a relay connection with the internal server as requested.

SOCKS Contd.

- Only after the client request is granted should the client be allowed to forward the UDP packet to the internal host using the SOCKS server as a relay node. This packet is forwarded through TCP connection which is established between the SOCKS server and client.



Selecting the Right Firewall

- When selecting firewall, consider a number of factors:
 - What firewall offers right balance between protection and cost for needs of organization?
 - What features are included in base price ? What features are available at extra cost? Are all the factors known?
 - How easy to setup and configuration the firewall? How accessible are staff technicians who can configure the firewall?
 - Can the candidate firewall adapt to the growing network in the target organization?
- Second most important factor is cost. Cost may keep a certain make or type out of reach. As with all security decision, certain compromises may be necessary in order to provide a possible solution under the budgetary constraints stipulated by management.

Best Practices for Firewalls

- All traffic from trusted network is allowed out. This allows the members of the organization to access the service they need. Filtering and logging of outbound traffic can be implemented when required by specific organizational policies.
- Firewall device is never directly accessed from public network for configuration or management purpose. Almost all administrative access to the firewall device is denied to internal users as well. Only authorized firewall administrators access the device through secure authentication mechanism, preferable via a method that is based on cryptographically strong authentication and uses two-factor access control technique.
- Simple Mail Transport Protocol (SMTP) data allowed to enter through the firewall, but it is routed to a well configured SMTP gateway to filter and route messaging traffic securely.
- All Internet Control Message Protocol (ICMP) data should be denied. Known as the ping service, ICMP is a common method for hacker to reconnaissance and should be turned off to prevent snooping.

Best Practices for Firewalls Contd.

- When Web services are offered outside the firewall , HTTP traffic should be blocked from your internal networks through the use of some form of proxy access or DMZ architecture. That way, if any employees are running Web servers for internal use on their desktops, the service are invisible to the outside Internet at large to view it. The best solution is to place the Web servers containing critical data inside the network and use proxy services from DMZ (screened network management) and also to restrict Web traffic bound for internal network address to allow only those request that originated from internal address
- All data is not verifiably authentic should be denied. When attempting to convince packet filtering firewall to permit malicious traffic, attacker will frequently put an internal address in the source field. To avoid this problem, set rules so that the external firewall blocks all inbound traffic with an organizational source address.

IT352-Class-3 and Class-4

Firewall Rules

- Operate by examining data packets and performing comparison with predetermined logical rules
- Logic based on set of guidelines most commonly referred to as firewall rules, rule base, or firewall logic
- Most firewalls use packet header information to determine whether specific packet should be allowed or denied.
- Some firewalls can filter packets protocol name as opposed to protocol port number. For, instance, Telnet protocol packets usually go to TCP port 23, but can sometimes be redirected to another much higher port number in an attempt to conceal the activity.
- The system (or well known) ports are those from ‘0’ to 1023, User (or Registered) ports are those from 1024 to 49151, and dynamic (or Private) ports are those from 59152 to 65535

Standard Port Numbers

- Simple TCP/IP services usually listen on the following ports

Port	TCP/IP Service
7	Echo (Ping)
9	Discard
13	Daytime
17	Quote of the day
19	Character Generator

Standard Port Numbers

Internet server usually listen on the following ports

Port	Server
21	File Transfer Protocol
23	Telnet
70	Gopher
80	World Wide Web
119	Net News (NNTP)
22	Secure Shell
443	Secure HTTP (HTTPS)

Packet Filtering Rules

- The packet filter is typically setup as a list of rules based on matches to fields in the IP or TCP header. If there is match to one of the rules, that rule is invoked to determine whether to forward to discard the packet. If there is not match to any rule, then default action is taken. Two default policies are possible.
- Default = discard : That which is not expressly permitted to prohibited.
- Default = Forward : That which is not expressly prohibited is permitted.
- Default discard policy is more conservative. Initially, everything is blocked, and service must be added on a case by case. 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 user but provides reduced security; the security administrator must in essence, react to each new security threat as it becomes known.

Packet Filtering Rules

- The packet filter is typically setup as a list of rules based on matches to fields in the IP or TCP header. If there is match to one of the rules, that rule is invoked to determine whether to forward to discard the packet. If there is not match to any rule, then 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.
- Default discard policy is more conservative. Initially, everything is blocked, and service must be added on a case by case. 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 user but provides reduced security; the security administrator must in essence, react to each new security threat as it becomes known.

Packet Filtering Rules

Assume that default = ‘discard’ policy in force.

1. Inbound mail is allowed (port 25 is for SMTP incoming , but only to gateway host. However, packets from a particular external host, SPIGOT are blocked because that host has a history of sending massive file in e-mail message.
2. This rule is intended to specify that any inside host can send mail to the outside. A TCP packet with a destination port of 25 is routed to the SMTP server on the destination machine. The problem with this rule is that the use of port 25 for SMTP receipt is only a default; an outside machine could be configured to have some other application linked to port 25. As this rule is written, an attacker could gain access to internal machines by sending packets with a TCP source port number of 25

Packet Filtering Rules Contd.

Assume that default = ‘discard’ policy in force.

3. This rule set achieves the intended result that was not achieved in previous rule. The rules take advantage of a feature of TCP connections. Once a connection is setup, the ACK flag of TCP segment is set to acknowledge segments sent from the other side. Thus, this rule set states that it allow IP packets where the source IP address is one of the list of designated internal host and the destination TCP port number is 25. It also allows incoming packets with a source port number of 25 that includes ACK flag in TCP segment.

4. This rule set is one approach to handling FTP connections. With FTP, two TCP connections are used. With FTP, two TCP connections are used: a control connection is set up the file transfer and a data connection for the actual file transfer. The data connection uses a different port number that is dynamically assigned for the transfer.

Packet Filtering Rules Contd.

3. This rule set is one approach to handling FTP connections. With FTP, two TCP connections are used: a control connection is set up for the file transfer and a data connection for the actual file transfer. The data connection uses a different port number that is dynamically assigned for the transfer.

Action	Source	Port	Destination	Port	Flags	Comments
Allow	{Our Hosts}	*	*	*		Our Outgoing Calls
Allow	*	*	*	*	ACK	Replies to our calls
Allow	*	*	*	>1024		Traffic to non servers

Rule Set 1

Response to internal requests are allowed. In most firewall implementation it is desirable to allow a response to an internal request for information. In dynamic or stateful, this is most easily accomplished by matching the incoming traffic to an outgoing request in a state table. In simple packet filtering, this can be accomplished with the f

Source Address	Port	Destination Address	Port	Action
Any	Any	10.10.10.0	>1024	Allow

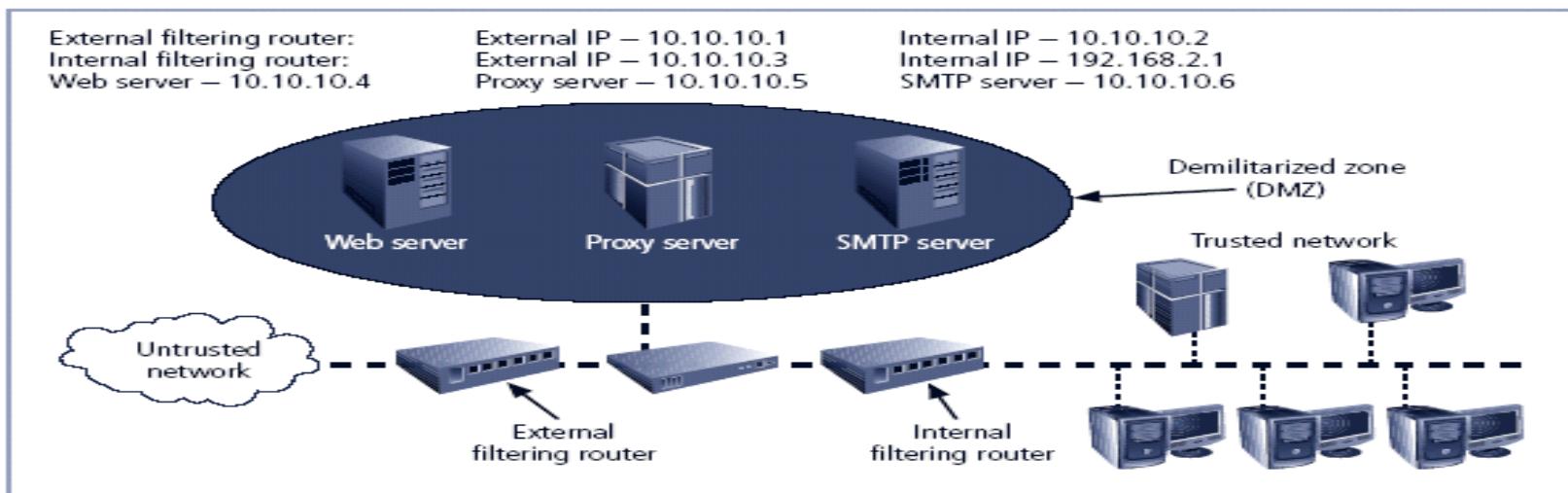


FIGURE 6-14 Example Network Configuration

Rule Set 2

Firewall device is never accessible directly from the public network. If attacker can directly access the firewall , they may be able to modify or delete rules and allow unwanted traffic through.

Source Address	Port	Destination Address	Port	Action
Any	Any	10.10.10.1	Any	Deny
Any	Any	10.10.10.2	Any	Deny
10.10.10.1	Any	Any	Any	Deny
10.10.10.2	Any	Any	Any	Deny

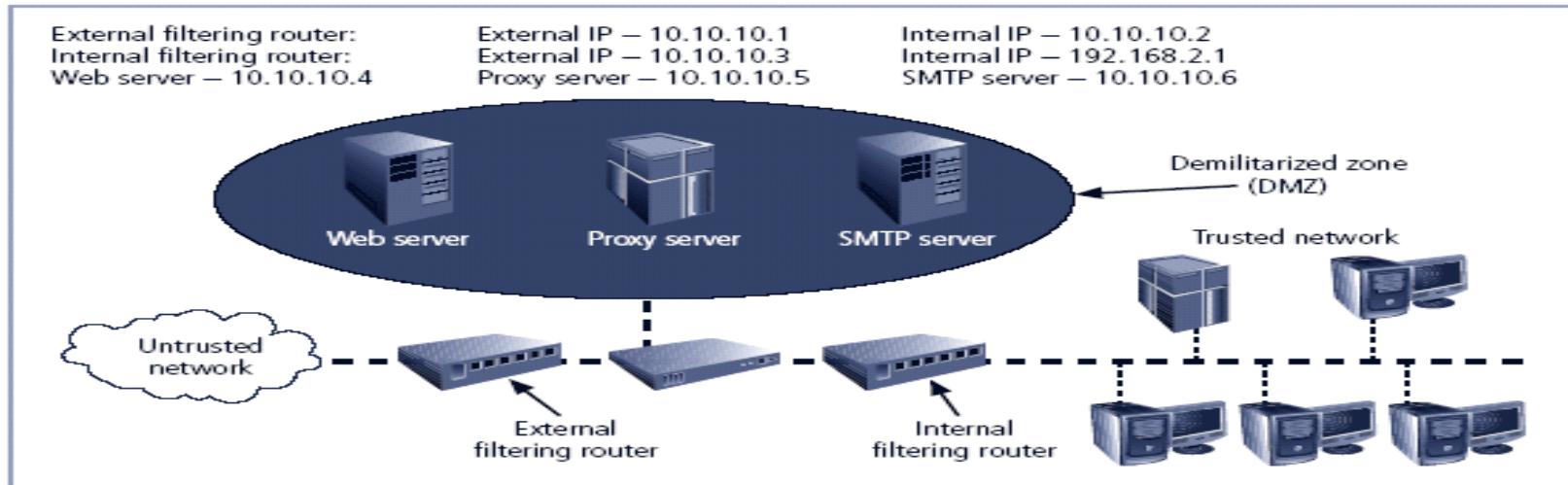


FIGURE 6-14 Example Network Configuration

Rule Set 3

All the traffic from the trusted network is allowed out. As a general rule it is wise not to restrict outbound traffic, unless a separate router is configured to handle it, to avoid overloading the firewall. If an organization wants control over outbound traffic, it should use a separate filtering device.

Source Address	Port	Destination Address	Port	Action
10.10.10.0	Any	Any	any	Allow

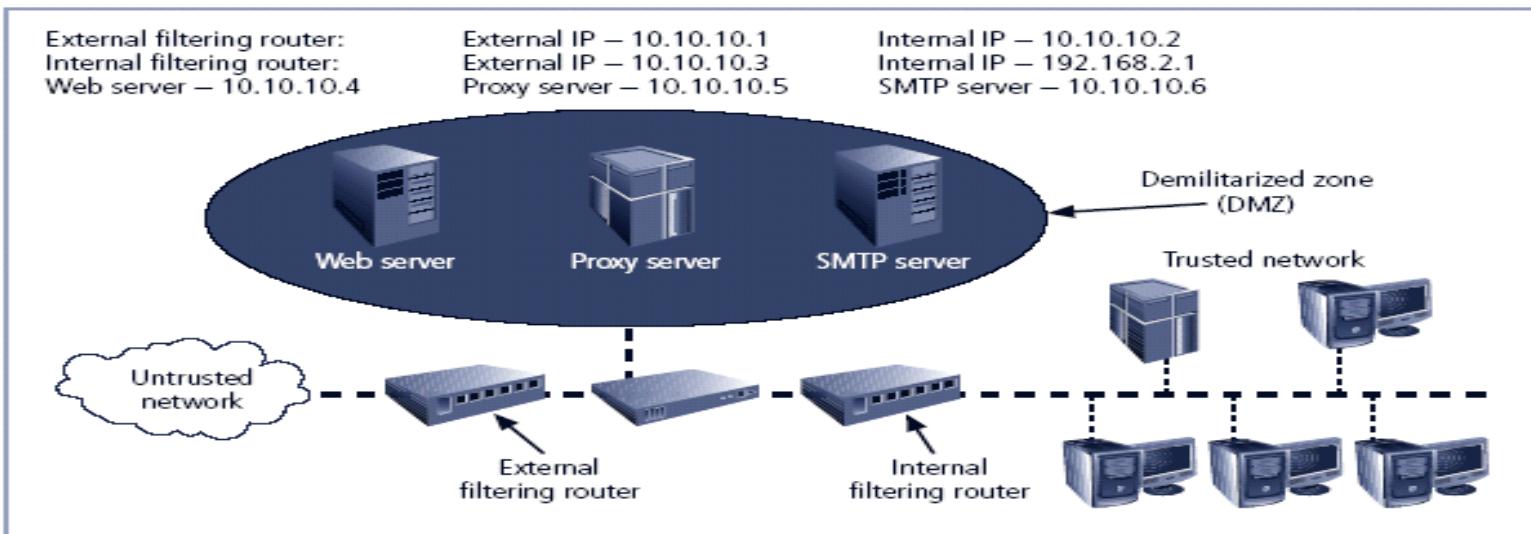


FIGURE 6-14 Example Network Configuration

Rule Set 4

The rule set for Simple Mail Transport Protocol.

The packets governed by this rule are allowed to pass through the firewall, but are all routed to a well configured SMTP gateway. It is important that e-mail traffic reach your email traffic server and only your email server.

Source Address	Port	Destination Address	Port	Action
Any	Any	10.10.10.6	25	Allow

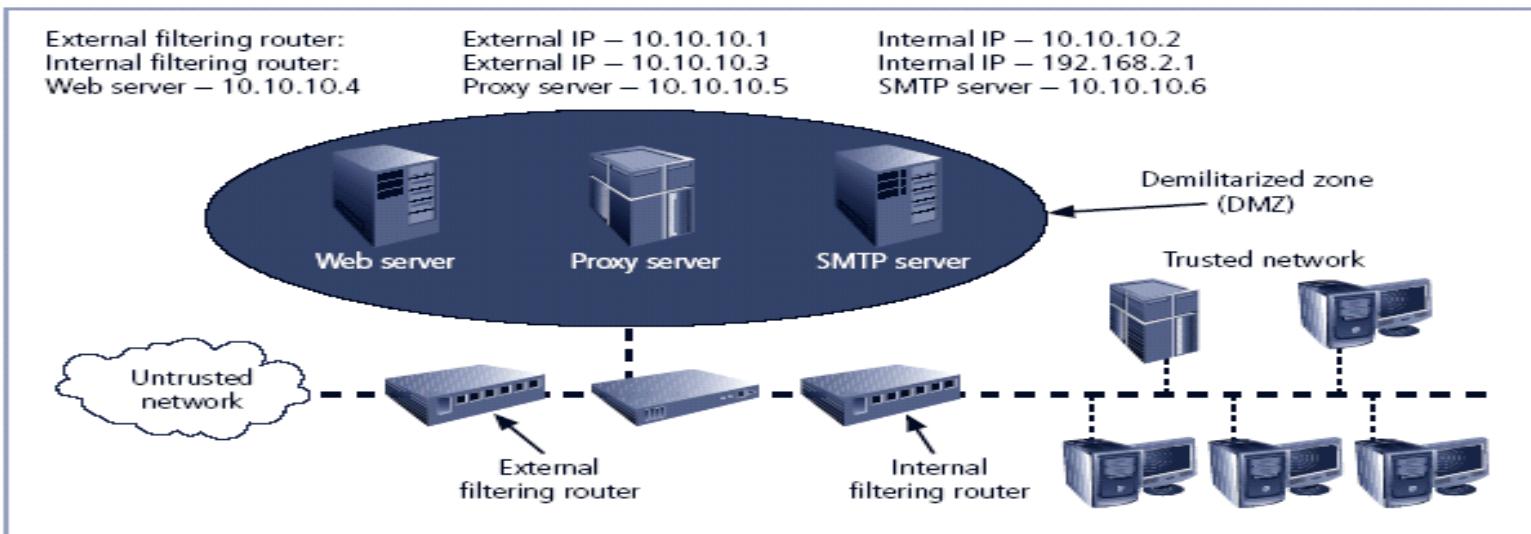


FIGURE 6-14 Example Network Configuration

Rule Set 5

All Internet Control Message Protocol (ICMP) data should be denied. Ping, formally known as ICMP echo request are used by internal system administrator to ensure that the clients and servers can communicate.

Source Address	Port	Destination Address	Port	Action
10.10.10.0	Any	Any	7	Allow
Any	Any	10.10.10.0	7	Deny

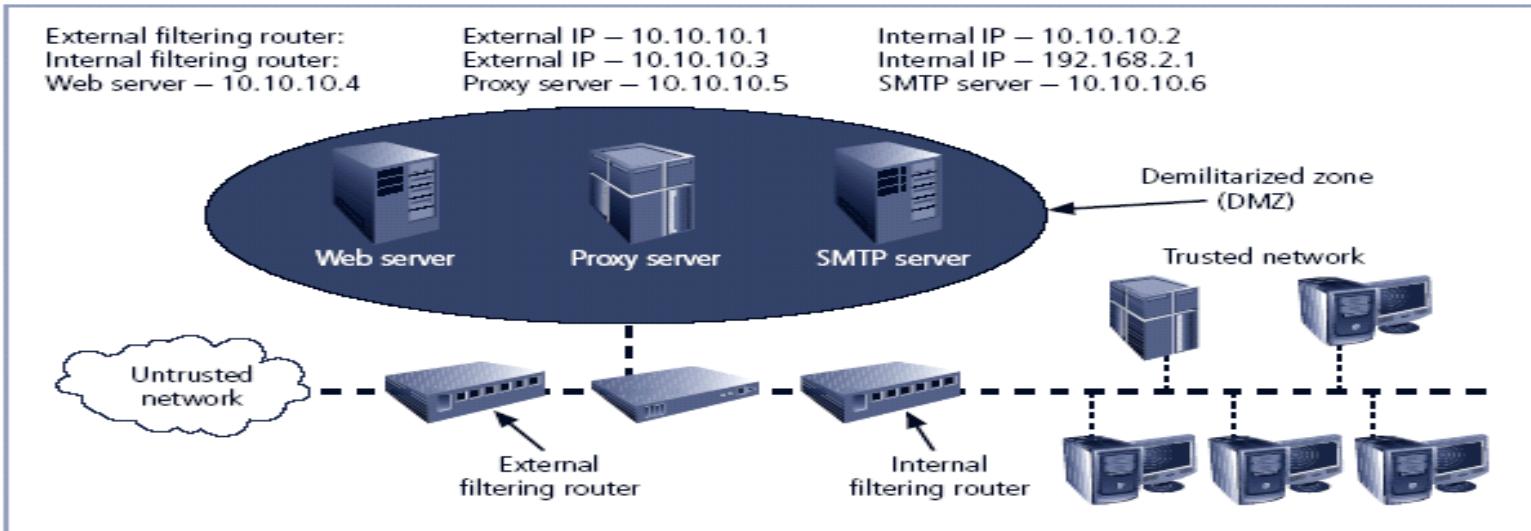


FIGURE 6-14 Example Network Configuration

Rule Set 6

Telnet (Terminal emulation) access to all internal servers from the public networks should be blocked. Though not used much in windows environments. Telnet is still useful to system administrator on Unix/Linux system.

Source Address	Port	Destination Address	Port	Action
10.10.10.0	Any	10.10.10.0	23	Allow
Any	Any	10.10.10.0	23	Deny

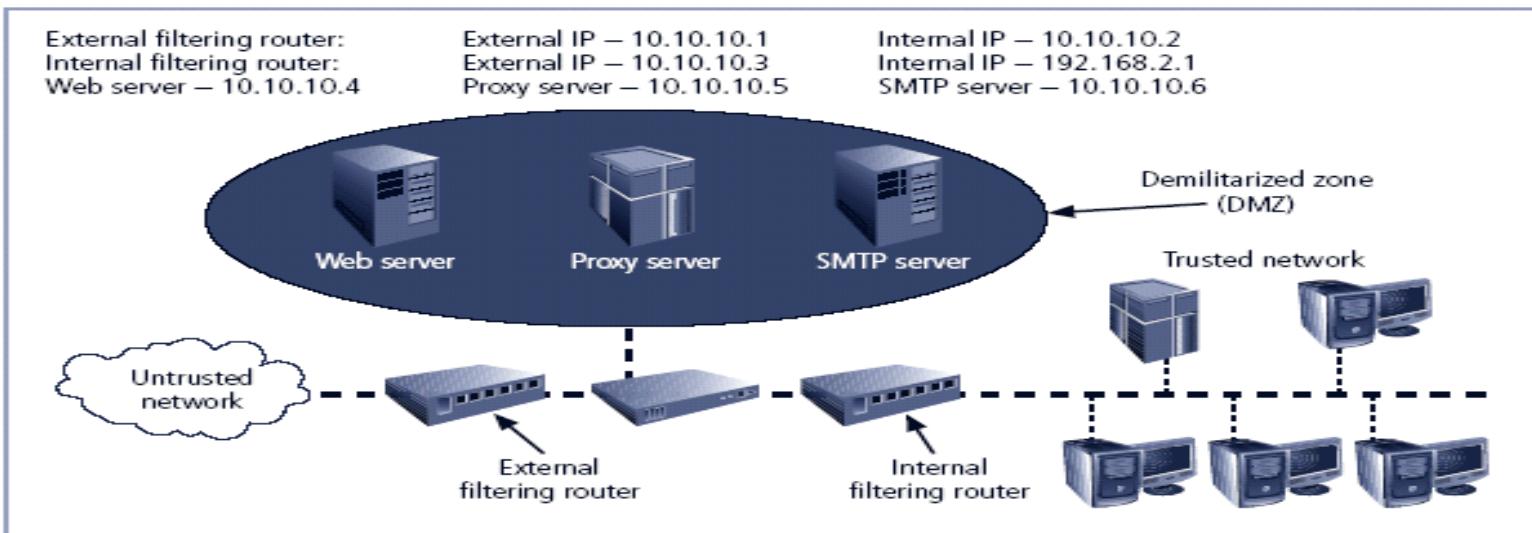


FIGURE 6-14 Example Network Configuration

Rule Set 7

When a web service are offered outside the firewall, HTTP traffic should be blocked from the internal network via the use of some form of proxy access or DMZ architecture.

With a Web server

Source Address	Port	Destination Address	Port	Action
Any	Any	10.10.10.4	80	Allow

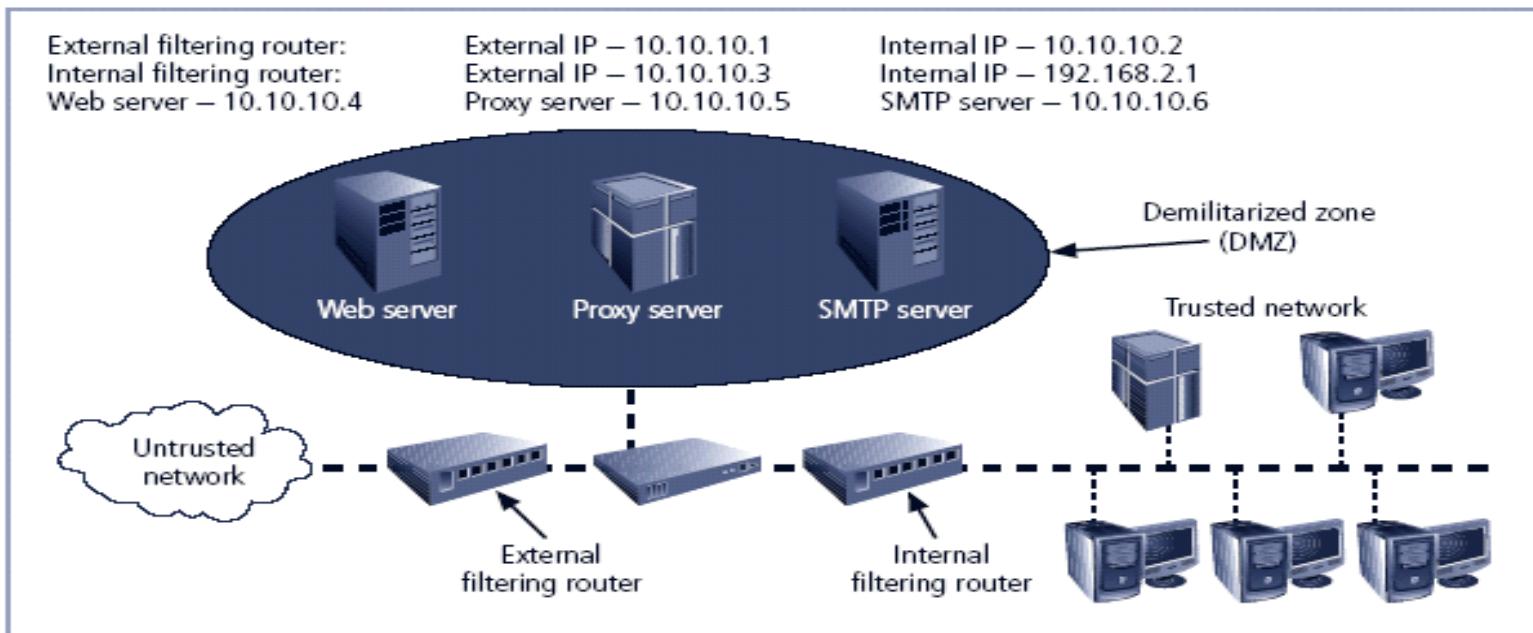


FIGURE 6-14 Example Network Configuration

Rule Set 8

The **Cleanup Rule** As a general practice in firewall rule construction, if a request for a service is not explicitly allowed by policy, that request should be denied by a rule.

Additional rules restricting access to a specific servers or device can be added, but they must be sequenced before the cleanup

Source Address	Port	Destination Address	Port	Action
Any	Any	10.10.10.4	80	Allow

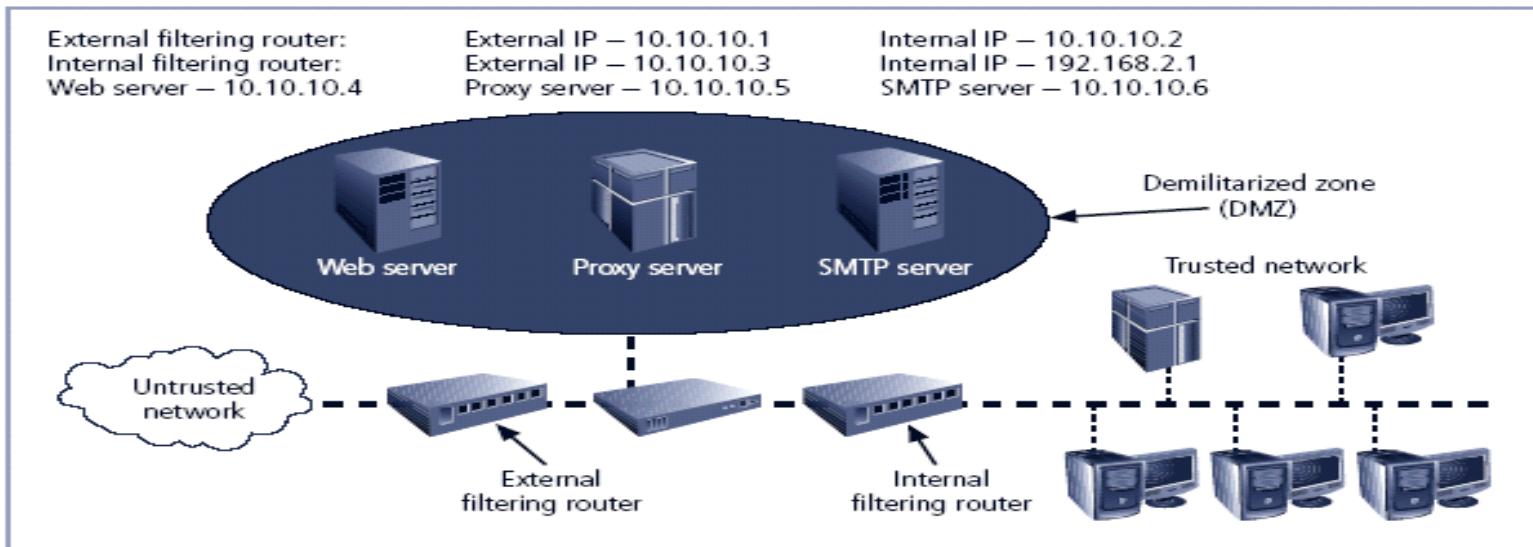


FIGURE 6-14 Example Network Configuration

TABLE 6-16 External Filtering Firewall Rule Set

Rule #	Source Address	Source Port	Destination Address	Destination Port	Action
1	Any	Any	10.10.10.0	>1023	Allow
2	Any	Any	10.10.10.1	Any	Deny
3	Any	Any	10.10.10.2	Any	Deny
4	10.10.10.1	Any	Any	Any	Deny
5	10.10.10.2	Any	Any	Any	Deny
6	10.10.10.0	Any	Any	Any	Allow
7	Any	Any	10.10.10.6	25	Allow
8	Any	Any	10.10.10.0	7	Deny
9	Any	Any	10.10.10.0	23	Deny
10	Any	Any	10.10.10.4	80	Allow
11	Any	Any	Any	Any	Deny

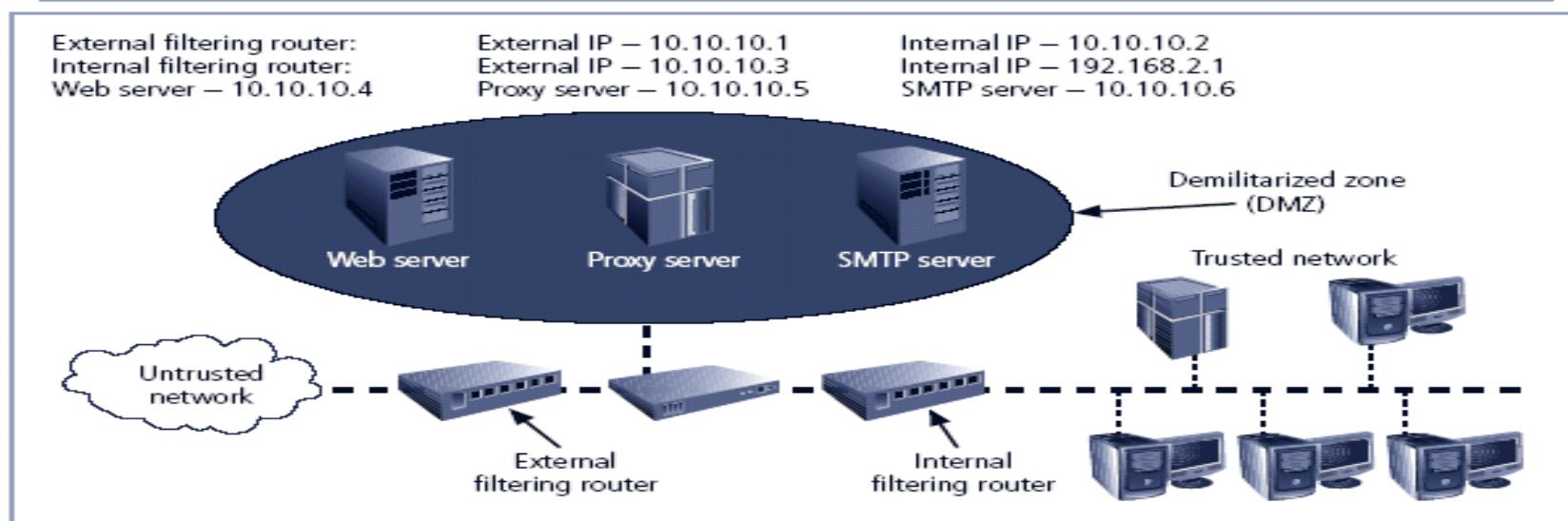
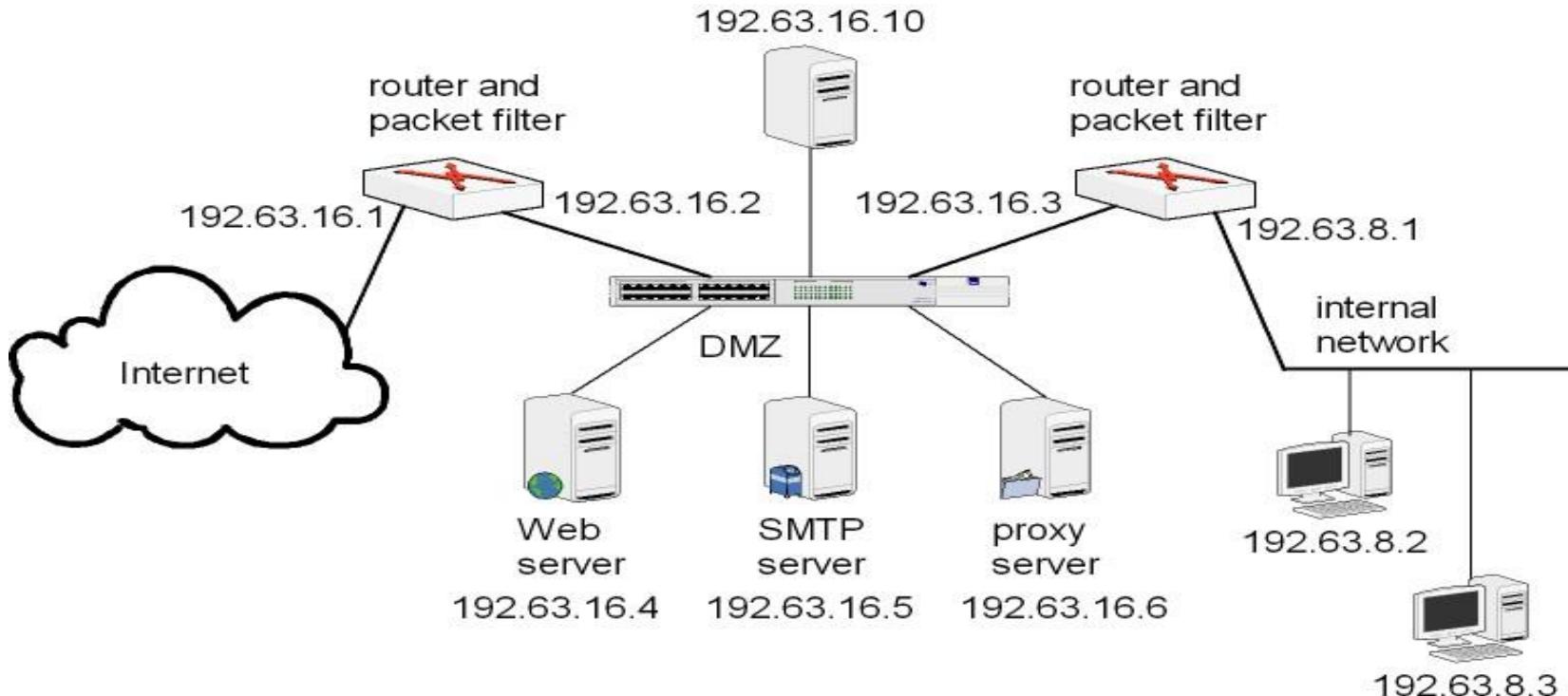
**FIGURE 6-14** Example Network Configuration

TABLE 6-17 Internal Filtering Firewall Rule Set

Rule #	Source Address	Source Port	Destination Address	Destination Port	Action
1	Any	Any	10.10.10.0	>1023	Allow
2	Any	Any	10.10.10.3	Any	Deny
3	Any	Any	192.168.2.1	Any	Deny
4	10.10.10.3	Any	Any	Any	Deny
5	192.168.2.1	Any	Any	Any	Deny
6	192.168.2.0	Any	Any	Any	Allow
7	10.10.10.5	Any	192.168.2.0	Any	Allow
8	Any	Any	Any	Any	Deny

Problem No. 3



In the above figure, assume that the router uses ACL rules given in table 7.7 and the inner router uses ACL rules given in Table 7.8. In addition to port 25, other ports in the tables are defined as follows: port 80 used for Web server program HTTP, port 7 is used for server program echo, port 23 is used for server program telnet and port 22 is used for server program SSH.

1. Explain what each ACL rule is intended to do
2. Point out which ACL rule is used for egress packet and which ACL rule is used for ingress packets.

Table 7.7 The ACL rules contained in the outer router in Fig. 7.12

line number	source addr	source port	dest addr	dest port	action
1	*	*	192.63.16.10	> 1023	allow
2	*	*	192.63.16.1	*	block
3	*	*	192.63.16.2	*	block
4	192.63.16.1	*	*	*	block
5	192.63.16.2	*	*	*	block
6	192.63.16.10	*	*	*	allow
7	*	*	192.63.16.5	110	allow
8	*	*	192.63.16.10	7	block
9	*	*	192.63.16.10	23	block
10	*	*	192.63.16.10	22	allow
11	*	*	192.63.16.4	80	allow
12	*	*	*	*	block

Table 7.8 The ACL rules contained in the inner router in Fig. 7.12

line number	source addr	source port	dest addr	dest port	action
1	*	*	192.63.16.10	> 1023	allow
2	*	*	192.63.16.10	25	allow
3	*	*	192.63.16.3	*	block
4	*	*	192.63.8.1	*	block
5	192.63.16.3	*	*	*	block
6	192.63.8.1	*	*	*	block
7	192.63.8.2	*	*	*	allow
8	192.63.8.3	*	*	*	allow
9	192.63.16.6	*	192.63.8.2	*	block
10	192.63.16.6	*	192.63.8.3	*	block
11	*	*	*	*	block

Problem No. 4

If an ingress packet from an external network to the internal network, its source address is an internal IP address or its private network address, should this packet be allowed or blocked? Why?

Packet Filters Contd.

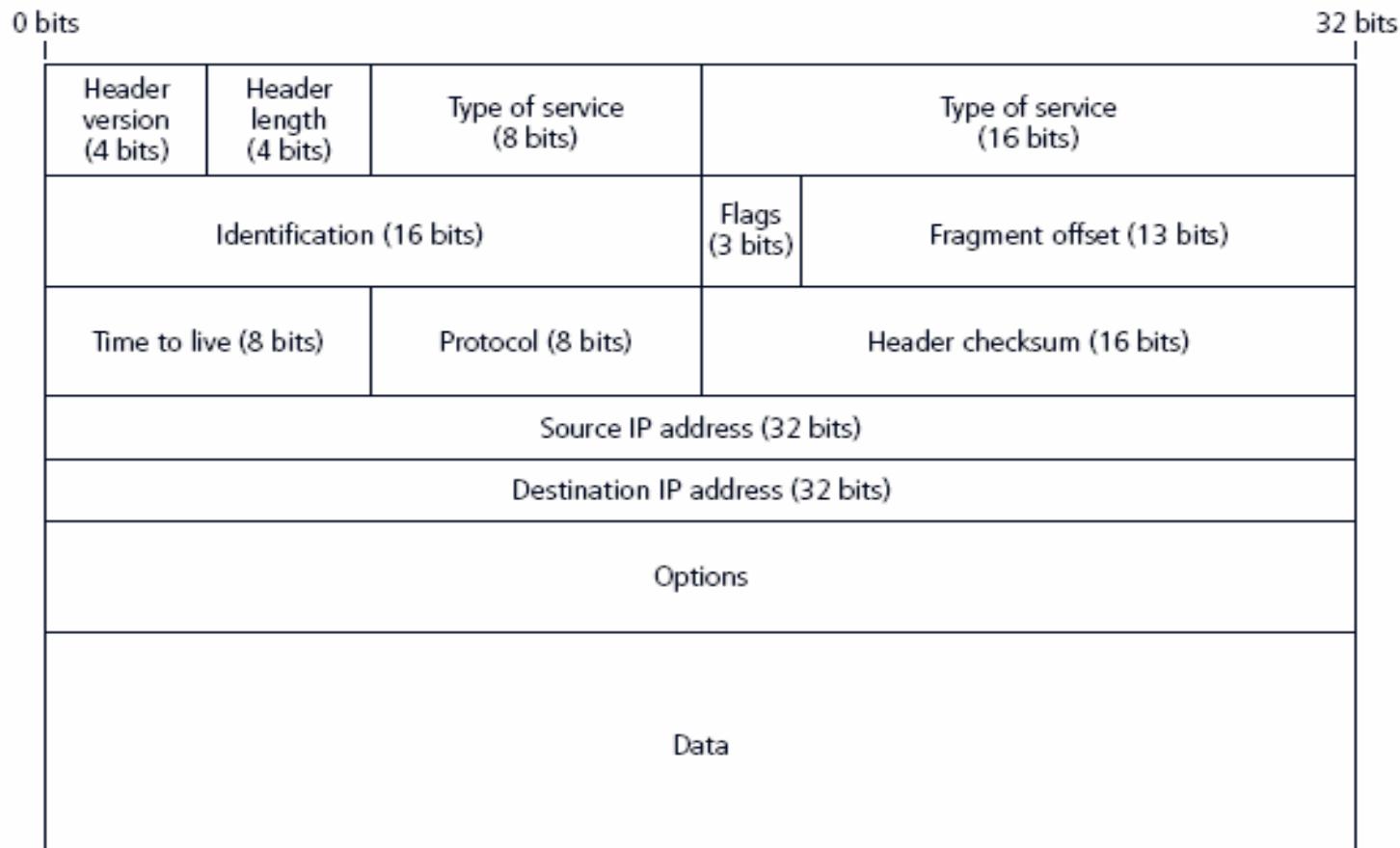


FIGURE 6-1 IP Packet Structure

Packet Filters Contd.

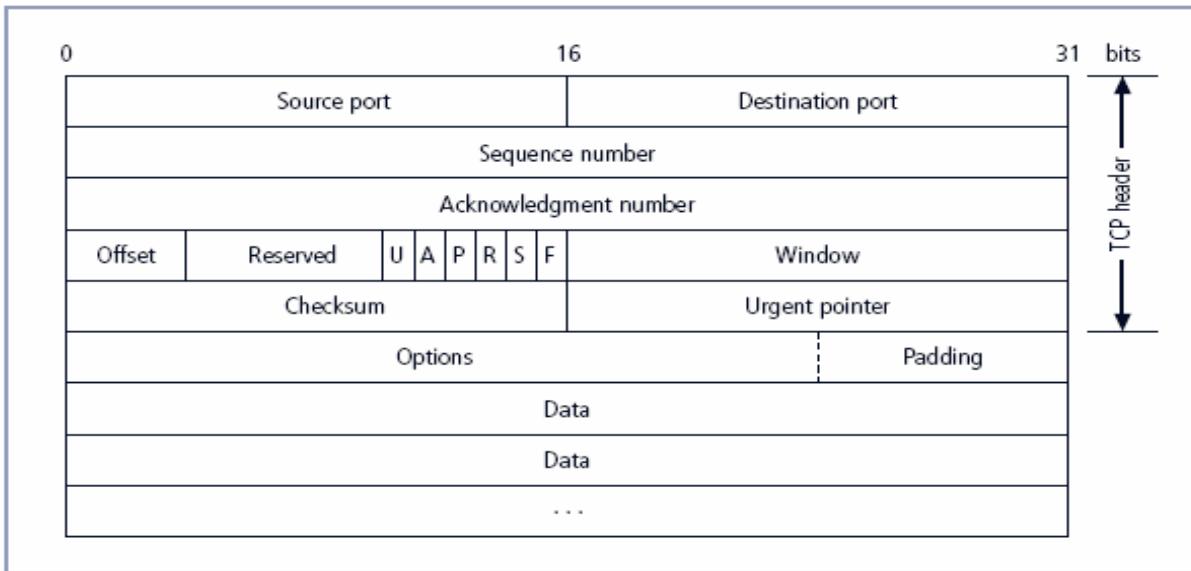


FIGURE 6-2 TCP Packet Structure

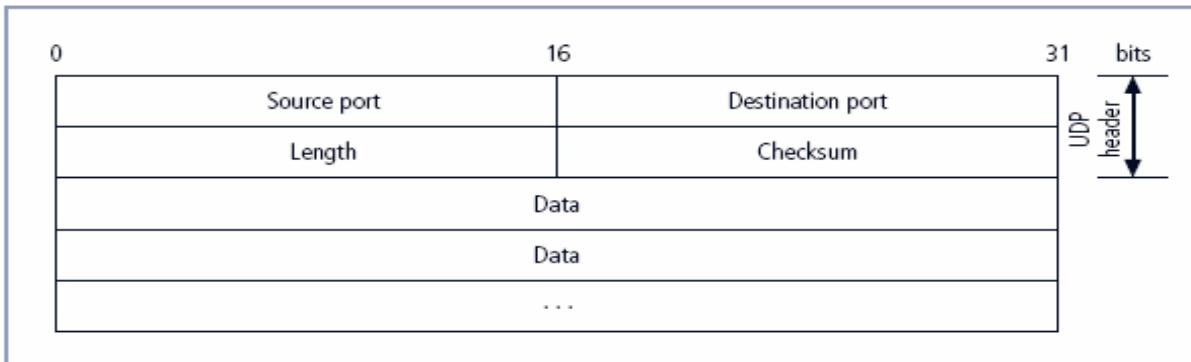


FIGURE 6-3 UDP Datagram Structure

What is a Firewall?

- A firewall is a system of hardware and software components designed to restrict access between or among networks, most often between the Internet and a private Internet.
- The firewall is part of an overall security policy that creates a perimeter defense designed to protect the information resources of the organization.

In other words

“A data entry at the gateway to your network, combining the power of multiple firewall technologies to deliver powerful perimeter security”



What a Firewall does

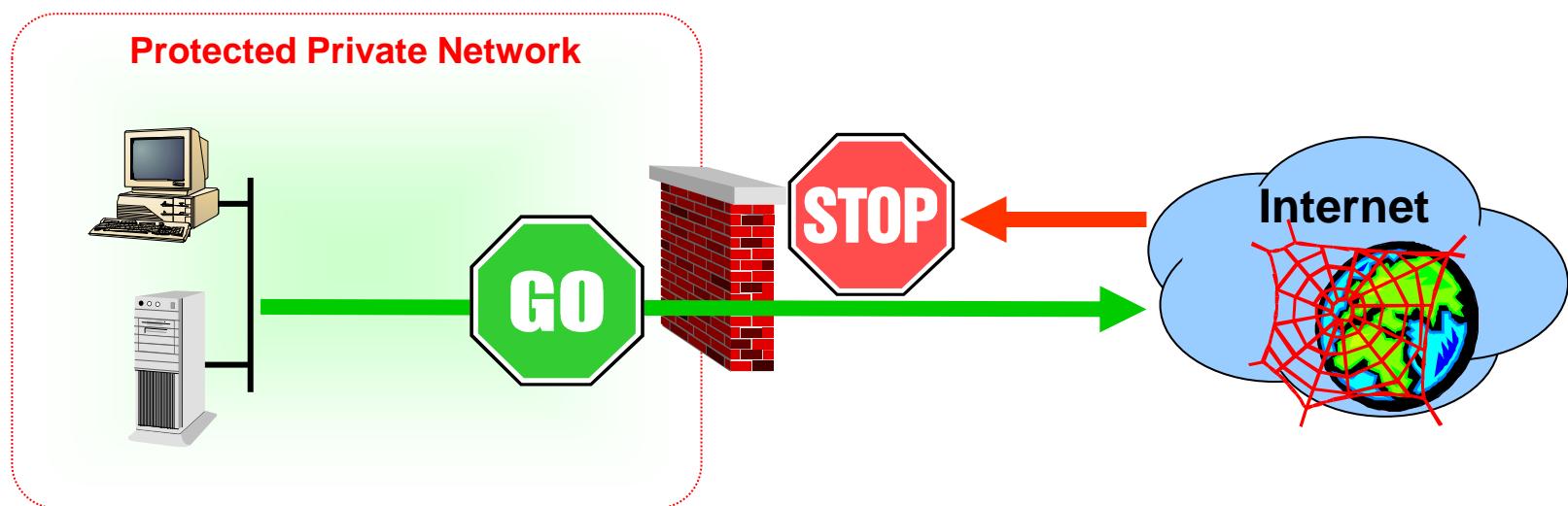
- Implement security policies at a single point
- Monitor security-related events (audit, log)
- Provide strong authentication
- Allow virtual private networks

What a Firewall does not do

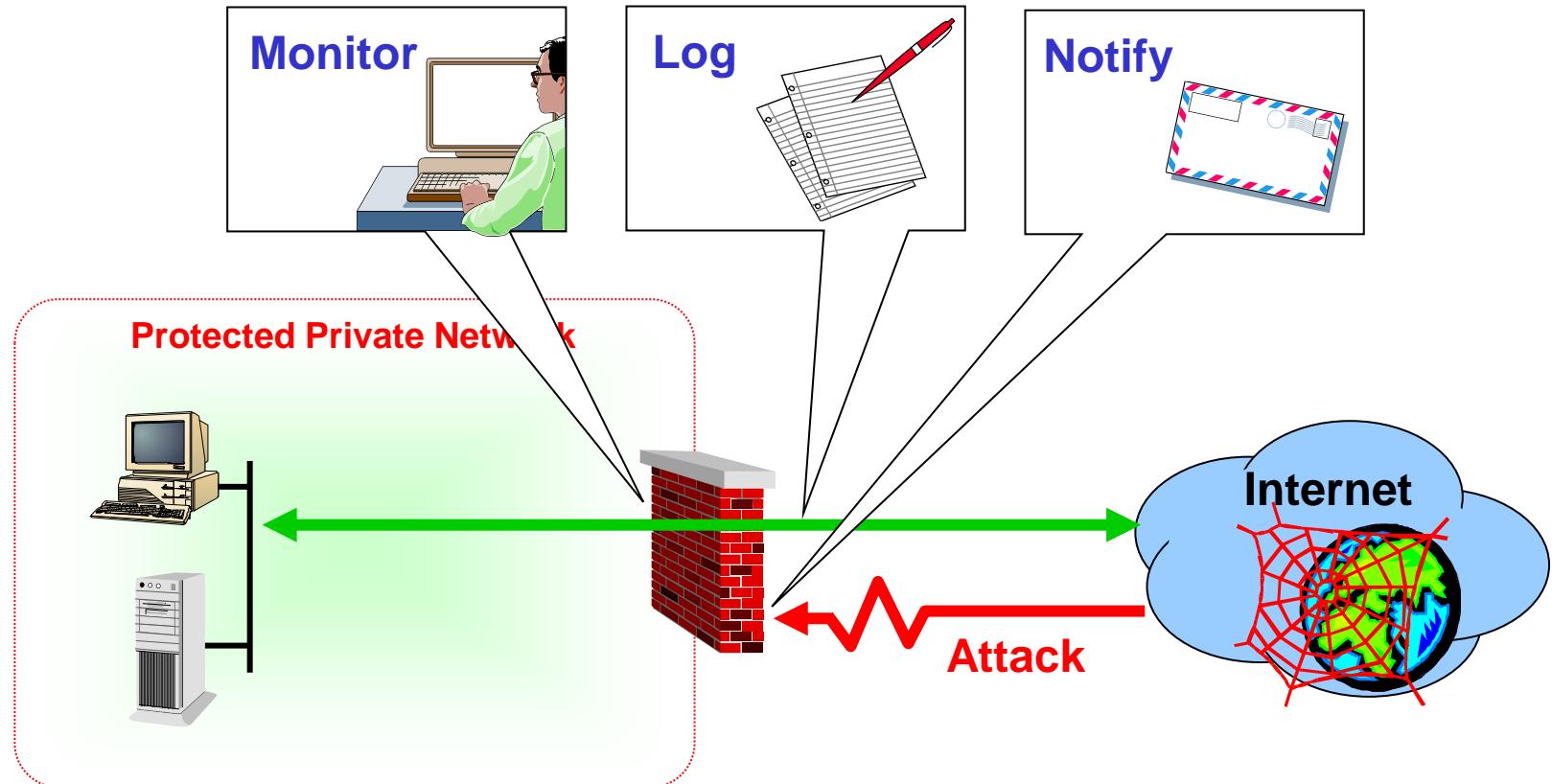
- Protect against attacks that bypass the firewall
 - Dial-out from internal host to an ISP
- Protect against internal threats
 - disgruntled employee
 - Insider cooperates with an external attacker
- Protect against the transfer of virus-infected programs or files

Firewall - Typical layout

A firewall denies or permits access based on policies and rules



Watching for attack



Firewall technologies

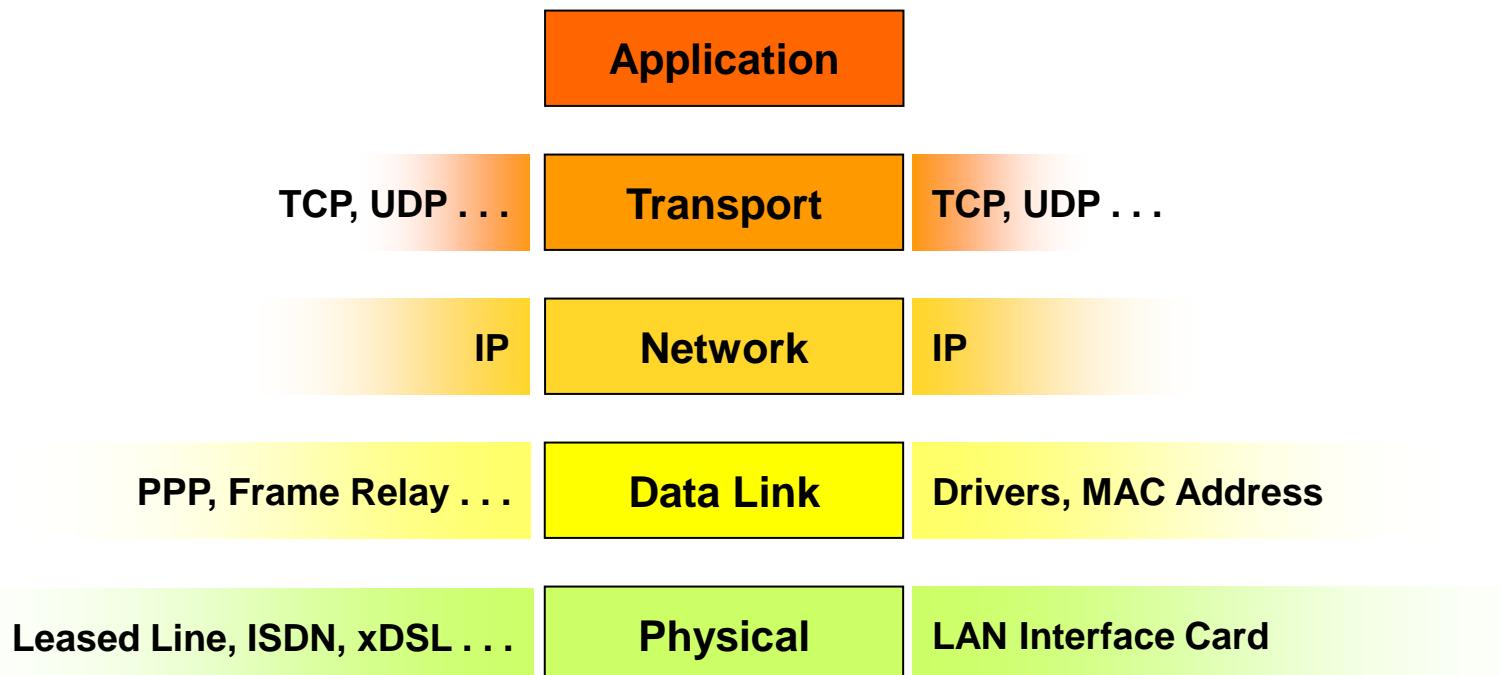
Common firewall technologies:

- They may be classified into four categories:
 - Packet Filtering Firewalls
 - Circuit Level Firewalls
 - Application Gateway Firewalls (or proxy servers)
 - Stateful Inspection Firewalls (dynamic packet filtering firewalls)

These technologies operate at different levels of detail, providing varying degrees of network access protection.

These technologies are not mutually exclusive as some firewall products may implement several of these technologies simultaneously.

The Internet protocol stack

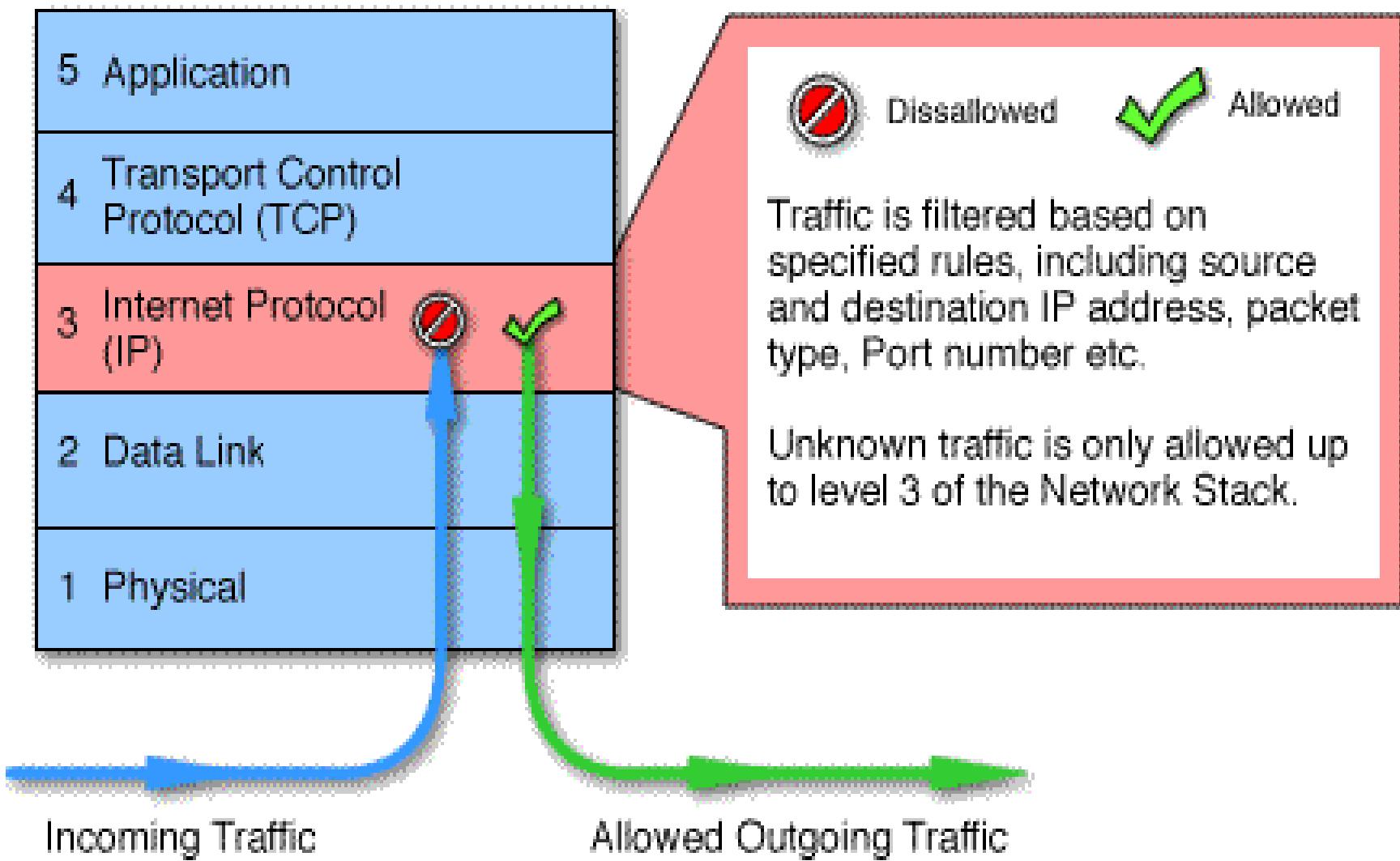


Packet Filtering Firewalls

Packet Filtering firewalls

- The original firewall
- Works at the network level of the OSI model
- Applies packet filters based on access rules
 - Source address
 - Destination address
 - Application or protocol
 - Source port number
 - Destination port number

Packet Filtering firewalls



Packet Filtering firewalls

- Packet Filtering is usually an integrated function of a router.
- Packet filtering relies on Network Layer and Transport Layer information contained in the headers of data packets to police traffic.
- This information includes source IP address and port number, destination IP address and port number, and protocol used (e.g., TCP, UDP, ICMP). This information is used as the criteria in network access rules. These rules are organized into several “filter sets” and each set handles traffic coming to the firewall over a specific interface.

Packet Filtering Policy Example

	My host		Other host		
action	name	port	name	port	comments
block	*	*	microsoft.com	*	Block everything from MS
allow	My-gateway	25	*	*	Allow incoming mail

Packet Filtering Policy Example

Rule	Direction	Source Address	Destination Address	Protocol	# Source Port	# Destin. Port	Action
1	Out	*	10.56.199*	*	*	*	Drop
2	Out	10.56*	10.122*	TCP	*	23 (Telnet)	Pass
3	In	10.122*	10.56.199*	TCP	23 (Telnet)	*	Pass
4	In & Out	*	10.56.199*	TCP	*	25 (Mail)	Pass
5	In	*	*	TCP	*	513 (rlogin)	Drop
6	In	201.32.4.76	*	*	*	*	Drop
7	Out	*	*	TCP	*	20 (FTP)	Pass
8	In	*	10.56.199*	TCP	*	20 (FTP)	Drop

Slide 16

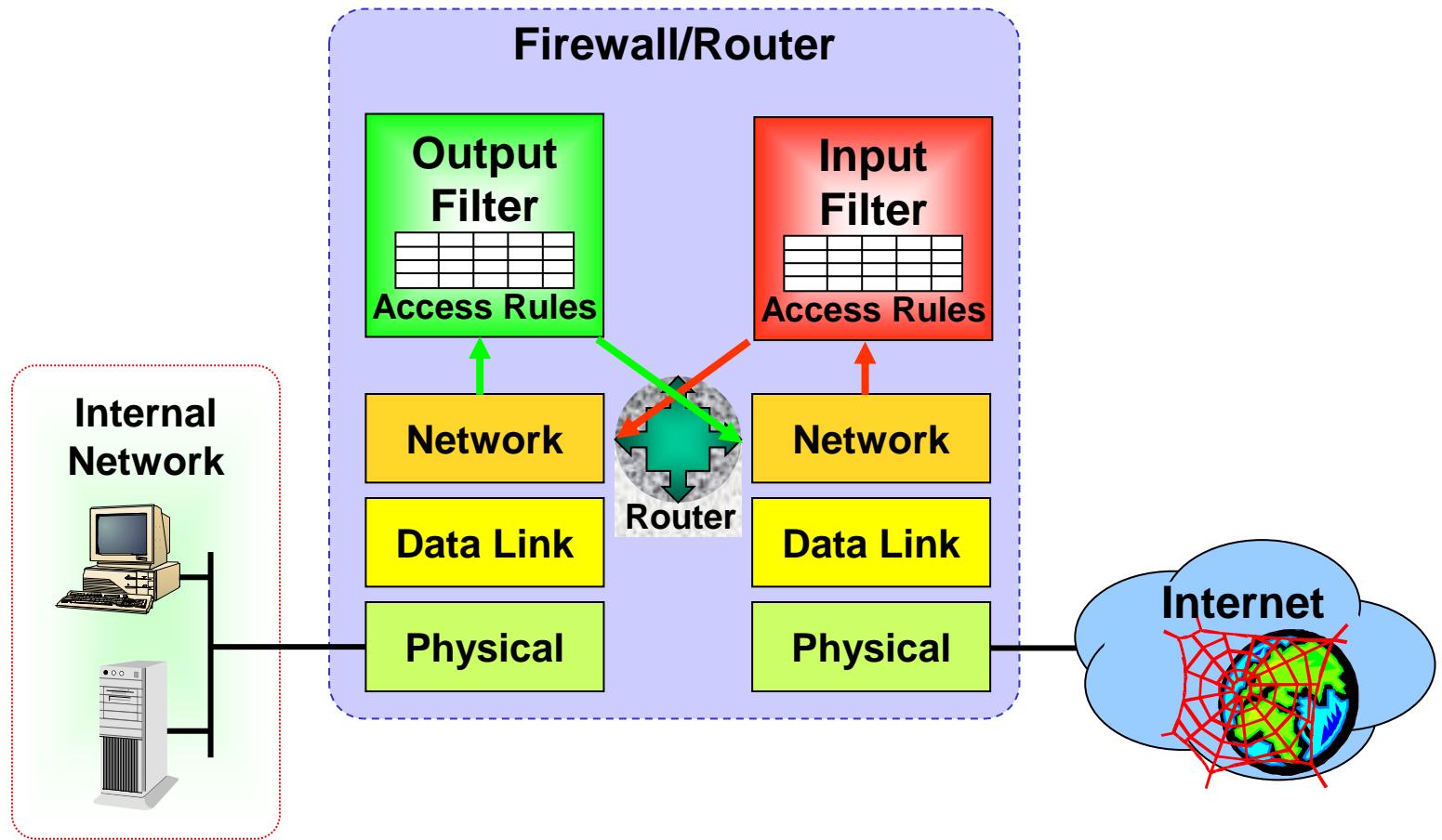
Web Access Through a Packet Filter Firewall

TABLE 14.1 Web Access Through a Packet Filter Firewall

Action	Src	Port	Dest	Port	Flags	Comment
block	*	*	*	*	*	Block all by default
allow	{Internal net}	*	*	80	*	Outgoing Web
allow	*	80	*	*	ACK	Incoming Web
allow	{Internal net}	*	*	21	*	Outgoing FTP control channel
allow	*	21	*	*	ACK	Incoming FTP control channel
allow	{Internal net}	*	*	≥1024	*	Outgoing FTP data
allow	*	≥1024	*	*	ACK	Incoming FTP data
allow	{Internal net}	*	*	443	*	Outgoing SSL
allow	*	443	*	*	ACK	Incoming SSL
allow	{Internal net}	*	*	70	*	Outgoing Gopher
allow	*	70	*	*	ACK	Incoming Gopher

ACK: = positive acknowledgement message for the sender from the receiver. Typically just one bit.

Packet Filtering Firewalls



Packet Filtering Firewalls: pros and cons

Advantages

- Simple, low cost, transparent to user

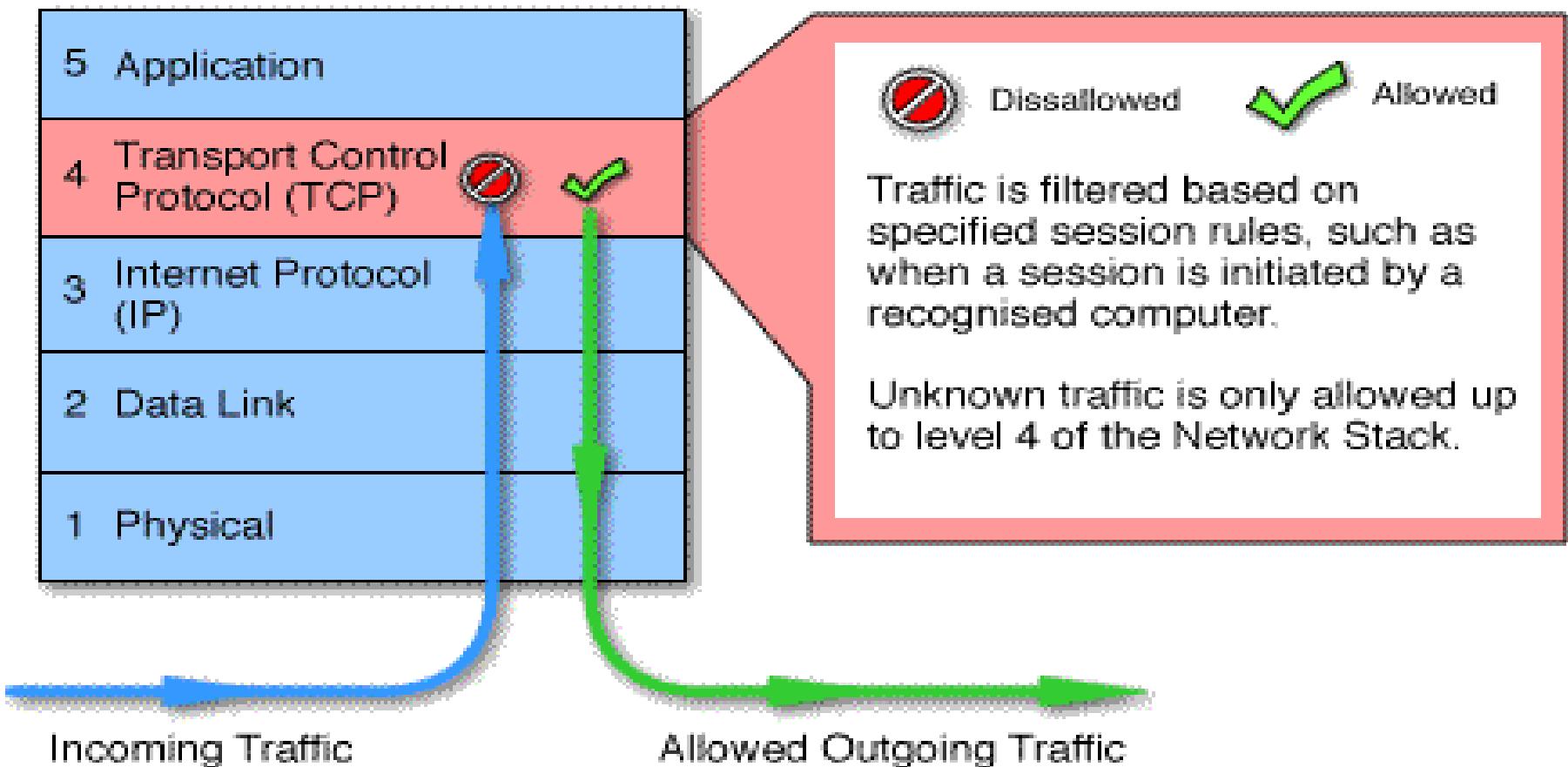
Disadvantages

- Hard to configure filtering rules
- Hard to test filtering rules
- Don't hide network topology (due to transparency)
- May not be able to provide enough control over traffic

Circuit Level Firewalls (Circuit Level Gateways)

- Circuit level gateways work at the session layer of the OSI model, or the TCP layer of TCP/IP
- Monitor TCP handshaking between packets to determine whether a requested session is legitimate.

Circuit Level Firewalls

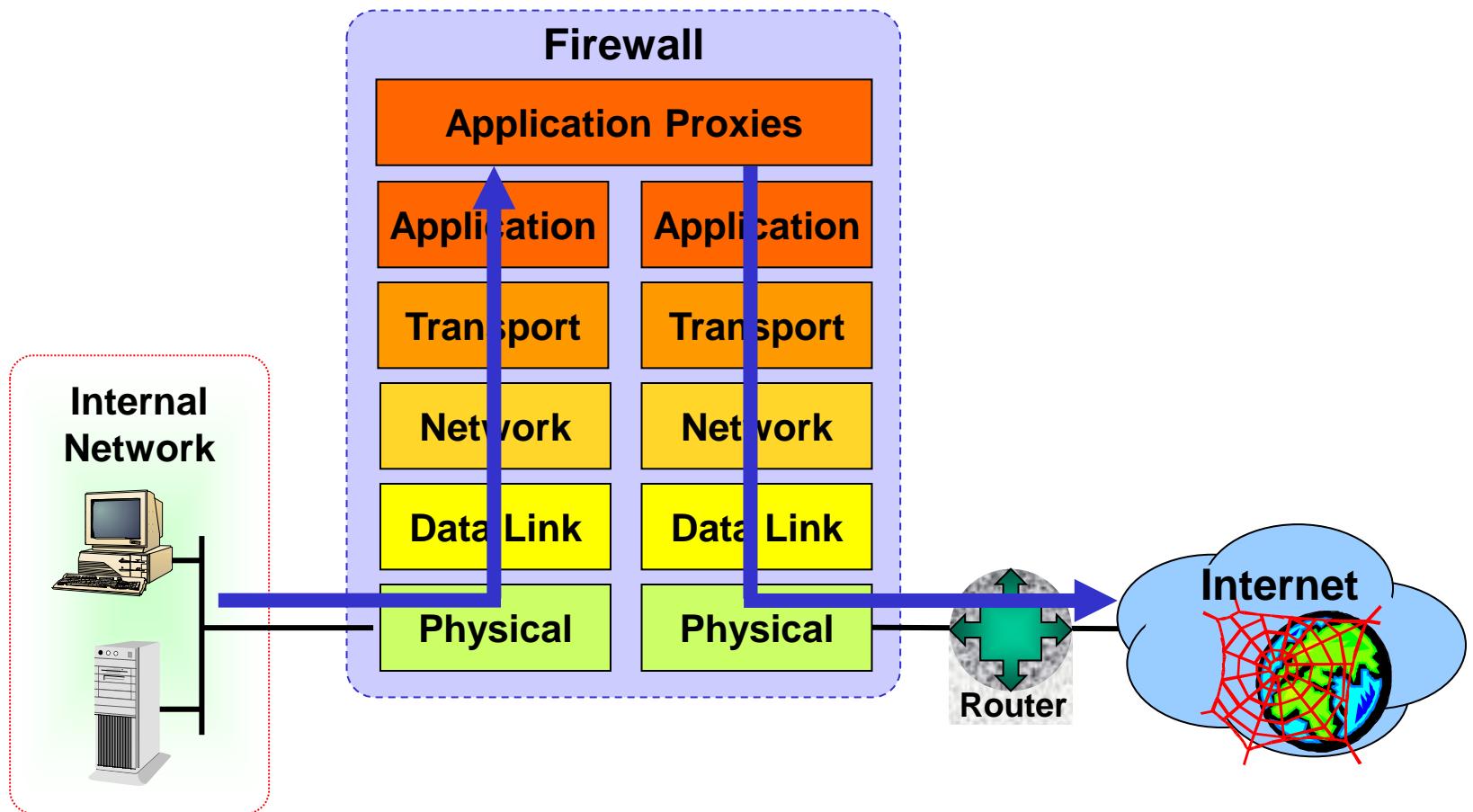


Application Gateway Firewalls (Proxy Firewalls)

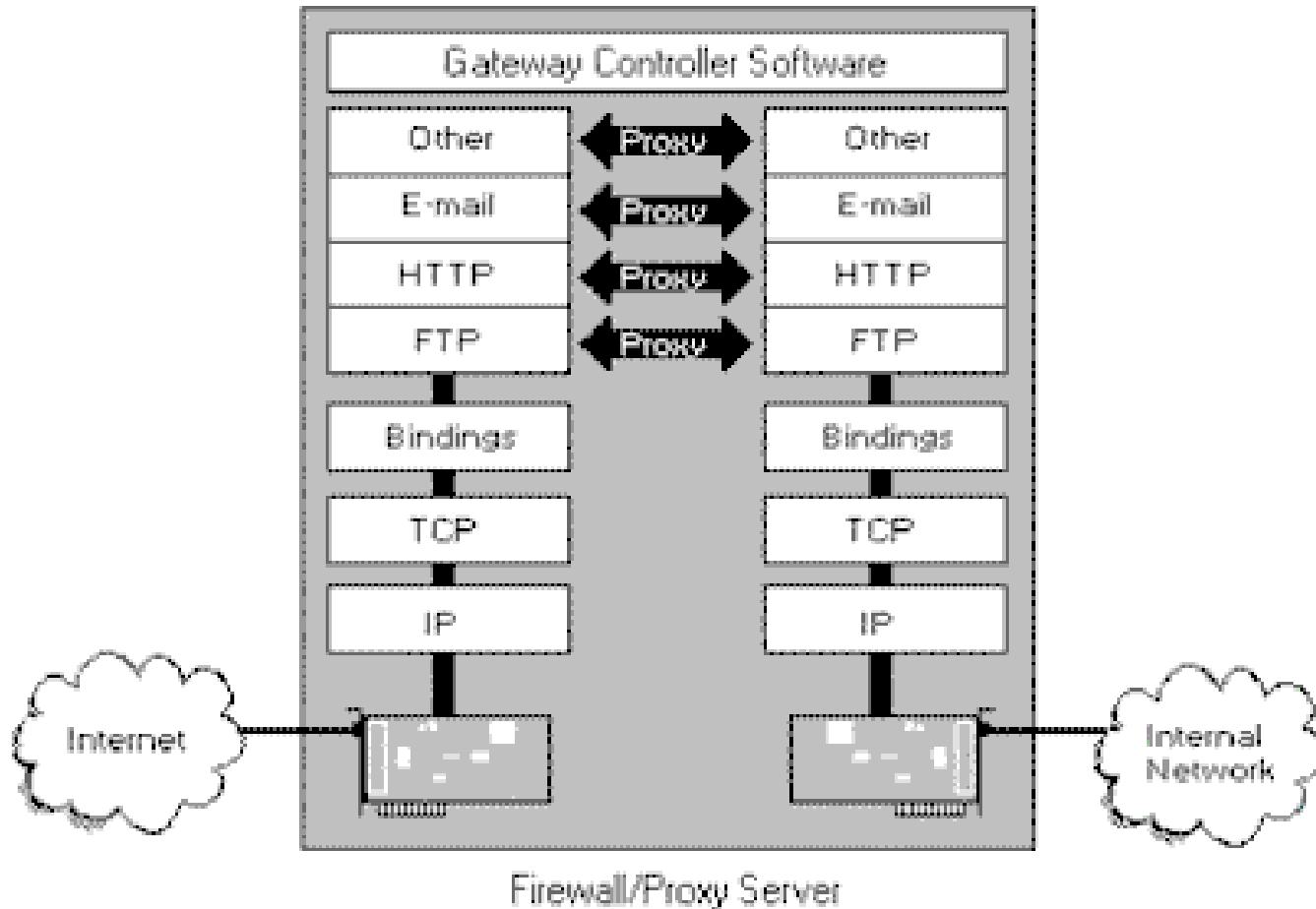
Application Gateway firewalls

- Similar to circuit-level gateways except that they are application specific.
- Every connection between two networks is made via an application program called a proxy
- Proxies are application or protocol specific
- Only protocols that have specific proxies configured are allowed through the firewall; all other traffic is rejected.
- Gateway that is configured to be a web proxy will not allow any ftp, gopher, telnet or other traffic through

Application Gateway Firewalls



Application Gateway Firewalls



Application Gateway Strengths

- Very secure if used in conjunction with an intelligent packet filtering firewall
- Well designed proxies provide excellent security

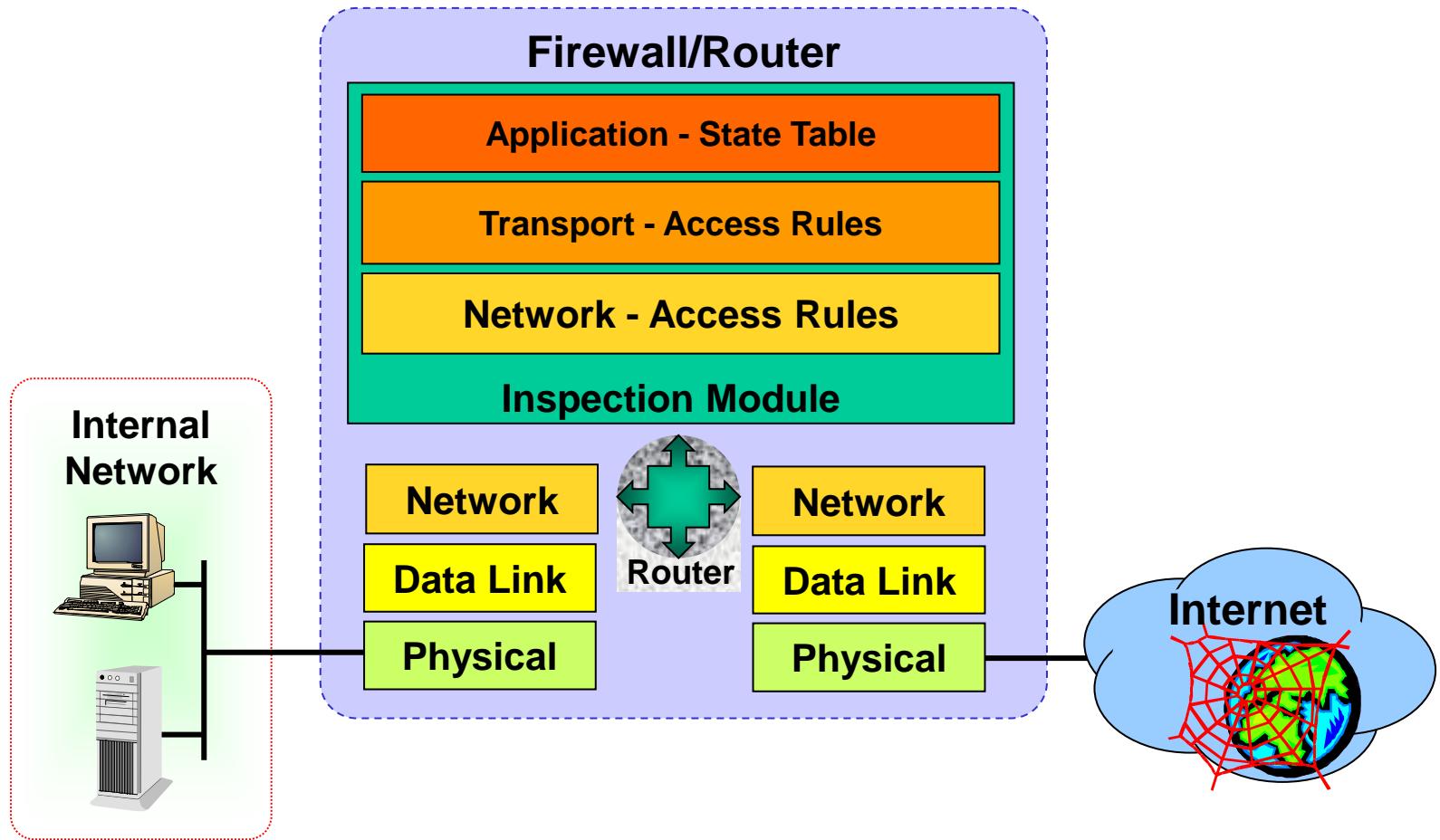
Application Gateway weaknesses

- Very CPU intensive
- Requires high performance host computer
- Host operating system liable to attack
- Many proxies are transparent to application
- Not transparent to users
- Expensive

Stateful Inspection Firewalls

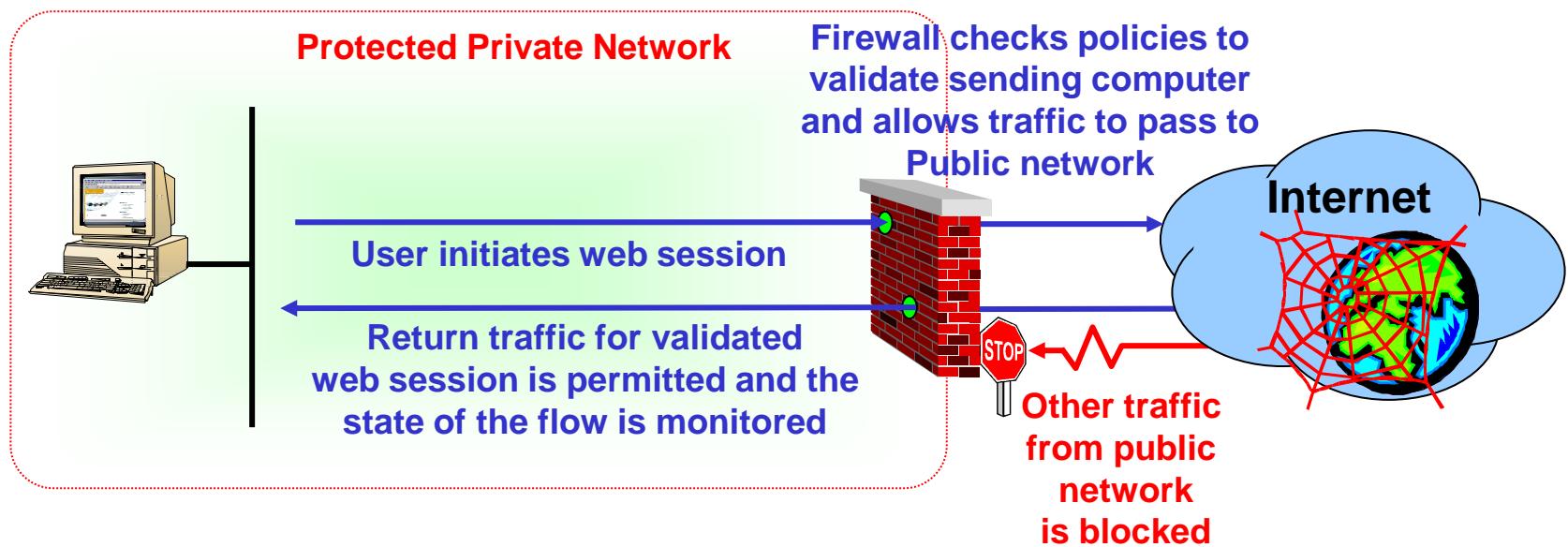
- Third generation firewall technology, often referred to as dynamic packet filtering
- Understands data in packets from the network layer (IP headers) up to the Application Layer
- Tracks the state of communication sessions

Stateful Inspection Firewalls



Dynamic Filtering

Stateful Inspection firewalls
dynamically open and close
ports (application specific
connection points) based
on access policies.



Stateful Inspection Strengths

- Monitors the state of all data flows
- Dynamically adapts filters based on defined policies and rules
- Easily adapted to new Internet applications
- Transparent to users
- Low CPU overheads

Stateful Inspection Weaknesses

- Need to provide new client program
- Might have problems with the availability of source code for various platforms

Stateful Inspection Firewalls

**These are among the most
secure firewalls available today**

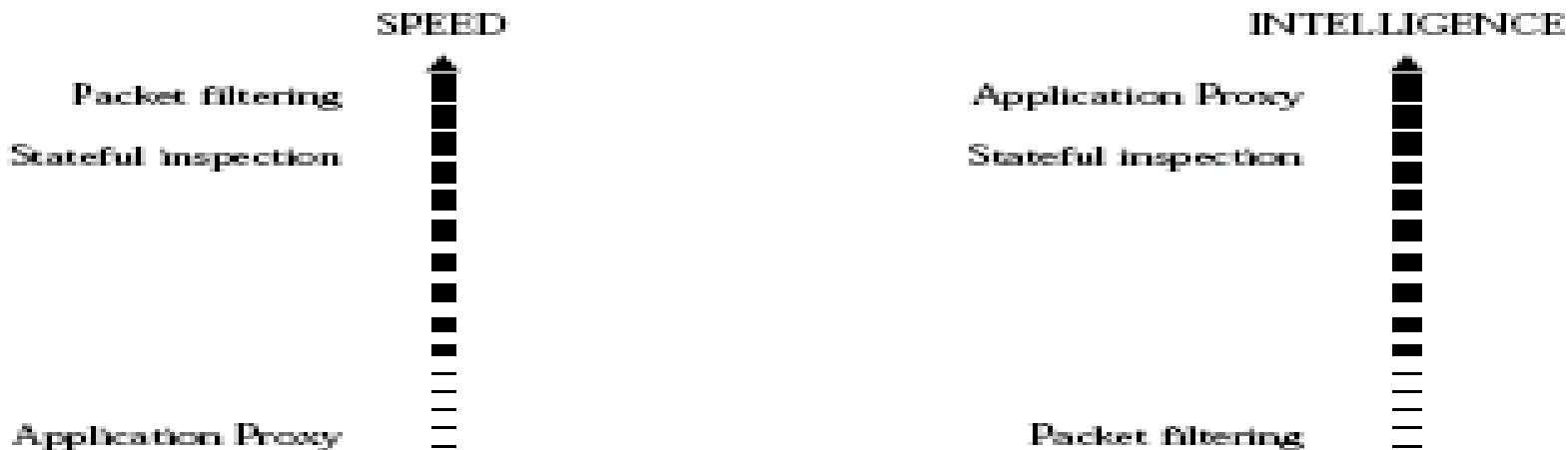
“fooling them can be a lot of work”

*Jon McCown, network security analyst for the - U.S.
National Computer Security Agency (NCSA)*

General Performance

FIREWALL PERFORMANCE SUMMARY

Technology	Speed	Flexibility	Intelligence
Packet filtering	V. Good	V.Good	Low
Application Proxy	Low	Low	V. Good
Stateful inspection	Good	Good	Good
Circuit gateway	Low	Low	Low

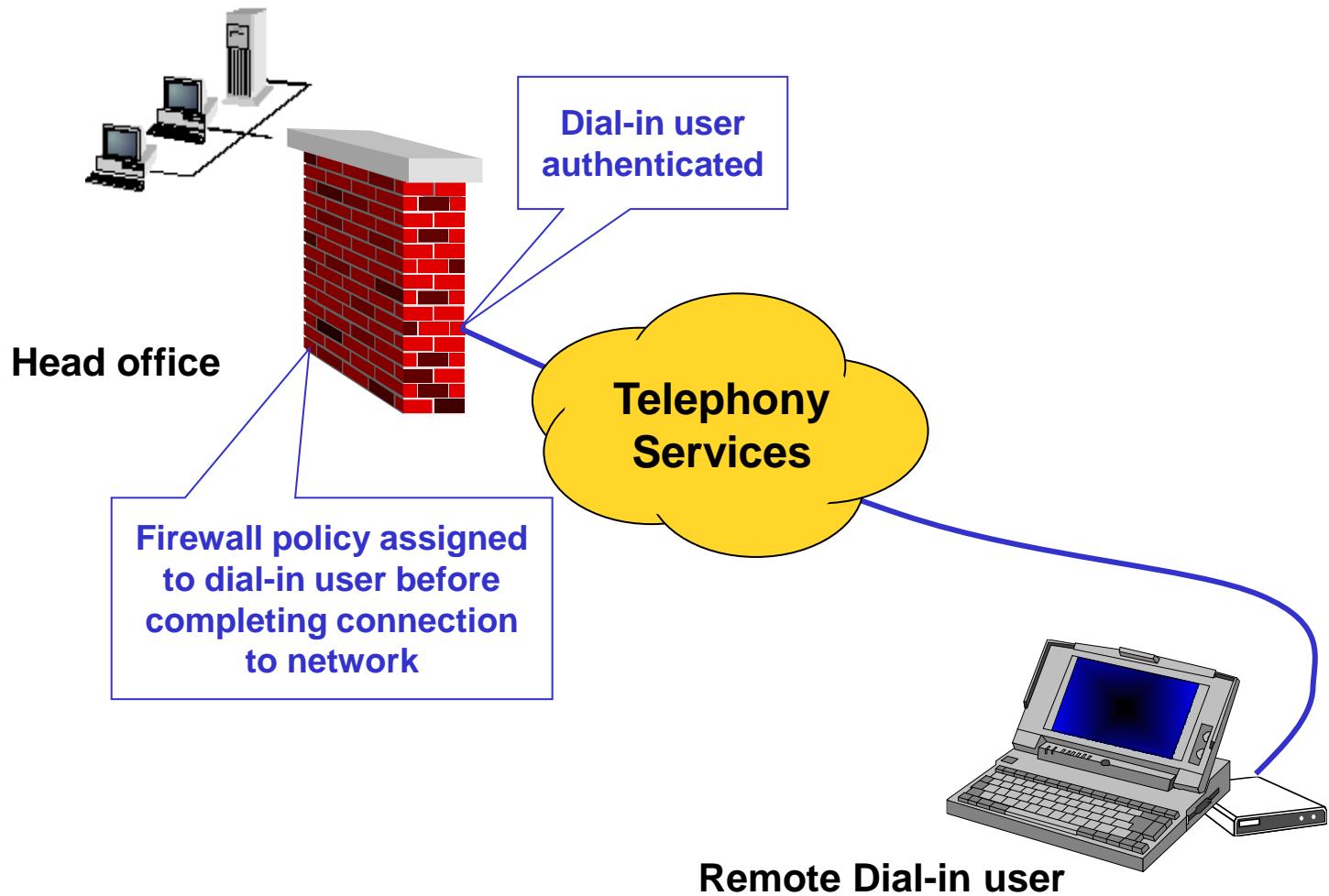


Other Issues about Firewalls

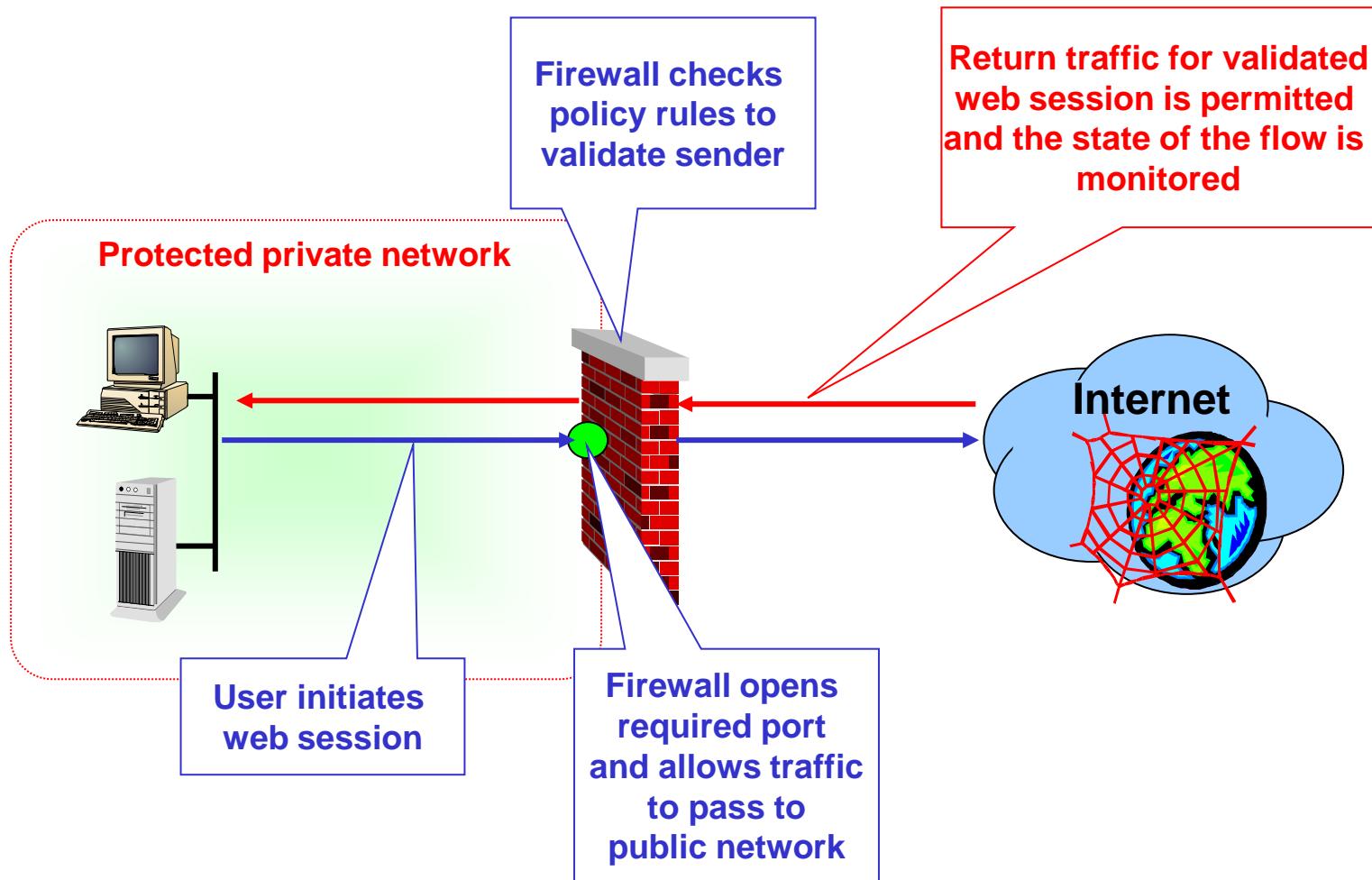
RADIUS Support

- Remote Authentication Dial-In User Services
 - A single, central security database for all system users
 - Centralised management of access lists

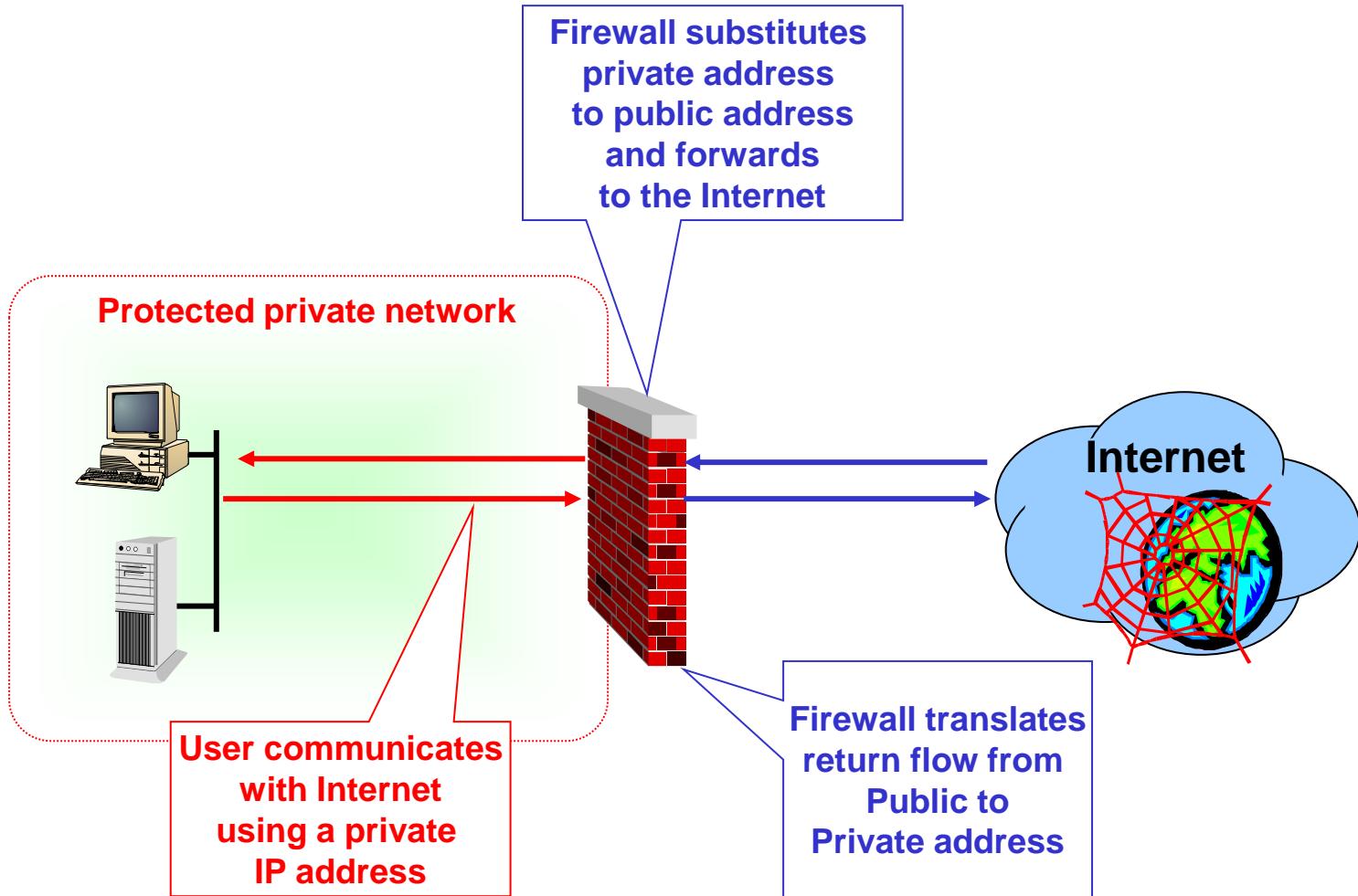
Remote access security



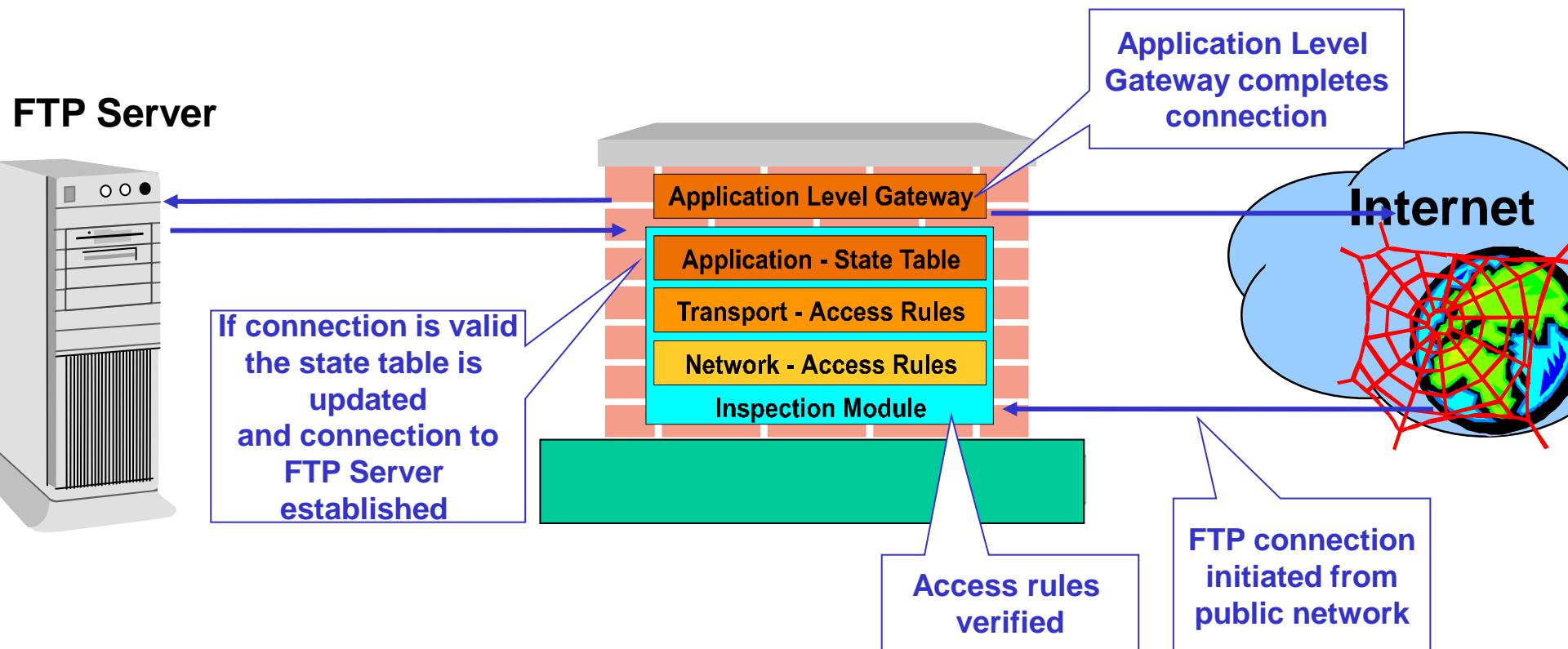
Stateful Inspection Implementation



Network Address Translation



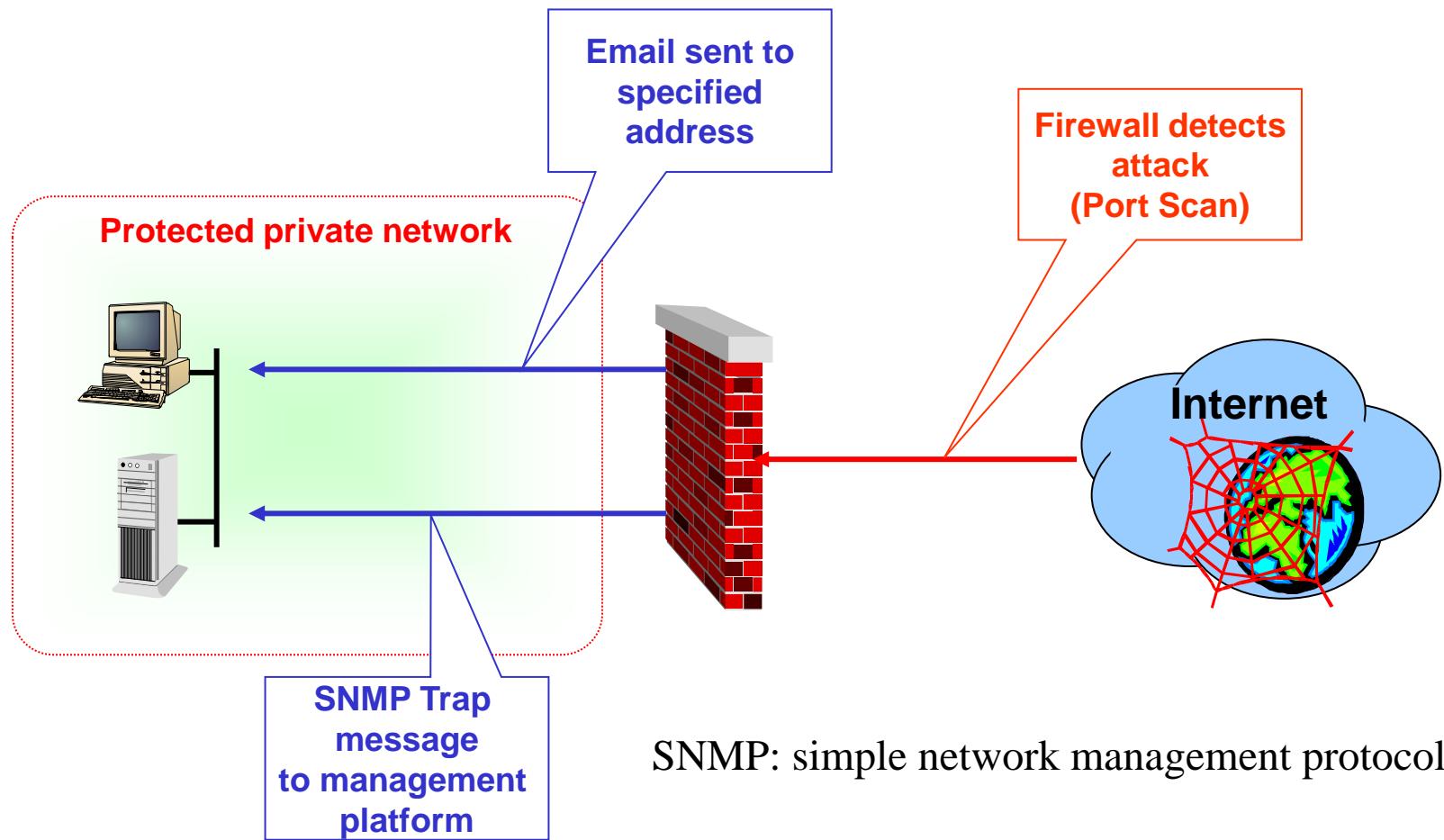
Application Level Gateway Example



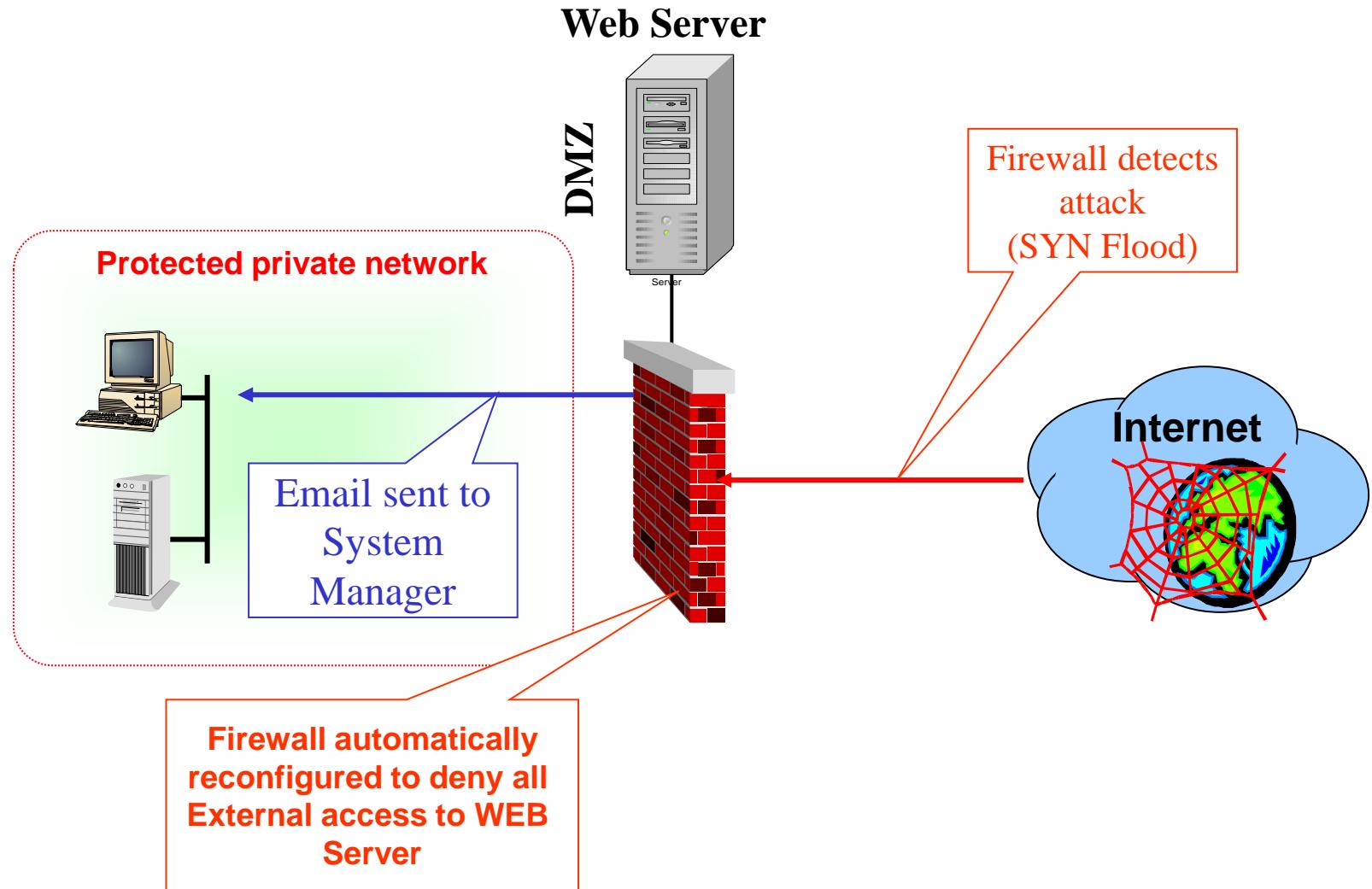
Session Logging

- The firewall can be configured to log an extensive range of events Including:
 - All denied packets
 - All allowed packets
 - Selected allowed and denied packet types
 - Etc.

Notification SNMP/SMTP



Notification and Reconfiguration



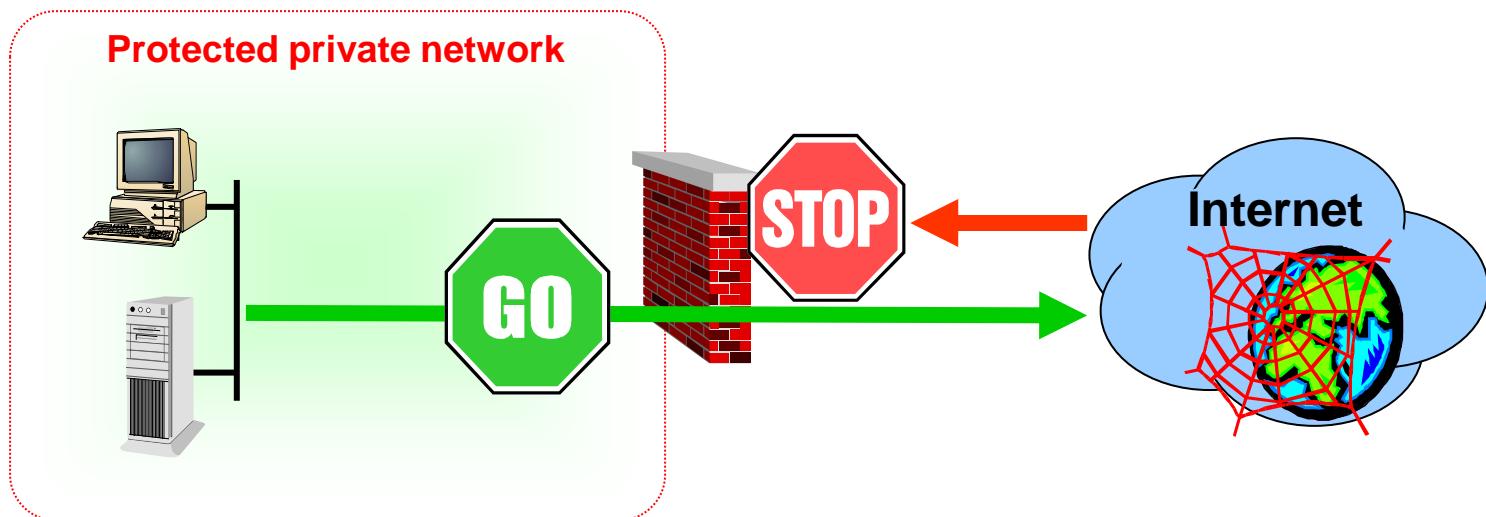
Secure Management

- Secure encrypted and authenticated remote management
 - Secure Shell “SSH”
 - RSA encryption keys 512 - 2048 bits
 - DES and Triple DES encryption for SSH sessions
 - Can limit access to specific user addresses

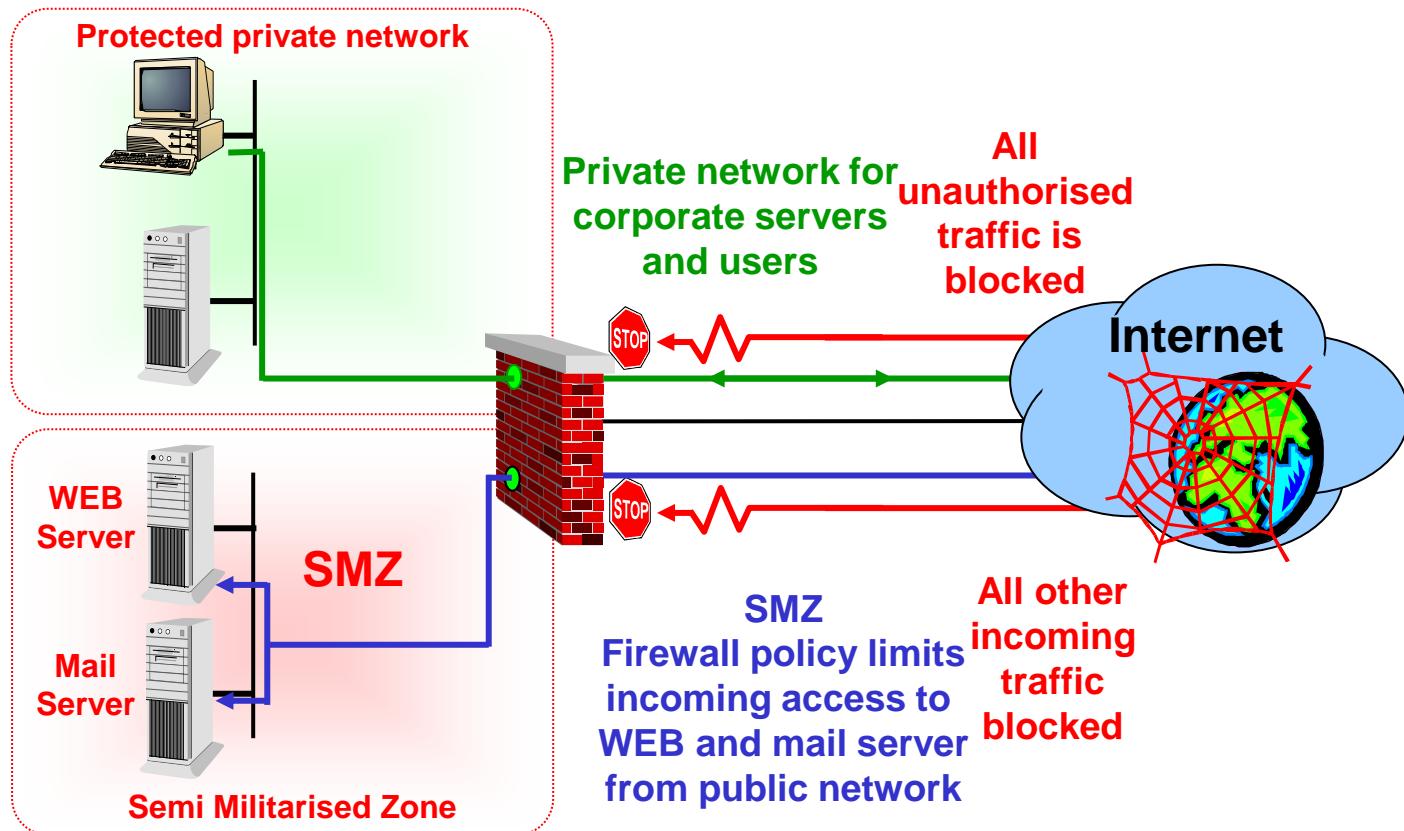
Network Configuration Examples

Protected Private Network

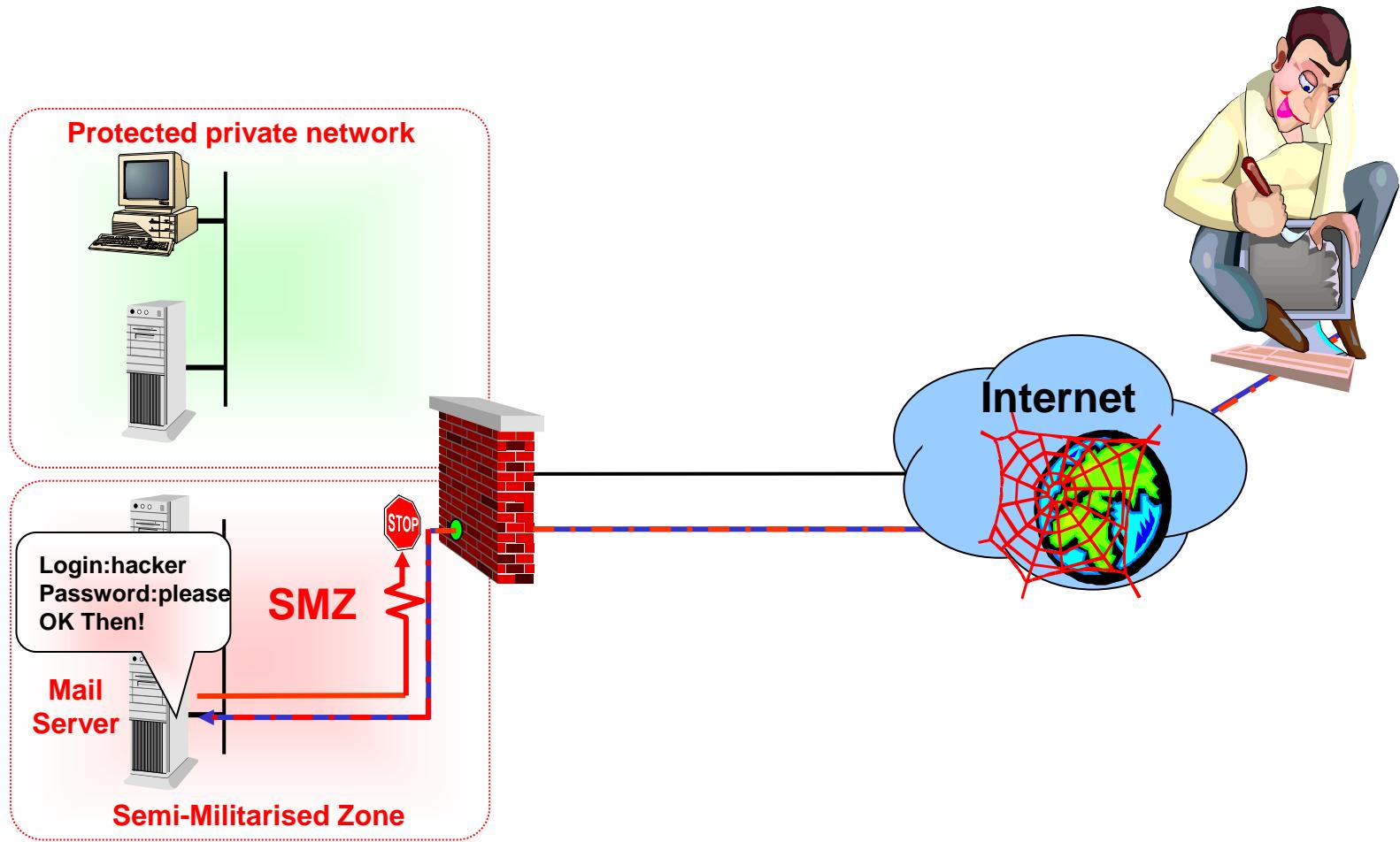
- Allow all access from private network to the Internet
- Deny all access from the Internet to the private network



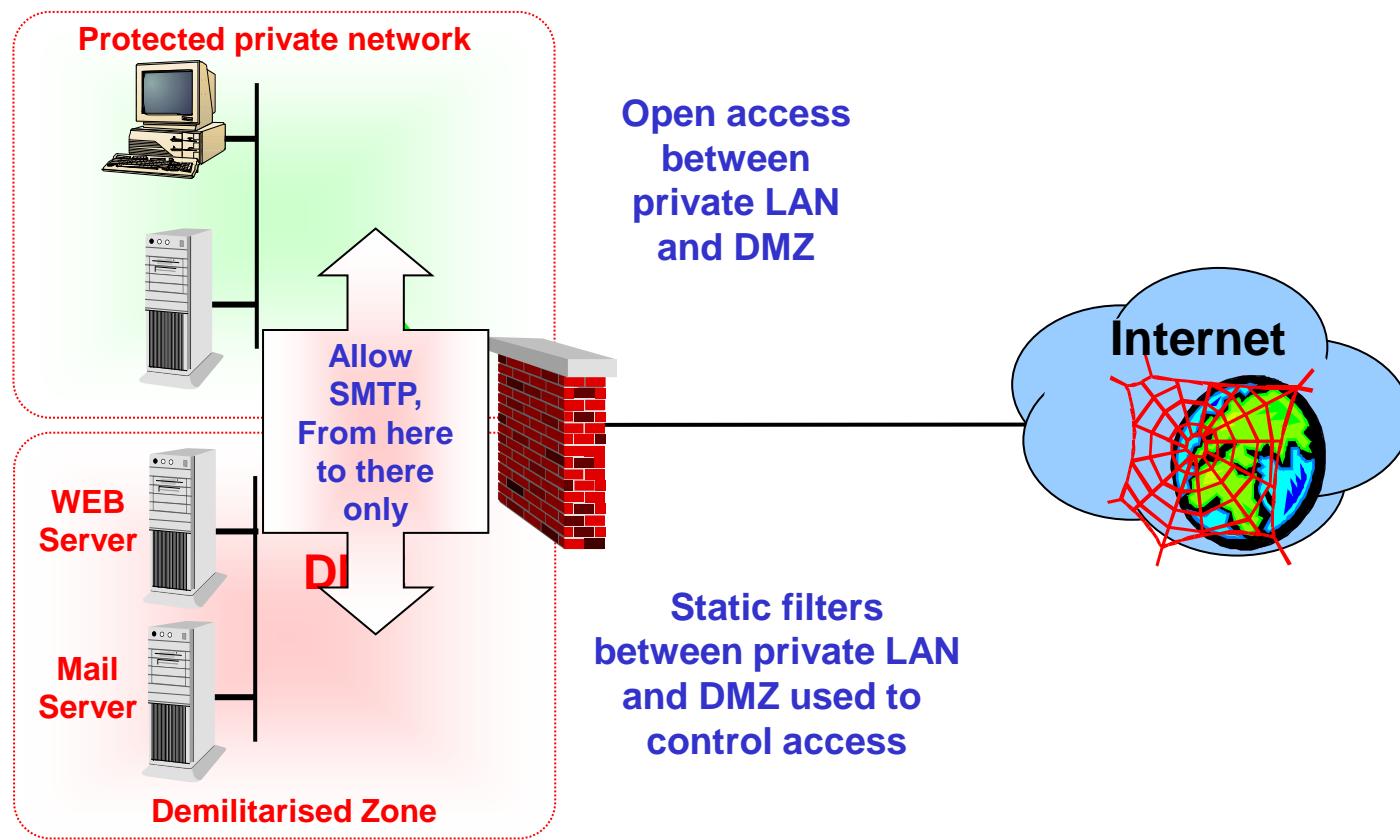
Semi-Militarised Zone



Private LAN Stays Secure



Demilitarised Zone



Concluding Remarks

- All that a firewall can do it's to control network activities between OSI levels 2 and 7.
- They cannot keep out data carried inside applications, such as viruses within email messages: there are just too many way of encoding data to be able to filter out this kind of threat.
- Although Firewalls provide a high level of security in today's Private Networks to the outside world we still need the assistance of other related Security components in order to guarantee proper network security.

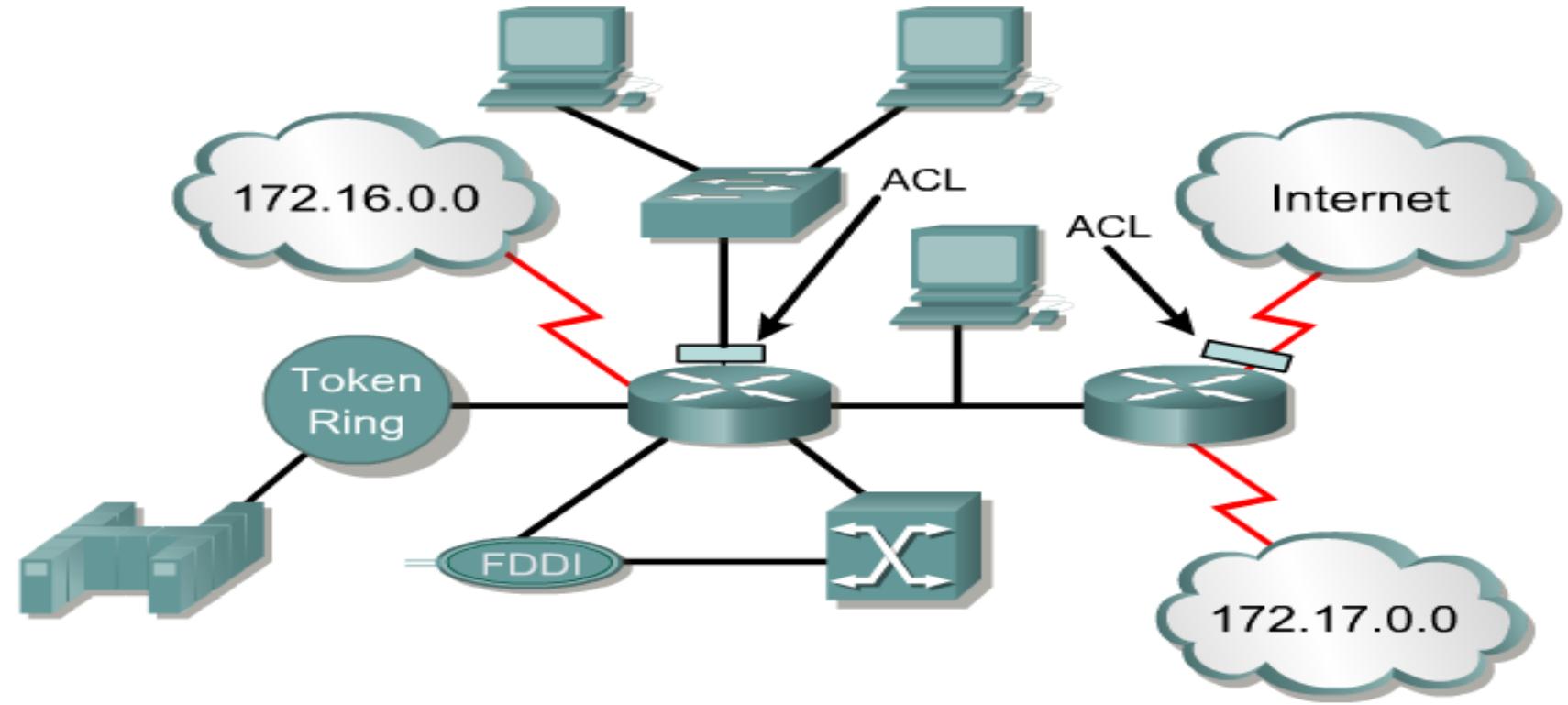
Overview of Access Control List (ACL)

- Network administrators must figure out how to deny unwanted access to the network while allowing internal users appropriate access to necessary services.
- Although security tools, such as passwords, and physical security devices are helpful, they often lack the flexibility of basic traffic filtering and the specific controls most administrators prefer.
- For example, a network administrator may want to allow users access to the Internet, but not permit external users telnet access into the LAN.
- Routers provide basic traffic filtering capabilities, such as blocking Internet traffic, with **access control lists (ACLs)**.
- An ACL is a sequential list of permit or deny statements that apply to addresses or upper-layer protocols.
- This presentation explains *standard* and *extended ACLs* as a means to control network traffic, and how **ACLs are used as part of a security solution**.

Overview of Access Control List (ACL) Contd.

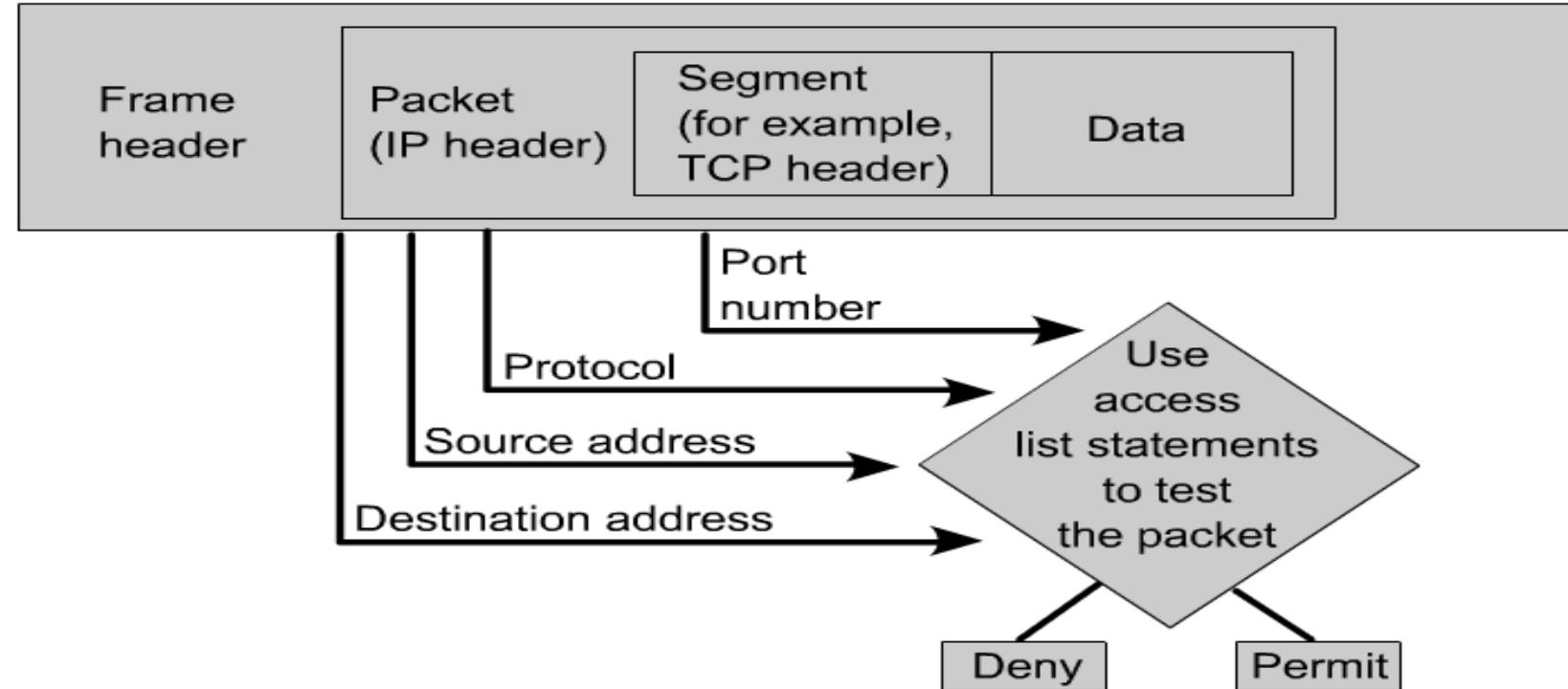
- In addition, it includes:
 - Tips, considerations, recommendations, and general guidelines on how to use ACLs,
 - Commands and configurations needed to create ACLs.
 - Examples of standard and extended ACLs
 - How to apply ACLs to router interfaces.

What are ACLs?



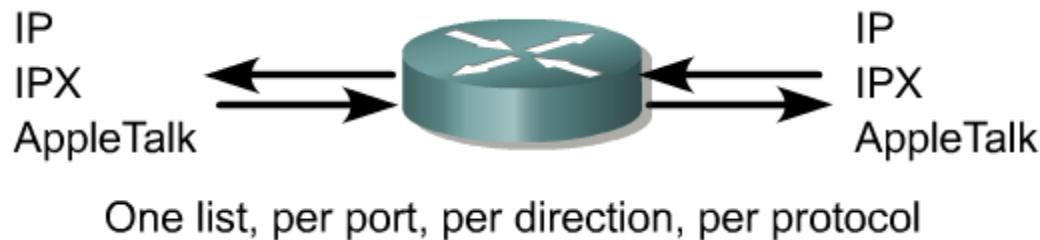
- **Note:** Much of the beginning of this are concepts. These concepts will become much clearer once we begin configuring ACLs.
- An access list is a sequential series of commands or filters.
- These lists tell the router what types of packets to:
 - **accept or deny**
- Acceptance and denial can be based on specified conditions.
- ACLs applied on the router's interfaces.

What are ACLs?



- The router examines each packet to determine whether to forward or drop it, based on the conditions specified in the ACL.
- *Some* ACL decision points are:
 - IP source address
 - IP destination addresses
 - UDP or TCP protocols
 - upper-layer (TCP/UDP) port numbers

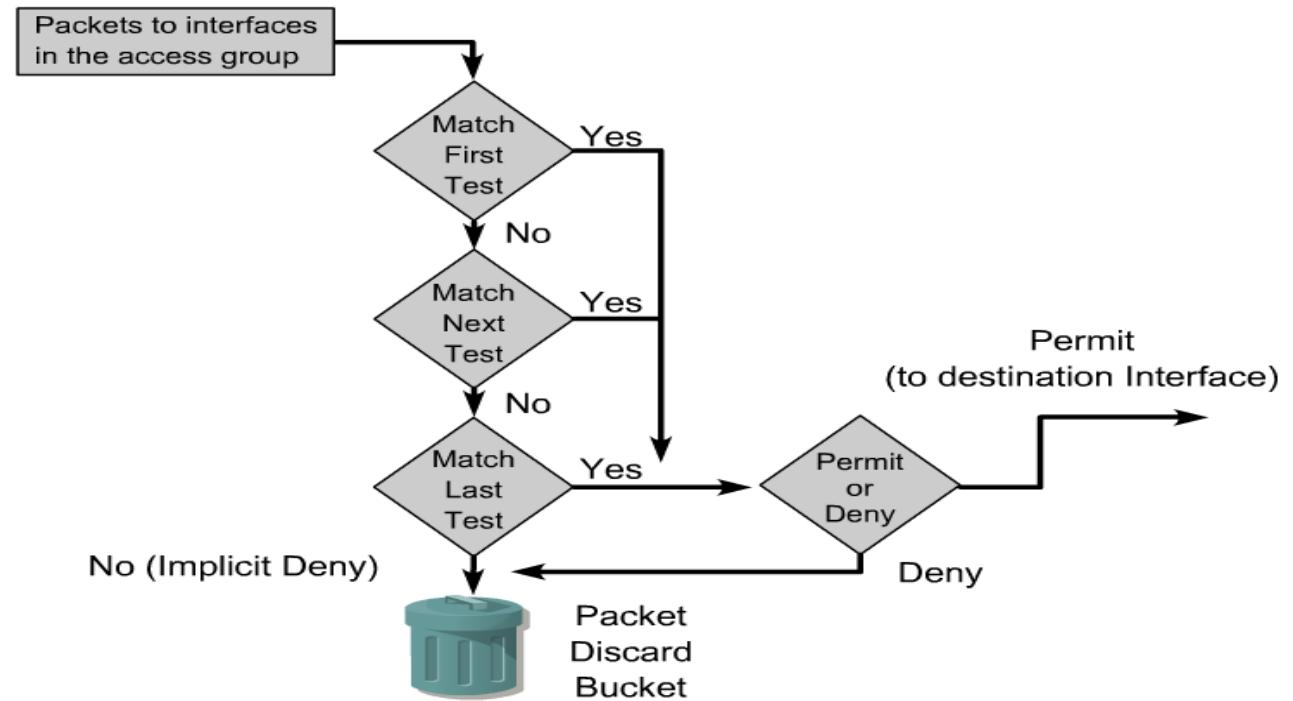
What are ACLs? Contd.



With two interfaces and three protocols running, this router could have a total of 12 separate ACLs applied.

- ACLs must be defined on a:
 - per-protocol (IP, IPX, AppleTalk)
 - per direction (in or out)
 - per port (interface) basis.
- ACLs control traffic in one direction at a time on an interface.
- A separate ACL would need to be created for each direction, one for inbound and one for outbound traffic.
- Finally every interface can have multiple protocols and directions defined.

How ACLs Work



- An ACL is a group of statements that define whether packets are accepted or rejected coming into an interface or leaving an interface.
- ACL statements operate in sequential, logical order.
- If a condition match is true, the packet is permitted or denied and the rest of the ACL statements are not checked.
- If all the ACL statements are unmatched, an implicit "***deny any***" statement is placed at the end of the list by ***default***. (not visible)
- When first learning how to create ACLs, it is a good idea to add the **implicit deny** at the end of ACLs to reinforce the dynamic presence of the command line.

How ACLs work Contd.

- **Access list statements** operate in sequential, logical order.
- They evaluate packets from the **top down**.
- Once there is an access list statement **match**, the packet skips the rest of the statements.
 - If a condition **match is true**, the packet is permitted or denied.
- There can be **only one access list** per protocol per interface.
- There is an implicit “deny any” at the end of every access list.
- **ACLs do not block packets that originate within the router. (ie. pings, telnets, etc.)**

Two types of ACLs

- Standard IP ACLs
 - Can only filter on source IP addresses
- Extended IP ACLs
 - Can filter on:
 - Source IP address
 - Destination IP address
 - Protocol (TCP, UDP)
 - Port Numbers (Telnet – 23, http – 80, etc.)
 - *and other parameters*

Creating Standard ACLs – 2 Steps

Step 1

Define the ACL by using the following command:

```
Router(config) #access-list access-list-number  
    {permit | deny} {test-conditions}
```

A global statement identifies the ACL. Specifically, the 1-99 range is reserved for standard IP. This number refers to the type of ACL. In Cisco IOS Release 11.2 or newer, ACLs can also use an ACL name, such as education_group, rather than a number.

The **permit** or **deny** term in the global ACL statement indicates how packets that meet the test conditions are handled by Cisco IOS software. **permit** usually means the packet will be allowed to use one or more interfaces that you will specify later. The final term or terms specifies the test conditions used by the ACL statement.

Step 2

Next, you need to apply ACLs to an interface by using the **access-group** command, as in this example:

```
Router(config-if) #{protocol} access-group access-list-number
```

All the ACL statements identified by *access-list-number* are associated with one or more interfaces. Any packets that pass the ACL test conditions can be permitted to use any interface in the access group of interfaces.

Creating ACLs – 2 Steps

Step 1

Define the ACL by using the following command:

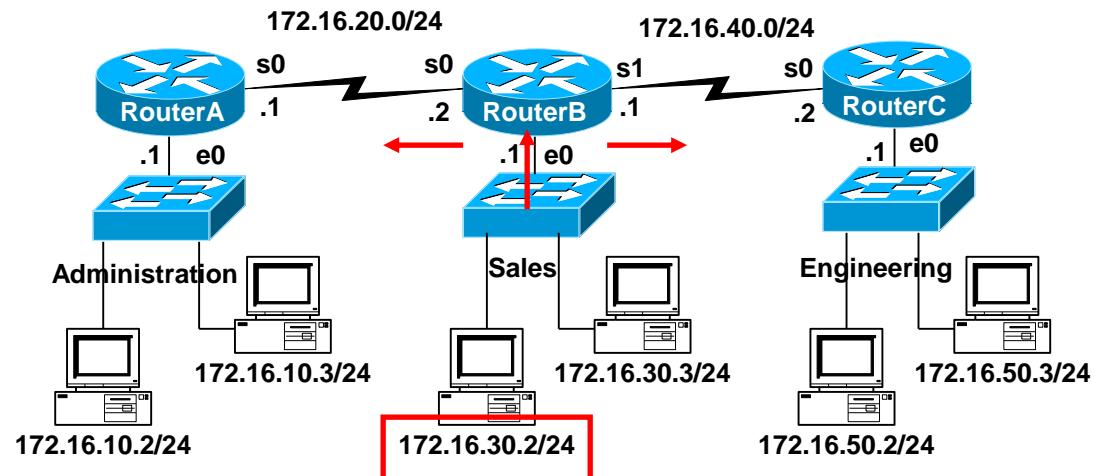
```
Router (config) #access-list access-list-number  
{permit | deny} {test-conditions}
```

A global statement identifies the ACL. Specifically, the 1-99 range is reserved for standard IP. This number refers to the type of ACL. In Cisco IOS Release 11.2 or newer, ACLs can also use an ACL name, such as education_group, rather than a number.

The **permit** or **deny** term in the global ACL statement indicates how packets that meet the test conditions are handled by Cisco IOS software. **permit** usually means the packet will be allowed to use one or more interfaces that you will specify later. The final term or terms specifies the test conditions used by the ACL statement.

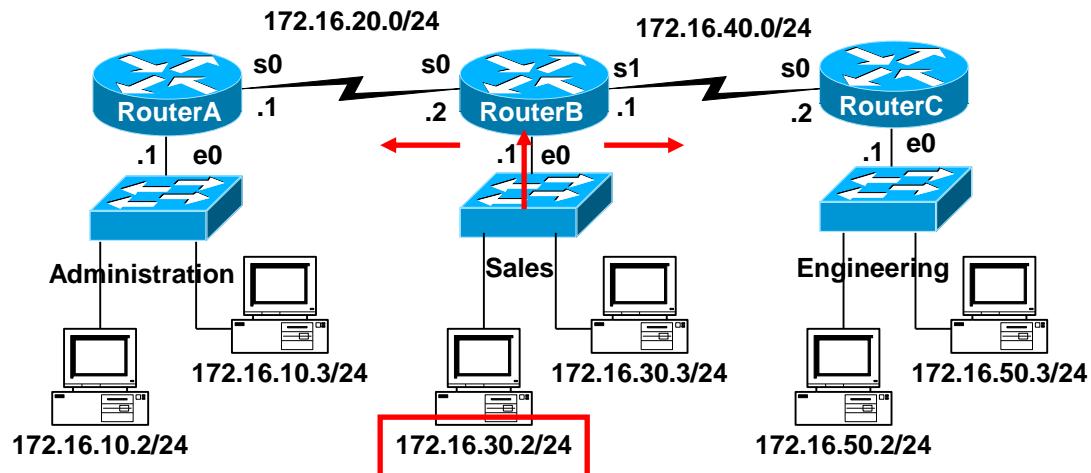
Protocol	Range
IP (Standard IP)	1-99
Extended IP	100-199
AppleTalk	600-699
IPX	800-899
Extended IPX	900-999
IPX Service Advertising Protocol	1000-1099

Learn by example!



- Task:
 - Permit only the host 172.16.30.2 from exiting the Sales network.
 - Deny all other hosts on the Sales network from leaving the 172.16.30.0/24 network.

Learn by example!



Step 1 – ACL statements Implicit deny any, which is automatically added.

Test Condition

```
RouterB(config) #access-list 10 permit 172.16.30.2
```

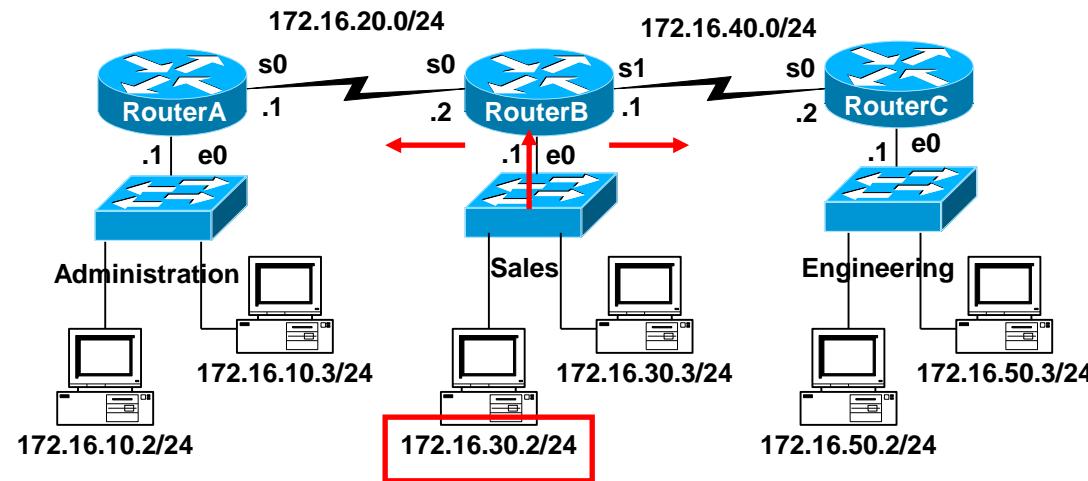
Implicit "deny any" -do not need to add this, discussed later

```
RouterB(config) #access-list 10 deny 0.0.0.0 255.255.255.255
```

```
Router(config) #access-list access-list-number
{permit | deny} {test-conditions}
```

Protocol	Range
IP	(Standard IP)

From Cisco Web Site



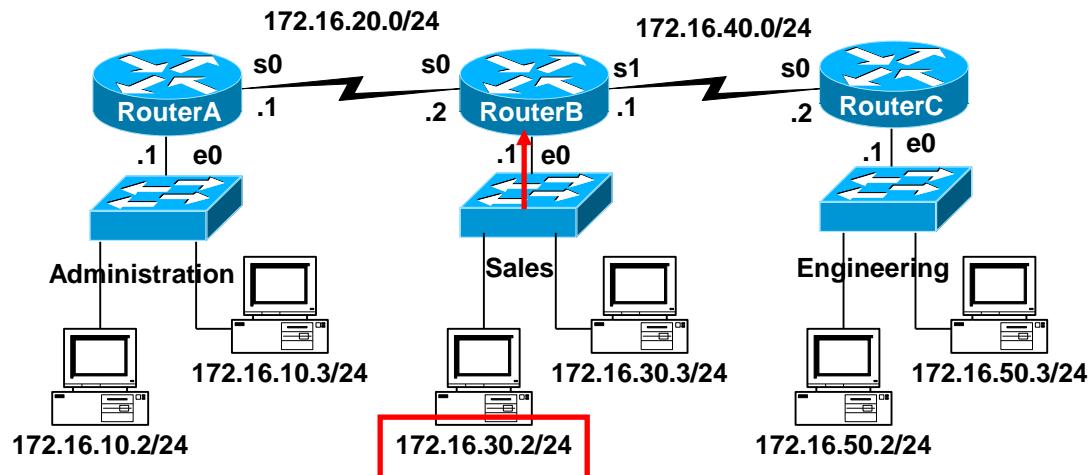
Applying ACLs

- You can define ACLs without applying them.
- However, **the ACLs will have no effect until they are applied to the router's interface.**
- It is a good practice to apply the Standard ACLs on the interface closest to the destination of the traffic and Extended ACLs on the interface closest to the source.

Defining In, Out, Source, and Destination

- **Out** - Traffic that has already been routed by the router and is leaving the interface
- **In** - Traffic that is arriving on the interface and which will be routed router.

Learn by example!



Step 2 – Apply to an interface(s)

```
RouterB(config) #access-list 10 permit 172.16.30.2
```

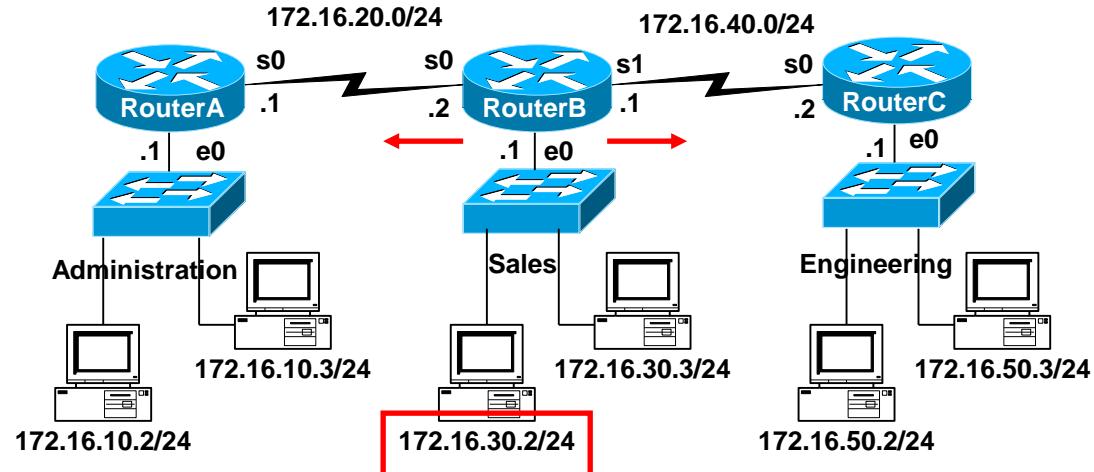
Implicit "deny any" -do not need to add this, discussed later

```
RouterB(config) #access-list 10 deny 0.0.0.0 255.255.255.255
```

```
RouterB(config) # interface e 0
```

```
Router(config-if)#{protocol} access-group access-list-number
```

Learn by example!



Step 2 – Or the outgoing interfaces... Which is preferable and why?

```
RouterB(config)#access-list 10 permit 172.16.30.2
```

Implicit "deny any" - do not need to add this, discussed later

```
RouterB(config)#access-list 10 deny 0.0.0.0 255.255.255.255
```

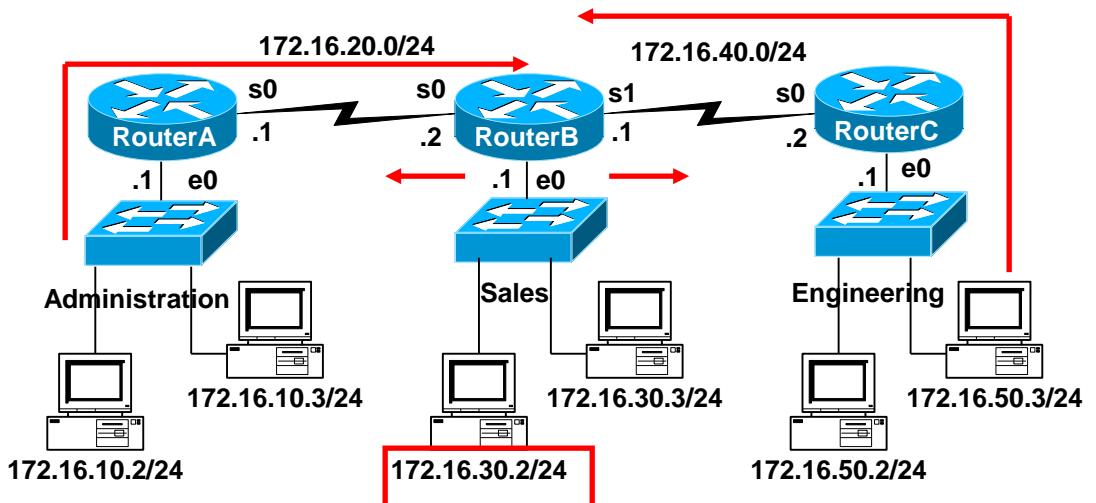
```
RouterB(config)# interface s 0
```

```
RouterB(config-if)# ip access-group 10 out
```

```
RouterB(config)# interface s 1
```

```
RouterB(config-if)# ip access-group 10 out
```

Learn by example!

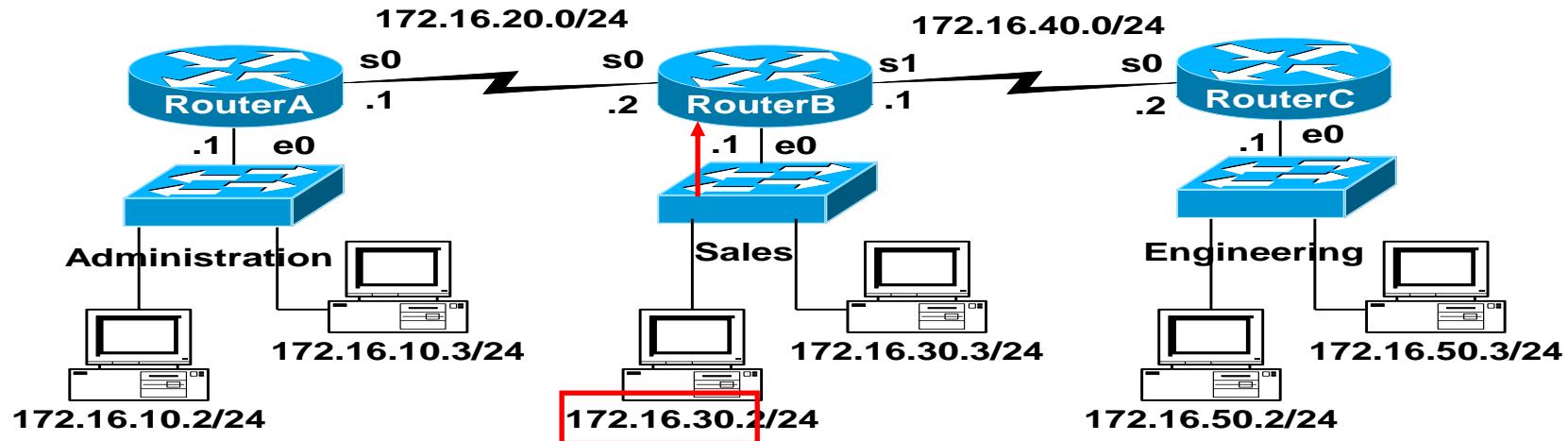


Because of the implicit deny any, this has an adverse affect of also denying packets from Administration from reaching Engineering, and denying packets from Engineering from reaching Administration.

```
RouterB(config)#access-list 10 permit 172.16.30.2
Implicit "deny any" -do not need to add this, discussed later
RouterB(config)#access-list 10 deny 0.0.0.0 255.255.255.255

RouterB(config)# interface s 0
RouterB(config-if)# ip access-group 10 out
RouterB(config)# interface s 1
RouterB(config-if)# ip access-group 10 out
```

Learn by example!



Preferred, this access list will work to all existing and new interfaces on RouterB.

```
RouterB(config)#access-list 10 permit 172.16.30.2
```

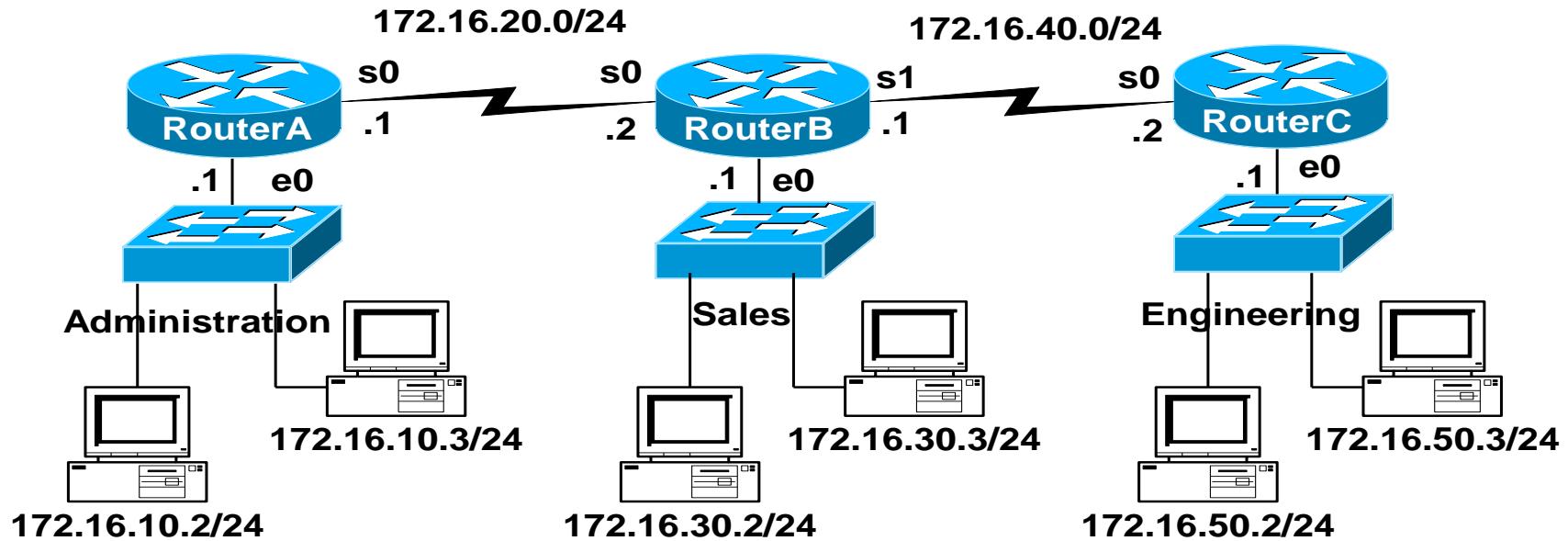
Implicit “deny any” -do not need to add this, discussed later

```
RouterB(config)#access-list 10 deny 0.0.0.0 255.255.255.255
```

```
RouterB(config)# interface e 0
```

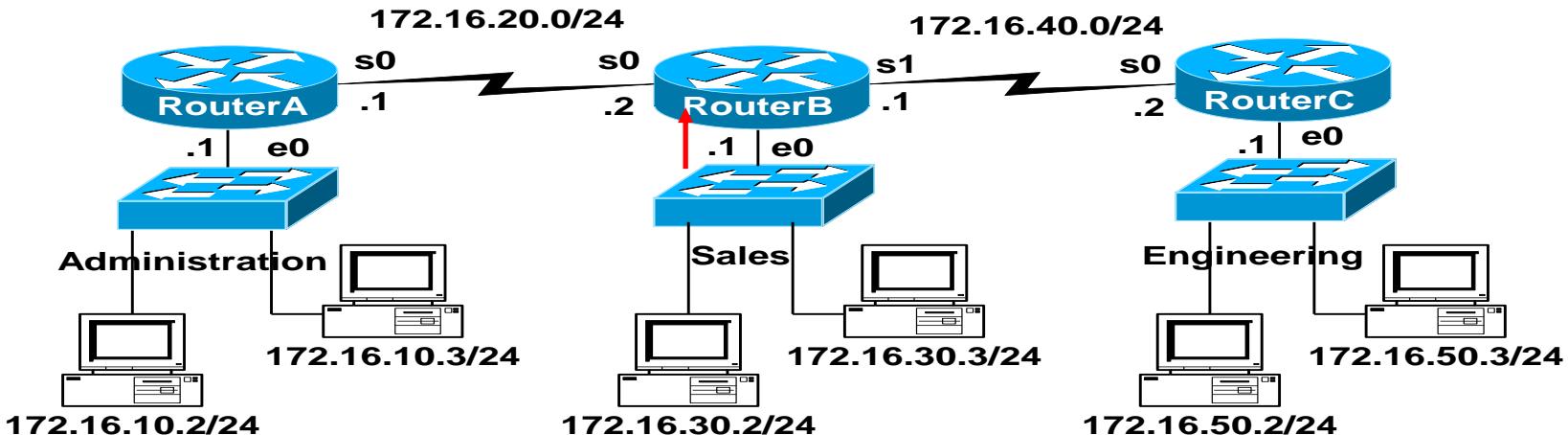
```
RouterB(config-if)# ip access-group 10 in
```

Example 2



- Task:
 - Permit only the hosts 172.16.30.2, 172.16.30.3, 172.16.30.4, 172.16.30.5 from exiting the Sales network.
 - Deny all other hosts on the Sales network from leaving the 172.16.30.0/24 network.

Example 2

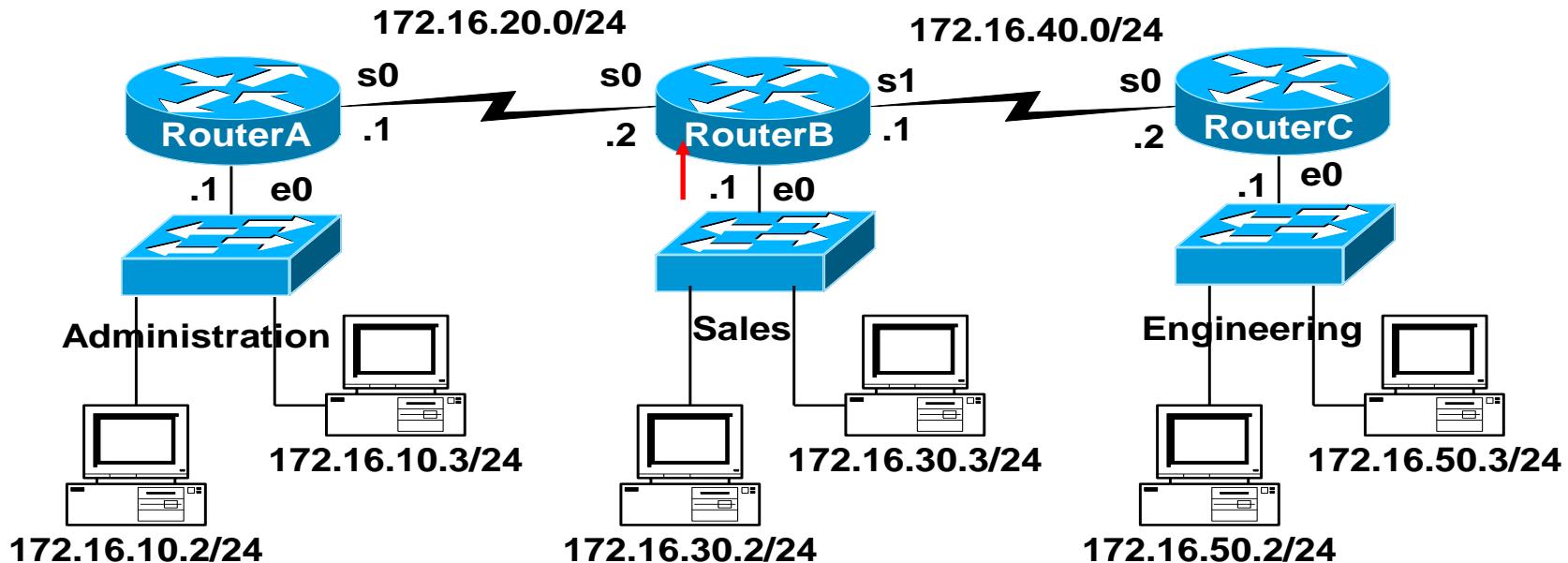


Once a condition is met, all other statements are ignored, so the implicit *deny any* only applies to not-matched packets.

```
RouterB(config)#access-list 10 permit 172.16.30.2
RouterB(config)#access-list 10 permit 172.16.30.3
RouterB(config)#access-list 10 permit 172.16.30.4
RouterB(config)#access-list 10 permit 172.16.30.5
Implicit "deny any" -do not need to add this, discussed later
RouterB(config)#access-list 10 deny 0.0.0.0 255.255.255.255

RouterB(config)# interface e 0
RouterB(config-if)# ip access-group 10 in
```

Example 2



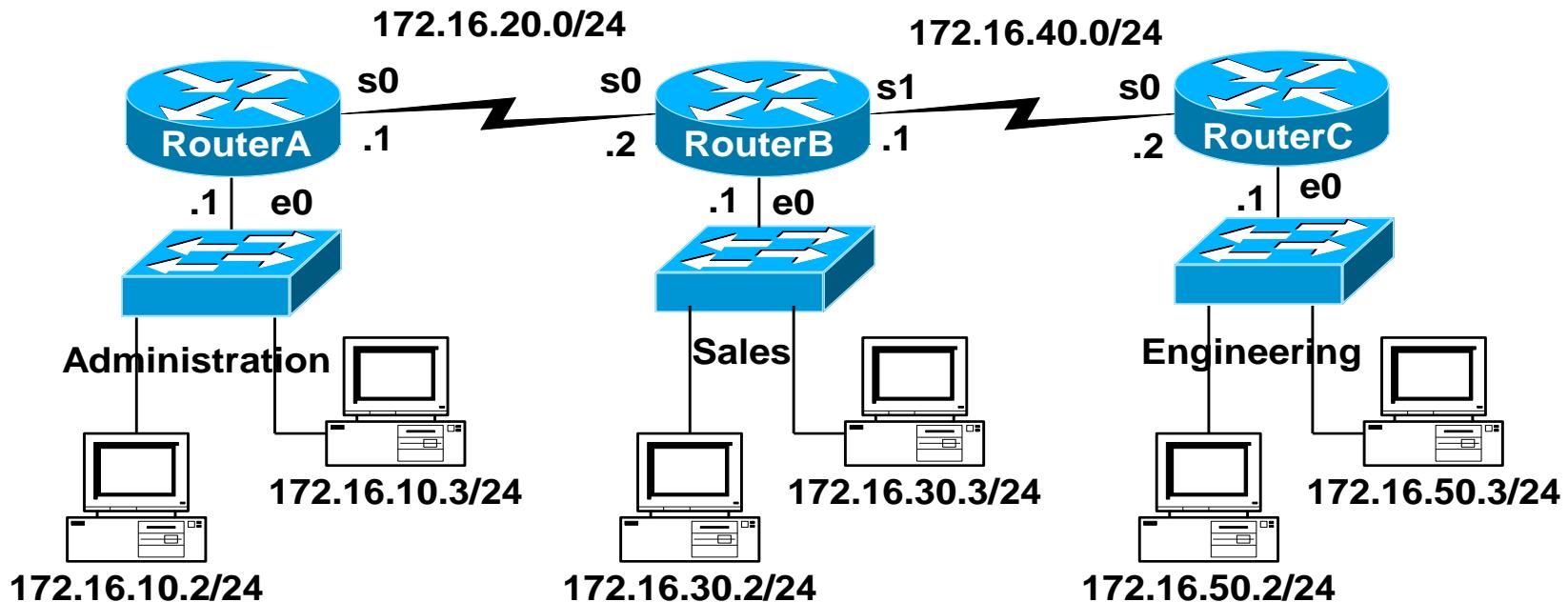
To remove an Access List, use the **no access-list** command. Removing the **access-group** only from the interface leaves the access list, but they are not currently being applied. Usually, best to remove it from both.

```
RouterB(config) #no access-list 10
```

```
RouterB(config) # interface e 0
```

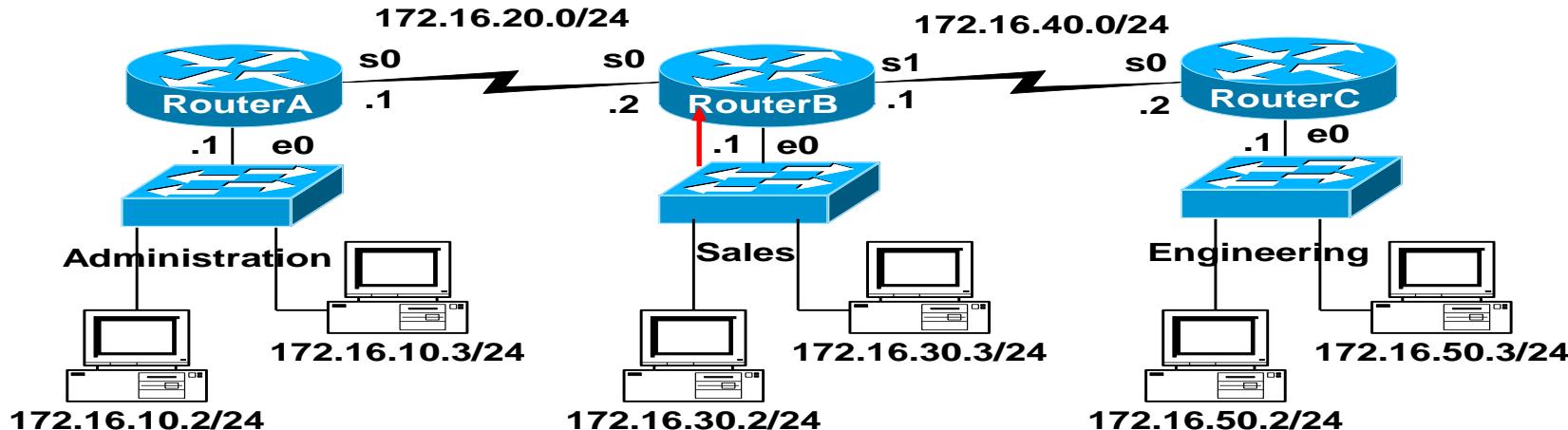
```
RouterB(config-if) # no ip access-group 10 in
```

Example 3



- Task:
 - Deny only the host 172.16.30.2 from exiting the Sales network.
 - Permit all other hosts on the Sales network to leave the 172.16.30.0/24 network.
- Keyword “any” can be used to represent all IP Addresses.

Example 3

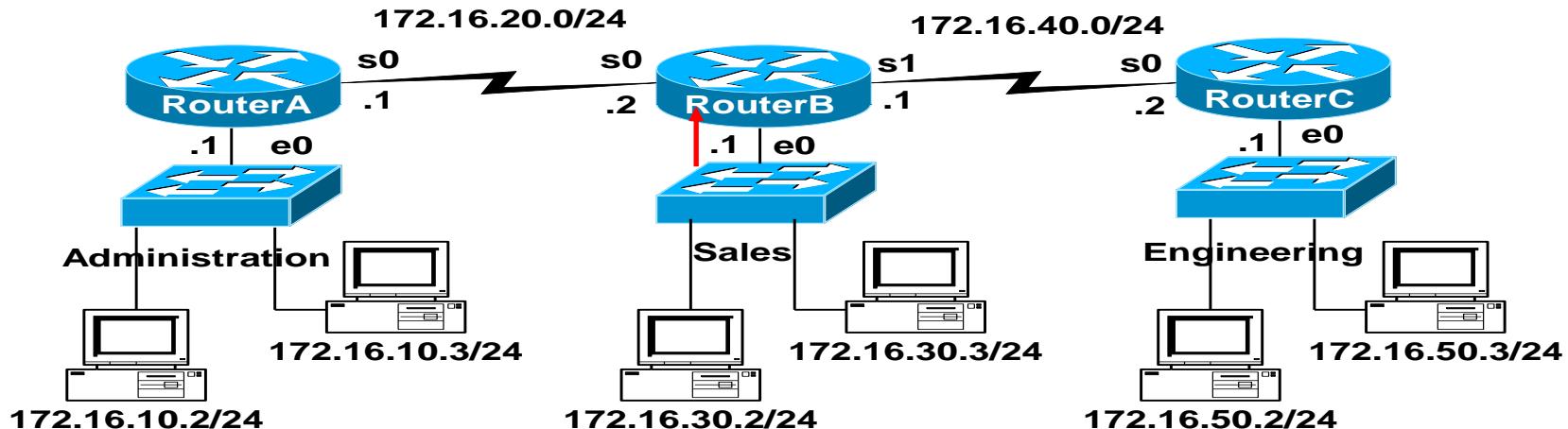


Order matters! What if these two statements were reversed? Does the *implicit deny any* ever get a match? No, the permit any will cover all other packets.

```
RouterB(config) #access-list 10 deny 172.16.30.2
RouterB(config) #access-list 10 permit any
Implicit "deny any" -do not need to add this, discussed later
RouterB(config) #access-list 10 deny 0.0.0.0 255.255.255.255

RouterB(config) # interface e 0
RouterB(config-if) # ip access-group 10 in
```

Example 3



Order matters! In this case all packets would be permitted, because all packets would match the first access list statement. Once a condition is met, all other statements are ignored. The second access list statement and the implicit deny any would never be used. This would not do what we want.

```
RouterB(config)#access-list 10 permit any
```

```
RouterB(config)#access-list 10 deny 172.16.30.2
```

Implicit “deny any” -do not need to add this, discussed later

```
RouterB(config)#access-list 10 deny 0.0.0.0 255.255.255.255
```

```
RouterB(config)# interface e 0
```

```
RouterB(config-if)# ip access-group 10 in
```

Note on Inbound Access Lists

- When an access lists applied to an **inbound** interface, the packets are checked against the access list before any routing table lookup process occurs.
- We will see how **outbound access list** work in a moment, but they are applied after the forwarding decision is made, after the routing table lookup process takes place and an exit interface is determined.
- **Once a packet is denied by an ACL, the router sends an ICMP “Destination Unreachable” message, with the code value set to “Administratively Prohibited” to the source of the packet.**

```
RouterB(config)#access-list 10 deny 172.16.30.2
```

```
RouterB(config)#access-list 10 permit any
```

Implicit “deny any” (do not need to add this, discussed later):

```
RouterB(config)#access-list 10 deny 0.0.0.0 255.255.255.255
```

```
RouterB(config)# interface e 0
```

```
RouterB(config-if)# ip access-group 10 in
```

Notes from www.cisco.com

- Traffic coming into the router is compared to ACL entries based on the order that the entries occur in the router.
- New statements are added to the end of the list.
- The router keeps looking until it has a match.
- If no matches are found when the router reaches the end of the list, the traffic is denied.
- For this reason, you should have the frequently hit entries at the top of the list.
- There is an "implied deny" for traffic that is not permitted.
- A single-entry ACL with only one "deny" entry has the effect of denying all traffic.
- You must have at least one "permit" statement in an ACL or all traffic will be blocked.

access-list 10 permit 10.1.1.1 0.0.0.255

access-list 10 deny ip any (implicit)

Time for Wildcard Masks!

Access-list 1 permit 172.16.0.0 0.0.255.255

A **wildcard mask** address:

- Tells how much of the packet's source IP address (or destination IP address) needs to match for this condition to be true.

Time for Wildcard Masks!

Access-list 1 permit 172.16.0.0 0.0.255.255

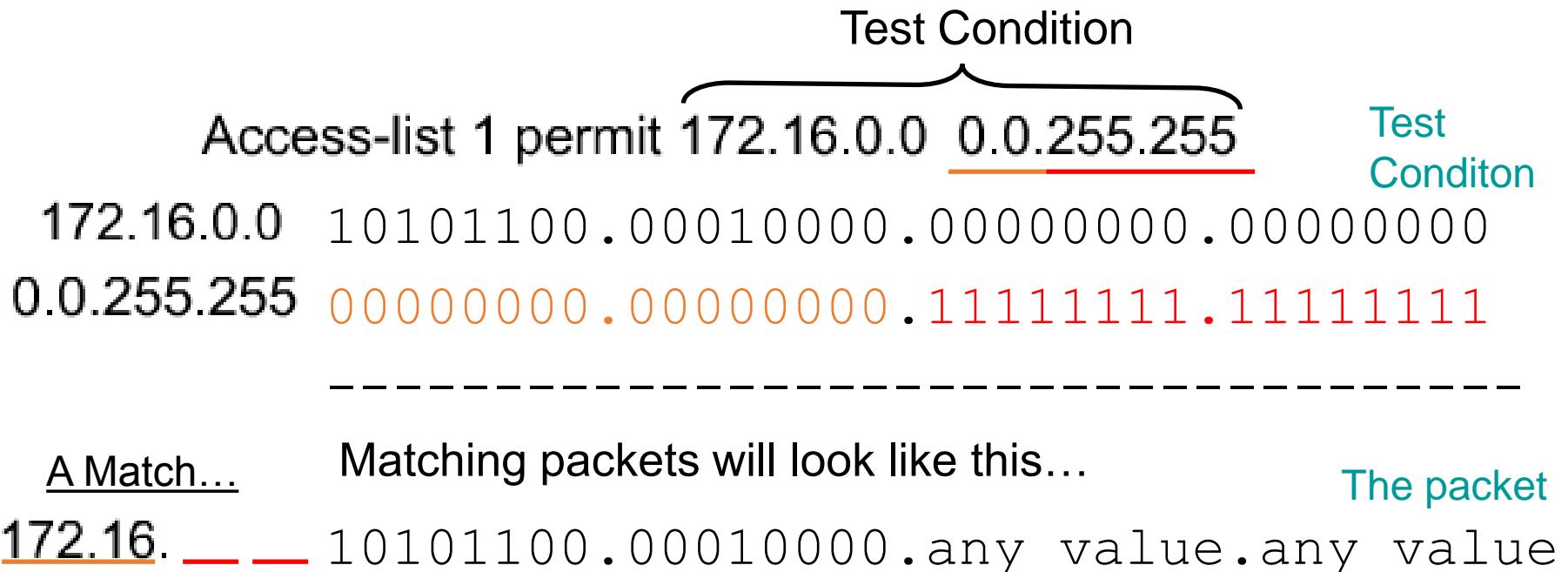
- A **wildcard mask** is a 32-bit quantity that is divided into four octets.
- A wildcard mask is paired with an IP address.
- The numbers one and zero in the mask are used to identify how to treat the corresponding IP address bits.
- The term wildcard masking is a nickname for the ACL mask-bit matching process and comes from of an analogy of a wildcard that matches any other card in the game of poker.
- **Wildcard masks have no functional relationship with subnet masks.**
 - They are used for different purposes and follow different rules.
- Subnet masks start from the left side of an IP address and work towards the right to extend the network field by borrowing bits from the host field.
- Wildcard masks are designed to filter individual or groups of IP addresses permitting or denying access to resources based on the address.

Wildcard Masks!

Access-list 1 permit 172.16.0.0 0.0.255.255

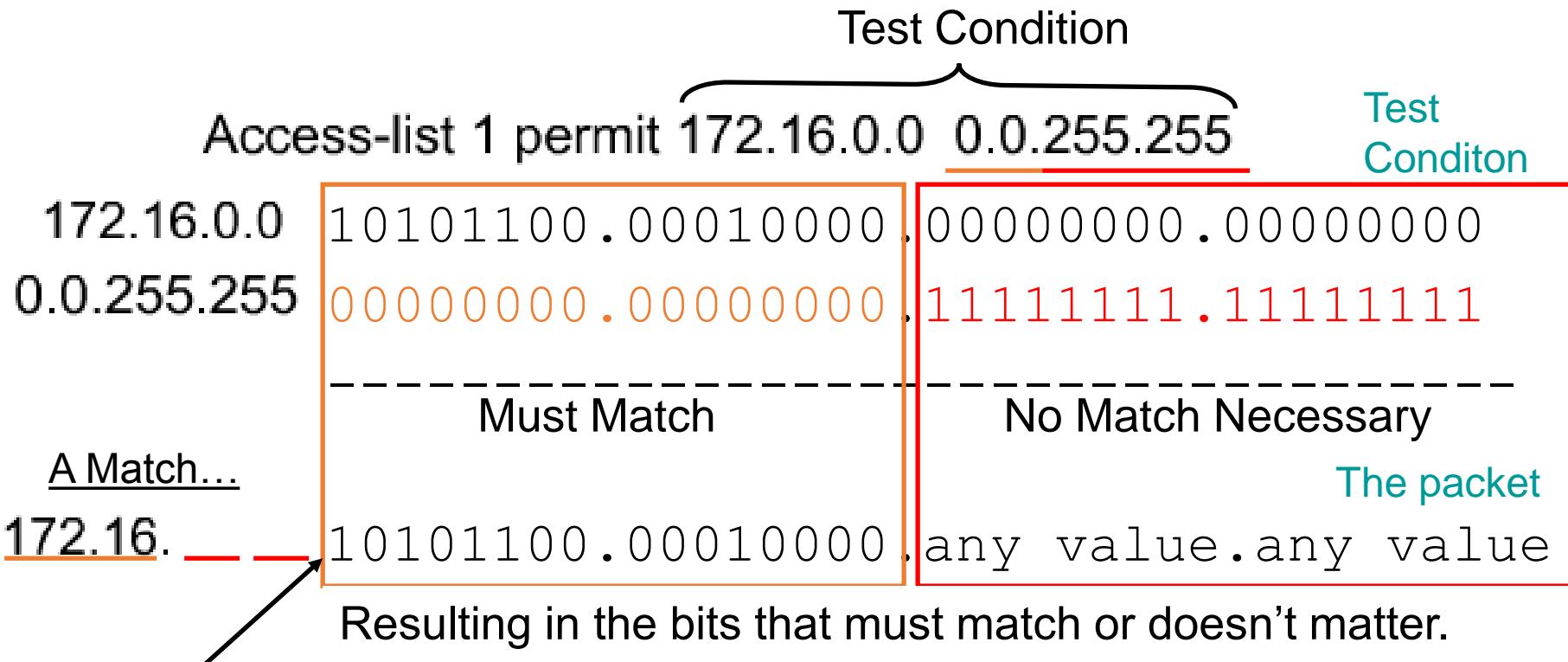
- “Trying to figure out how wildcard masks work by relating them to subnet masking will only confuse the entire matter. The only similarity between a wildcard mask and a subnet mask is that they are both thirty-two bits long and use ones and zeros for the mask.”
- This is not entirely true.
- Although it is very important that you understand how a wildcard mask works, it can also be thought as an **inverse subnet mask**.

Wildcard Masks!



- Wildcard masking used to identify how to treat the corresponding IP address bits.
 - **0** - “**check** the corresponding bit value.”
 - **1** - “**do not check** (ignore) that corresponding bit value.”
- A **zero** in a bit position of the access list mask indicates that the corresponding bit in the address must be checked and must match for condition to be true.
- A **one** in a bit position of the access list mask indicates the corresponding bit in the address is not “interesting”, does not need to match, and can be ignored.

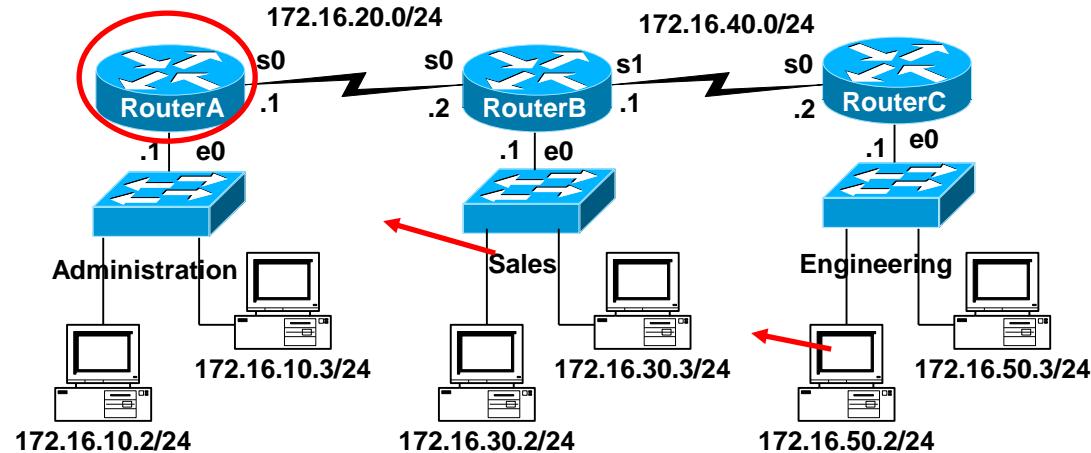
Wildcard Masks!



Matching packets will look like this.

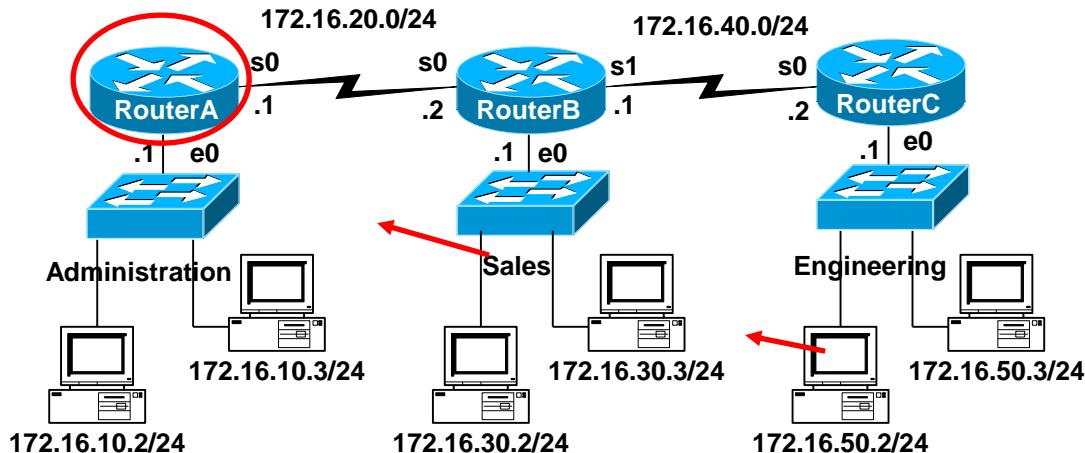
- **0** - “check the corresponding bit value.”
- **1** - “do not check (ignore) that corresponding bit value.”

Example 4 – Using Wildcard Masks



- Task:
 - Want Router A to permit entire sales network and just the 172.16.50.2 station.
 - Deny all other traffic from entering Administrative network.

Example 4 – Using Wildcard Masks



```
RouterA(config)#access-list 11 permit 172.16.30.0 0.0.0.255  
RouterA(config)#access-list 11 permit 172.16.50.2 0.0.0.0
```

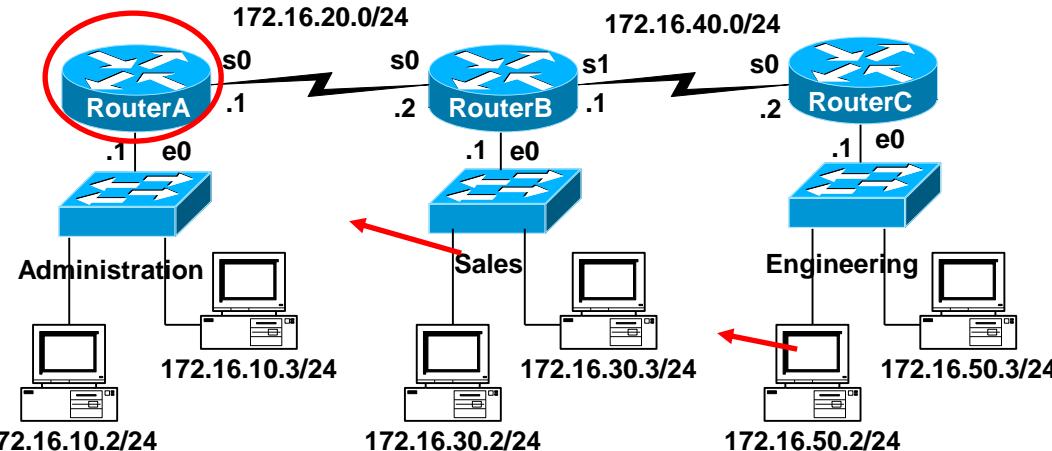
172.16.30.0 **0.0.0.255**

- 0 check - make sure first octet is 172
- 0 check - make sure second octet is 16
- 0 check - make sure third octet is 30
- 255 - don't check (*permit* any fourth octet)

172.16.50.2 **0.0.0.0**

- 0 check - make sure first octet is 172
- 0 check - make sure second octet is 16
- 0 check - make sure third octet is 50
- 0 check - make sure fourth octet is 2

Example 4 – Using Wildcard Masks

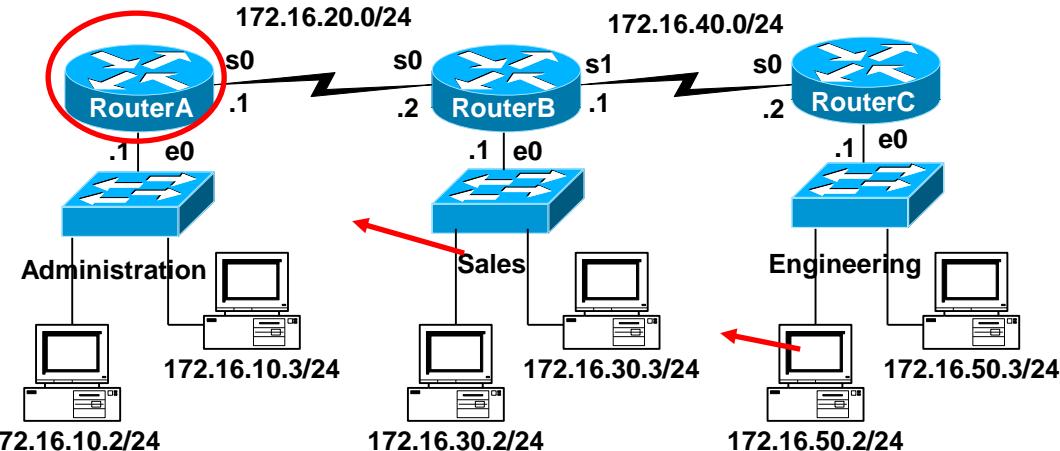


```
RouterA(config) #access-list 11 permit 172.16.30.0 0.0.0.255
```

0 = check, we want this to match, 1 = don't check (don't care)

172.16.30.0	10101100 . 00010000 . 00011110 . 00000000	Test Condition
0.0.0.255	00000000 . 00000000 . 00000000 . 11111111	
-----	-----	-----
172.16.30.0	10101100 . 00010000 . 00011110 . 00000000	The
172.16.30.1	10101100 . 00010000 . 00011110 . 00000001	packet(s)
	... (through)	
172.16.30.255	10101100 . 00010000 . 00011110 . 11111111	

Example 4 – Using Wildcard Masks

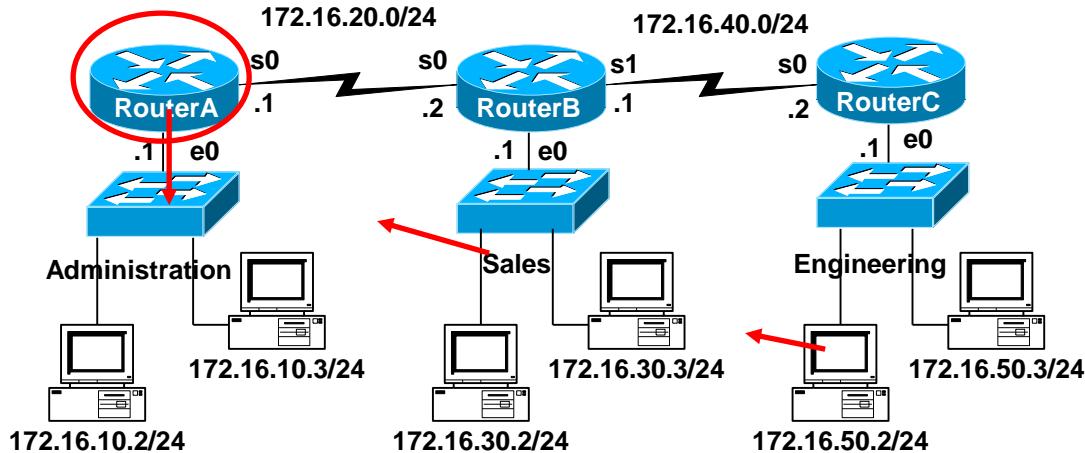


```
RouterA(config)#access-list 11 permit 172.16.50.2 0.0.0.0
```

0 = check, we want this to match, 1 = don't check (don't care)

172.16.50.2	10101100 . 00010000 . 00110010 . 00000010	Test Condition
0.0.0.0	00000000 . 00000000 . 00000000 . 00000000	
<hr/>		
172.16.50.2	10101100 . 00010000 . 00110010 . 00000010	The packet(s)

Example 4 – Using Wildcard Masks

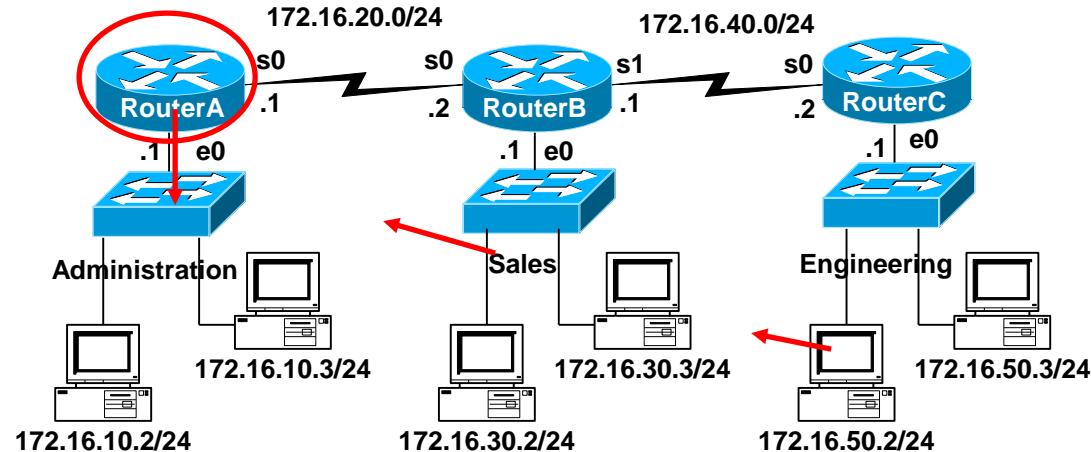


Don't forget to apply the access-list to an interface.

```
RouterA(config)#access-list 11 permit 172.16.30.0 0.0.0.255  
RouterA(config)#access-list 11 permit 172.16.50.2 0.0.0.0
```

```
RouterA(config)# interface e 0  
RouterA(config-if)#ip access-group 11 out
```

Example 4 – Using Wildcard Masks

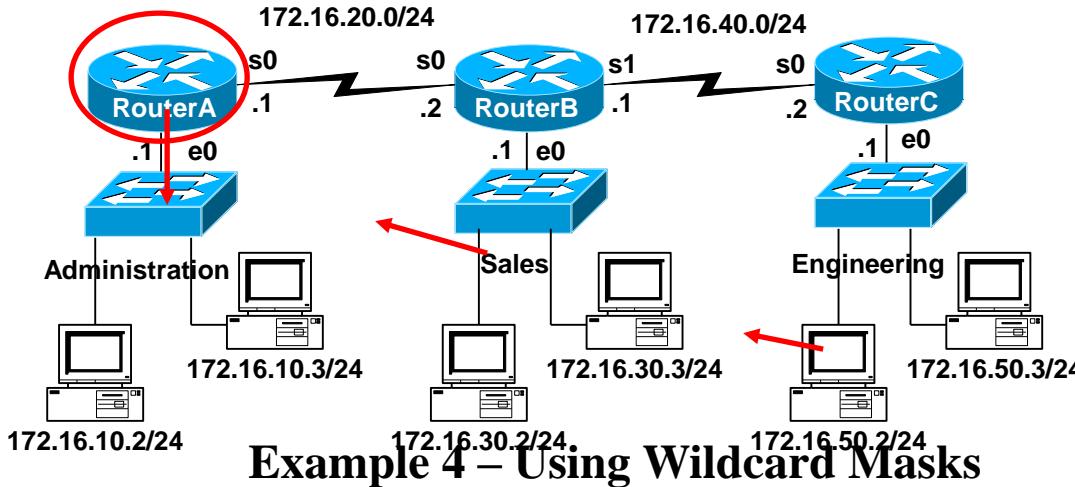


Remember that implicit deny any? It's a good idea for beginners to include the deny any statement just as a reminder.

```
RouterA(config) #access-list 11 permit 172.16.30.0 0.0.0.255
RouterA(config) #access-list 11 permit 172.16.50.2 0.0.0.0
RouterA(config) #access-list 11 deny 0.0.0.0 255.255.255.255
```

```
RouterA(config) # interface e 0
RouterA(config-if) #ip access-group 11 out
```

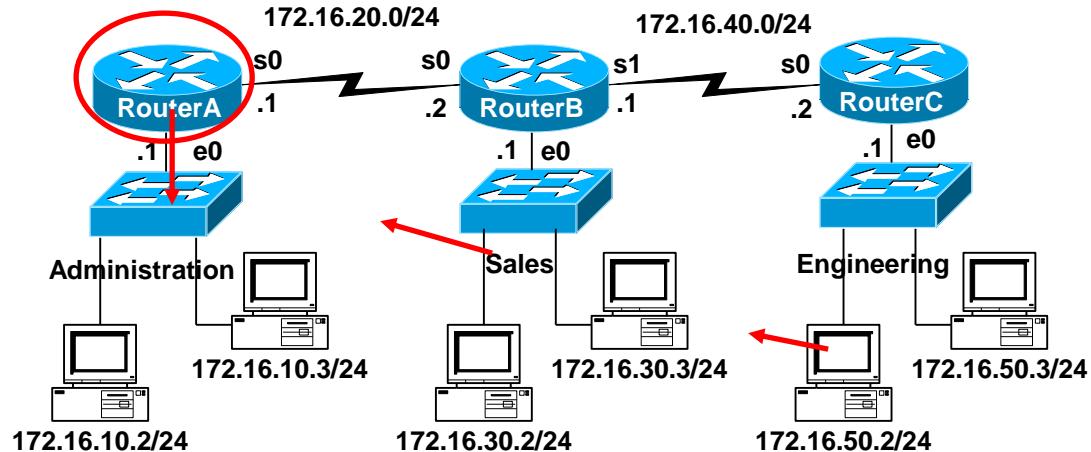
Example 4 – Using Wildcard Masks



```
RouterA(config) #access-list 11 deny 0.0.0.0 255.255.255.255  
0 = check, we want this to match, 1 = don't check (don't care)
```

0.0.0.0	00000000 . 00000000 . 00000000 . 00000000	Test Condition
255.255.255.255	11111111 . 11111111 . 11111111 . 11111111	
0.0.0.0	00000000 . 00000000 . 00000000 . 00000000	The packet(s)
0.0.0.1	00000000 . 00000000 . 00000000 . 00000001	
	... (through)	
255.255.255.255	11111111 . 11111111 . 11111111 . 11111111	

“any” keyword



```
RouterA(config)#access-list 11 deny 0.0.0.0 255.255.255.255
```

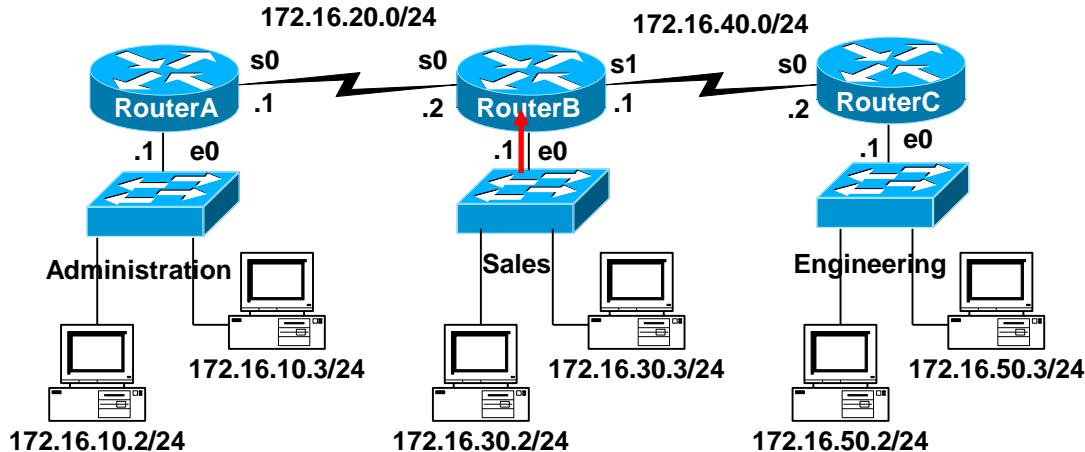
Or

```
RouterA(config)#access-list 11 deny any
```

any = 0.0.0.0 255.255.255.255

- Simply put, the **any** option substitutes 0.0.0.0 for the IP address and 255.255.255.255 for the wildcard mask.
- This option will match any address that it is compared against.

“any” keyword – From Example 3

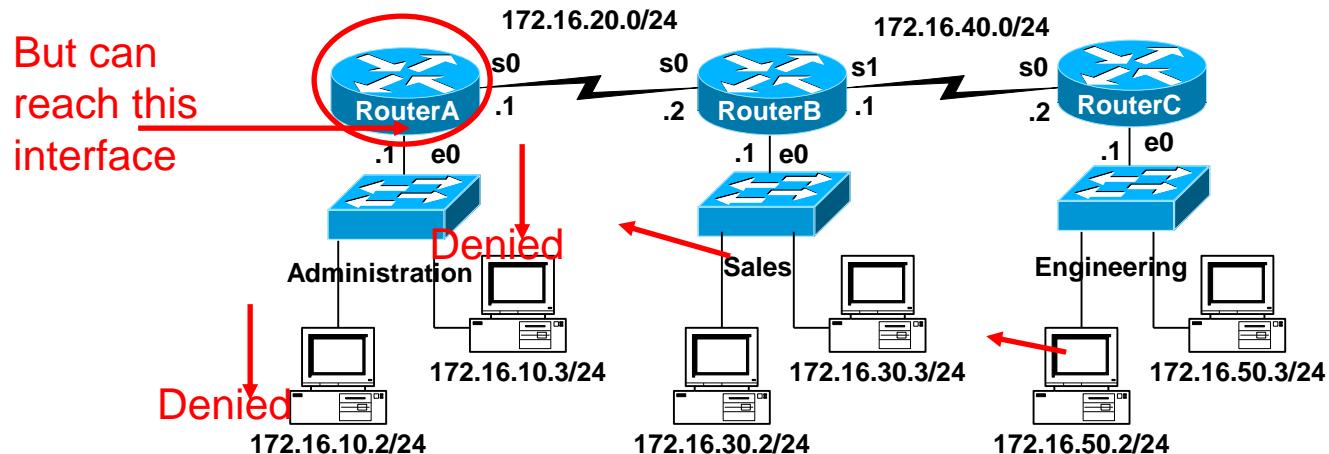


```
RouterB(config)#access-list 10 deny 172.16.30.2
RouterB(config)#access-list 10 permit any
or
RouterB(config)#access-list 10 permit 0.0.0.0 255.255.255.255
```

Previous example:

- Deny only the host 172.16.30.2 from exiting the Sales network.
- Permit all other hosts on the Sales network to leave the 172.16.30.0/24 network.
- Keyword “any” can be used to represent all IP Addresses.

A note about outbound access lists



```
RouterA(config) #access-list 11 permit 172.16.30.0 0.0.0.255
RouterA(config) #access-list 11 permit 172.16.50.2 0.0.0.0
RouterA(config) #access-list 11 deny 0.0.0.0 255.255.255.255
RouterA(config) # interface e 0
RouterA(config-if)#ip access-group 11 out
```

This will deny packets from 172.16.30.0/24 from reaching all devices in the 172.16.10.0/24 Administration LAN, except RouterA's Ethernet 0 interface, of 172.16.10.1. The access list will need to be applied on Router A's Serial 0 interface for it to be denied on RouterA's Ethernet 0 interface. A better solution is to use an Extended Access list. (coming)

Extended Access Lists

```
Router(config)#access-list access-list-number {permit | deny}  
protocol source  
[source-mask destination destination-mask operator operand]  
[established]
```

Parameter	Description
<i>access-list-number</i>	Identifies the list using a number in the range 100 to 199.
permit deny	Indicates whether this entry allows or blocks the specified address.
protocol	The protocol, such as IP, TCP, UDP, ICMP, GRE, or IGRP.
source and destination	Identifies source and destination addresses.
<i>source-mask</i> and <i>destination-mask</i>	Wildcard mask; zeros indicate positions that must match, ones indicate do not care positions.
operator <i>operand</i>	lt, gt, eq, neq (less than, greater than, equal, not equal), and a port number.
established	Allows TCP traffic to pass if the packet uses an established connection (for example, has ACK bits set).

Extended Access Lists

```
Router(config)#access-list access-list-number {permit | deny}  
protocol source  
[source-mask destination destination-mask operator operand]  
[established]
```

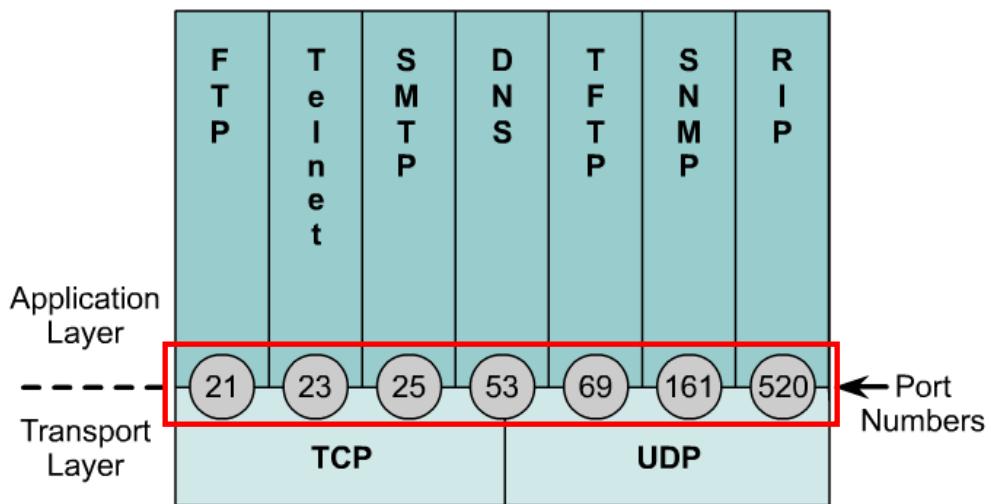
Protocol	Range
IP	1-99
Extended IP	100-199
AppleTalk	600-699
IPX	800-899
Extended IPX	900-999
IPX Service Advertising Protocol	1000-1099

- Extended ACLs are used more often than standard ACLs because they provide a **greater range of control**.
- Extended ACLs **check the source and destination packet addresses** as well as being able to **check for protocols and port numbers**.
- This gives **greater flexibility** to describe what the ACL will check.
- Packets can be permitted or denied access based on where the packet originated and its destination as well as protocol type and port addresses.

Extended Access Lists

```
Router(config)#access-list access-list-number {permit | deny}  
protocol source  
[source-mask destination destination-mask operator operand]  
[established]
```

protocol	The protocol, such as IP, TCP, UDP, ICMP, GRE, or IGRP.
operator operand	lt, gt, eq, neq (less than, greater than, equal, not equal), and a port number.



- **Operator and operand** can also refer to ICMP Types and Codes or whatever the **protocol** is being checked.
- If the **operator and operand** follow the **source address** it refers to the **source port**
- If the **operator and operand** follow the **destination address** it refers to the **destination port**.

Extended Access Lists - Examples

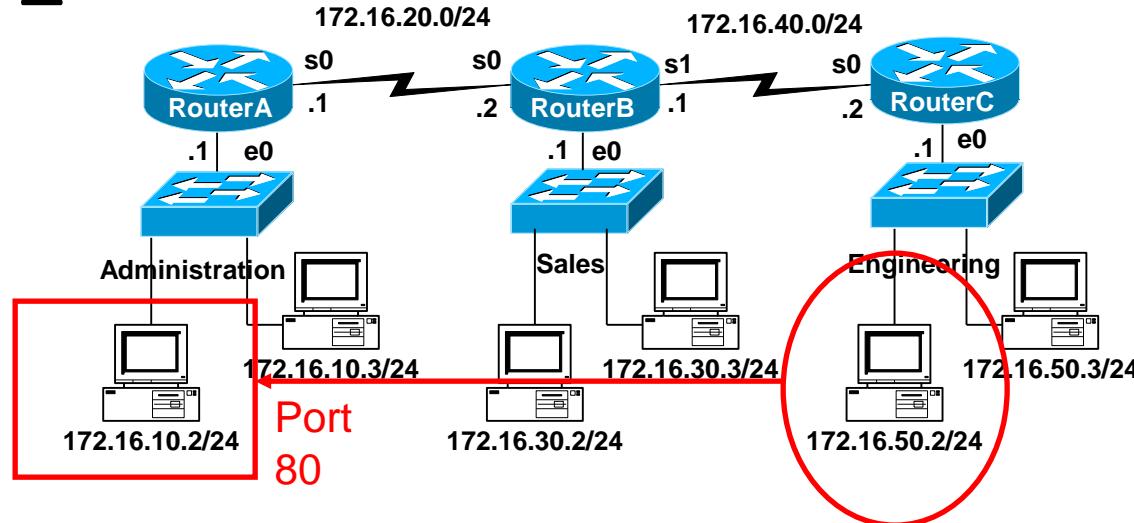
```
access-list 114 permit tcp 172.16.6.0 0.0.0.255 any eq telnet  
access-list 114 permit tcp 172.16.6.0 0.0.0.255 any eq ftp  
access-list 114 permit tcp 172.16.6.0 0.0.0.255 any eq ftp-data
```

port number or protocol name

- Access list number range of 100-199
- Source destination IP address
- Layer 4 protocol number
- Applied to port closest to source host

- The **ip access-group** command links an existing extended ACL to an interface.
- Remember that only one ACL per interface, per direction, per protocol is allowed. The format of the command is:
- Router(config-if)#**ip access-group** access-list-number {in | out}

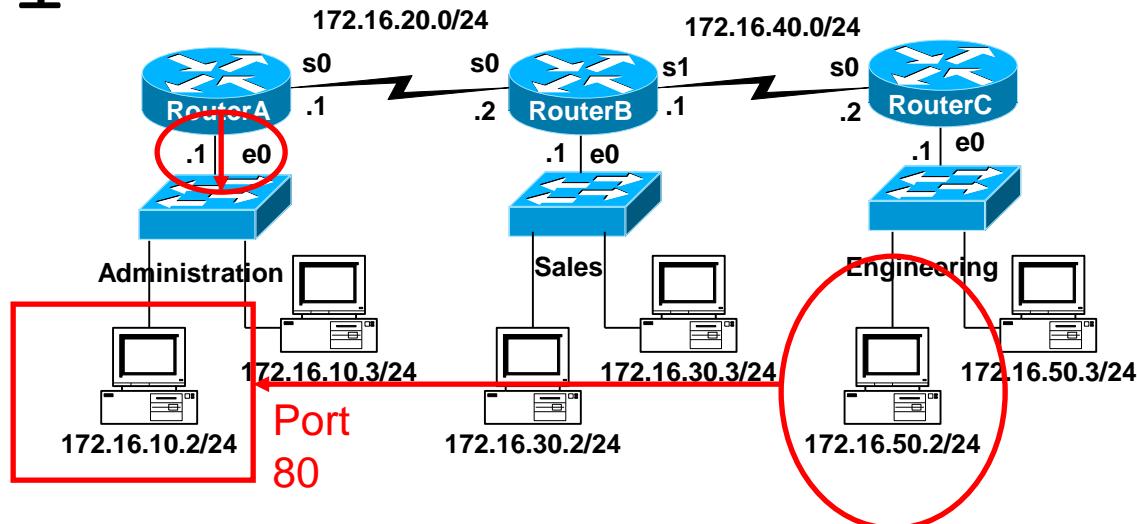
Example 1



Task

- What if we wanted Router A to permit only the Engineering workstation 172.16.50.2 to be able to access the web server in Administrative network with the IP address 172.16.10.2 and port address 80.
- All other traffic is denied.

Example 1



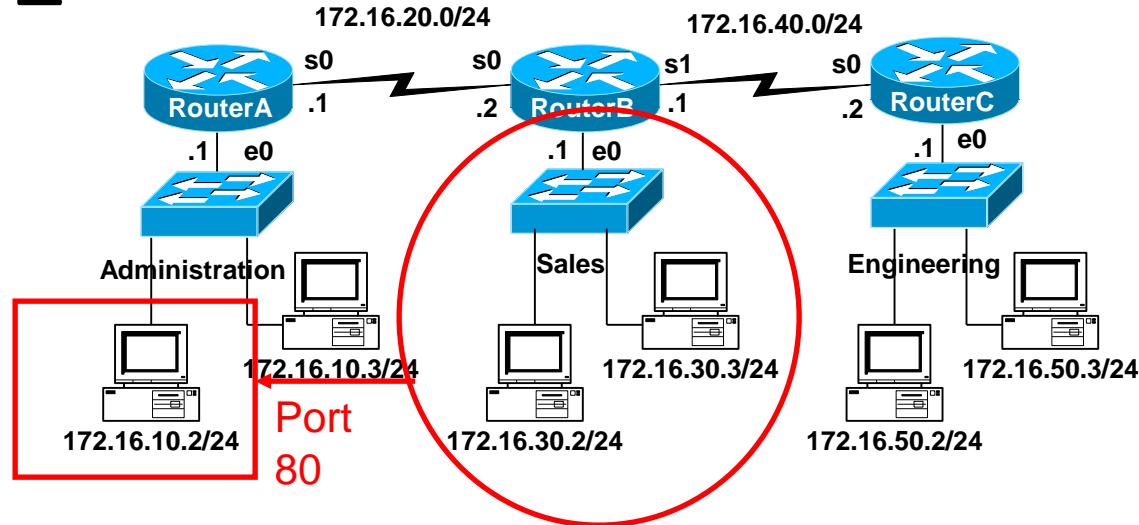
```
RouterA(config)#access-list 110 permit tcp host 172.16.50.2  
host 172.16.10.2 eq 80
```

```
RouterA(config)#inter e 0
```

```
RouterA(config-if)#ip access-group 110 out
```

- Why is better to place the ACL on RouterA instead of RouterC?
- Why is the e0 interface used instead of s0 on RouterA?
- We'll see in a moment!

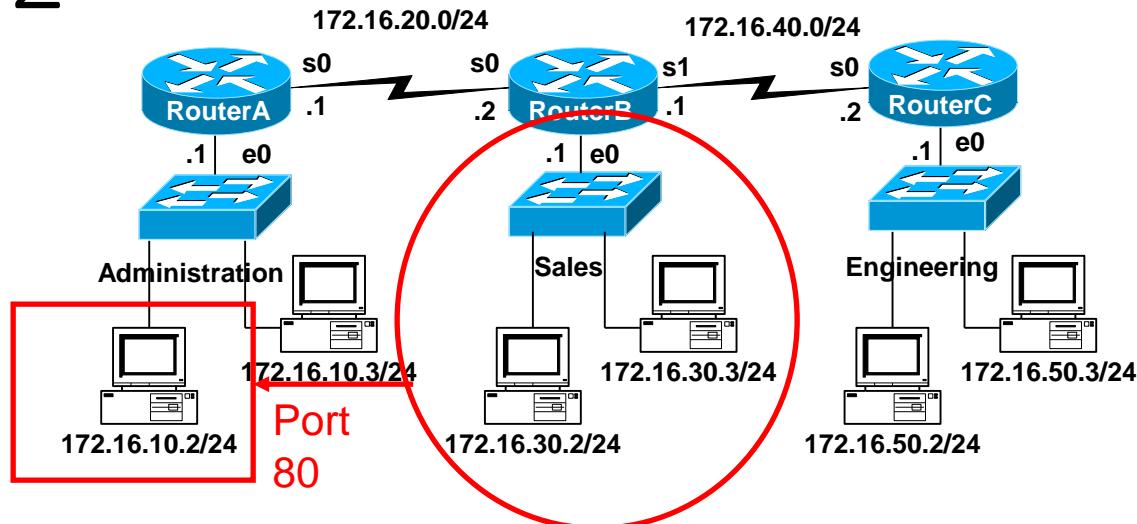
Example 2



Task

- What if we wanted Router A to permit any workstation on the Sales network be able to access the web server in Administrative network with the IP address 172.16.10.2 and port address 80.
- All other traffic is denied.

Example 2



```
RouterA(config)#access-list 110 permit tcp 172.16.30.0  
0.0.0.255 host 172.16.10.2 eq 80
```

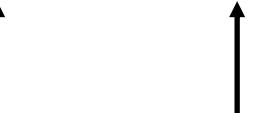
```
RouterA(config)#inter e 0
```

```
RouterA(config-if)#ip access-group 110 out
```

- When configuring access list statements, use the "?" to walk yourself through the command!

Practice

RouterB(config) #access-list 10 permit _____



Permit the following networks:

- | <u>Network/Subnet Mask</u> | <u>Address/Wildcard Mask</u> |
|-------------------------------|------------------------------|
| A. 172.16.0.0 255.255.0.0 | |
| B. 172.16.1.0 255.255.255.0 | |
| C. 192.168.1.0 255.255.255.0 | |
| D. 172.16.16.0 255.255.240.0 | (hmmm . . .?) |
| E. 172.16.128.0 255.255.192.0 | (hmmm . . .?) |

Permit the following hosts:

- | <u>Network/Subnet Mask</u> | <u>Address/Wildcard Mask</u> |
|----------------------------|------------------------------|
| A. 172.16.10.100 | |
| B. 192.168.1.100 | |
| C. All hosts | |

Practice – Do you see a relationship?

RouterB(config)#access-list 10 permit _____
_____ ↑ ↑

Permit the following networks:

<u>Network/Subnet Mask</u>	<u>Address/Wildcard Mask</u>
A. 172.16.0.0 255.255.0.0	172.16.0.0 <u>0.0.255.255</u>
B. 172.16.1.0 255.255.255.0	172.16.1.0 <u>0.0.0.255</u>
C. 192.168.1.0 255.255.255.0	192.168.1.0 <u>0.0.0.255</u>
D. 172.16.32.0 255.255.240.0	172.16.32.0 <u>0.0.15.255</u>
E. 172.16.128.0 255.255.192.0	172.16.128 <u>0.0.63.255</u>

Permit the following hosts:

<u>Network/Subnet Mask</u>	<u>Address/Wildcard Mask</u>
A. 172.16.10.100	172.16.10.100 0.0.0.0
B. 192.168.1.100	192.168.1.100 0.0.0.0
C. All hosts	0.0.0.0 <u>255.255.255.255</u>

Answers Explained

A. 172.16.0.0 0.0.255.255

RouterB(config) #access-list 10 permit 172.16.0.0 0.0.255.255

0 = check, we want this to match

1 = don't check, this can be any value, does not need to match
Test Condition

172.16.0.0	10101100 . 00010000 . 00000000 . 00000000
0.0.255.255	00000000 . 00000000 . 11111111 . 11111111

172.16.0.0	10101100 . 00010000 . 00000000 . 00000000
------------	---

172.16.0.1	10101100 . 00010000 . 00000000 . 00000001
------------	---

172.16.0.2	10101100 . 00010000 . 00000000 . 00000010
------------	---

... (through)

172.16.255.255	10101100 . 00010000 . 11111111 . 11111111
----------------	---

Matching packets will look like this.

The
packet(s)

Answers Explained

D. 172.16.32.0 255.255.240.0

RouterB(config) #access-list 10 permit 172.16.32.0 0.0.15.255

0 = check, we want this to match

1 = don't check, this can be any value, does not need to match
Test Condition

172.16.16.0	10101100 . 00010000 . 00100000 . 00000000
0.0.15.255	00000000 . 00000000 . 00001111 . 11111111

172.16.16.0	10101100 . 00010000 . 00100000 . 00000000
-------------	---

172.16.16.1	10101100 . 00010000 . 00100000 . 00000001
-------------	---

172.16.16.2	10101100 . 00010000 . 00100000 . 00000010
-------------	---

... (through) The packet(s)

172.16.16.255 10101100 . 00010000 . 00101111 . 11111111

Packets belonging to the 172.16.32.0/20 network will match this condition because they have the same 20 bits in common.

There is a relationship! Bitwise-not on the Subnet Mask

D. **172.16.32.0 255.255.240.0**

RouterB(config)#access-list 10 permit 172.16.32.0 0.0.15.255

Subnet Mask: 255 . 255 . 240 . 0

Wildcard Mask: + 0 . 0 . 15 . 255

255 . 255 . 255 . 255

So, we could calculate the Wildcard Mask by:

255 . 255 . 255 . 255

Subnet Mask: - 255 . 255 . 240 . 0

Wildcard Mask: 0 . 0 . 15 . 255

255.255.255.255 – Subnet = Wildcard

```
RouterB(config)#access-list 10 permit _____ _____
```

Permit the following networks:

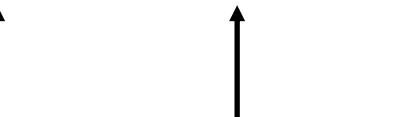
- | 255.255.255.255. - Subnet Mask | = | Wildcard Mask |
|---------------------------------------|----------|----------------------|
| A. 255.255.255.255 - 255.255.0.0 | = | 0.0.255.255 |
| B. 255.255.255.255 - 255.255.255.0 | = | 0.0.0.255 |
| C. 255.255.255.255 - 255.255.255.0 | = | 0.0.0.255 |
| D. 255.255.255.255 - 255.255.240.0 | = | 0.0.15.255 |
| E. 255.255.255.255 - 255.255.192.0 | = | 0.0.63.255 |

Permit the following hosts: (host routes have a /32 mask)

- | 255.255.255.255. - /32 Mask | = | Wildcard Mask |
|--------------------------------------|----------|----------------------|
| A. 255.255.255.255 - 255.255.255.255 | = | 0.0.0.0 |
| B. 255.255.255.255 - 255.255.255.255 | = | 0.0.0.0 |

255.255.255.255 – Subnet = Wildcard

RouterB (config) #access-list 10 permit _____



Permit the following networks:

<u>Network/Subnet Mask</u>	<u>Address/Wildcard Mask</u>
A. 172.16.0.0 255.255.0.0	172.16.0.0 <u>0.0.255.255</u>
B. 172.16.1.0 255.255.255.0	172.16.1.0 <u>0.0.0.255</u>
C. 192.168.1.0 255.255.255.0	192.168.1.0 <u>0.0.0.255</u>
D. 172.16.32.0 255.255.240.0	172.16.32.0 <u>0.0.15.255</u>
E. 172.16.128.0 255.255.192.0	172.16.128 <u>0.0.63.255</u>

Permit the following hosts:

<u>Network/Subnet Mask</u>	<u>Address/Wildcard Mask</u>
A. 172.16.10.100	172.16.10.100 0.0.0.0
B. 192.168.1.100	192.168.1.100 0.0.0.0
C. All hosts or “any”	0.0.0.0 <u>255.255.255.255</u>

“host” option

```
RouterB(config)#access-list 10 permit 192.168.1.100 0.0.0.0
```

```
RouterB(config)#access-list 10 permit host 192.168.1.100
```

Permit the following hosts:

<u>Network/Subnet Mask</u>	<u>Address/Wildcard Mask</u>
A. 172.16.10.100	172.16.10.100 0.0.0.0
B. 192.168.1.100	192.168.1.100 0.0.0.0

- The **host** option substitutes for the 0.0.0.0 mask.
- This mask requires that all bits of the ACL address and the packet address match.
- The host keyword precedes the IP address.
- This option will match just one address.

172.16.10.100 0.0.0.0 replaced by host 172.16.10.100

192.168.1.100 0.0.0.0 replaced by host 192.168.1.100

Ranges with Wildcard Masks - Extra

- Wildcard masks can be used to define “some” ranges of IP address.
- Note:
 - It is possible to get overly complicate your access lists when trying to do a range.
 - Many times using multiple access lists are easier to configure, easier to understand, and you are less likely to make a mistake.
 - We will do our best to understand this, but it is not imperative that you do.
 - If you are with me so far, but I lose you here, don’t worry about it.
- For example:
 - The administrator wants to use IP wildcard masking bits to permit, match subnets **172.30.16.0 to 172.30.31.0**.
 - access-list 20 permit 172.30.16.0 0.0.15.255

Ranges with Wildcard Masks

```
Match subnets 172.30.16.0 to 172.30.31.0  
access-list 20 permit 172.30.16.0 0.0.15.255
```

What's happening (*we'll see its easier than this*):

- The easiest way to see how we did this is to show it in binary...

Ranges with Wildcard Masks

Match subnets 172.30.16.0 to 172.30.31.0

```
access-list 20 permit 172.30.16.0 0.0.15.255
```

172.30.16.0 10101100 . 00011110 . 00010000 . 00000000

0.0.15.255 00000000 . 00000000 . 00001111 . 11111111

172.30.16.0 10101100 . 00011110 . 00010000 . 00000000

172.30.16.1 10101100 . 00011110 . 00010000 . 00000001

through . . .

172.30.31.254 10101100 . 00011110 . 00011111 . 11111110

172.30.31.255 10101100 . 00011110 . 00011111 . 111111115

Ranges with Wildcard Masks

Match subnets 172.30.16.0 to 172.30.31.0

```
access-list 20 permit 172.30.16.0 0.0.15.255
```

Must match

172.30.16.0	10101100 . 00011110 . 00010000 . 00000000
0.0.15.255	00000000 . 00000000 . 00001111 . 11111111

172.30.16.0 10101100 . 00011110 . 00010000 . 00000000

172.30.16.1 10101100 . 00011110 . 00010000 . 00000001

through . . .

172.30.31.254 10101100 . 00011110 . 00011111 . 11111110

172.30.31.255 10101100 . 00011110 . 00011111 . 111111115

Ranges with Wildcard Masks

Match subnets 172.30.16.0 to 172.30.31.0

```
access-list 20 permit 172.30.16.0 0.0.15.255
```

		Any Value
172.30.16.0	10101100 . 00011110 . 00010000 . 00000000	
0.0.15.255	00000000 . 00000000 . 00001111 . 11111111	

172.30.16.0	10101100 . 00011110 . 00010000 . 00000000	
172.30.16.1	10101100 . 00011110 . 00010000 . 00000001	
through . . .		
172.30.31.254	10101100 . 00011110 . 00011111 . 11111110	
172.30.31.255	10101100 . 00011110 . 00011111 . 11111111	

Ranges with Wildcard Masks

Match subnets 172.30.16.0 to 172.30.31.0

```
access-list 20 permit 172.30.16.0 0.0.15.255
```

	Must match	Any Value
172.30.16.0	10101100 . 00011110 . 00010000 . 00000000	
0.0.15.255	00000000 . 00000000 . 00001111 . 11111111	

172.30.16.0	10101100 . 00011110 . 00010000 . 00000000	
172.30.16.1	10101100 . 00011110 . 00010000 . 00000001	
		through . . .
172.30.31.254	10101100 . 00011110 . 00011111 . 11111110	
172.30.31.255	10101100 . 00011110 . 00011111 . 11111111	

Ranges with Wildcard Masks

Match subnets 172.30.16.0 to 172.30.31.0

```
access-list 20 permit 172.30.16.0 0.0.15.255
```

	Must match	Any Value
172.30.16.0	10101100 . 00011110 . 00010000 . 00000000	
0.0.15.255	00000000 . 00000000 . 00001111 . 11111111	

172.30.16.0 10101100 . 00011110 . 00010000 . 00000000

through . . .

172.30.31.255 10101100 . 00011110 . 00011111 . 11111111

- The subnets 172.30.16.0 *through* 172.30.31.0 have the subnet mask 255.255.240.0 in common.
- This gives us the wildcard mask: 0.0.15.255 (255.255.255.255 – 255.255.240.).
- Using the first permitted subnet, 172.30.16.0, gives us the address for our test condition.
- This will not work for all ranges but does in some cases like this one.

Verifying Access Lists

```
Router#show ip interface
```

```
FastEthernet0/0 is up, line protocol is down
    Internet address is 192.168.1.1/24
    Broadcast address is 255.255.255.255
    MTU is 1500 bytes
    Helper address is not set
    Directed broadcast forwarding is disabled
    Outgoing access list is not set
    Inbound access list is 2

Serial0/0 is down, line protocol is down
    Internet address is 200.200.2.1/24
    Broadcast address is 255.255.255.255
    MTU is 1500 bytes
    Helper address is not set
    Directed broadcast forwarding is disabled
    Outgoing access list is not set
    Inbound access list is 101
```

Verifying Access Lists

```
Router#show access-lists
Standard IP access list 2
    deny  172.16.1.1
    permit 172.16.1.0, wildcard bits 0.0.0.255
    deny  172.16.0.0, wildcard bits 0.0.255.255
    permit 172.0.0.0, wildcard bits 0.255.255.255
Extended IP access list 101
    permit tcp 192.168.6.0 0.0.0.255 any eq telnet
    permit tcp 192.168.6.0 0.0.0.255 any eq ftp
    permit tcp 192.168.0.0 0.0.0.255 any eq ftp-data
Router#
```

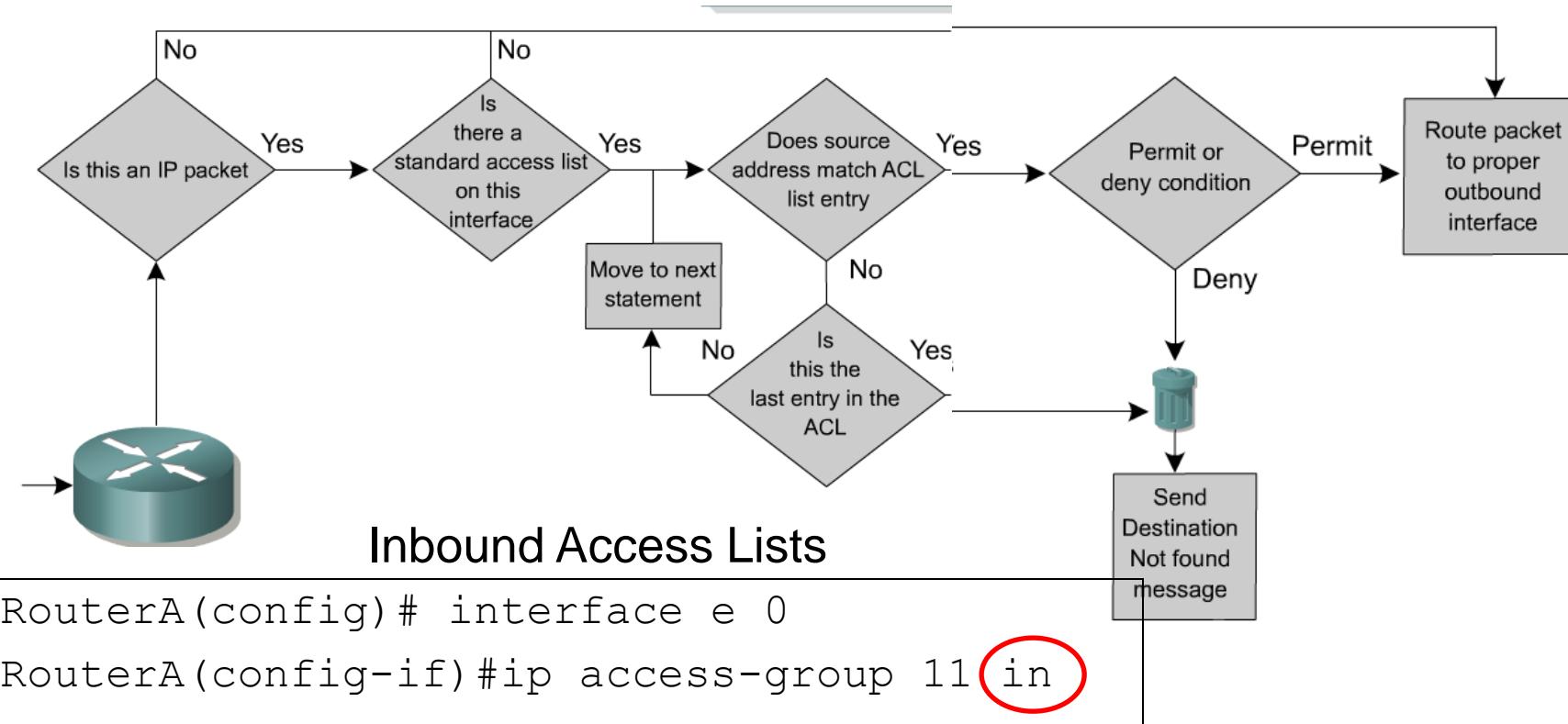
Verifying Access Lists

```
Router#show running-config
Current configuration : 953 bytes
!
interface FastEthernet0/0
    ip address 192.168.1.1 255.255.255.0
→ ip access-group 2 in
!
interface Serial0/0
    ip address 200.200.2.1 255.255.255.0
→ ip access-group 101 in
!
{ access-list 2 deny 172.16.1.1
{ access-list 2 permit 172.16.1.0 0.0.0.255
{ access-list 2 deny 172.16.0.0 0.0.255.255
{ access-list 2 permit 172.0.0.0 0.255.255.255
{ access-list 101 permit tcp 192.168.6.0 0.0.0.255 any
eq telnet
{ access-list 101 permit tcp 192.168.6.0 0.0.0.255 any
eq ftp
!
```

- Note: More than one interface can use the same access-list.

Part 2: ACL Operations

Inbound Standard Access Lists



- With **inbound** Access Lists the IOS checks the packets *before* it is sent to the Routing Table Process.
- With **outbound** Access Lists, the IOS checks the packets *after* it is sent to the Routing Table Process, except destined for the router's own interface.
 - This is because the output interface is not known until the forwarding decision is made.

Standard ACL

```
access-list 2 deny 172.16.1.1
access-list 2 permit 172.16.1.0 0.0.0.255
access-list 2 deny 172.16.0.0 0.0.255.255
access-list 2 permit 172.0.0.0 0.255.255.255
```

- Access list number range of 1-99
- Filter only on source IP address
- Wildcard masks
- Applied to port closest to destination ← We will see why in a moment.

The **full syntax** of the standard ACL command is:

```
Router(config)#access-list access-list-number {deny | permit} source  
[source-wildcard] [log]
```

The **no form** of this command is used to remove a standard ACL. This is the syntax: (Deletes entire ACL!)

```
Router(config)#no access-list access-list-number
```

Extended Access Lists

```
Router(config)#access-list access-list-number {permit | deny}  
protocol source  
[source-mask destination destination-mask operator operand]  
[established]
```

Parameter	Description
<i>access-list-number</i>	Identifies the list using a number in the range 100 to 199.
permit deny	Indicates whether this entry allows or blocks the specified address.
protocol	The protocol, such as IP, TCP, UDP, ICMP, GRE, or IGRP.
source and destination	Identifies source and destination addresses.
<i>source-mask</i> and <i>destination-mask</i>	Wildcard mask; zeros indicate positions that must match, ones indicate do not care positions.
operator <i>operand</i>	lt, gt, eq, neq (less than, greater than, equal, not equal), and a port number.
established	Allows TCP traffic to pass if the packet uses an established connection (for example, has ACK bits set).

Extended Access Lists

```
Router(config)#access-list access-list-number {permit | deny}  
protocol source  
[source-mask destination destination-mask operator operand]  
[established]
```

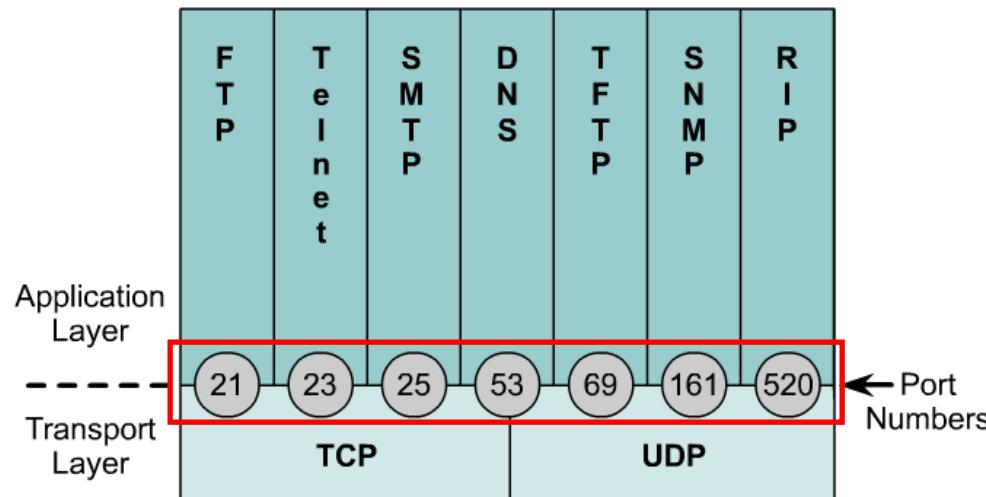
Protocol	Range
IP	1-99
Extended IP	100-199
AppleTalk	600-699
IPX	800-899
Extended IPX	900-999
IPX Service Advertising Protocol	1000-1099

- Extended ACLs are used more often than standard ACLs because they provide a **greater range of control**.
- Extended ACLs **check the source and destination packet addresses** as well as being able to **check for protocols and port numbers**.
- This gives **greater flexibility** to describe what the ACL will check.
- Packets can be permitted or denied access based on where the packet originated and its destination as well as protocol type and port addresses.

Extended Access Lists

```
Router(config)#access-list access-list-number {permit | deny}  
protocol source  
[source-mask destination destination-mask operator operand]  
[established]
```

protocol	The protocol, such as IP, TCP, UDP, ICMP, GRE, or IGRP.
operator operand	lt, gt, eq, neq (less than, greater than, equal, not equal), and a port number.



- **Operator and operand** can also refer to ICMP Types and Codes or whatever the **protocol** is being checked.
- If the **operator and operand** follow the **source address** it refers to the **source port**
- If the **operator and operand** follow the **destination address** it refers to the **destination port**.

Extended Access Lists - Examples

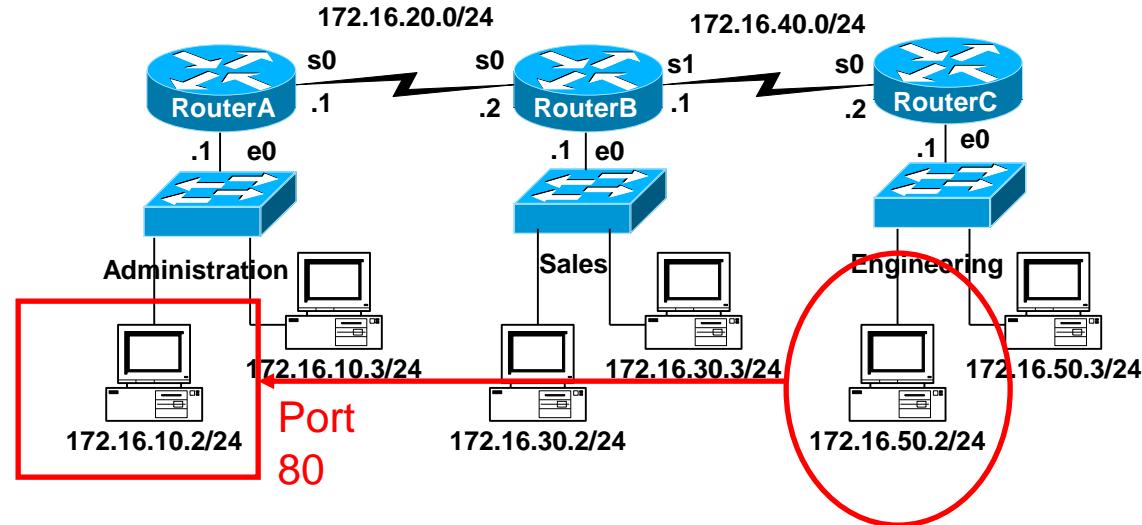
```
access-list 114 permit tcp 172.16.6.0 0.0.0.255 any eq telnet  
access-list 114 permit tcp 172.16.6.0 0.0.0.255 any eq ftp  
access-list 114 permit tcp 172.16.6.0 0.0.0.255 any eq ftp-data
```

port number or protocol name

- Access list number range of 100-199
- Source destination IP address
- Layer 4 protocol number
- Applied to port closest to source host

- The **ip access-group** command links an existing extended ACL to an interface.
- Remember that only one ACL per interface, per direction, per protocol is allowed. The format of the command is:
- Router(config-if)#**ip access-group** *access-list-number* {in | out}

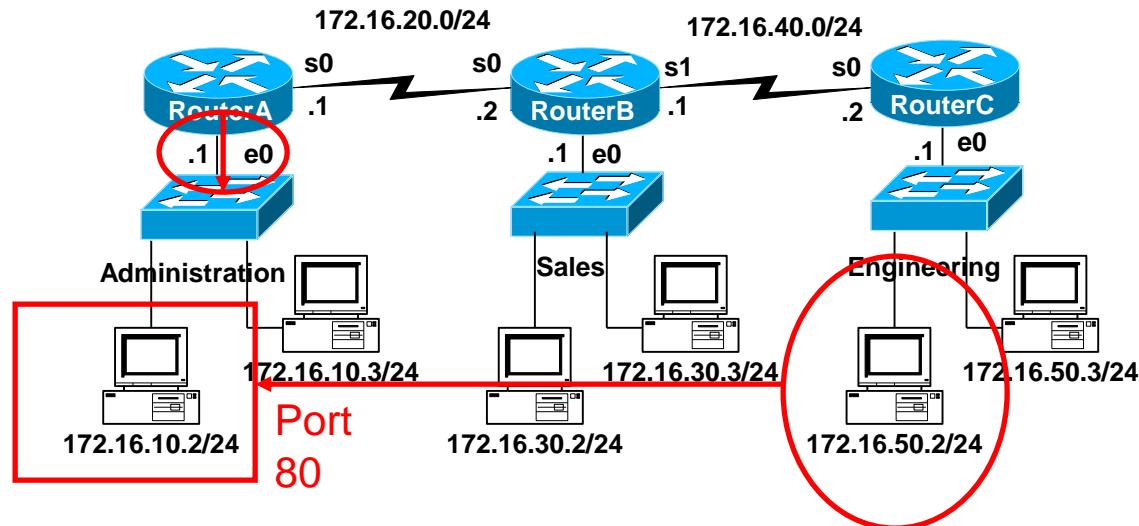
Example 1



Task

- What if we wanted Router A to permit only the Engineering workstation 172.16.50.2 to be able to access the web server in Administrative network with the IP address 172.16.10.2 and port address 80.
- All other traffic is denied.

Example 1



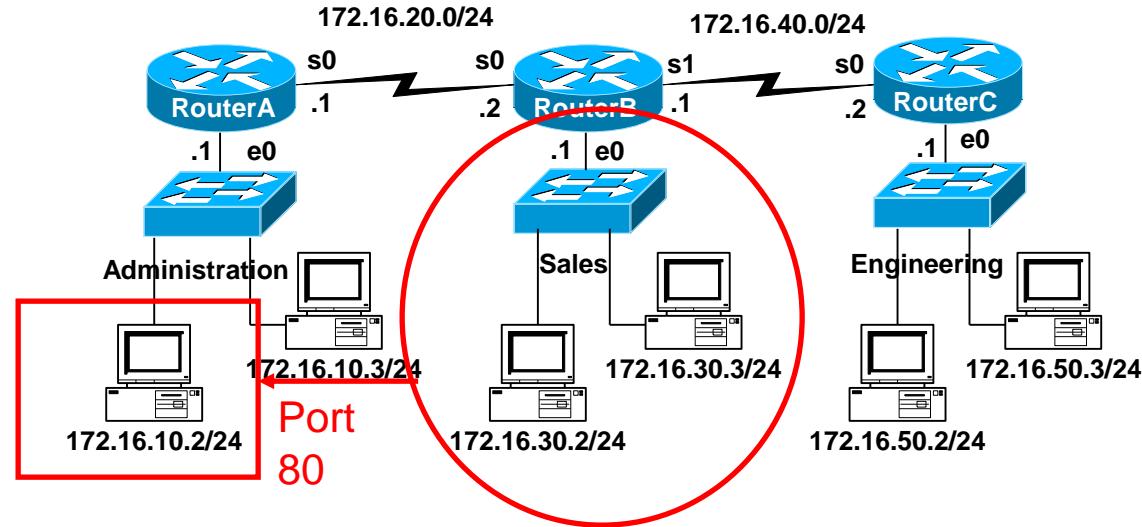
```
RouterA(config)#access-list 110 permit tcp host 172.16.50.2  
host 172.16.10.2 eq 80
```

```
RouterA(config)#inter e 0
```

```
RouterA(config-if)#ip access-group 110 out
```

- Why is better to place the ACL on RouterA instead of RouterC?
- Why is the e0 interface used instead of s0 on RouterA?
- We'll see in a moment!

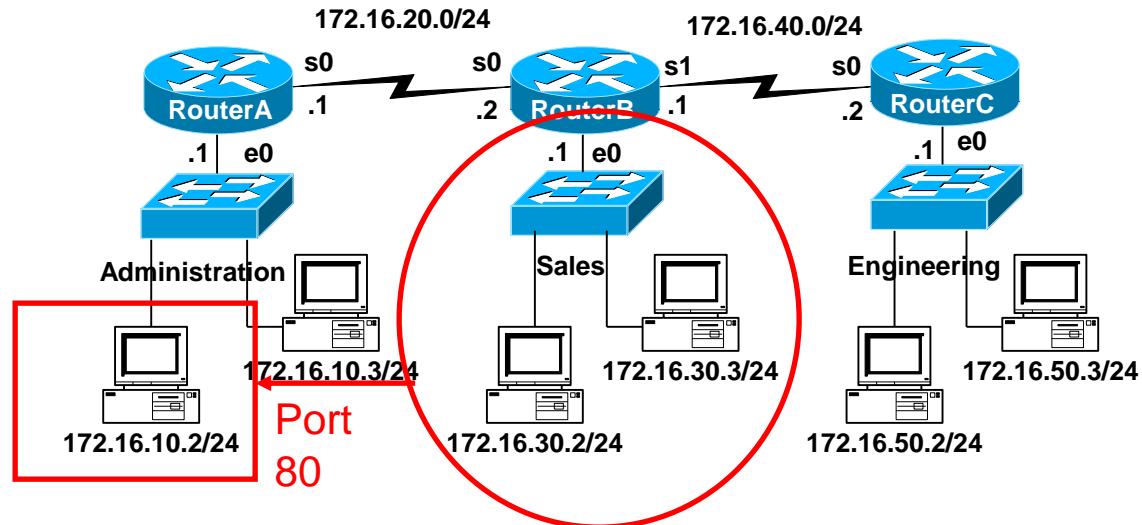
Example 2



Task

- What if we wanted Router A to permit any workstation on the Sales network be able to access the web server in Administrative network with the IP address 172.16.10.2 and port address 80.
- All other traffic is denied.

Example 2



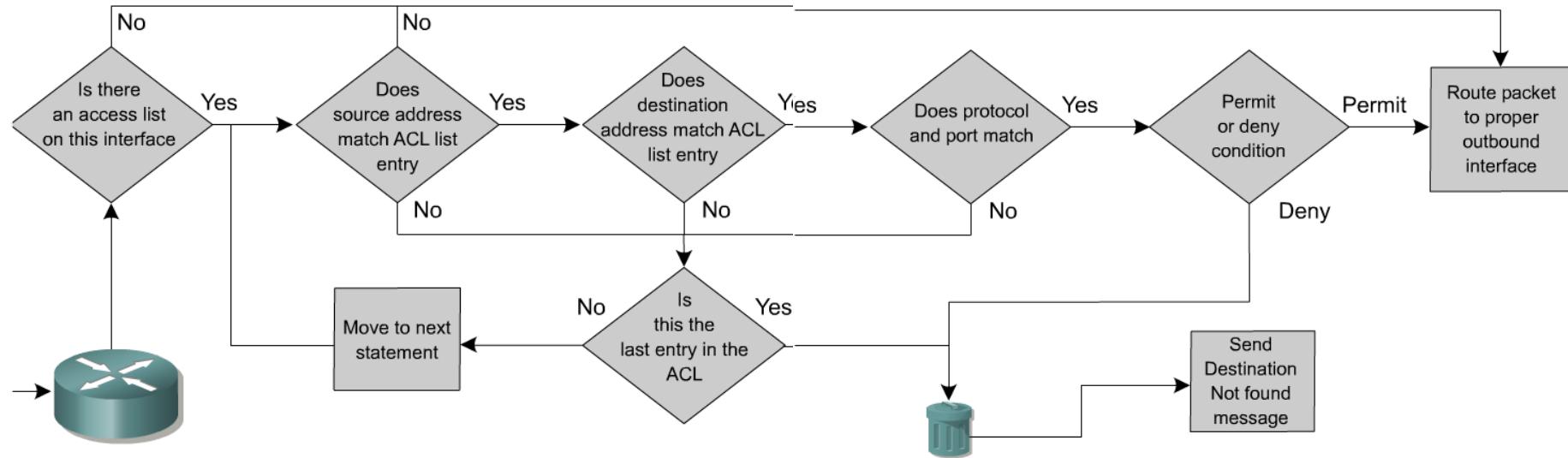
```
RouterA(config)#access-list 110 permit tcp 172.16.30.0  
0.0.0.255 host 172.16.10.2 eq 80
```

```
RouterA(config)#inter e 0
```

```
RouterA(config-if)#ip access-group 110 out
```

- When configuring access list statements, use the “?” to walk yourself through the command!

Inbound Extended Access Lists



Inbound Access Lists

```
RouterA(config)# interface e 0
RouterA(config-if)#ip access-group 11 in
```

- With **inbound** Access Lists the IOS checks the packets *before* it is sent to the Routing Table Process.
- With **outbound** Access Lists, the IOS checks the packets *after* it is sent to the Routing Table Process.
 - This is because the output interface is not known until the forwarding decision is made.

Notes from www.cisco.com

- In the following example, the last entry is sufficient.
- You do not need the first three entries because TCP includes Telnet, and IP includes TCP, User Datagram Protocol (UDP), and Internet Control Message Protocol (ICMP).

```
access-list 101 permit tcp host 10.1.1.2 host 172.16.1.1 eq telnet
access-list 101 permit tcp host 10.1.1.2 host 172.16.1.1
access-list 101 permit udp host 10.1.1.2 host 172.16.1.1
access-list 101 permit ip 10.1.1.0 0.0.0.255 172.16.1.0 0.0.0.255
```

Named ACLs

```
ip access-list {extended|standard} name
```

- IP named ACLs were introduced in Cisco IOS Software Release 11.2.
- Allows standard and extended ACLs to be given names instead of numbers.
- The advantages that a named access list provides are:
 - Intuitively identify an ACL using an alphanumeric name.
 - Eliminate the limit of 798 simple and 799 extended ACLs
 - Named ACLs provide the ability to modify ACLs without deleting and then reconfiguring them.
 - It is important to note that a named access list will allow the deletion of statements but will only allow for statements to be inserted at the end of a list.
 - Even with named ACLs it is a good idea to use a text editor to create them.

Named ACLs

```
Rt1(config)#ip access-list extended server-access
Rt1(config-ext-nacl)#permit TCP any host 131.108.101.99 eq
smtp
Rt1(config-ext-nacl)#permit UDP any host 131.108.101.99 eq
domain
Rt1(config-ext-nacl)#deny ip any any log
Rt1(config-ext-nacl)#^Z
Applying the named list:
Rt1(config)#interface fastethernet 0/0
Rt1(config-if)#ip access-group server-access out
Rt1(config-if)#^Z
```

- A named ACL is created with the **ip access-list** command.
- This places the user in the ACL configuration mode.

Named ACLs

```
router(config-ext-nacl)#permit|deny protocol source  
source-wildcard [operator [port]] destination  
destination-wildcard [operator [port]] [established]  
[precedence precedence] [tos tos] [log] [time-range time-  
range-name]
```

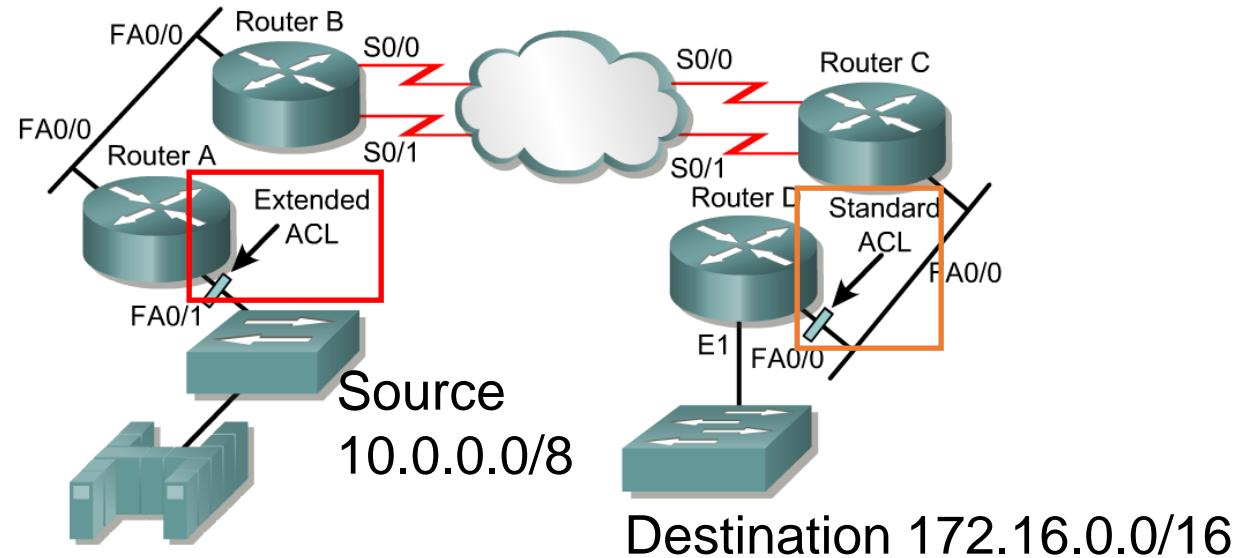
- In ACL configuration mode, specify one or more conditions to be permitted or denied.
- This determines whether the packet is passed or dropped when the ACL statement matches.

Named ACLs

```
ip interface ethernet0/5
ip address 192.168.5.1 255.255.255.0
ip access-group Internetfilter out
ip access-group marketinggroup in
...
{ ip access-list standard Internetfilter
  permit 10.1.1.1
  deny any
}
{ ip access-list extended marketing_group
  permit tcp any 172.30.0.0.0.255.255.255 eq telnet
  deny udp any any
  deny udp any 171.30.0.0.0.255.255.255 1t 1024
  deny ip any log
}
```

The configuration shown creates a standard ACL named `internetfilter` and an extended ACL named `marketing_group`. The lists are applied to the Ethernet Interface 0/5.

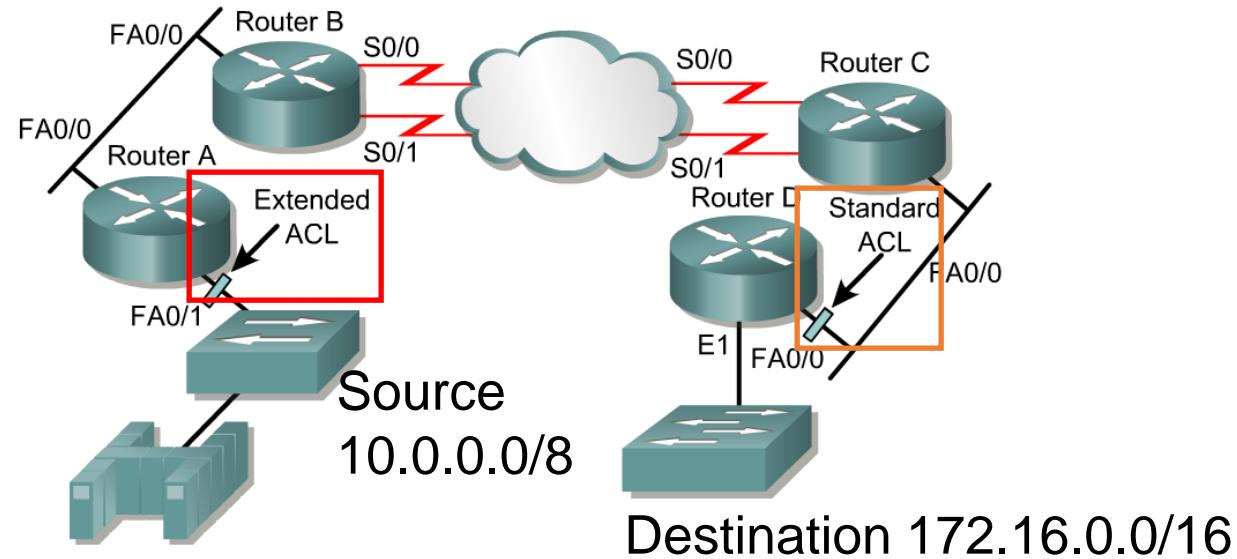
Placing ACLs



The general rule:

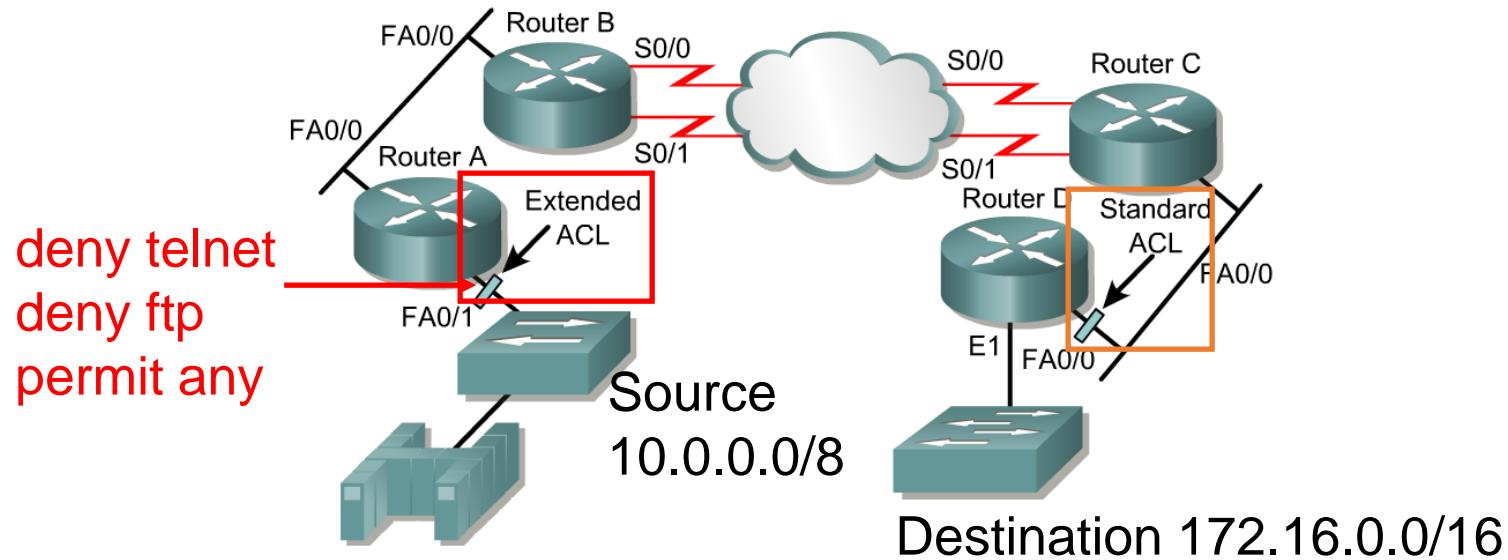
- **Standard ACLs** do not specify destination addresses, so they should be placed as close to the destination as possible.
- Put the **extended ACLs** as close as possible to the source of the traffic denied.

Placing ACLs



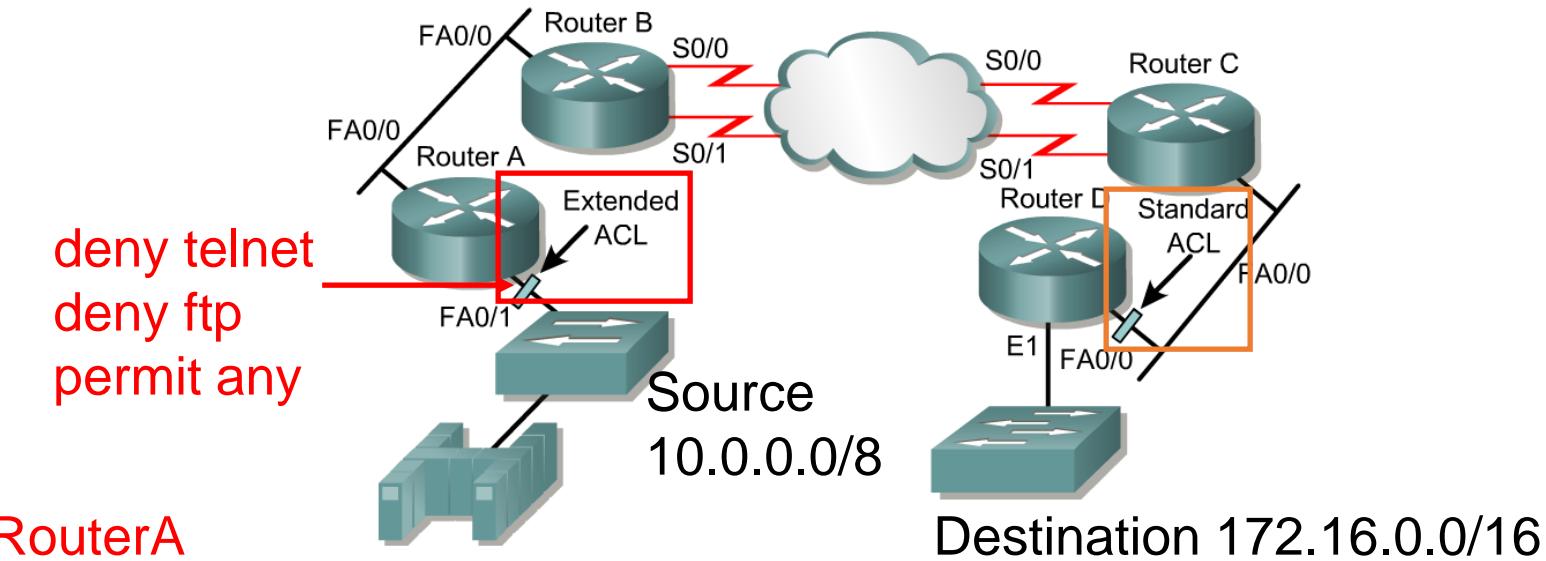
- If the ACLs are placed in the proper location, not only can traffic be filtered, but it can make the whole network more efficient.
- If traffic is going to be filtered, the ACL should be placed where it has the greatest impact on increasing efficiency.

Placing ACLs – Extended Example



- Policy is to deny telnet or FTP Router A LAN to Router D LAN.
- All other traffic must be permitted.
- Several approaches can accomplish this policy.
- The recommended approach uses an extended ACL specifying both source and destination addresses.

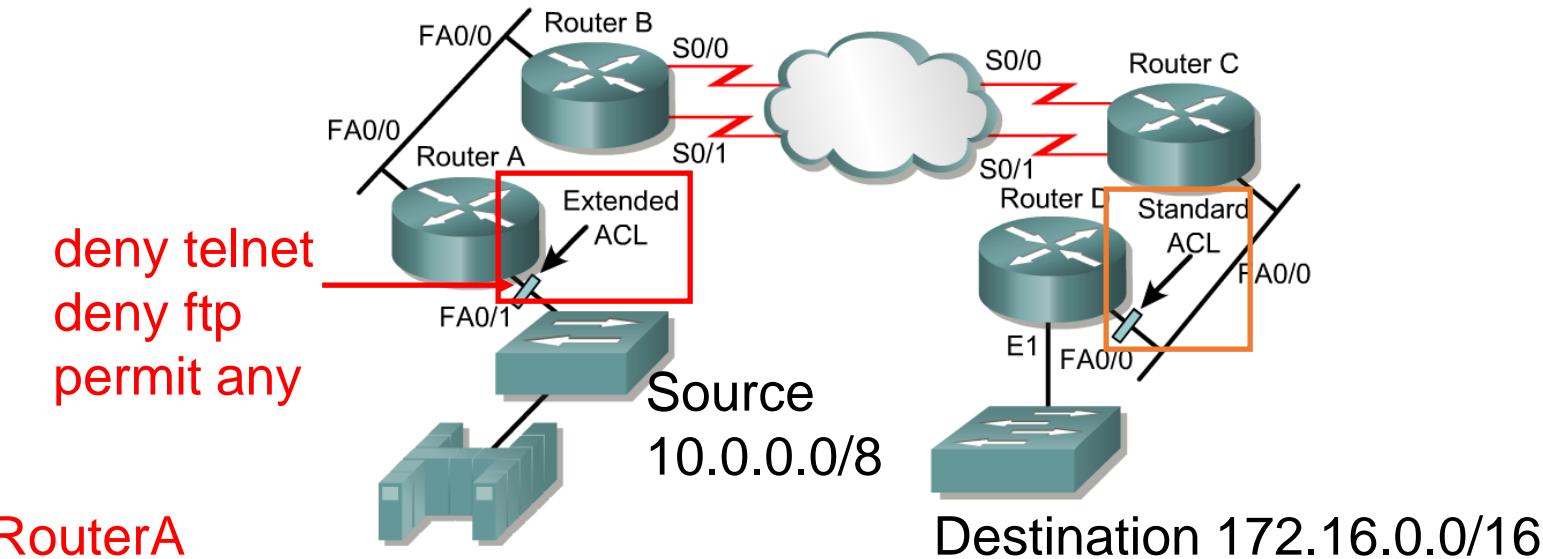
Placing ACLs – Extended Example



```
interface fastethernet 0/1
    access-group 101 in
access-list 101 deny tcp any 172.16.0.0 0.0.255.255 eq telnet
access-list 101 deny tcp any 172.16.0.0 0.0.255.255 eq ftp
access-list 101 permit ip any any
```

- Place this extended ACL in Router A.
- Then, packets do not cross Router A's Ethernet, do not cross the serial interfaces of Routers B and C, and do not enter Router D.
- Traffic with different source and destination addresses will still be permitted.

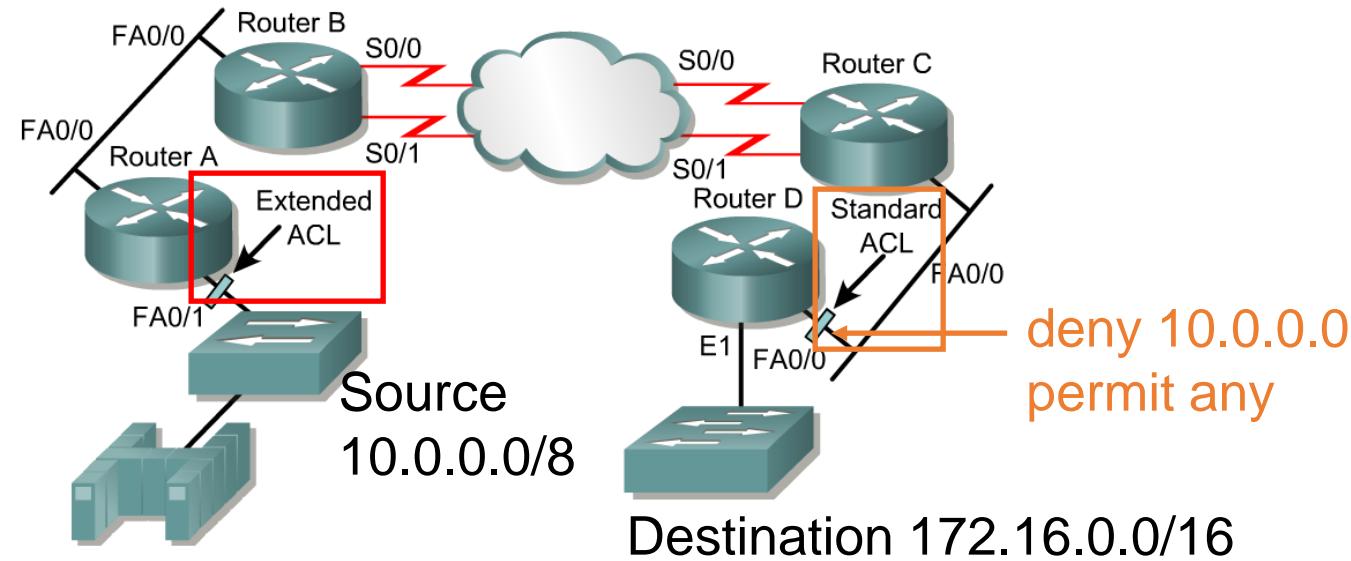
Placing ACLs – Extended Example



```
interface fastethernet 0/1
    access-group 101 in
access-list 101 deny tcp any 172.16.0.0 0.0.255.255 eq telnet
access-list 101 deny tcp any 172.16.0.0 0.0.255.255 eq ftp
access-list 101 permit ip any any
```

- If the **permit ip any any** is not used, then no traffic is permitted.
- Be sure to **permit ip** and not just **tcp** or **all udp** traffic will be denied.

Placing ACLs – Standard Example

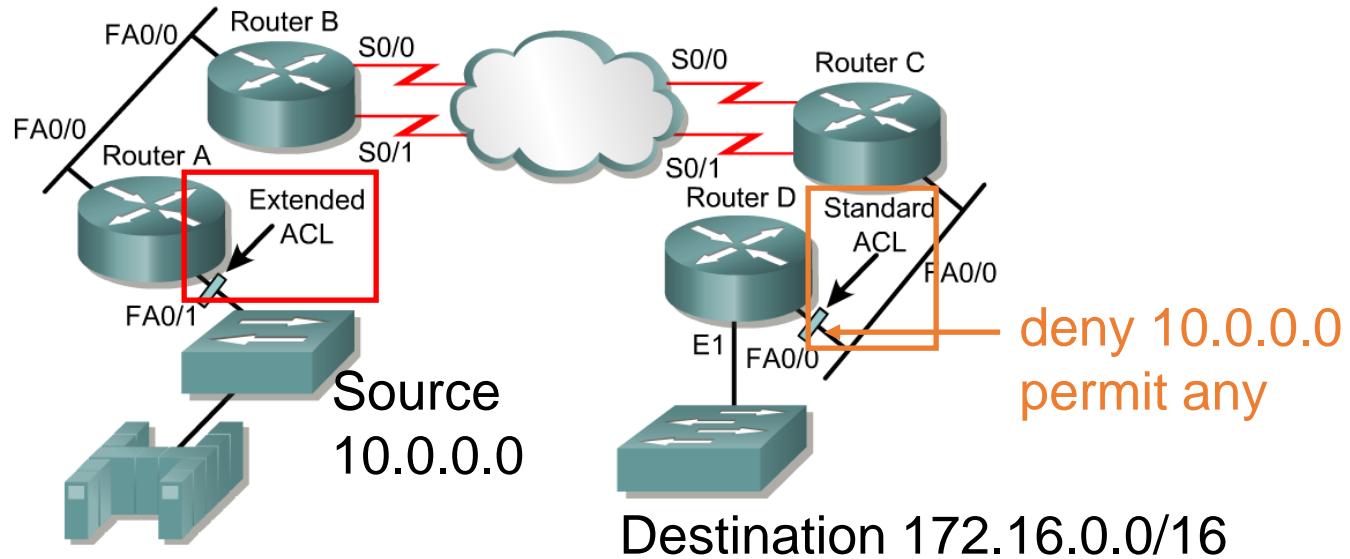


RouterD

```
interface fastethernet 0/0
  access-group 10 in
access-list 10 deny 10.0.0.0 0.255.255.255
access-list 10 permit any
```

- Standard ACLs do not specify destination addresses, so they should be placed as close to the destination as possible.
- If a **standard** ACL is put too close to the source, it will not only deny the intended traffic, but all other traffic to all other networks.⁸⁹

Placing ACLs – Standard Example

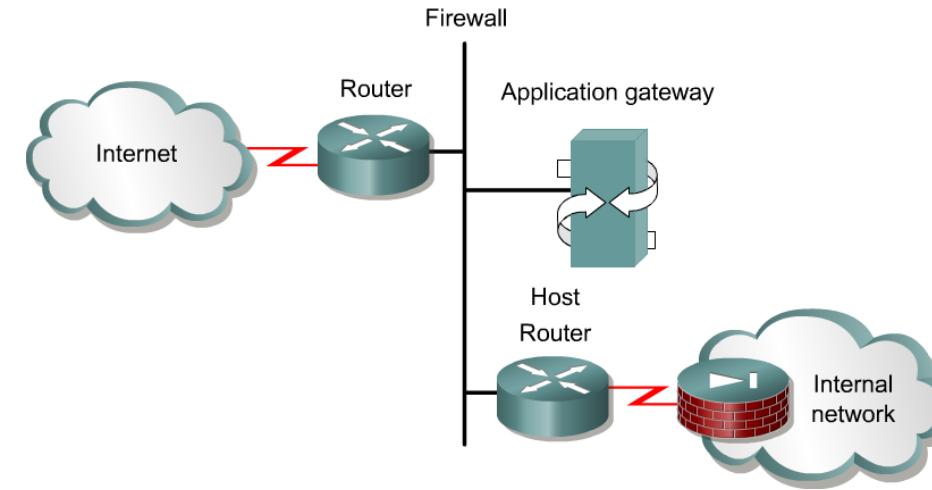


RouterD

```
interface fastethernet 0/0
  access-group 10 in
access-list 10 deny 10.0.0.0 0.255.255.255
access-list 10 permit any
```

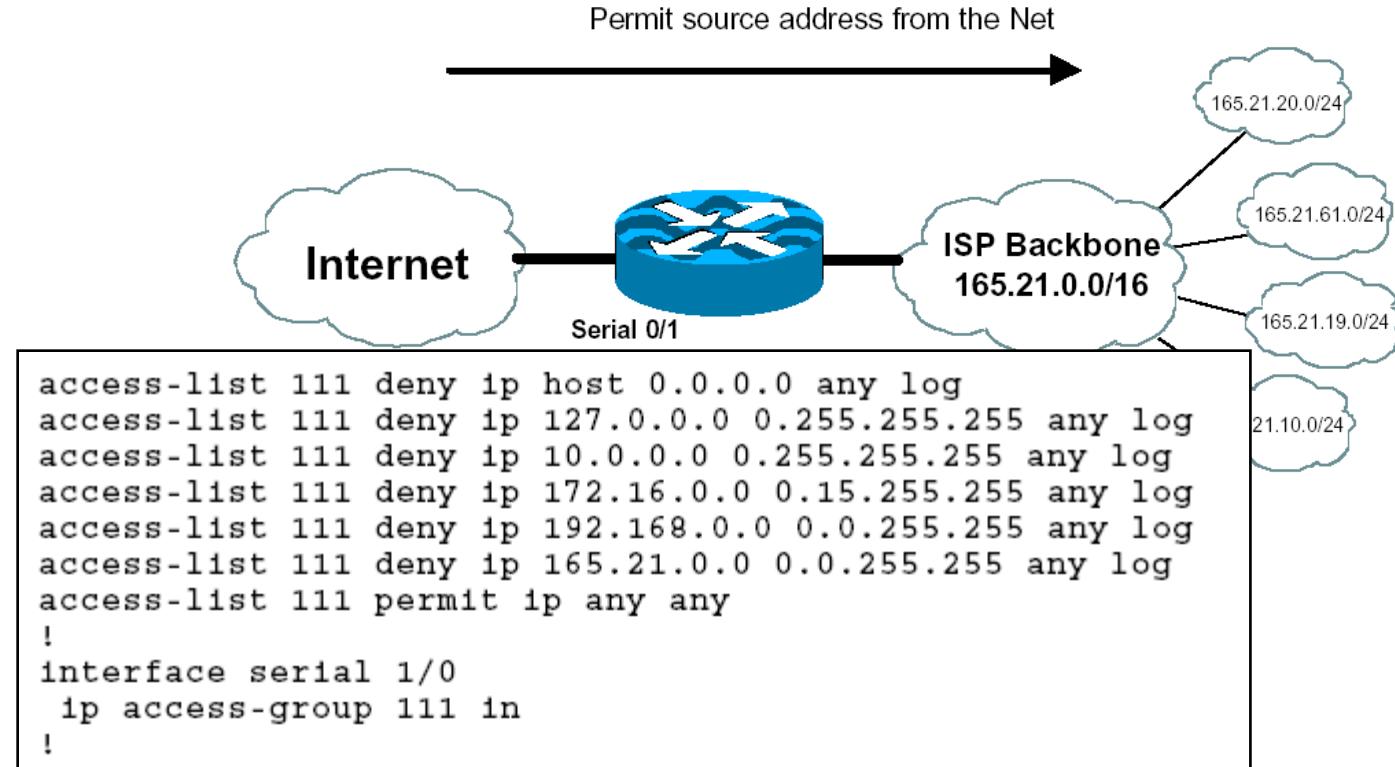
- Better to use extended access lists, and place them close to the source, as this traffic will travel all the way to RouterD before being denied.

Firewalls



- A firewall is an architectural structure that exists between the user and the outside world to protect the internal network from intruders.
- In most circumstances, intruders come from the global Internet and the thousands of remote networks that it interconnects.
- Typically, a network firewall consists of several different machines that work together to prevent unwanted and illegal access.
- ACLs should be used in firewall routers, which are often positioned between the internal network and an external network, such as the Internet.
- The firewall router provides a point of isolation so that the rest of the internal network structure is not affected.
- ACLs can be used on a router positioned⁹¹ between the two parts of the network to

Firewalls



- ISPs use ACLs to deny RFC 1918 addresses into their networks as these are non-routable Internet addresses.
- IP packets coming into your network should never have a source addresses that belong to your network. (This should be applied on all network entrance routers.)
- There are several other simple access lists which should be added to network entrance routers.
- See Cisco IP Essentials White Paper for more information.

Restricting Virtual Terminal Access to a Router

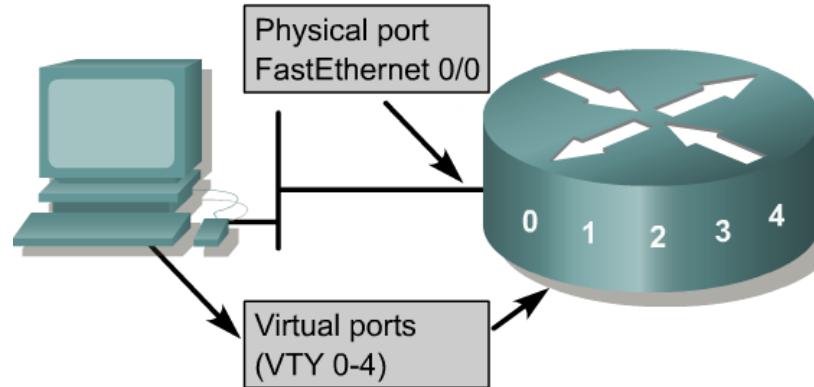
```
Creating the standard list:
```

```
Rt1(config)#access-list 2 permit 172.16.1.0 0.0.0.255  
Rt1(config)#access-list 2 permit 172.16.2.0 0.0.0.255  
Rt1(config)#access-list 2 deny any
```

```
Applying the access list:
```

```
Rt1(config)#line vty 0 4  
Rt1(config)#login  
Rt1(config)#password secret  
Rt1(config)#access-class 2 in  
Rt1(config-line)#

```



- The purpose of restricted vty access is increased network security.
- Access to vty is also accomplished using the Telnet protocol to make a nonphysical connection to the router.
- As a result, there is only one type of vty access list. Identical restrictions should be placed on all vty lines as it is not possible to control which line a user will connect on.

Restricting Virtual Terminal Access to a Router

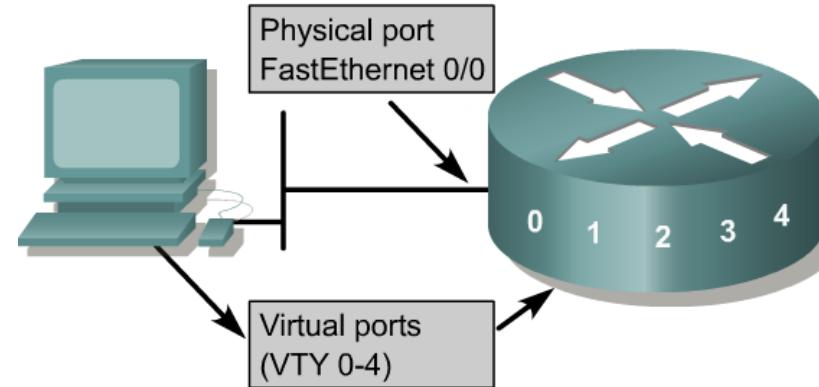
```
Creating the standard list:
```

```
Rt1(config)#access-list 2 permit 172.16.1.0 0.0.0.255  
Rt1(config)#access-list 2 permit 172.16.2.0 0.0.0.255  
Rt1(config)#access-list 2 deny any
```

```
Applying the access list:
```

```
Rt1(config)#line vty 0 4  
Rt1(config)#login  
Rt1(config)#password secret  
Rt1(config)#access-class 2 in  
Rt1(config-line)#

```



- Standard and extended access lists apply to packets traveling through a router.
- **ACLs do not block packets that originate within the router.**
- An outbound Telnet extended access list does not prevent router initiated Telnet sessions, by default.

Standard ACLs

- Use rules based only the packet's **source address**
- 1-99

Extended ACLs

- Provide more precise (finer tuned) packet selection based on:
 - Source and destination addresses
 - Protocols
 - Port numbers
- 100-199

Steps to Configure ACLs

1) Create ACL (global config mode)

- The list may contain many rules, each on one line.
- The list is identified by a number or name.

2) Apply to an interface (interface config mode)

How do ACLs work?

- Processing occurs line by line from top to bottom of the list.
- Each line tests a packet for a “match”.
- If there is a match, a “permit” or “deny” rule is applied.
- When a “match” occurs, no further rules are checked.
- Invisible last line of an ACL is an implicit “deny any.”

How do ACLs work?

- ACL example:

```
oak#sh ru
oak#...
oak#access-list 10 deny 192.168.1.0 0.0.0.255
oak#access-list 10 permit any
oak#access-list 10 deny any (implicit)
oak#...
```

How does a Standard ACL work?

- Permits or **denies** if source IP address is matched:
 - Permit – packet is allowed
 - Deny – packet is dropped
 - Implicit Deny – If a packet's address does not match an earlier statement, an implicit **deny any** occurs at the end of every ACL and the packet is dropped.

Wildcard Masks

- Are used to specify (by bits) the part of the ip address to be matched.
- Looks like a subnet mask but it its not!
- Example:

172.16.0.0



The network address to be matched

0.0.255.255



The wildcard bitmask

Wildcard Masks

- Specify the part of the ip address to be matched.
- Use 0s to match, 1s to ignore. (Reverse of subnet masks!)
- In the example below, only the 1st
2 octets will be examined for a match:

172.16.0.0 0.0.255.255

172.16.0.0 0.0.255.255



Match this part of the address This is the wildcard bitmask

Wildcard Masks

172.16.0.0 0.0.255.255

Address to Match

Wildcard Bitmask

	172	16	0	0
Address	10101100	00010000	00000000	00000000
Wildcard Mask	00000000	00000000	11111111	11111111
	0	0	255	255

Check for a Match

Ignore

Wildcard Masks

Example

- which octets 172.16.5.0 0.0.0.255 will be examined for a match?
- The first 3:

172.16.5.0 0.0.0.255
↑↑↑

Match this part of the address

Wildcard Masks

- In this example, which octets 172.16.5.2 0.0.0.0 will be examined for a match?
- All 4 octets:

172.16.5.2 0.0.0.0
↑ ↑ ↑ ↑

Match the entire address
(permit or deny this specific host)

Wildcard Masks

- In Cisco 2, we will work only with wildcard bitmasks that are 0 or 255 for an entire octet.
- In Cisco 3, you'll work with masks where the change from 0 to 1 does not fall on an octet boundary:
 - e.g. 0.0.15.255

Keyword: “any”

- Identical statements
 - access-list 22 permit 0.0.0.0 255.255.255.255
 - access-list 22 permit **any**



Keyword: “host”

- Identical statements
 - Access-list 23 permit 172.16.1.1 0.0.0.0
 - Access-list 23 permit host 172.16.1.1



Standard IP ACL Command

access-list *ACL-number* {permit |deny} *source-ip-address wildcard-mask*

- ACL number: 1-99
- Global Config mode

Standard ACL Example

- To permit all packets from the network number 172.16.0.0

```
access-list 20 permit 172.16.0.0 0.0.255.255
```

- To permit traffic from the host 172.16.1.1 only

```
access-list 20 permit 172.16.1.1 0.0.0.0
```

OR

```
access-list 20 permit host 172.16.1.1
```

Standard ACL Example

- To permit traffic from any source address.

```
access-list 20 permit 0.0.0.0 255.255.255.255
```

OR

```
access-list 20 permit any
```

How does an Extended ACL work?

- Permits or denies if all conditions match:
 - Source Address
 - Destination Address
 - Protocol
 - Port No. or Protocol Options

Extended IP ACL command

access-list *ACL-number* {**permit|deny**} *protocol* *source-ip-address source-wildcard-mask* *destination-ip-address destination-wildcard-mask* **eq** *port-number*

- ACL number: 100-199
- Global Config mode

Extended ACL Example

- To permit traffic from the network 192.168.1.0 to the host 192.168.3.10 only on telnet:

```
access-list 101 permit tcp 192.168.1.0 0.0.0.255 192.168.3.10 0.0.0.0 eq telnet
```

- More about extended ACLs later...

Major differences

- Standard ACL
 - Use only source address
 - Requires fewer CPU cycles.
 - Place as close to destination as possible. (because they can only check source address)
- Extended ACL
 - Uses source, destination, protocol, port
 - Requires more CPU cycles.
 - Place as close to source as possible. (This stops undesired traffic early.)

Command to apply IP ACL

ip access-group *ACL-number* {in |out}

- Interface Config mode
- The group of rules in the list is applied to the interface being configured.
- Use “**in**” and “**out**” as if looking at the interface from inside the router.

Do I place an ACL in?

In

- Coming into the router.
- Requires less CPU processing because every packet bypasses processing before it is routed.
- Filtering decision is made prior to the routing table.

Do I place an ACL out?

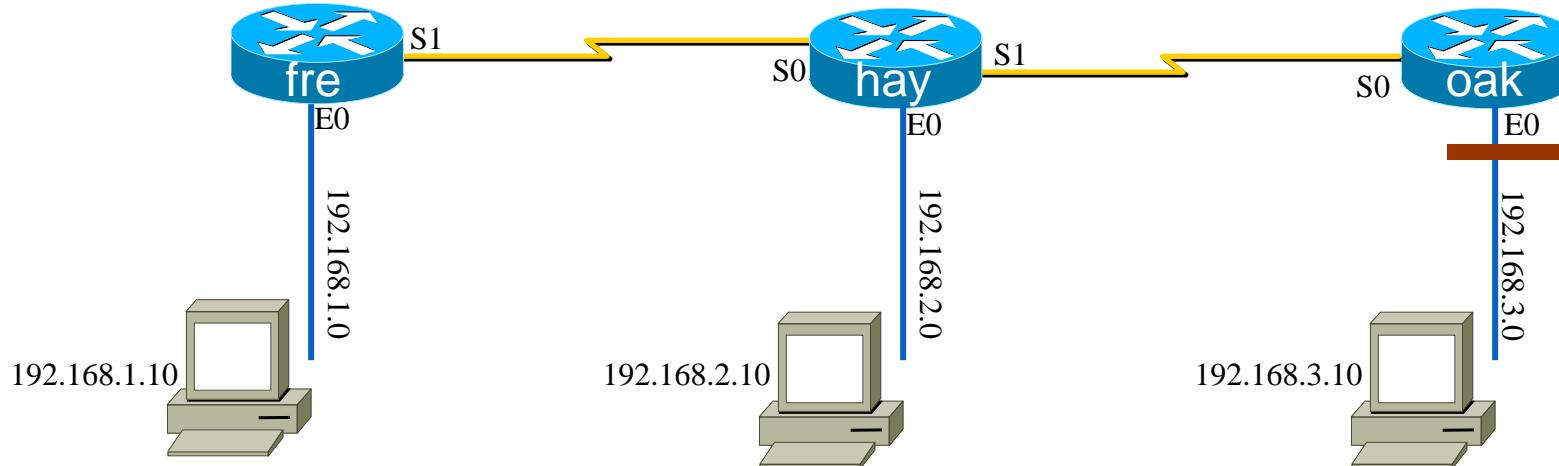
Out

- Going out of the router.
- Routing decision has been made and the packet is switched to the proper outbound interface **before** it is tested against the access list.
- ACLs are outbound unless otherwise specified.

ACL Configuration Example

What will this list do?

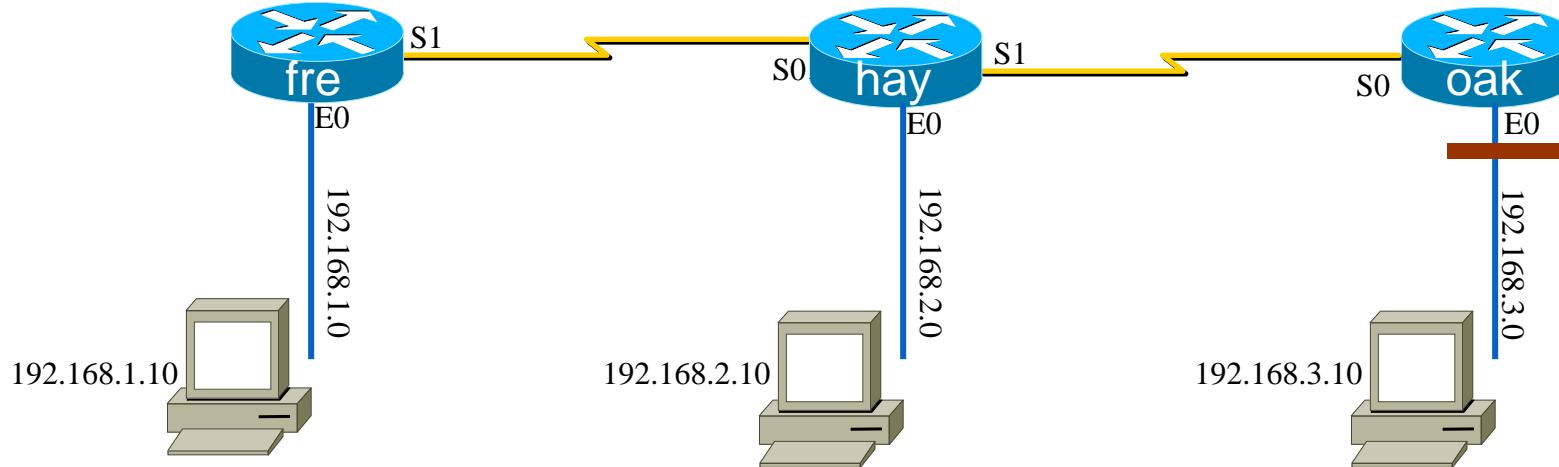
```
oak(config)#access-list 10 permit 192.168.1.0 0.0.0.255
oak(config)#access-list 10 permit 192.168.2.10 0.0.0.0
oak(config)#int e0
oak(config-if)#ip-access group 10 out
oak(config-if)#^z
```



ACL Configuration Example

What's the problem here?

```
oak(config)#access-list 10 permit any
oak(config)#access-list 10 deny 192.168.2.10 0.0.0.0
oak(config)#int e0
oak(config-if)#ip-access group 10 out
oak(config-if)#^z
```



Commands to show ACLs

show access-lists

- Privileged exec mode
- Displays the ACLs on the router.

show ip interface

- Privileged exec mode
- Shows which ACLs are set on that interface.

Extended ACLs

- Provide more precise (finer tuned) packet selection based on:
 - Source and destination addresses
 - Protocols
 - Port numbers
- 100-199

Steps to Configure ACLs

- 1) Create ACL (global config mode)
- 2) Apply to an interface (interface config mode)

Extended ACL operation

- Permits or denies if all conditions match:
 - Source Address
 - Destination Address
 - Protocol
 - Port No. or Protocol Options

Extended IP ACL command

```
access-list ACL-number {permit|deny} protocol source-ip-address source-wildcard-mask destination-ip-address destination-wildcard-mask eq port-number
```

- ACL number: 100-199
- Global Config mode

Extended ACL Example

- To permit traffic from the network 192.168.1.0 to the host 192.168.3.10 only on telnet:

```
access-list 101 permit tcp 192.168.1.0 0.0.0.255 192.168.3.10 0.0.0.0 eq 23
```

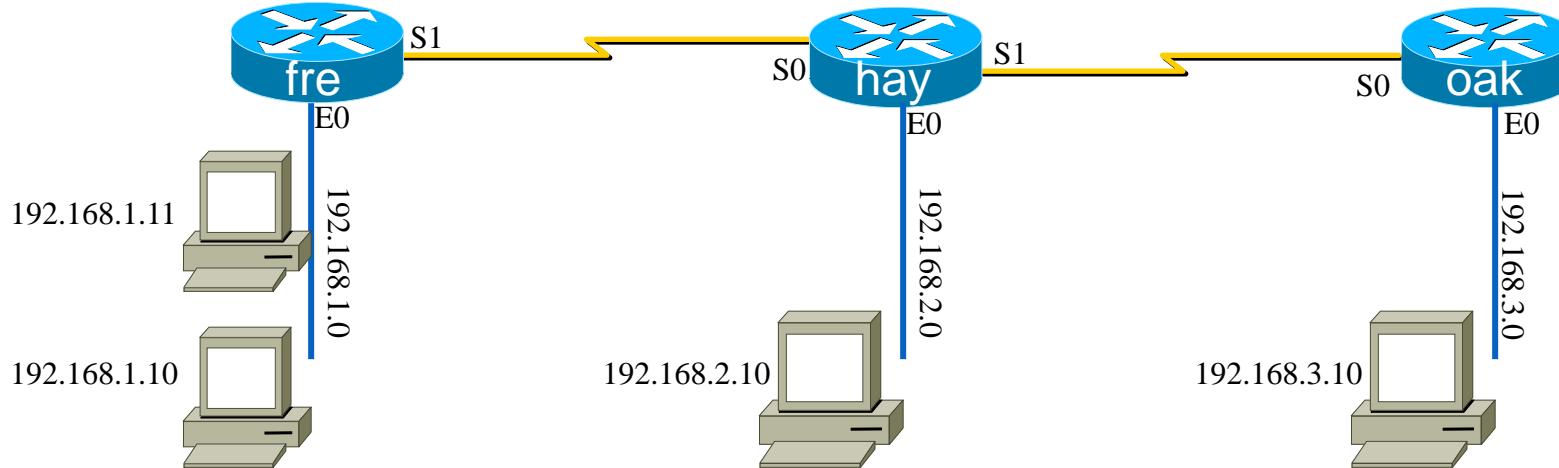
Some Protocols with Port Numbers

- FTP – 21
- Telnet – 23
- SMTP – 25
- DNS – 53
- TFTP – 69
- WWW, HTML – 80
- POP3 - 110
- SNMP - 161

ACL Configuration Example

What will this list do?

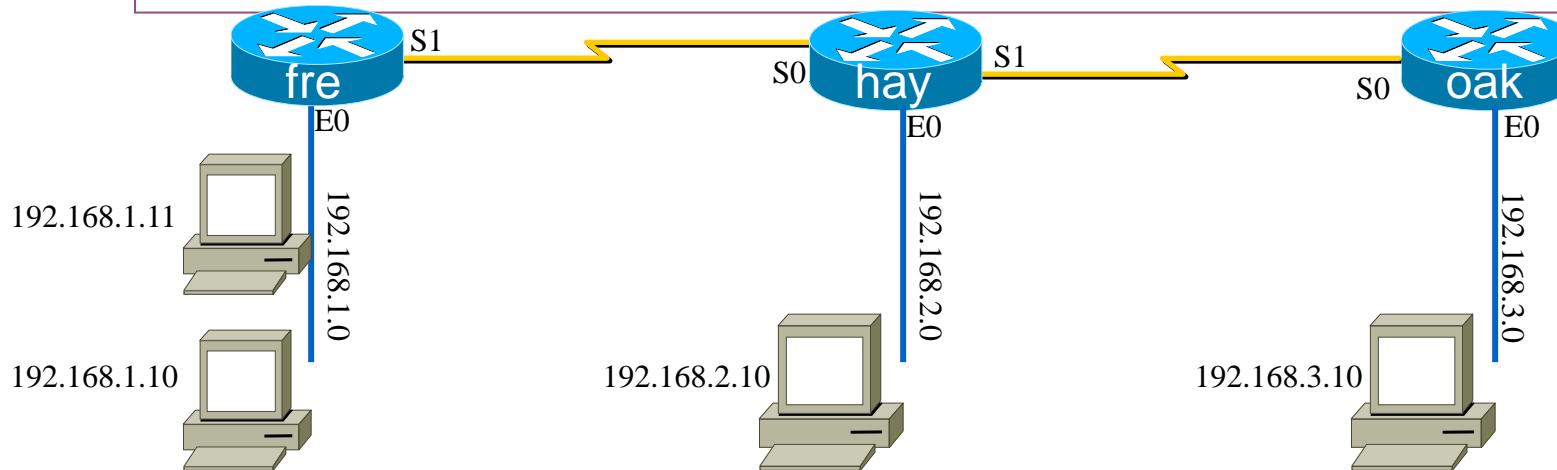
```
fre(config)#access-list 101 deny tcp any 192.168.3.10  
0.0.0.0 eq 80  
fre(config)#access-list 101 permit ip any any  
fre(config)#int e0  
fre(config-if)#ip-access group 101 in  
fre(config-if)#^z
```



ACL Configuration Example

What will this list do?

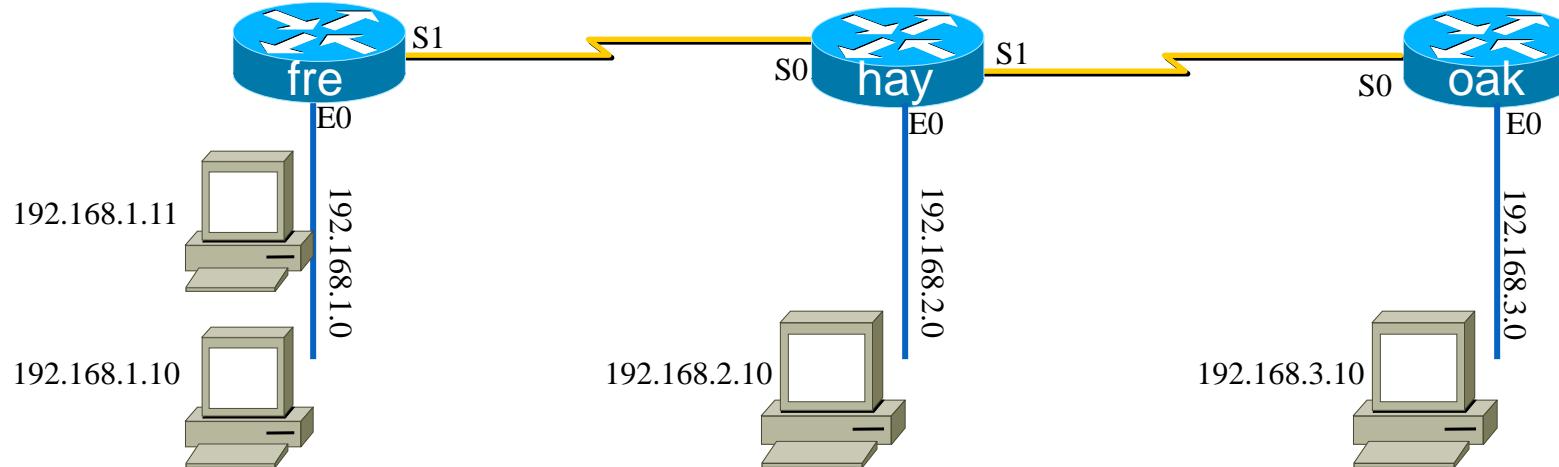
```
fre(config)#access-list 101 deny tcp 192.168.1.10  
0.0.0.0 any eq 80  
fre(config)#access-list 101 deny tcp 192.168.1.0  
0.0.0.255 any eq 21  
fre(config)#access-list 101 permit ip any any  
fre(config)#int e0  
fre(config-if)#ip-access group 101 in  
fre(config-if)#^z
```



ACL Configuration Example

What will this list do? (What's wrong here?)

```
fre(config)#access-list 101 deny tcp 192.168.1.10  
0.0.0.0 any eq 80  
fre(config)#int e0  
fre(config-if)#ip-access group 101 in  
fre(config-if)#^z
```

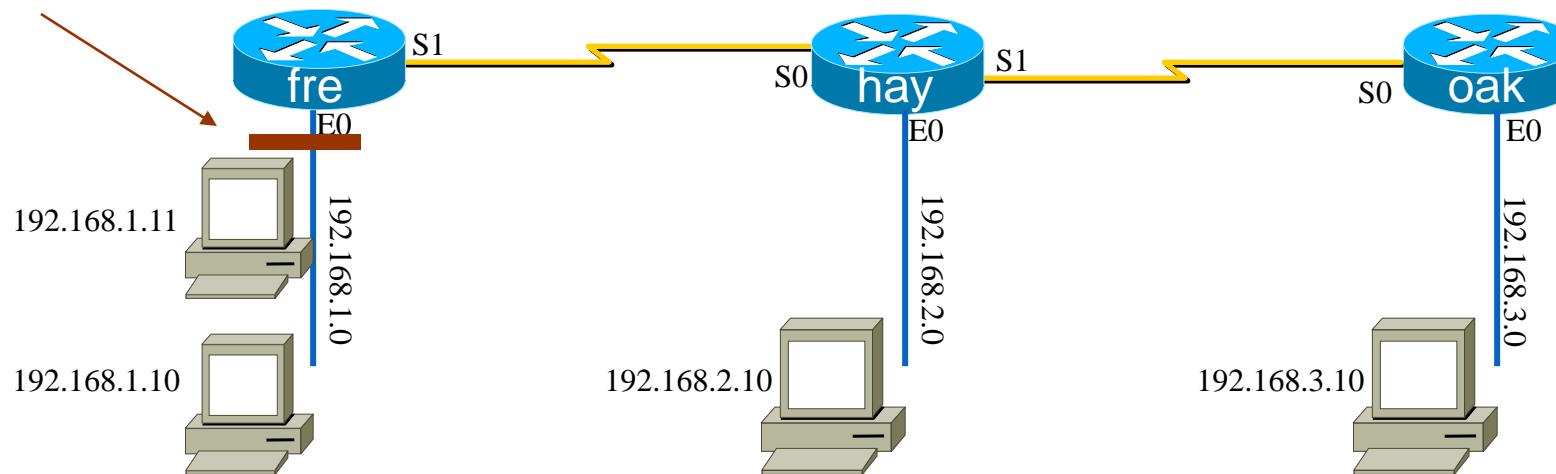


Extended ACL Placement

Blocking traffic from Fremont LAN to Oakland PC

Place extended ACL close to source.

```
fre(config)#access-list 101 deny ip any host 192.168.3.10
fre(config)#access-list 101 permit ip any any
fre(config)#int e0
fre(config-if)#ip-access group 101 in
```

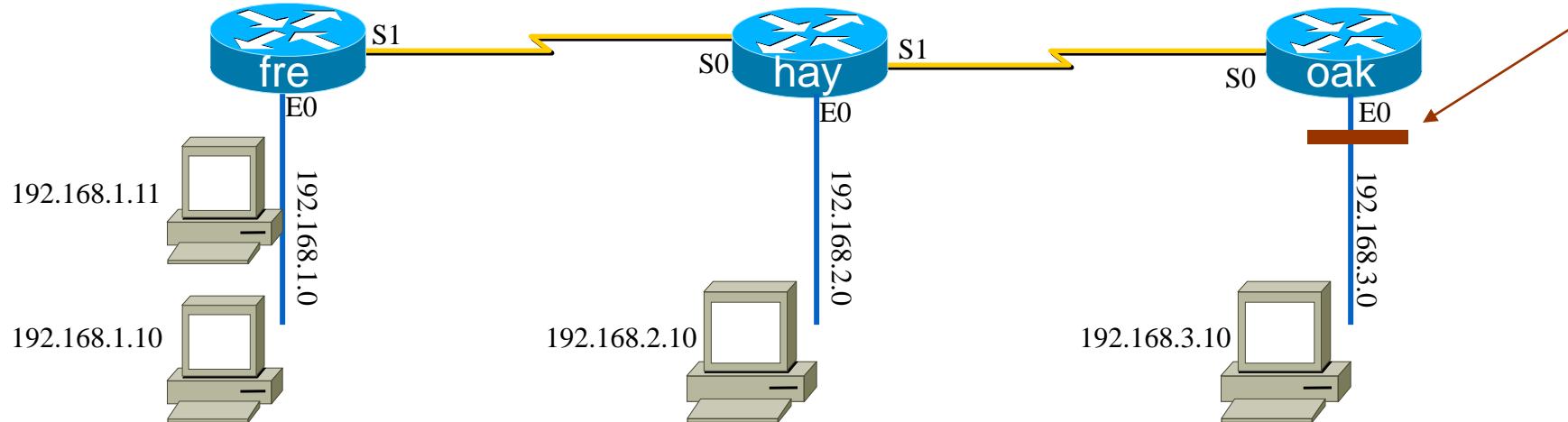


Standard ACL Placement

Blocking traffic from Fremont LAN to Oakland LAN

Place standard ACL close to destination.

```
oak(config)#access-list 10 deny 192.168.1.0 0.0.0.255  
oak(config)#access-list 10 permit any  
oak(config)#int e0  
oak(config-if)#ip-access group 10 out
```



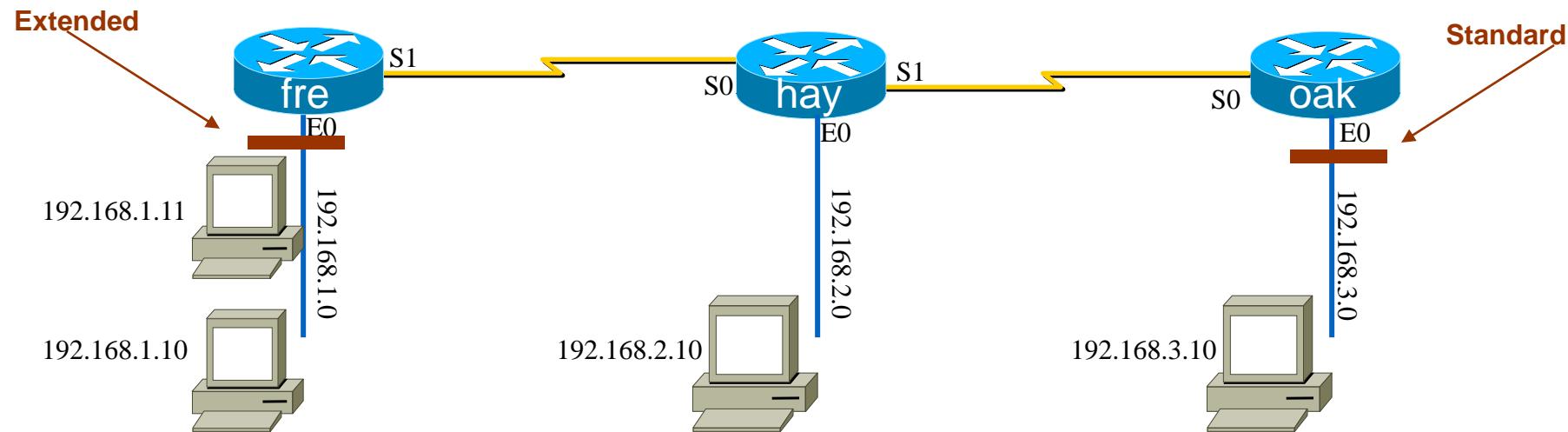
ACL Placement

Blocking traffic from Fremont LAN to Oakland PC

Standard or Extended ACL

Which seems more efficient?

Why?



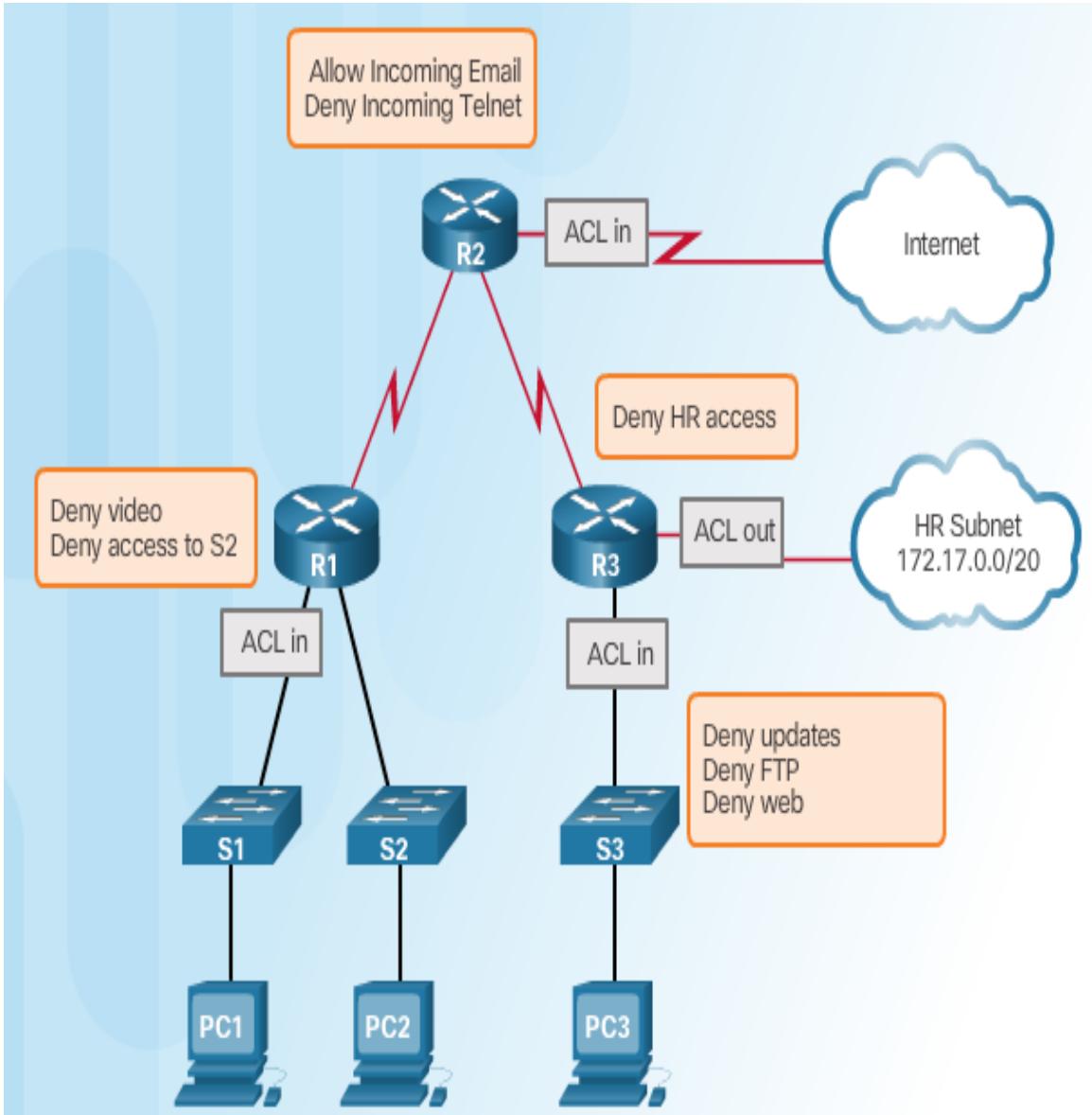
Editing ACLs

- The exec adds new lines (rules) to an ACL at the end; probably not where you want them.
- To change lines in the middle, you must delete the entire list and re-enter it.
- Or - dump your config out to a text file & edit it as follows:

ACL Syntax Summary

- Network Computing has published a great summary chart of the “[anatomy of an ACL](#)”

Introduction to Access Control Lists



Protocol	Range
IP	1-99, 1300-1999
Extended IP	100-199, 2000-2699
Ethernet type code	200-299
DECnet and Extended DECnet	300-399
XNS	400-499
Extended XNS	500-599
AppleTalk	600-699
Ethernet address	700-799
IPX	800-899
Extended IPX	900-999
IPX SAP	1000-1099
Extended transparent bridging	1100-1199

Configuring Numbered and Named ACLs

Standard Numbered ACL Syntax

```
access-list {acl-#} {permit | deny | remark} source-addr [source-wildcard] [log]
```

Extended Numbered ACL Syntax

```
access-list acl-# {permit | deny | remark} protocol source-addr [source-wildcard]  
dest-addr [dest-wildcard] [operator port] [established]
```

Named ACL Syntax

```
Router(config)# ip access-list [standard | extended] name_of_ACL
```

Standard ACE Syntax

```
Router(config-std-nacl)# {permit | deny | remark} {source [source-wildcard] | any}
```

Extended ACE Syntax

```
Router(config-ext-nacl)# {permit | deny | remark} protocol source-addr [source-wildcard]  
dest-address [dest-wildcard] [operator port]
```

Applying an ACL

Syntax - Apply an ACL to an interface

```
Router(config-if)# ip access-group {acl-#|name} {in|out}
```

Syntax - Apply an ACL to the VTY lines

```
Router(config-line)# access-class {acl-#|name} {in|out}
```

Example - Named Standard ACL

```
R1(config)# ip access-list standard NO_ACCESS
R1(config-std-nacl)# deny host 192.168.11.10
R1(config-std-nacl)# permit any
R1(config-std-nacl)# exit
R1(config)# interface g0/0
R1(config-if)# ip access-group NO_ACCESS out
```

```
R1(config)# ip access-list extended SURFING
R1(config-ext-nacl)# permit tcp 192.168.10.0 0.0.0.255 any eq 80
R1(config-ext-nacl)# permit tcp 192.168.10.0 0.0.0.255 any eq 443
R1(config-ext-nacl)# exit
R1(config)# ip access-list extended BROWSING
R1(config-ext-nacl)# permit tcp any 192.168.10.0 0.0.0.255 established
R1(config-ext-nacl)# exit
R1(config)# interface g0/0
R1(config-if)# ip access-group SURFING in
R1(config-if)# ip access-group BROWSING out
```

Example - Named Extended ACL

Applying an ACL (Cont.)

Syntax - Apply an ACL to the VTY lines

```
Router(config-line)# access-class {acl-#|name} {in|out}
```

Example - Named ACL on VTY lines with logging

```
R1(config)# ip access-list standard VTY_ACCESS
R1(config-std-nacl)# permit 192.168.10.10 log
R1(config-std-nacl)# deny any
R1(config-std-nacl)# exit
R1(config)# line vty 0 4
R1(config-line)# access-class VTY_ACCESS in
R1(config-line)# end
R1#
R1#!The administrator accesses the vty lines from 192.168.10.10
R1#
*Feb 26 18:58:30.579: %SEC-6-IPACCESSLOGNP: list VTY_ACCESS permitted 0
192.168.10.10 -> 0.0.0.0, 5 packets
R1# show access-lists
Standard IP access list VTY_ACCESS
    10 permit 192.168.10.10 log (6 matches)
    20 deny any
```

ACL Configuration Guidelines

- Create an ACL globally and then apply it.
- Ensure the last statement is an implicit deny any or deny any any.
- Remember that statement order is important because ACLs are processed top-down. As soon as a statement is matched the ACL is exited.
- Ensure that the most specific statements are at the top of the list.
- Remember that only one ACL is allowed per interface, per protocol, per direction.
- Remember that new statements for an existing ACL are added to the bottom of the ACL by default.
- Remember that router generated packets are not filtered by outbound ACLs.
- Place standard ACLs as close to the destination as possible.
- Place extended ACLs as close to the source as possible.

Editing Existing ACLs

Existing access list has three entries

```
Router# show access-lists
Extended IP access list 101
    10 permit tcp any any
    20 permit udp any any
    30 permit icmp any any
```

Access list has been edited, which adds a new ACE and replaces ACE line 20.

```
Router(config)# ip access-list extended 101
Router(config-ext-nacl)# no 20
Router(config-ext-nacl)# 5 deny tcp any any eq telnet
Router(config-ext-nacl)# 20 deny udp any any
```

Updated access list has four entries

```
Router# show access-lists
Extended IP access list 101
    5 deny tcp any any eq telnet
    10 permit tcp any any
    20 deny udp any any
    30 permit icmp any any
```

Sequence Numbers and Standard ACLs

Existing access list has four entries

```
router# show access-lists
Standard IP access list 19
 10 permit 192.168.100.1
 20 permit 10.10.10.0, wildcard bits 0.0.0.255
 30 permit 201.101.110.0, wildcard bits 0.0.0.255
 40 deny any
```

Access list has been edited, which adds a new ACE that permits a specific IP address.

```
router(config)# ip access-list standard 19
router(config-std-nacl)# 25 permit 172.22.1.1
```

Updated access list places the new ACE before line 20

```
router# show access-lists
Standard IP access list 19
 10 permit 192.168.100.1
 25 permit 172.22.1.1
 20 permit 10.10.10.0, wildcard bits 0.0.0.255
 30 permit 201.101.110.0, wildcard bits 0.0.0.255
 40 deny any
```

Packet Filters Contd.

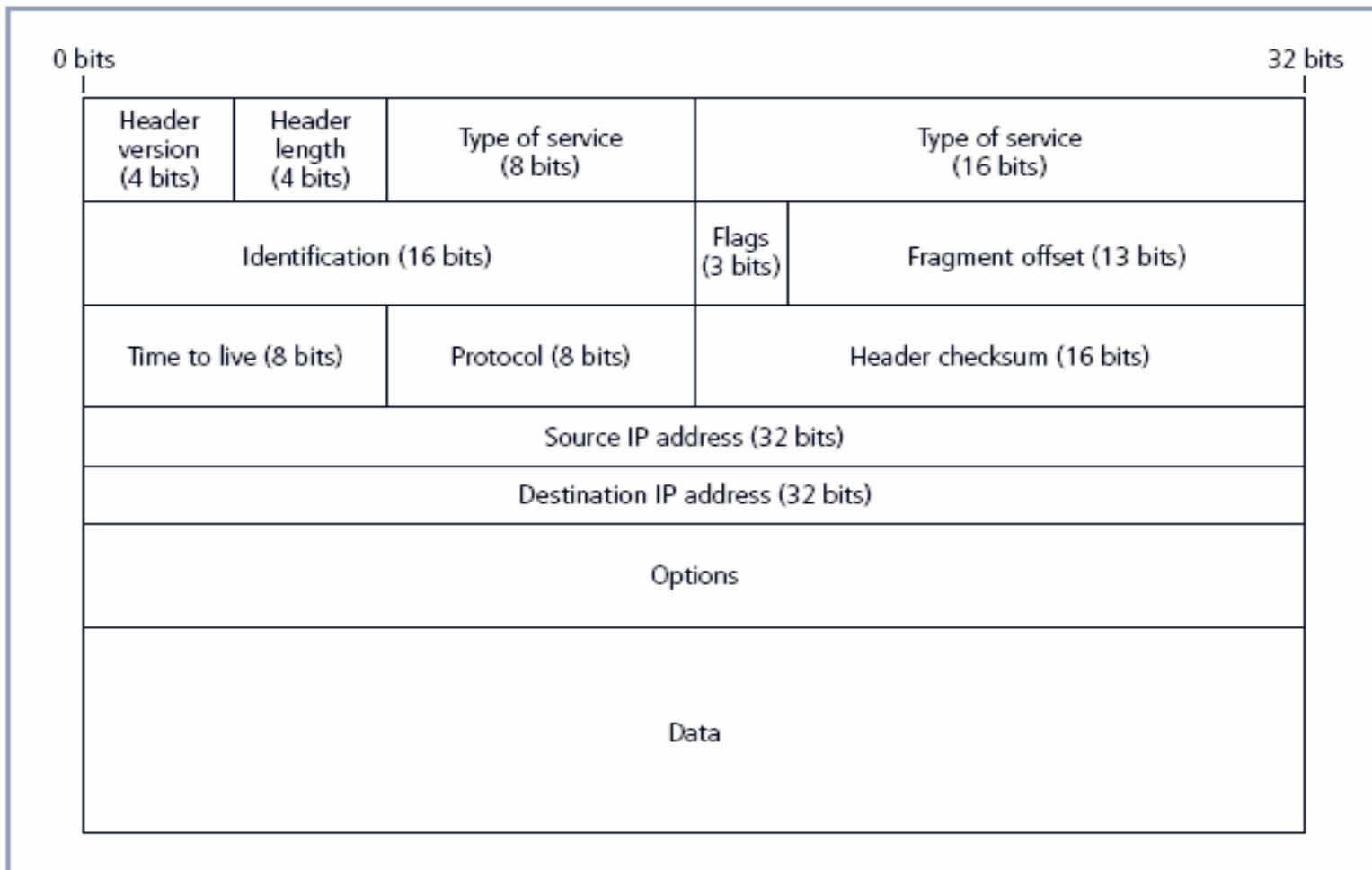


FIGURE 6-1 IP Packet Structure

Packet Filters Contd.

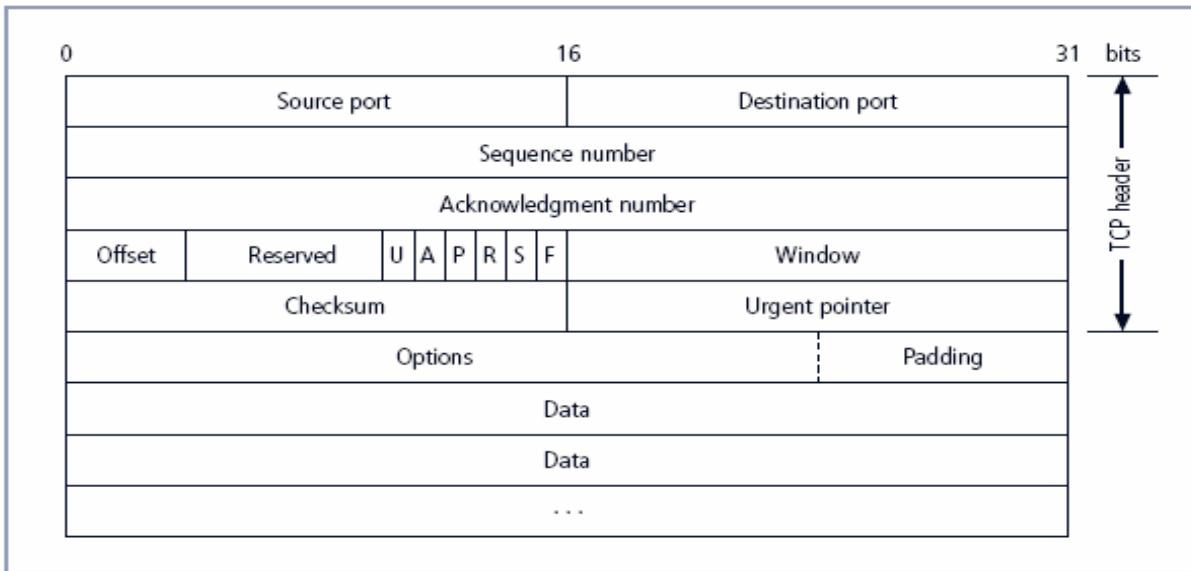


FIGURE 6-2 TCP Packet Structure

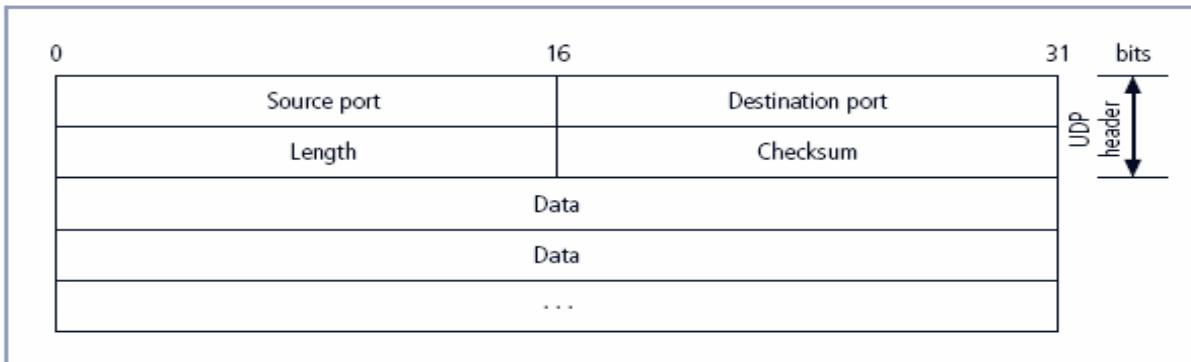


FIGURE 6-3 UDP Datagram Structure

IP Datagram Fragmentation

- **Identification** is selected by sender host, and is unique for each packet sent by that host in recent past.
- **M bit:** shows whether there are more fragments for that packet following (b)
- **Offset:** deals with possibly reordered fragments (counts 8-byte words; why not count bytes?)
- A packet can be fragmented at multiple routers
- Reassembly is only performed at the receiving host
- When should receiver give up on the reassembly of a packet?

(a)

Start of header			
Ident= x		0	Offset= 0
Rest of header			
1400 data bytes			

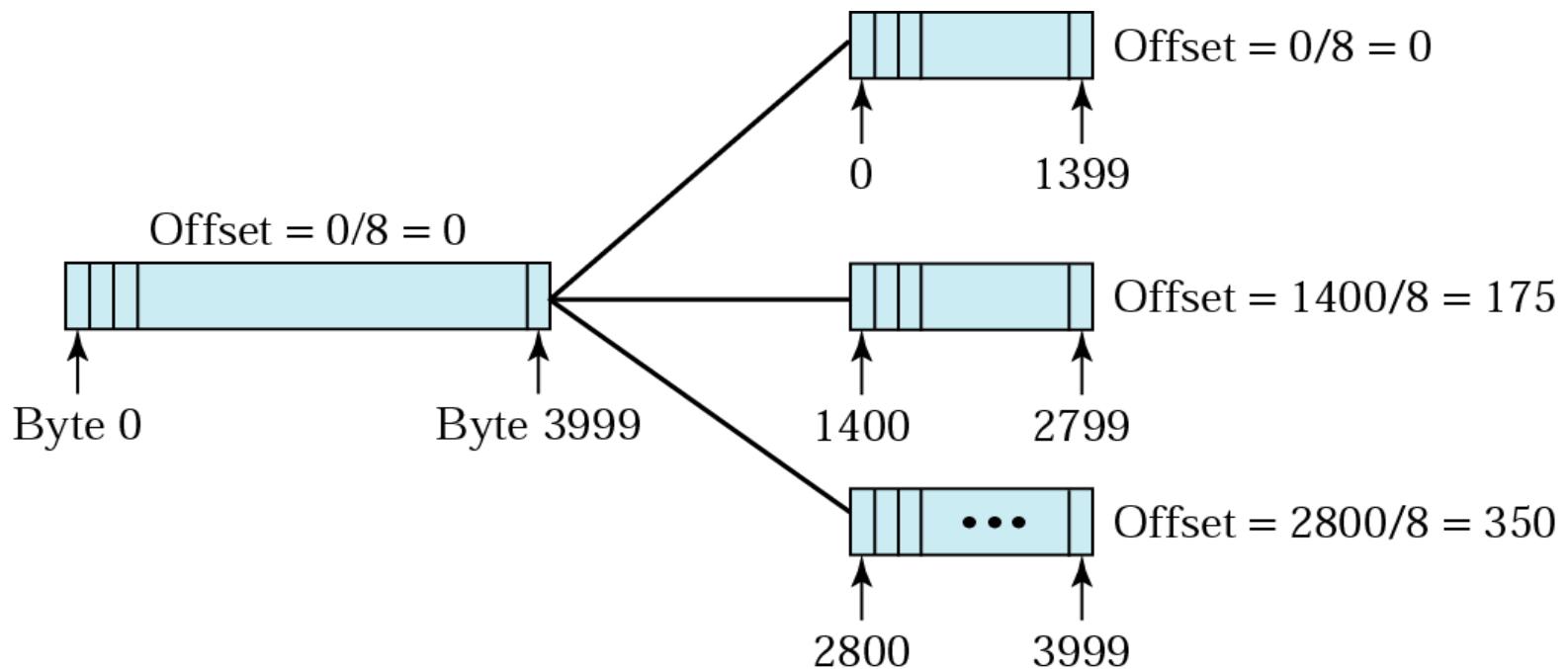
(b)

Start of header			
Ident= x		1	Offset= 0
Rest of header			
512 data bytes			

Start of header			
Ident= x		1	Offset= 64
Rest of header			
512 data bytes			

Start of header			
Ident= x		0	Offset= 128
Rest of header			
376 data bytes			

IP Fragmentation Example



Problem No. 1

Problem 1. (IP fragmentation)

Suppose an IP datagram containing 256 bytes of data is divided into two fragments, each containing 128 bytes of data. Fill in the table for the header fields for the two fragments.

Header field	Datagram	Fragment 1	Fragment 2
header length	5		
total length	276		
identification	3		
MF	0		
fragment offset	0		

Solution to Problem No. 1

An IP datagram of 256 bytes of data is divided into two fragments, each containing 128 bytes of data. The header length is 20 bytes but measured in units of 4 bytes. Total length is the sum of header and data bytes. Identification number is the same in all fragments to indicate that they belong to the same original datagram. MF flag is always 1 for all fragments except for the last fragment. Fragment offset indicates where the fragment's data belongs in the original datagram measured in units of 8 bytes.

Header field	Datagram	Fragment 1	Fragment 2
header length	5	5	5
total length	276	148	148
identification	3	3	3
MF	0	1	0
fragment offset	0	0	16

Problem No. 1

One approach to defeat the tiny fragment attack is to enforce a minimum length of the transport header that must contained in the first fragment of an IP packet. If the first fragment is rejected, all the subsequent fragment can be rejected. However, the nature of the IP is such that fragments may arrive out of order. Thus, an intermediate fragment may pass through the filter before the initial fragment is rejected. How can this situation be handled.

Answer to Problem No. 1

It will be impossible for the destination host to complete reassembly of the packet if the first fragment is missing, and therefore the entire packet will be discarded by the destination after a time-out.

Problem No. 2

In an IPv4 packet, the size of the payload in the first fragment, in octet is equal to Total Length-(4*IHL). If this value is less than the required minimum (8 octets for TCP) then this fragment and the entire packet are rejected. Suggest an alternative method of achieving the same result using only the Fragment offset Field.

Solution to Problem No. 2

When a TCP packet is fragmented so as to force interesting header fields out of the zero-offset fragment, there must exist a fragment with FO equal to 1. If a packet with $FO = 1$ is seen, conversely, it could indicate the presence, in the fragment set, of a zero-offset fragment with a transport header length of eight octets Discarding this one-offset fragment will block reassembly at the receiving host and be as effective as the direct method described above.

Problem No. 5

- If an ingress packet has 127.0.0.0 as both of its source IP address and its destination IP address, should the packet filer block this packet? Why? (Note: 127.0.0.1 is the address of localhost)

Problem No. 6

- If an ingress packet has 0.0.0.0 as its source address or its destination IP address, should the packet filter block this packet? Why? (Note: 0.0.0.0 is the address for broadcasting message)

Problem No. 7

If an ingress packet belongs to an existing TCP connection, should this packet be blocked? Why?

Problem No. 8

If an ingress packet has 25 or 80 as its destination port number, should this packet be blocked? Why?

Problem No. 9

SMTP (Simple Mail Transfer Protocol) is the standard protocol for transferring mail between hosts over TCP. A TCP connection is set up between a user agent and a server program. The server listens on TCP port 25 for incoming connection requests. The user end of the connection is on a TCP port number above 1023. Suppose you wish to build a packet filter rule set allowing inbound and outbound SMTP traffic. You generate the following ruleset. Describe the effect of each rule.

Rule	Direction	Src Addr	Dest Addr	Protocol	Dest Port	Action
A	In	External	Internal	TCP	25	Permit
B	Out	Internal	External	TCP	>1023	Permit
C	Out	Internal	External	TCP	25	Permit
D	In	External	Internal	TCP	>1023	Permit
E	Either	Any	Any	Any	Any	Deny

Circuit Gateways

Can encrypted packet be relayed through a circuit gateway?

Packet Filters Contd.

Can encrypted packet pass through Packet Filter Firewall ?

Intrusion Detection and Prevention System

Introduction

- An intrusion is a type of attack on information assets in which the instigator attempts to gain entry into a system or disrupt the normal operations of a system with, almost always, the intent to do malicious harm.
- **Incident response** is the identification of, classification of, response to, and recovery from an incident, and it is frequently discussed in terms of prevention, detection, reaction, and correction.
- **Intrusion prevention** consists of activities that seek to deter an intrusion from occurring.
- Intrusion detection consists of procedures and systems that are created and operated to detect system intrusions.

Introduction Contd.

- Intrusion correction activities finalize the restoration of operations to a normal state, and by seeking to identify the source and method of the intrusion in order to ensure that the same type of attack cannot occur again, they return to intrusion prevention—thus closing the incident response loop.
- **Alert or Alarm:** An indication that a system has just been attacked and/or continues to be under attack.
- **False Attack Stimulus:** An event that triggers alarms and causes a false positive when no actual attacks are in progress. Scenarios that test the configuration of IDPS may use false attack stimuli to determine if the IDPS's can distinguish between these stimuli and real attack.

IDPS Terminology

- **False Negative:** The failure of an IDPS to react to an actual attack event. This is the most grievous failure, since the purpose of an IDPS is to detect and respond to attacks.
- **False Positive:** An alarm or alert that indicates that an attack is in progress or that an attack has successfully occurred when, in fact, there has been no such attack. False positive tend to make users insensitive to alarms and thus reduce their reactivity to actual intrusion events.

IDPS Terminology Contd.

- **Noise:** The ongoing activity from alarm events that are accurate and noteworthy but not necessarily significant as potentially successful attacks.
- **Site Policy:** The rules and configuration guidelines governing the implementation and operation of IDSs within the organization.
- **Site Policy Awareness:** An IDS's ability to dynamically modify its site policies in reaction or response to environmental activity.

IDPS Terminology Contd.

- **True Attack Stimulus:** An event that triggers alarms and causes an IDS to react as if a real attack was in progress. The event may be an actual attack in which an attacker is at work on a system compromise attempt, or it may be drill, in which security personal are using tools to conduct test of a network segment.

IDPS Terminology Contd.

- **Tuning** : The process of adjusting an IDPS to maximize its efficiency in detecting true positive while minimizing both false positive sand false negatives.
- **Confidence Value:** A value associated with an IDS's ability to detect and identify an attack correctly. The confidence value an organization places in the IDPS is based on experience and past performance measurements.

Intrusion Detection System (IDS)

IDS is a device, typically another computer that monitors the activity to identify malicious or suspicious events.

An IDS is a sensor, like a smoke detector, that arises an alarm if specific things occurs

- IDS performs a variety of functions
- Monitors system configuration for vulnerabilities and misconfigurations.
- Accessing the integrity of critical system and data files.
- Recognizing known attack patterns in system activity.
- Identifying abnormal activity through statistical analysis.
- Managing audit trails and highlighting user violations of policy or normal activity.
- Correcting system configuration errors.
- Installing and operating traps to record information about intruders.

Goals for Intrusion Detection System

Ideally, an IDS should be fast, simple and accurate, while at the same time being complete. It should detect all attacks with little performance penalty. As IDS could use some-or-all of the following design approaches.

- Filter on packet header
- Filter on packet content
- Maintain connection state
- Use complex, multipacket signature
- Use minimal number of signatures with maximum effect
- Filter in real time, online
- Hide its presence
- Use Optimal sliding time window size to match signatures

Why Use an IDPS?

- To prevent problem behaviors by increasing the perceived risk of discovery and punishment for those who would attack or otherwise abuse the system.
- To detect attacks and other security violations that are not prevented by other security measures.
- To detect and deal with the preambles to attacks
- To document the existing threat to an organization
- To act as quality control for security design and administration, especially of large and complex enterprises
- To provide useful information about intrusions that do take place, allowing improved diagnosis, recovery, and correction of causative factors.

Distinguishes between IPS and IDS

- IPS stops the attack itself. Examples of how this could be done as follows
 - Terminate the network connection or user session that is being used for attack.
 - Block access to the target (or possibly other likely targets) from the offending user account, IP address or other attacker attribute.
 - Block all access to the targeted host, service, application and other services.

Distinguishes between IPS and IDS

- IPS changes the security environment. The IPS could change configuration of other security controls to disrupt an attack. Common examples are reconfiguring a network device (e.g. firewall, router and switch) to block access from an attacker to the target and altering a host based firewall on a target to block incoming attacks.
- The IPS changes the attack content. Some IPS technologies can remove or replace malicious portion of an attack to make it benign.

Types of IDP Systems

- IDSs operate as network-based, host-based, or application-based systems.
 - A network-based IDS is focused on protecting network information assets.
 - A host-based version is focused on protecting the server or host's information assets.
 - The application-based model works on one or more host systems that support a single application and is oriented to defend that specific application from special forms of attack.

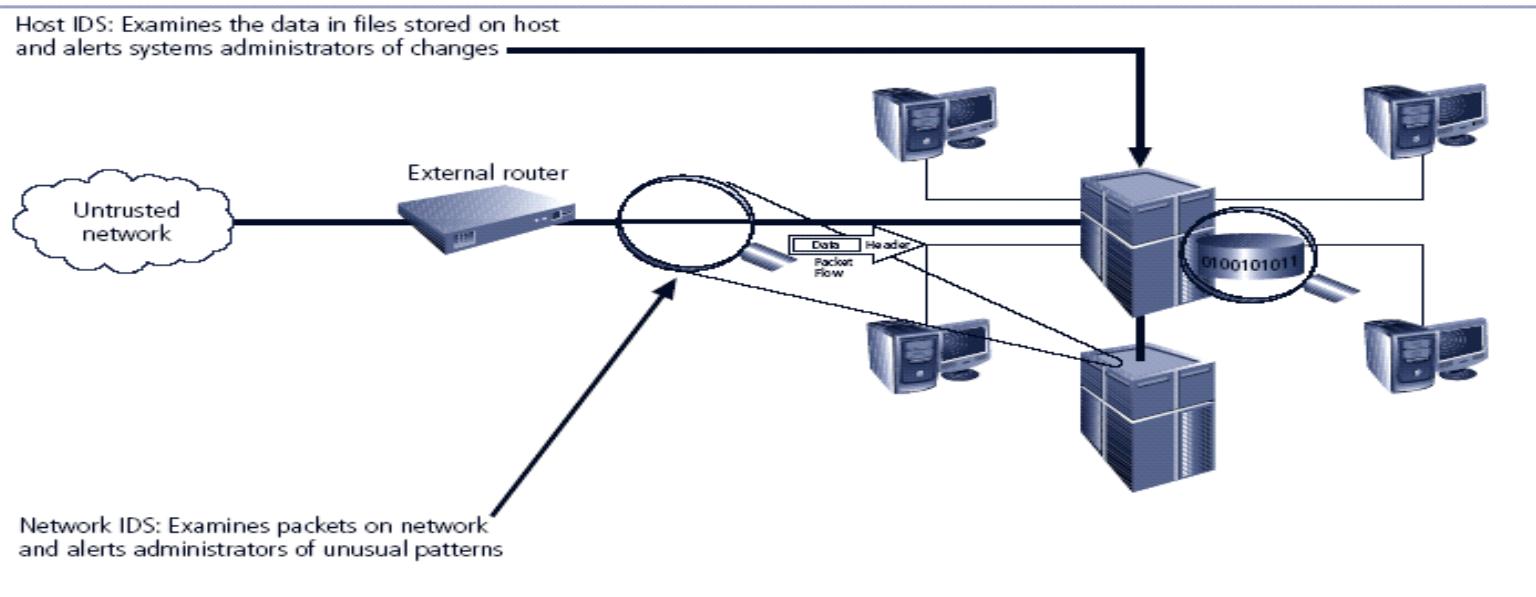


FIGURE 7-1 Intrusion Detection Systems

Host Based IDPS

- A host-based IDPS (HIDPS) resides on a particular computer or server, known as the host, and monitors activity only on that system.
- HIDPSs are also known as **system integrity verifiers**, as they benchmark and monitor the status of key system files and detect when an intruder creates, modifies, or deletes monitored files.
- Most HIDPSs work on the principle of configuration or change management, which means they record the sizes, locations, and other attributes of system files. **The HIDPS then triggers an alert when one of the following changes occurs:** file attributes change; new files are created; or existing files are deleted.

Host Based IDPS Contd.

- A HIDPS has an advantage over NIDPS in that it can usually be installed in such a way that it can access information that is encrypted when traveling over the network
- A HIDPS relies on the classification of files into various categories and then applies various notification actions, depending on the rules in the HIDPS configuration.
- Managed HIDPSs can monitor multiple computers simultaneously by creating a configuration file on each monitored host and by making each HIDPS report back to a master console system, which is usually located on the system administrator's computer.

Host Based IDPS Contd.

Advantages of HIDPSs

- A HIDPS can detect local events on host systems and also detect attacks that may elude a network-based IDPS.
- A HIDPS functions on the host system, where encrypted traffic will have been decrypted and is available for processing.
- The use of switched network protocols does not affect a HIDPS.
- A HIDPS can detect inconsistencies in how applications and systems programs were used by examining the records stored in audit logs.

Disadvantages of HIDPSs

- HIDPSs pose more management issues since they are configured and managed on each monitored host. A HIDPS is vulnerable both to direct attacks and to attacks against the host operating system.
- A HIDPS is not optimized to detect multi-host scanning, nor is it able to detect the scanning of non-host network devices, such as routers or switches.
- A HIDPS is susceptible to some denial-of-service attacks.
- A HIDPS can use large amounts of disk space to retain the host OS audit logs, and to function properly, it may require disk capacity to be added to the system.

Disadvantages of HIDPSs

- A HIDPS can inflict a performance overhead on its host systems, and in some cases, may reduce system performance below acceptable levels.

Why Use an IDS?

- To prevent problem behaviors by increasing the perceived risk of discovery and punishment for those who would attack or otherwise abuse the system
- To detect attacks and other security violations that are not prevented by other security measures
- To detect and deal with the preambles to attacks
- To document the existing threat to an organization
- To act as quality control for security design and administration, especially of large and complex enterprises
- To provide useful information about intrusions that do take place, allowing improved diagnosis, recovery, and correction of causative factors

Network-based IDPS (NIDPS)

- A Network-based IDPS (NIDPS) resides on a computer or appliance connected to a segment of an organization's network and monitors network traffic on that network segment, looking for indications of ongoing or successful attacks.
- When a situation occurs that the NIDS is programmed to recognize as an attack, it responds by sending notifications to administrators.

Network-based IDPS (NIDPS)

- When examining the packets transmitted through an organization's networks, a NIDPS looks for attack patterns within network traffic, such as large collections of related items that are of a certain type, which could indicate that a Denial-of-Service(DoS) attack is underway, or it looks for the exchange of a series of related packets in a certain pattern, which could indicate that a port scan is in progress.

Network-based IDPS (NIDPS)

- NIDPSs are installed at a specific place in the network (such as on the inside of an edge router) from where it is possible to watch the traffic going into and out of a particular network segment.
- The NIDPS can be deployed to watch a specific grouping of host computers on a specific network segment, or it can be installed to monitor all traffic between the systems that make up an entire network.
- To determine whether or not an attack has occurred or may be underway, NIDPSs look for attack patterns by comparing measured activity to known signatures in their knowledge base.

Network-based IDPS (NIDPS) Contd.

- This is accomplished by the comparison of captured network traffic using a special implementation of the TCP/IP stack that reassembles the packets and applies protocol stack or application protocol verification. In the process of protocol stack verification, the NIDSs look for invalid data packets.
- In application protocol verification, the higher-order protocols are examined for unexpected packet behavior, or improper use.

Advantages of NIDPSs.

- Good network design and placement of NIDPS devices can enable an organization to use a few devices to monitor a large network.
- NIDPSs are usually passive devices and can be deployed into existing networks with little or no disruption to normal network operations.
- **NIDPSs are not usually susceptible** to direct attack and, in fact, may not be detectable by attackers.

Disadvantages of NIDPSs

- A NIDPS can become overwhelmed by network volume and fail to recognize attacks it might otherwise have detected.
- NIDPSs require access to all traffic to be monitored.
- NIDPSs cannot analyze encrypted packets, making some of the network traffic invisible to the process.
- NIDPSs cannot reliably ascertain if an attack was successful or not.

Some forms of attack are not easily discerned by NIDPSs, specifically those involving fragmented packets.

Features Explored by an Attacker

An Intruder explores the following features to get into a system

- Password Cracking
- Software bugs and Buffer overflow
- Java scripts, Active X control and CGI scripts
- Weaknesses in Internet Protocol and services
- Domain Name Service Attack
- Attack through mail Protocol
- Pings a range of IP address to find which machines are alive.

Utilities that Aid Intruders

The utilities that aid an intruder may be categorized as general utilities and special utilities

General Utilities

Ping	:	To see if a host is alive
Traceroute	:	To find the route to a host
Nslookup/dig	:	To discover DNS information
Whois	:	To find out domain registration information
Finger	:	To find out which users are logged in and to collect information about them.
r-commands	:	Scans for rlogin, rsh

Utilities that Aid Intruders Cond.

Special Utilities

- | | | |
|-----------------|---|--|
| Crack Package | : | To crack password, steal password out of the system or to sniff them off the wire. |
| Sniff Utilities | : | To watch raw network traffic |
| Port Scanner | : | To Scan/strobe/probe TCP and UDP ports |
| Ping Sweepers | : | To Ping large numbers of machines to see which ones are active. |
| Exploit Packs | : | To know how to exploit holes on the systems when the users are already logged in. |
| War dialer | : | To dial several phone numbers to locate dial-in-ports. |

Signature-Based IDS

- A signature-based IDS (also known as a knowledge-based IDS) examines data traffic in search of patterns that match known signatures—that is, preconfigured, predetermined attack patterns.
- Signature-based IDS technology is widely used because many attacks have clear and distinct signatures. The problem with the signature-based approach is that as new attack strategies are identified, the IDS's database of signatures must be continually updated periodically; otherwise the attacks that use new strategies will not be recognized and might succeed.

Signature-Based IDS

- Another weakness of the signature-based method is that a slow, methodical attack might escape detection, if the relevant IDPS attack signature has a shorter time frame.

Signature-Based IDS Contd.

- A signature-based detections also known as **operational detections** .
- Signature detection inspects current events (i.e., event occurred in a small interval of the current time) and decide whether these events are acceptable.
- Signature detections are also referred to as **rule-based detections**. Because, these are associated with set of rules.
- Signature detection includes **Network Signature** and **Host Based signature**.

Signature-Based IDS Contd.

Network Signature

- Network signature provide information of packet behaviors that may affect the normal execution of the system.
- Network signature consists of **header signature** and **payload signature**. Payload signature also referred as **content signature**.

Payload Signature

Checking packet content is a basic intrusion detection technique in NIDS. It uses payload signature to determine which packets are acceptable and which packets are not acceptable.

Signature-Based IDS Contd.

Header Signature

- Checking packet header is another basic intrusion detection technique in NBD systems.
- It uses header signature to identify malicious packets.
- For example, in a broadcast attack, the attacker sends packets to the target in which the source address and destination address are the same. Broadcast may cause the target system to crash. Thus, when it finds a packet has the same source and destination address, the IDS should inform the alarm manager about it.

Signature-Based IDS Contd.

Host-based signature

- Host based signature provide information of event behaviors that may affect the normal execution of the system.
- For example, “three consecutive failed logins”
- Host-based signature can be characterized as

Signature-Based IDS Contd.

Single-event Signature

Suspicious behavior in a single command is a single-event signature. Listed below are several activities with single-event signature.

- A single command that modifies a system file
- A single command that reads another user's personal directory
- A single command that modifies another user's file
- A single command that duplicates the system files

Signature-Based IDS Contd.

Multi Event Signature

- The sequence of single event signature form a multi-event signature.
- For example, the three consecutive failed login is a multi-event signature.

Signature-Based IDS Contd.

Multi-Host Signature

- A multi-host signature is a signature formed from a sequence of single-event signature and multi-event signature on several different hosts.
- For example, the following activities have a multi-host signature

A user attempted to log on to machine A and failed. Then, the user attempted to log on to another machine B and failed again. After this attempted to log on to machine C and still failed.

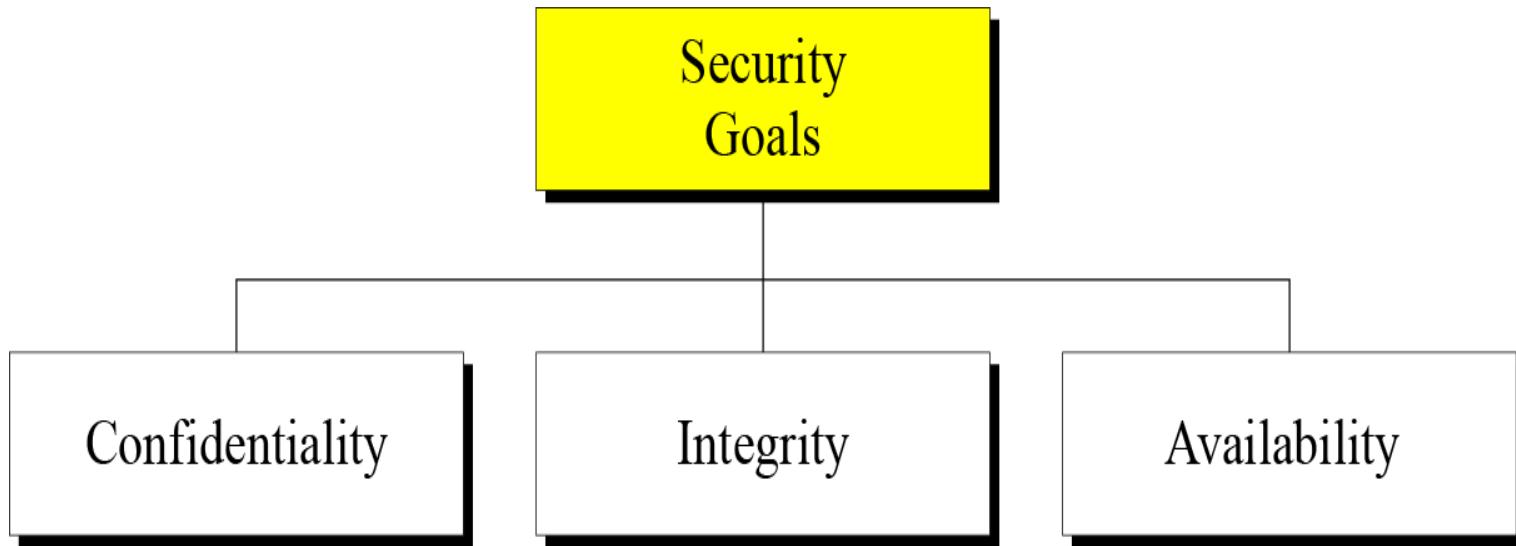
Statistical Anomaly-Based IDS

- The statistical anomaly-based IDS (stat IDS) or behavior-based IDS collects statistical summaries by observing traffic that is known to be normal. This normal period of evaluation establishes a performance baseline.
- Once the baseline is established, the stat IDS will periodically sample network activity, and, using statistical methods, compare the sampled network activity to this baseline. When the measured activity is outside the baseline parameters, it is said to exceed the **clipping level**, and the IDS will trigger an alert to notify the administrator.

Statistical Anomaly-Based IDS

- The data that is measured from the normal traffic and is used to prepare the baseline can include variables such as host memory or CPU usage, network packet types, and packet quantities.
- The advantage of the statistical anomaly-based approach is that IDPS can detect new types of attacks, since it looks for abnormal activity of any type. Unfortunately, these systems require much overhead and processing capacity than signature based IDPS, because they must constantly compare pattern of activity against the baseline.
- Another drawback is that these systems may not detect minor changes to system variables and may generate many false positive.

Security Goals



Confidentiality is probably the most common aspect of information security. We need to protect our confidential information. An organization needs to guard against those malicious actions that endanger the confidentiality of its information.

Integrity

Information needs to be changed constantly. **Integrity** means that changes need to be done only by authorized entities and through authorized mechanisms.

Availability

The information created and stored by an organization needs to be available to authorized entities. Information needs to be constantly changed, which means it must be accessible to authorized entities.

ATTACKS

The three goals of security—confidentiality, integrity, and availability—can be threatened by security attacks.

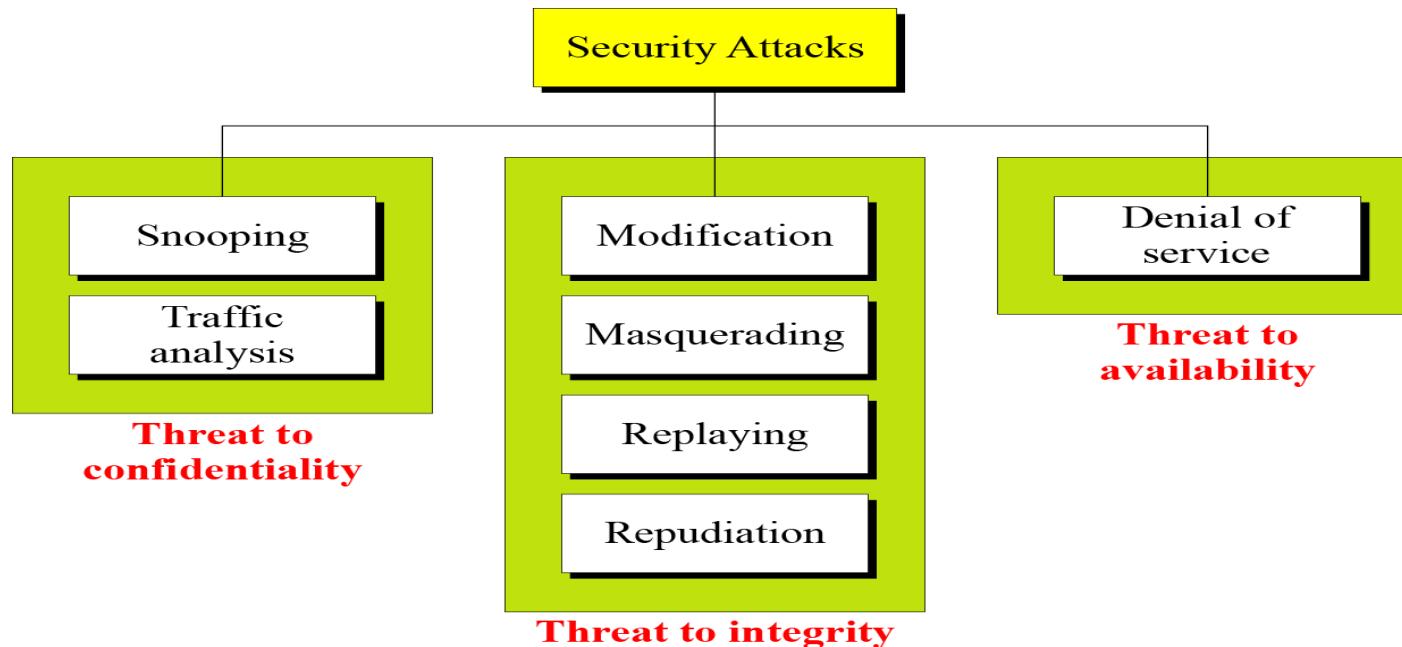
Attacks Threatening Confidentiality

Attacks Threatening Integrity

Attacks Threatening Availability

Passive versus Active Attacks

Taxonomy of attacks with relation to security goals



Attacks Threatening Confidentiality

Snooping refers to unauthorized access to or interception of data.

Traffic analysis refers to obtaining some other type of information by monitoring online traffic.

Attacks Threatening Availability

Denial of Service (DoS) is a very common attack. It may slow down or totally interrupt the service of a system.

Attacks Threatening Integrity

Modification means that the attacker intercepts the message and changes it.

Masquerading or **spoofing** happens when the attacker impersonates somebody else.

Replaying means the attacker obtains a copy of a message sent by a user and later tries to replay it.

Repudiation means that sender of the message might later deny that she has sent the message; the receiver of the message might later deny that he has received the message.

Categorization of Passive and Active Attacks

<i>Attacks</i>	<i>Passive/Active</i>	<i>Threatening</i>
Snooping Traffic analysis	Passive	Confidentiality
Modification Masquerading Replaying Repudiation	Active	Integrity
Denial of service	Active	Availability

SERVICES AND MECHANISMS

ITU-T provides some security services and some mechanisms to implement those services. Security services and mechanisms are closely related because a mechanism or combination of mechanisms are used to provide a service..

Security Services

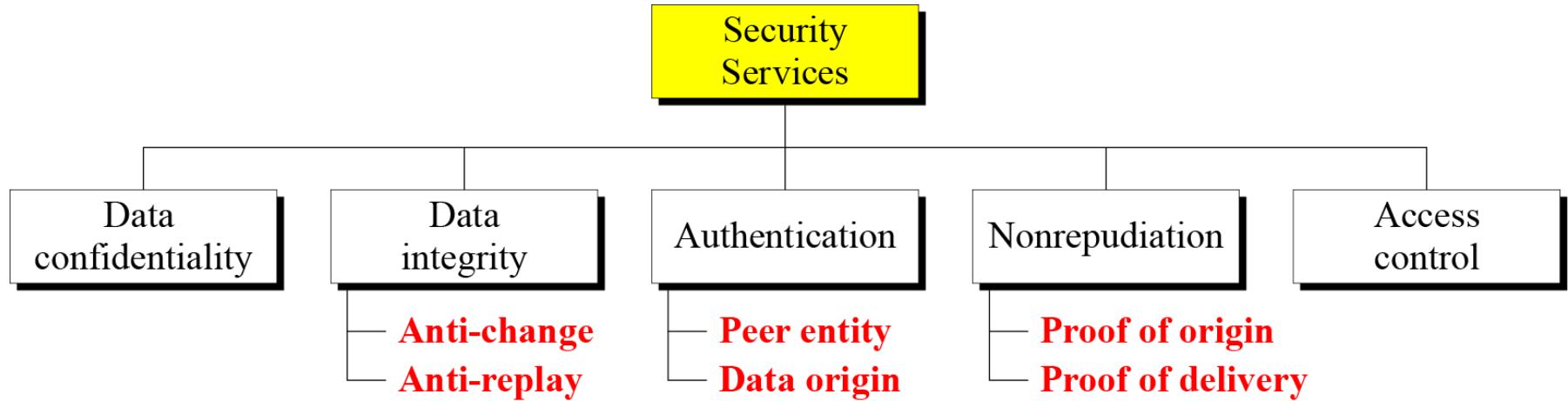
Security Mechanism

Relation between Services and Mechanisms

Security Attack : Any action that compromises the security of information owned by an organization.

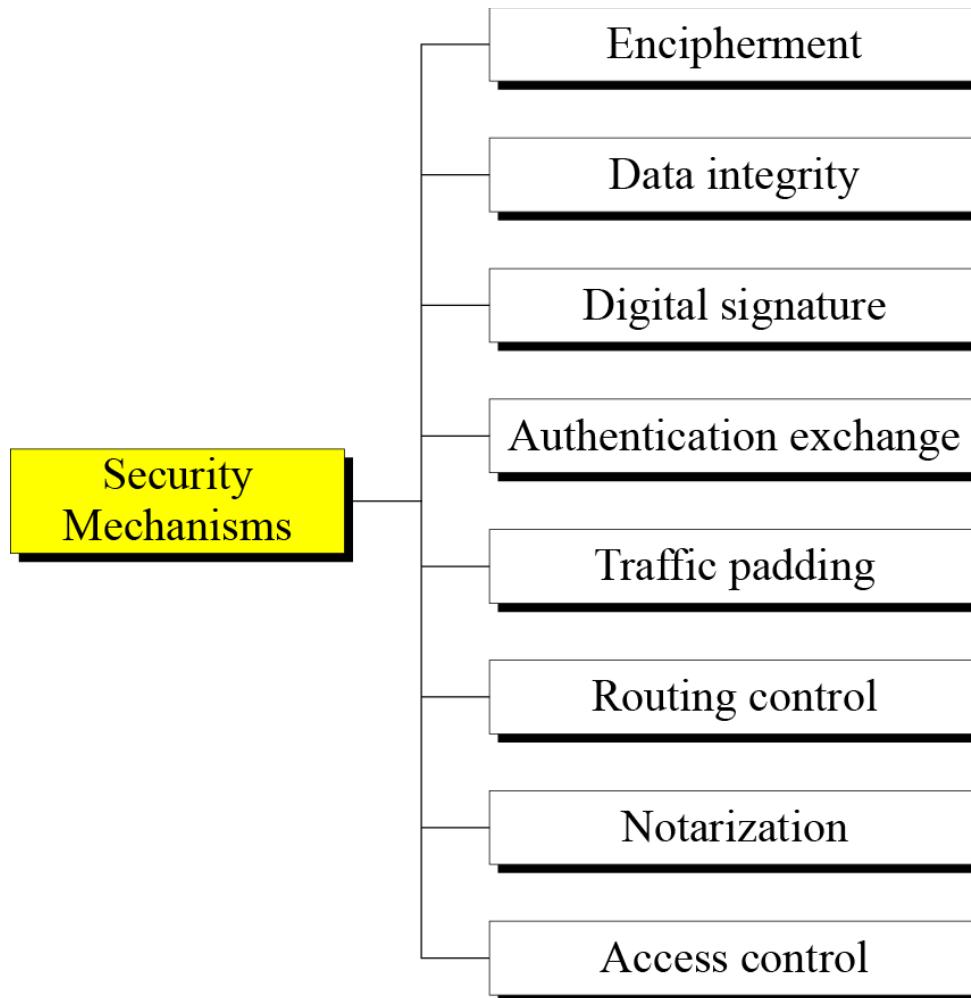
Security Services: A service that enhances the security of the data processing systems and the information transfers of an organization. The services are more intended to counter security attacks and they make use of one or more security mechanism to provide the service.

Security Services



Security Mechanisms

Security Mechanism: A mechanism that is designed to detect, prevent, or recover from a security attack.



- **Notarization** means selecting a third party to control the communication between two entities. This can be done for example, to prevent repudiation.

Relation between Services and Mechanisms

Security Service: A service that enhances the security of data processing systems and information transfers. A security service makes use of one or more security mechanisms.

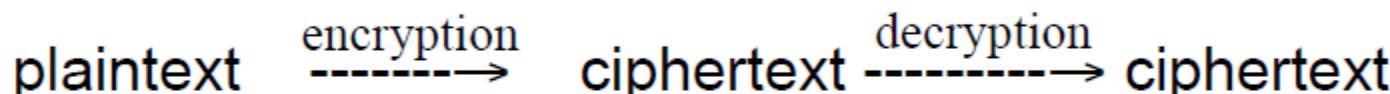
Security Mechanism: A mechanism that is designed to detect, prevent, or recover from a security attack.

<i>Security Service</i>	<i>Security Mechanism</i>
Data confidentiality	Encipherment and routing control
Data integrity	Encipherment, digital signature, data integrity
Authentication	Encipherment, digital signature, authentication exchanges
Nonrepudiation	Digital signature, data integrity, and notarization
Access control	Access control mechanism

What is a Cryptosystem?

Plaintext: data to be ‘hidden’

Ciphertext: “not-meaningful” data



Definition

A **cryptosystem** is a five-tuple (P, C, K, E, D) , s. t.:

1. P is a finite set of possible plaintexts
2. C is a finite set of possible ciphertexts
3. K , the keyspace, is the set of possible keys
4. For each $k \in K$, there are
 - encryption rule e_k , $e_k: P \rightarrow C$,
 - decryption rule d_k , $d_k: C \rightarrow P$,
 - s.t. $d_k(e_k(x)) = x$

Basic Terminology

- **plaintext** - the original message
- **ciphertext** - the coded message
- **cipher** - algorithm for transforming plaintext to ciphertext
- **key** - info used in cipher known only to sender/receiver
- **encipher (encrypt)** - converting plaintext to ciphertext
- **decipher (decrypt)** - recovering ciphertext from plaintext
- **cryptography** - study of encryption principles/methods
- **cryptanalysis (codebreaking)** - the study of principles/methods of deciphering ciphertext *without* knowing key
- **cryptology** - the field of both cryptography and cryptanalysis

Two Requirements for Secure use of Symmetric Encryption

- **Strong Encryption Algorithm**

The opponent should be unable to decrypt ciphertext or discover the key even if the opponent is in possession of a number of ciphertexts together with the plaintext that produced each ciphertext.

$$Y = E_K(X)$$

$$X = D_K(Y)$$

- **Secret key known only to Sender / Receiver**

Sender and Receiver must have obtained copies of the secrete key in a secure fashion and must keep the key secure. If someone can discover the key and knows the algorithm all communication using this key is readable.

Symmetric & Public Key Algorithms

Encryption and Decryption keys are known to both communicating parties (Alice and Bob).

They are usually related and it is easy to derive the decryption key once one knows the encryption key.

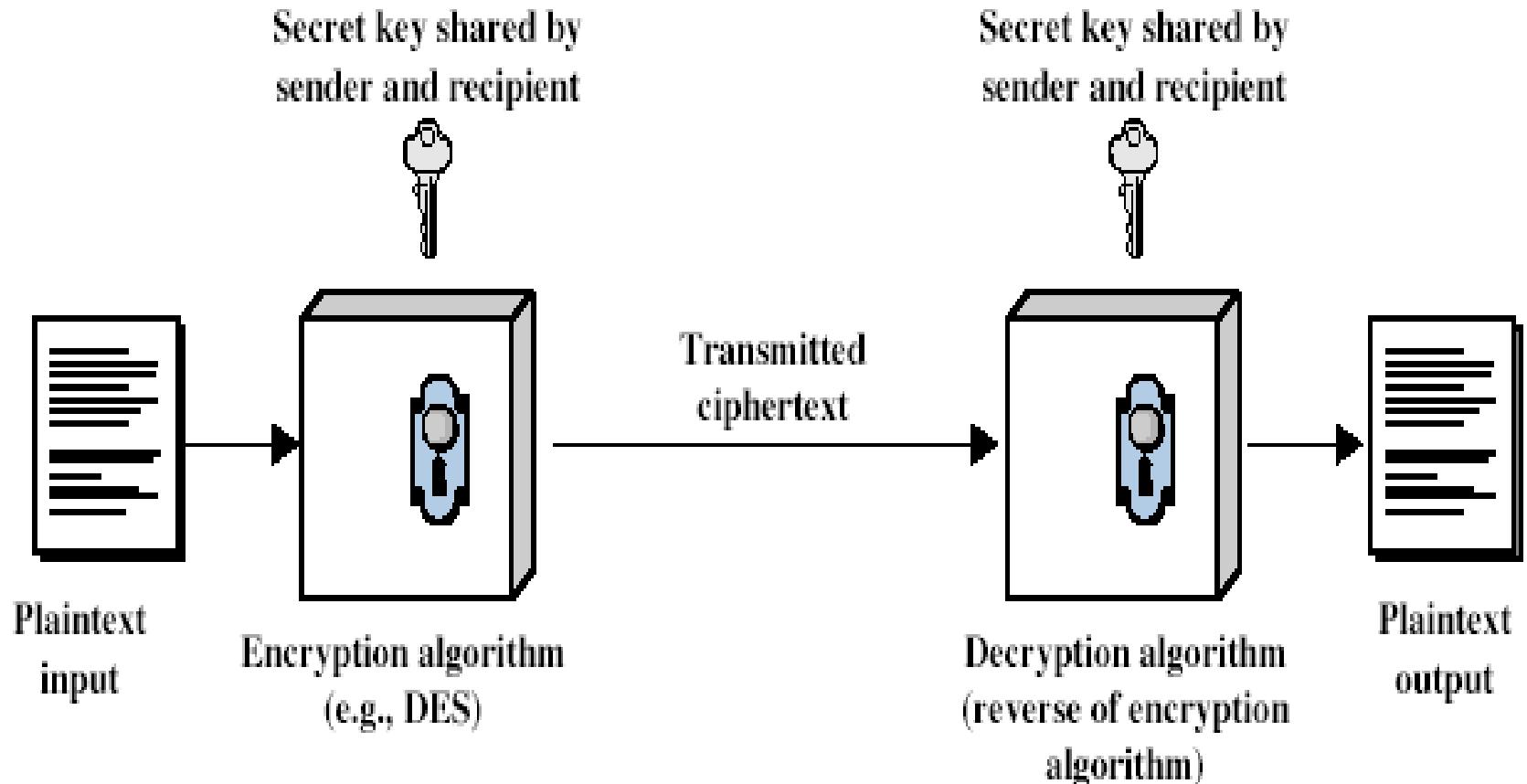
In most cases, they are identical. All of the classical (pre-1970) cryptosystems are symmetric. Examples : DES and AES (Rijndael)

A Secret should be shared (or agreed) between the communicating parties.

Symmetric Encryption

- or conventional / private-key / single-key
- sender and recipient share a common key
- all classical encryption algorithms are private-key
- was only type prior to invention of public-key in 1970's

Symmetric Cipher Model



Kerckhoff's Principle

While assessing the strength of a cryptosystem, one should always assume that the enemy knows the cryptographic algorithm used.

The security of the system, therefore, should be based on

- * the quality (strength) of the algorithm but not its obscurity
- * the key space (or key length)

The Number of Keys Used

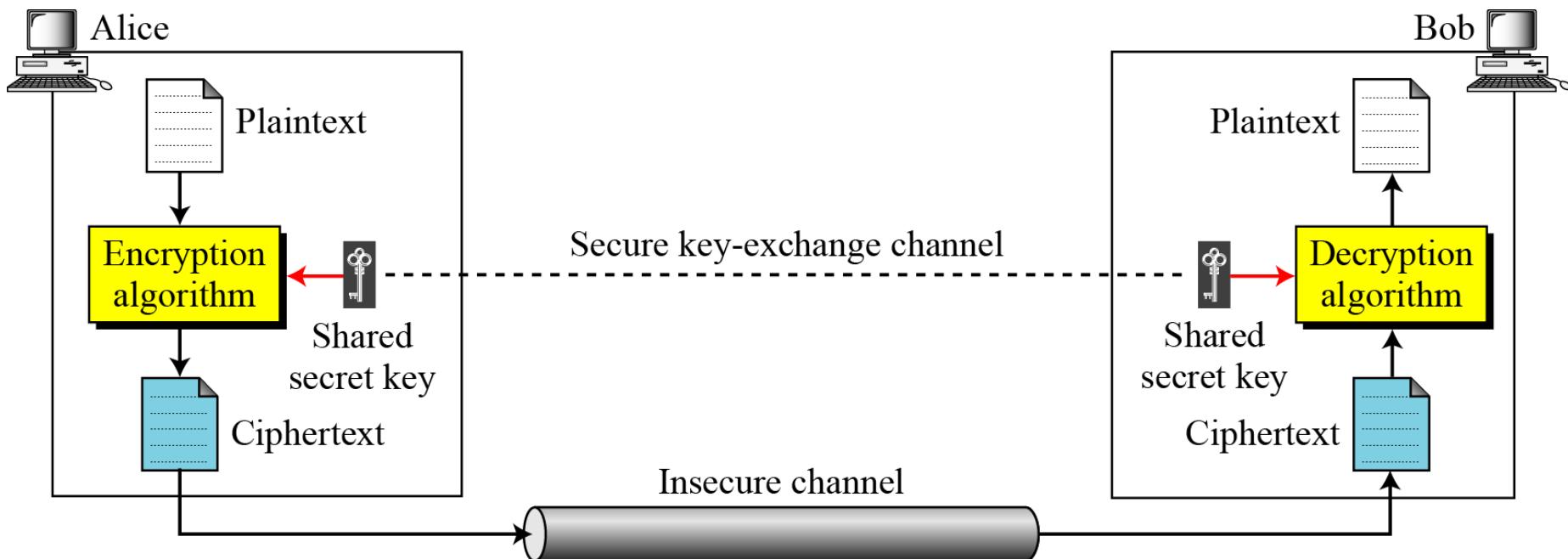
The Number of Keys Used

- If both sender and receiver use the same key, the system is referred to as symmetric key, single key, secret key or conventional encryption.
- If the sender and receiver use different keys, the system is referred to asymmetric, two key or public key encryption.

The way in which the plain text is processed

- A block cipher processes the input one block of elements at a time, producing an output block for each input block.
- A stream cipher processes the input elements continuously producing output one element at a time as it goes along.

General idea of Symmetric-key Cipher



General idea of Symmetric-key Cipher Contd.

If P is the plaintext, C is the ciphertext, and K is the key,

Encryption: $C = E_k(P)$

Decryption: $P = D_k(C)$

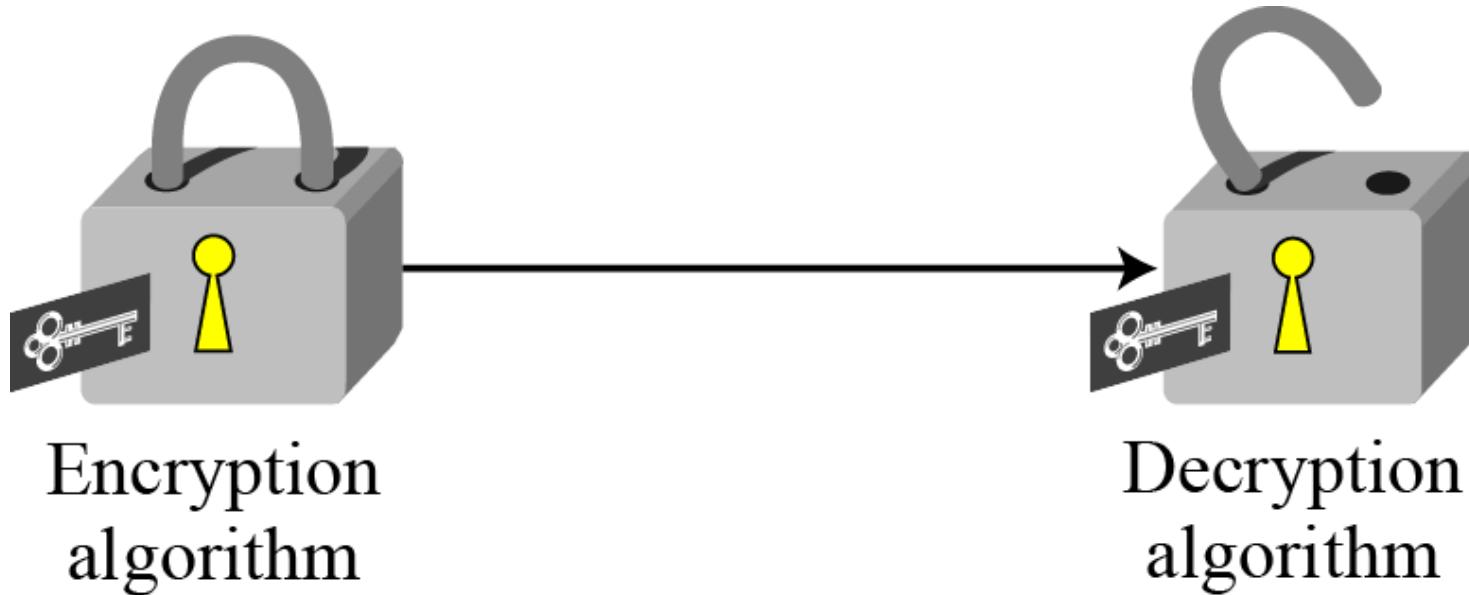
In which, $D_k(E_k(x)) = E_k(D_k(x)) = x$

We assume that Bob creates P_1 ; we prove that $P_1 = P$:

Alice: $C = E_k(P)$

Bob: $P_1 = D_k(C) = D_k(E_k(P)) = P$

Locking and Unlocking with the Same Key



Type of Operations used for Transforming Plain text to Cipher Text

All encryption algorithms are based on two general principles:

- substitution in which each element in the plain text (bit, letter, group of bits or letters) is mapped into another element, and transposition in which elements in the plaintext are rearranged. The fundamental requirement is that no information be lost (that is, that all operations are reversible).
- Most systems referred to as product systems, involve multiple stages of substitution and transposition.

SUBSTITUTION CIPHERS

A substitution cipher replaces one symbol with another. Substitution ciphers can be categorized as either monoalphabetic ciphers or polyalphabetic ciphers.

**A Substitution Cipher Replaces one Symbol
with Another.**

Monoalphabetic Ciphers

In monoalphabetic substitution, the relationship between a symbol in the plaintext to a symbol in the ciphertext is always one-to-one.

Monoalphabetic Ciphers Contd.

The following shows a plaintext and its corresponding ciphertext. The cipher is probably monoalphabetic because both l's (els) are encrypted as O's.

Plaintext: hello

Ciphertext: KHOOR

Monoalphabetic Ciphers Additive Cipher Contd.

The simplest monoalphabetic cipher is the additive cipher. This cipher is sometimes called a **shift cipher** and sometimes a **Caesar cipher**, but the term additive cipher better reveals its mathematical nature.

Plaintext and ciphertext in Z_{26}

Plaintext →	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Ciphertext →	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Value →	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Monoalphabetic Substitution Cipher

Because additive, multiplicative, and affine ciphers have small key domains, they are very vulnerable to brute-force attack.

A better solution is to create a mapping between each plaintext character and the corresponding ciphertext character. Alice and Bob can agree on a table showing the mapping for each character.

An example key for monoalphabetic substitution cipher

Plaintext →	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Ciphertext →	N	O	A	T	R	B	E	C	F	U	X	D	Q	G	Y	L	K	H	V	I	J	M	P	Z	S	W

Monoalphabetic Substitution Cipher Contd.

Example 3.13

We can use the key in Figure 3.12 to encrypt the message

this message is easy to encrypt but hard to find the key

The ciphertext is

ICFVQRVVNEFVRNVSIYRGAHSLIOJICNHTIYBFGTICRXRS

Autokey Cipher

$$P = P_1 P_2 P_3 \dots$$

$$C = C_1 C_2 C_3 \dots$$

$$k = (k_1, P_1, P_2, \dots)$$

$$\text{Encryption: } C_i = (P_i + k_i) \bmod 26$$

$$\text{Decryption: } P_i = (C_i - k_i) \bmod 26$$

Example

Assume that Alice and Bob agreed to use an autokey cipher with initial key value $k_1 = 12$. Now Alice wants to send Bob the message “Attack is today”. Enciphering is done character by character.

Plaintext:	a	t	t	a	c	k	i	s	t	o	d	a	y
P's Values:	00	19	19	00	02	10	08	18	19	14	03	00	24
Key stream:	12	00	19	19	00	02	10	08	18	19	14	03	00
C's Values:	12	19	12	19	02	12	18	00	11	7	17	03	24
Ciphertext:	M	T	M	T	C	M	S	A	L	H	R	D	Y

Vigenere Cipher

$$P = P_1 P_2 P_3 \dots$$

$$C = C_1 C_2 C_3 \dots$$

$$K = [(k_1, k_2, \dots, k_m), (k_1, k_2, \dots, k_m), \dots]$$

$$\text{Encryption: } C_i = P_i + k_i$$

$$\text{Decryption: } P_i = C_i - k_i$$

Example

We can encrypt the message “She is listening” using the 6-character keyword “PASCAL”.

Plaintext:	s	h	e	i	s	l	i	s	t	e	n	i	n	g
P's values:	18	07	04	08	18	11	08	18	19	04	13	08	13	06
Key stream:	15	00	18	02	00	11	15	00	18	02	00	11	15	00
C's values:	07	07	22	10	18	22	23	18	11	6	13	19	02	06
Ciphertext:	H	H	W	K	S	W	X	S	L	G	N	T	C	G

Vigenere Cipher

Example

Let us see how we can encrypt the message “She is listening” using the 6-character keyword “PASCAL”. The initial key stream is (15, 0, 18, 2, 0, 11). The key stream is the repetition of this initial key stream (as many times as needed).

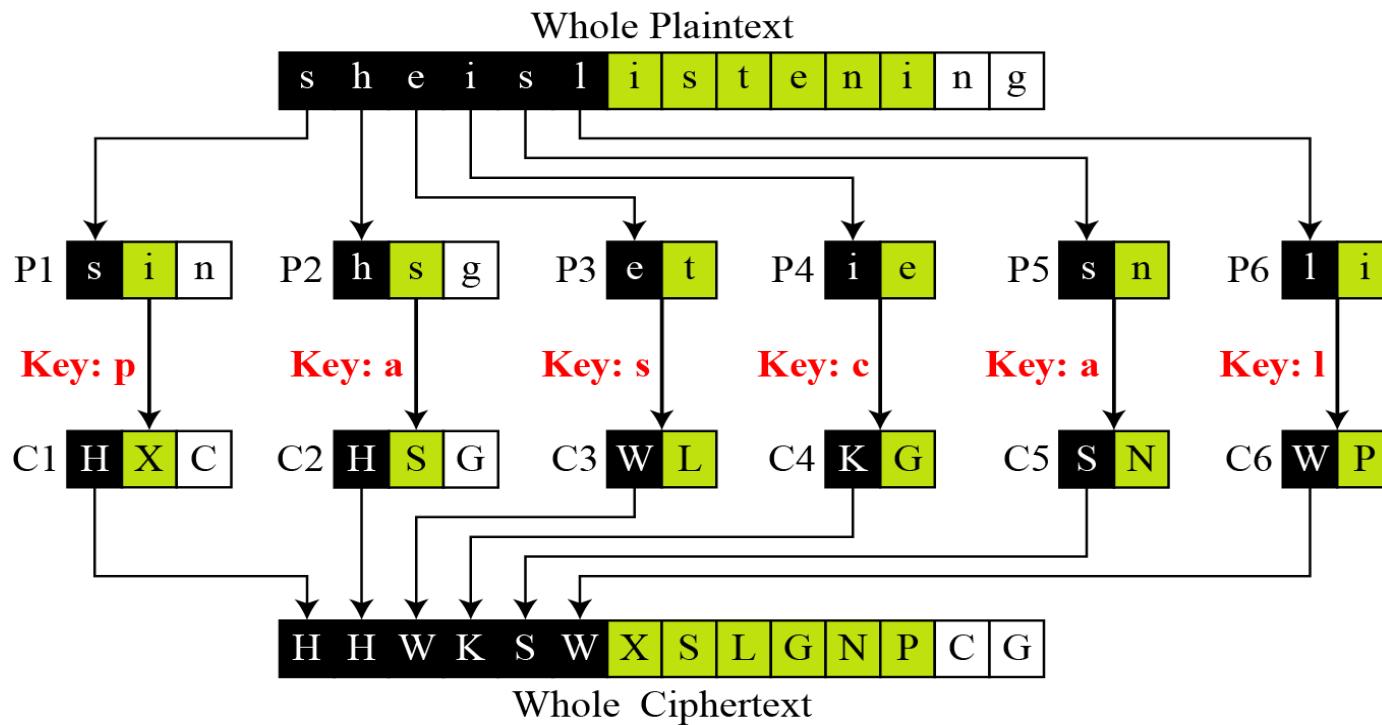
Plaintext:	s	h	e	i	s	l	i	s	t	e	n	i	n	g
P's values:	18	07	04	08	18	11	08	18	19	04	13	08	13	06
Key stream:	15	00	18	02	00	11	15	00	18	02	00	11	15	00
C's values:	07	07	22	10	18	22	23	18	11	6	13	19	02	06
Ciphertext:	H	H	W	K	S	W	X	S	L	G	N	T	C	G

Vigenere Cipher

Example

Vigenere cipher can be seen as combinations of m additive ciphers.

A Vigenere cipher as a combination of m additive ciphers



Vigenere Cipher Example

Using Example we can say that the additive cipher is a special case of Vigenere cipher in which $m = 1$.

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	v	v	w	x	y	z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z		
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z			
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z				
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z					
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z						
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z							
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z								
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z									
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z										
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z											
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z												
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z													
O	O	P	Q	R	S	T	U	V	W	X	Y	Z														
P	P	Q	R	S	T	U	V	W	X	Y	Z															
Q	Q	R	S	T	U	V	W	X	Y	Z																
R	R	S	T	U	V	W	X	Y	Z																	
S	S	T	U	V	W	X	Y	Z																		
T	T	U	V	W	X	Y	Z																			
U	U	V	W	X	Y	Z																				
V	V	W	X	Y	Z																					
W	W	X	Y	Z																						
X	X	Y	Z																							
Y	Y	Z																								
Z	Z																									

A Vigenere Tableau

Hill Cipher

- Hill Cipher developed by mathematician Lester Hill in 1929.
- The encryption algorithm takes m successive plaintext letters and substitutes for them m cipher text letters
- The substitution determined by m linear equation in which each character is assigned a numerical value ($a=0$, $b=1, \dots, Z=25$)

Hill Cipher Contd.

Key in the Hill cipher

$$K = \begin{bmatrix} k_{11} & k_{12} & \dots & k_{1m} \\ k_{21} & k_{22} & \dots & k_{2m} \\ \vdots & \vdots & & \vdots \\ k_{m1} & k_{m2} & \dots & k_{mm} \end{bmatrix}$$

$$C_1 = P_1 k_{11} + P_2 k_{21} + \dots + P_m k_{m1}$$

$$C_2 = P_1 k_{12} + P_2 k_{22} + \dots + P_m k_{m2}$$

...

$$C_m = P_1 k_{1m} + P_2 k_{2m} + \dots + P_m k_{mm}$$

The key matrix in the Hill cipher needs to have a multiplicative inverse.

Hill Cipher Contd.

1. Determinant of a matrix A , denoted by $\det A$:
 - if $A(a_{ij})$ is 2×2 , then $\det A = a_{11}a_{22} - a_{12}a_{21}$
 - if $A(a_{ij})$ is 3×3 , then $\det A = a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32} - a_{13}a_{22}a_{31} - a_{12}a_{21}a_{33} - a_{11}a_{23}a_{32}$

2. Theorem: suppose $K = \begin{pmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{pmatrix}$ with $k_{ij} \in \mathbb{Z}_{26}$

Then K has an inverse **if and only if** $\det K$ is invertible in \mathbb{Z}_{26}

if and only if $\gcd(\det K, 26) = 1$

Moreover,

$$K^{-1} = (\det K)^{-1} \begin{pmatrix} k_{22} & -k_{12} \\ -k_{21} & k_{11} \end{pmatrix} \text{ Where } \det K = k_{11}k_{22} - k_{12}k_{21}$$

Hill Cipher Contd.

- $C_1 = (K_{11}P_1 + K_{12}P_2 + K_{13}P_3) \text{ mod } 26$
- $C_2 = (K_{21}P_1 + K_{22}P_2 + K_{23}P_3) \text{ mod } 26$
- $C_3 = (K_{31}P_1 + K_{32}P_2 + K_{33}P_3) \text{ mod } 26$
- $C = E(K, P) = KP \text{ mod } 26$
- $P = D(K, C) = K^{-1}C \text{ mod } 26 = K^{-1}KP = P$

Hill Cipher (I)

A multiple-letter encryption method – encrypts m letters of plaintext at each step

- The encryption key K is a $m \times m$ matrix of coefficients
- To encrypt – multiply the matrix K by a vector of m plaintext letters to receive a vector of m ciphertext letters. (Arithmetic is modulo the size of the alphabet.)
- Example: $m = 3$

$$C_1 = K_{11} P_1 + K_{12} P_2 + K_{13} P_3$$

$$C_2 = K_{21} P_1 + K_{22} P_2 + K_{23} P_3$$

$$C_3 = K_{31} P_1 + K_{32} P_2 + K_{33} P_3$$

Hill Cipher (II)

The encryption key K is a $m \times m$ matrix of coefficients

- The decryption key K^{-1} is the $m \times m$ matrix of coefficients that is the inverse of matrix K :

$$K K^{-1} = I$$

- To decrypt – multiply the matrix K^{-1} by the vector of m ciphertext letters to receive the vector of m plaintext letters. (Arithmetic is modulo the size of the alphabet.)

TRANSPOSITION CIPHERS

A transposition cipher does not substitute one symbol for another, instead it changes the location of the symbols.

A transposition cipher reorders symbols.

Keyless Transposition Ciphers

Keyed Transposition Ciphers

Combining Two Approaches

TRANSPOSITION CIPHERS

Transposition techniques differ from substitution technique in the way that they do not simply replace one alphabet with another: they also performs some permutation over the plain text alphabets.

Rail Fence Technique

The rail fence technique is an example of transposition cipher. It uses a simple algorithm as

1. Write down the plain text message as a sequence of diagonals.
2. Read the plain text written in step1 as a sequence of rows.

TRANSPOSITION CIPHERS Contd.

Simple Columnar Transposition Technique

Variation of the basic transposition technique such as Rail Fence Technique exist. Such a scheme is can call as Simple Columnar Transposition Technique.

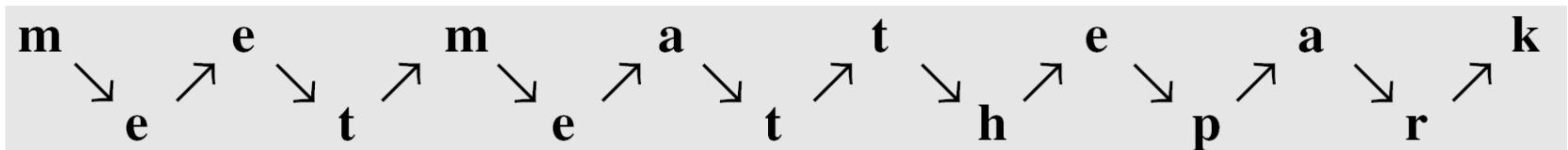
1. Write the plain text message row by row in a rectangle of predefined size.
2. Read the message column-by-column. However, it need not be in the order of Column 1, Column 2, It can be random order such as 2, 3 ect.
3. The message thus obtained is the cipher text message.

Keyless Transposition Ciphers

Simple transposition ciphers, which were used in the past, are keyless.

Example

A good example of a keyless cipher using the first method is the [rail fence cipher](#). The ciphertext is created reading the pattern row by row. For example, to send the message “**meet me at the park**” to Bob, Alice writes



She then creates the ciphertext “**MEMATEAKETETHPR**”.

Keyless Transposition Ciphers

Example

Alice and Bob can agree on the number of columns and use the second method. Alice writes the same plaintext, row by row, in a table of four columns.

m	e	e	t
m	e	a	t
t	h	e	p
a	r	k	

She then creates the ciphertext “MMTAEEHREAEKTTP”.

Example

The cipher in previous example is actually a transposition cipher. The following shows the permutation of each character in the plaintext into the ciphertext based on the positions.

01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
01	05	09	13	02	06	10	13	03	07	11	15	04	08	12

The second character in the plaintext has moved to the fifth position in the ciphertext; the third character has moved to the ninth position; and so on. Although the characters are permuted, there is a pattern in the permutation: (01, 05, 09, 13), (02, 06, 10, 13), (03, 07, 11, 15), and (08, 12). In each section, the difference between the two adjacent numbers is 4.

Keyed Transposition Ciphers

The keyless ciphers permute the characters by using writing plaintext in one way and reading it in another way.

The permutation is done on the whole plaintext to create the whole ciphertext.

Another method is to divide the plaintext into groups of predetermined size, called blocks, and then use a key to permute the characters in each block separately.

Keyed Transposition Ciphers

Alice needs to send the message “*enemy attacks tonight*” to Bob.

e n e m y a t t a c k s o n i g h t z

The key used for encryption and decryption is a permutation key, which shows how the characters are permuted.

Encryption ↓

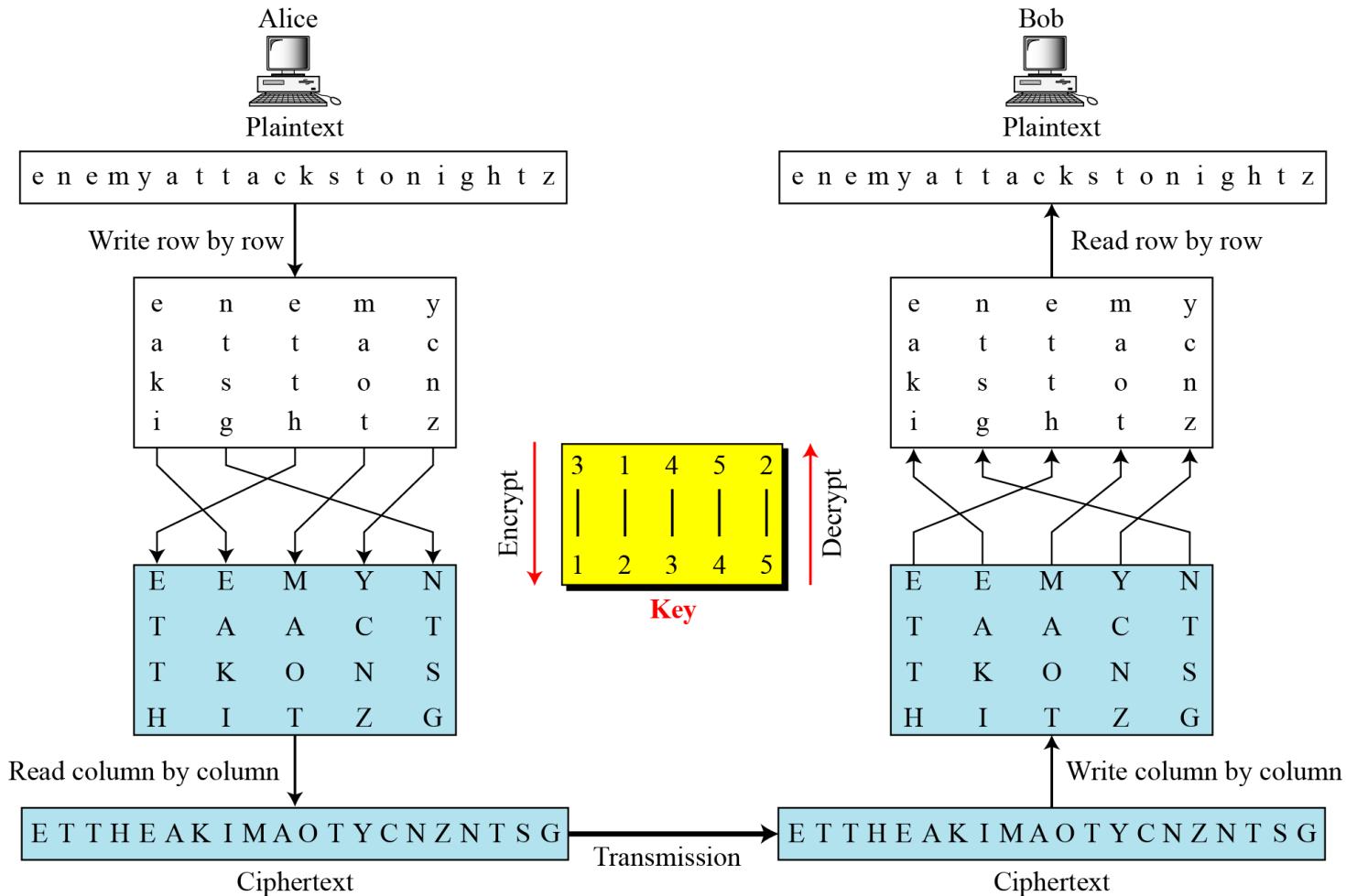
3	1	4	5	2
1	2	3	4	5

↑ Decryption

The permutation yields

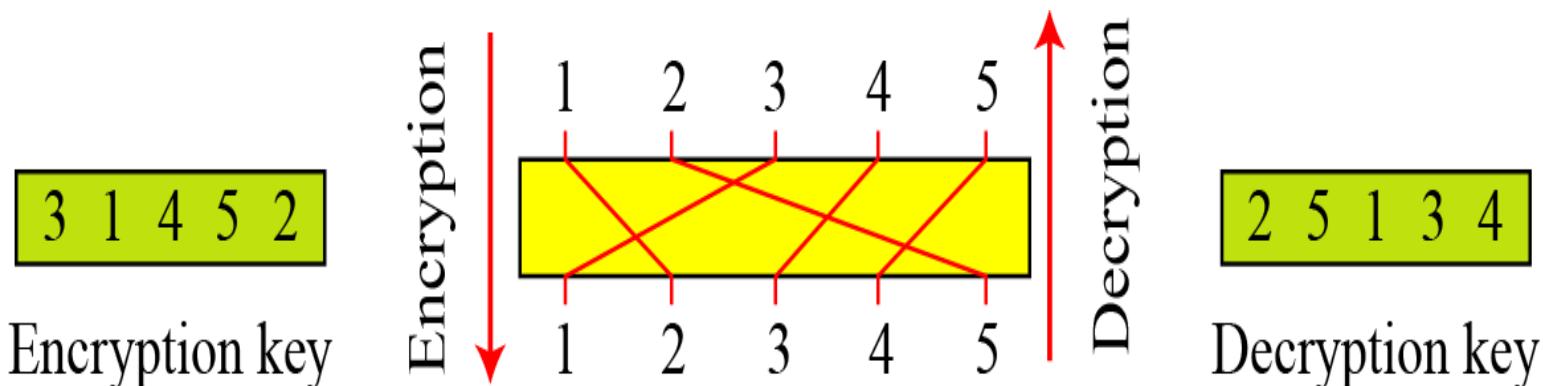
E E M Y N T A A C T T K O N S H I T Z G

Combining Two Approaches

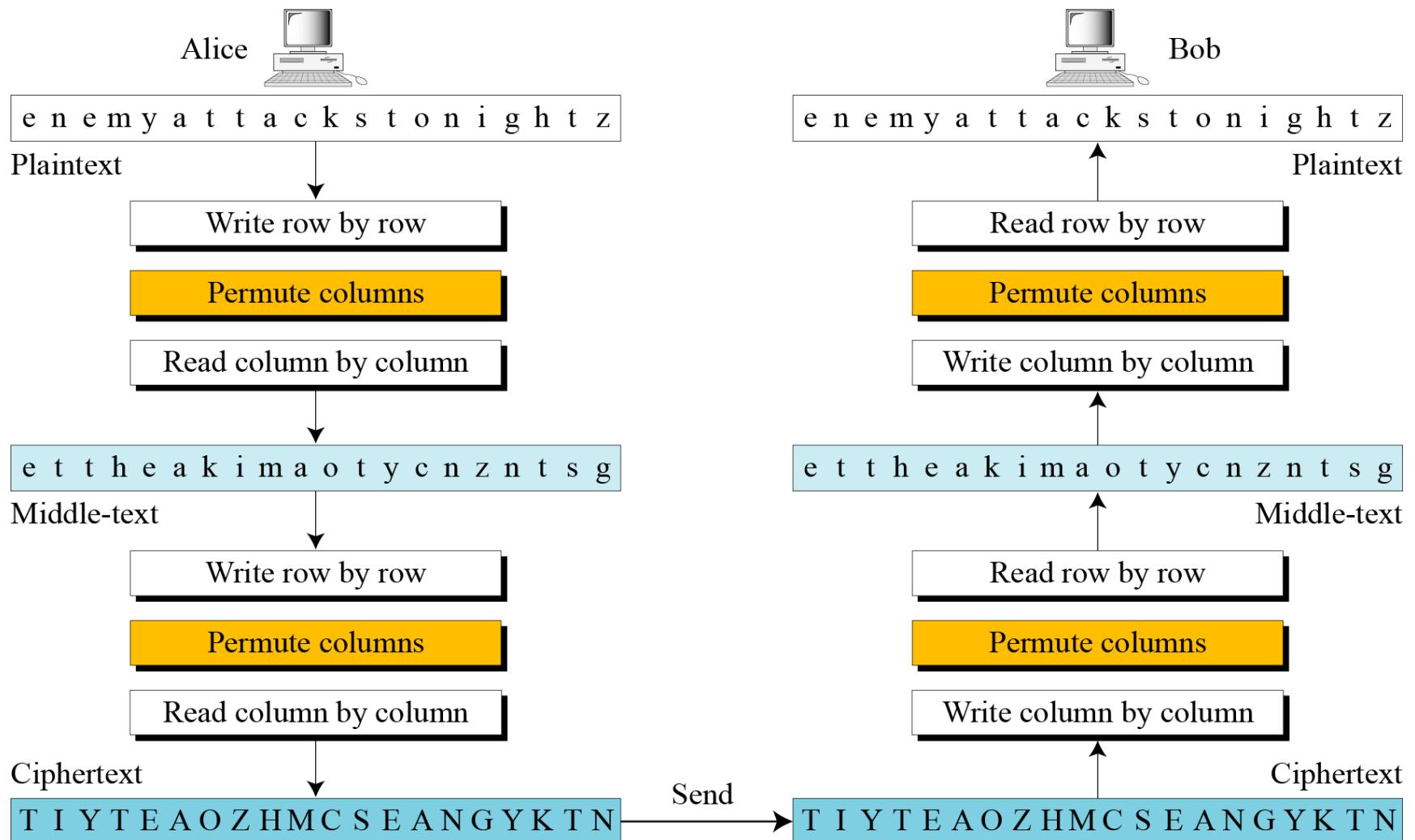


In previous example, a single key was used in two directions for the column exchange: downward for encryption, upward for decryption. It is customary to create two keys.

Encryption/Decryption keys in Transpositional Ciphers



Double Transposition Ciphers



There are Two Requirements for Secure use of Conventional Encryption

1. We need a strong encryption algorithm. At a minimum, we would like the algorithm to be such that an opponent who knows the algorithm and has access to one or more ciphertexts would be unable to decipher the ciphertext or figure out the key. This requirement is usually stated in a stronger form: The opponent should be unable to decrypt ciphertext or discover the key even if he or she is in possession of a number of ciphertexts together with the plaintext that produced each ciphertext.

2. Sender and receiver must have obtained copies of the secret key in a secure fashion and must keep the key secure. If someone can discover the key and knows the algorithm, all communication using this key is readable.

Cryptanalysis and Brute-Force Attack

Objective of attacking an encryption system is to recover the key in use rather than simply to recover the plaintext of a single ciphertext. There are two general approaches to attacking a conventional encryption scheme:

Cryptanalysis: Cryptanalytic attacks rely on the nature of the algorithm plus perhaps some knowledge of the general characteristics of the plaintext or even some sample plaintext–ciphertext pairs. This type of attack exploits the characteristics of the algorithm to attempt to deduce a specific plaintext or to deduce the key being used.

Brute-force attack: The attacker tries every possible key on a piece of ciphertext until an intelligible translation into plaintext is obtained. On average, half of all possible keys must be tried to achieve success.

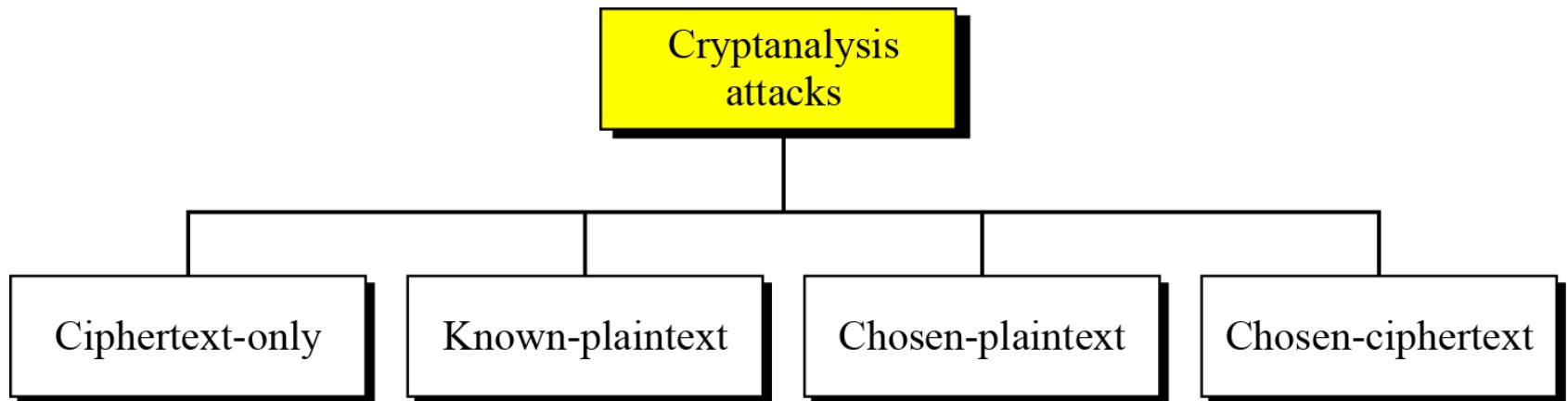
Attack Methods

1. **Ciphertext only:** Alice has only a copy of ciphertext
2. **Known Plaintext:** Eve has a copy of ciphertext and the corresponding plaintext and tries to deduce the key.
3. **Chosen Plaintext:** Eve has a copy of ciphertext corresponding to a copy of plaintext selected by Alice who believes it is useful to deduce the key.
4. **Chosen Ciphertext:** Eve has a copy of plaintext corresponding to a copy of ciphertext selected by Alice who believes it is useful to deduce the key.

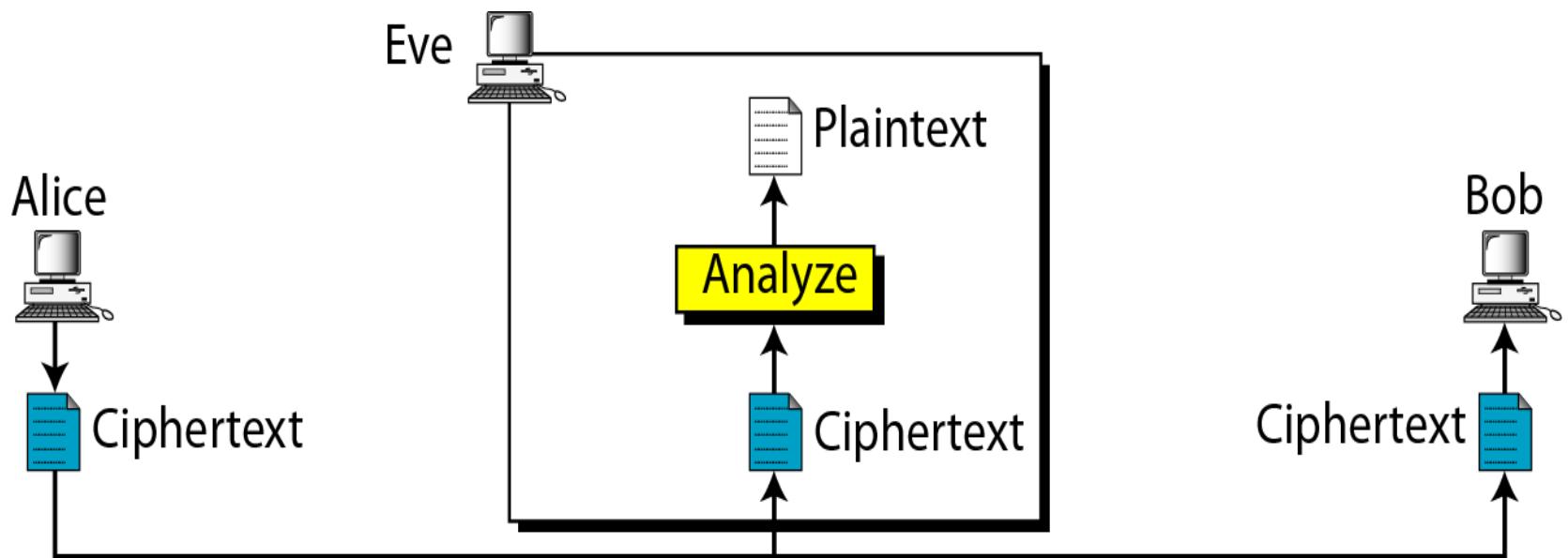
Cryptanalysis

As cryptography is the science and art of creating secret codes, **cryptanalysis** is the science and art of breaking those codes.

Cryptanalysis attacks

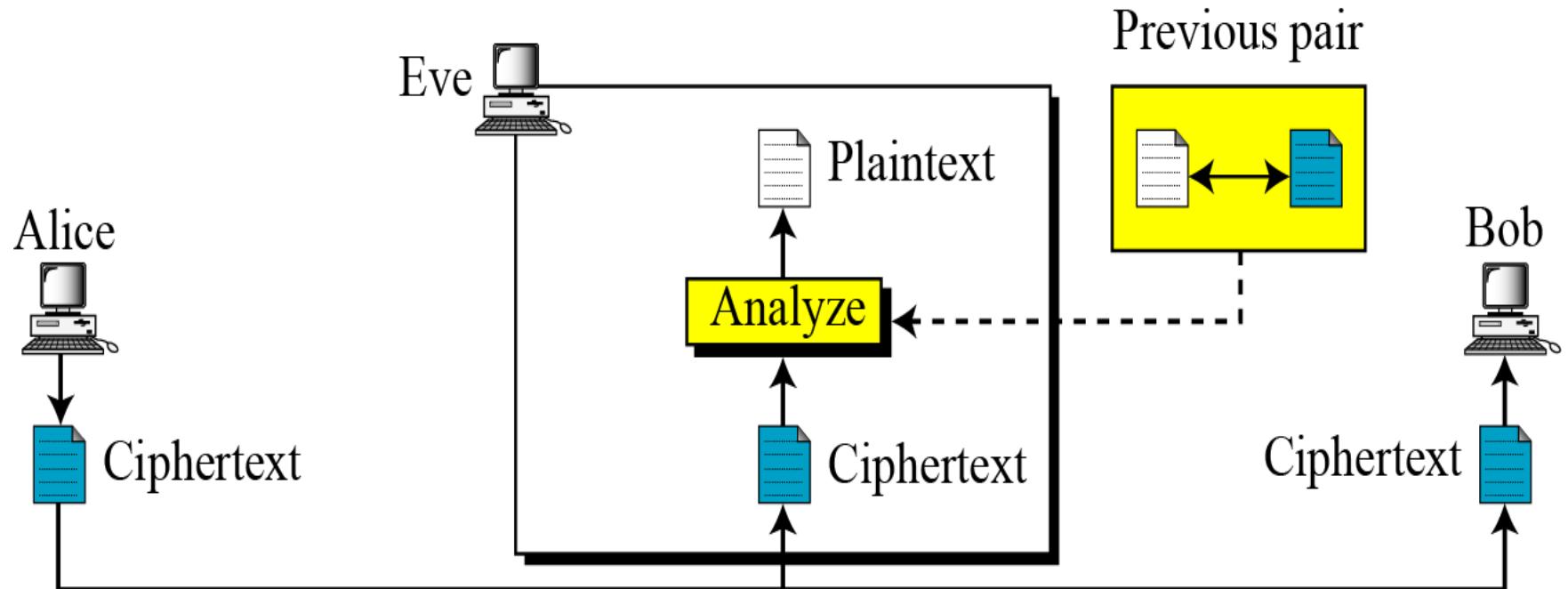


Ciphertext-Only Attack



Note: Refer/Read Text Book to get the description

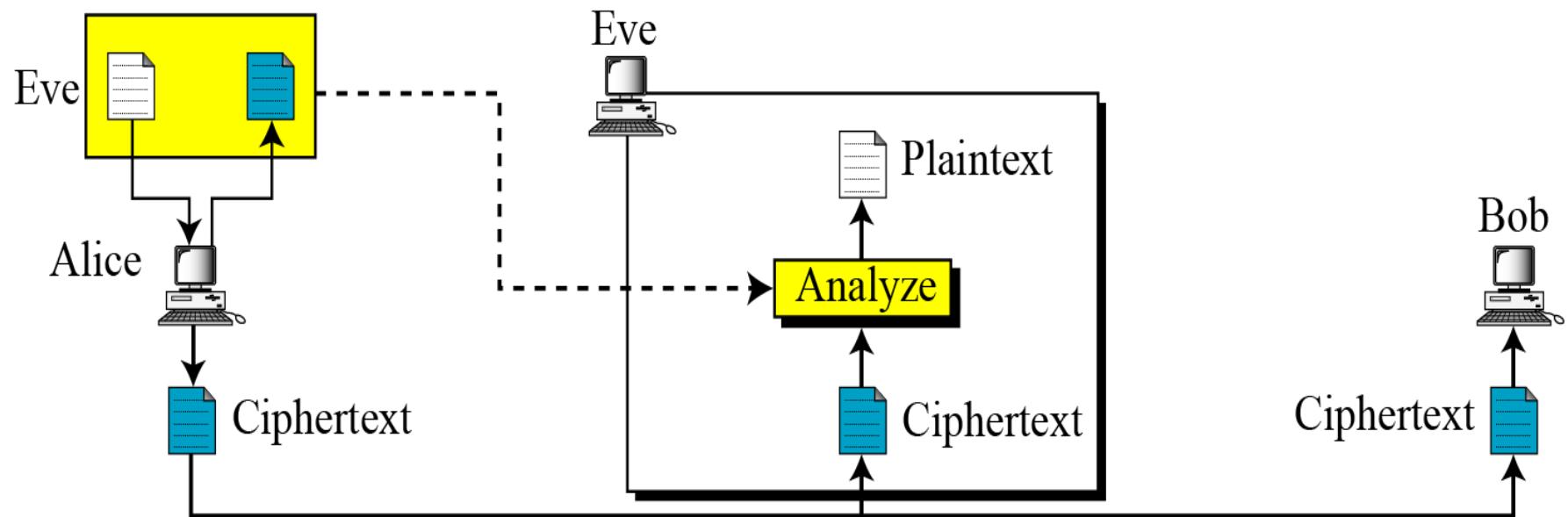
Known-Plaintext Attack



Note: Refer/Read Text Book to get the description

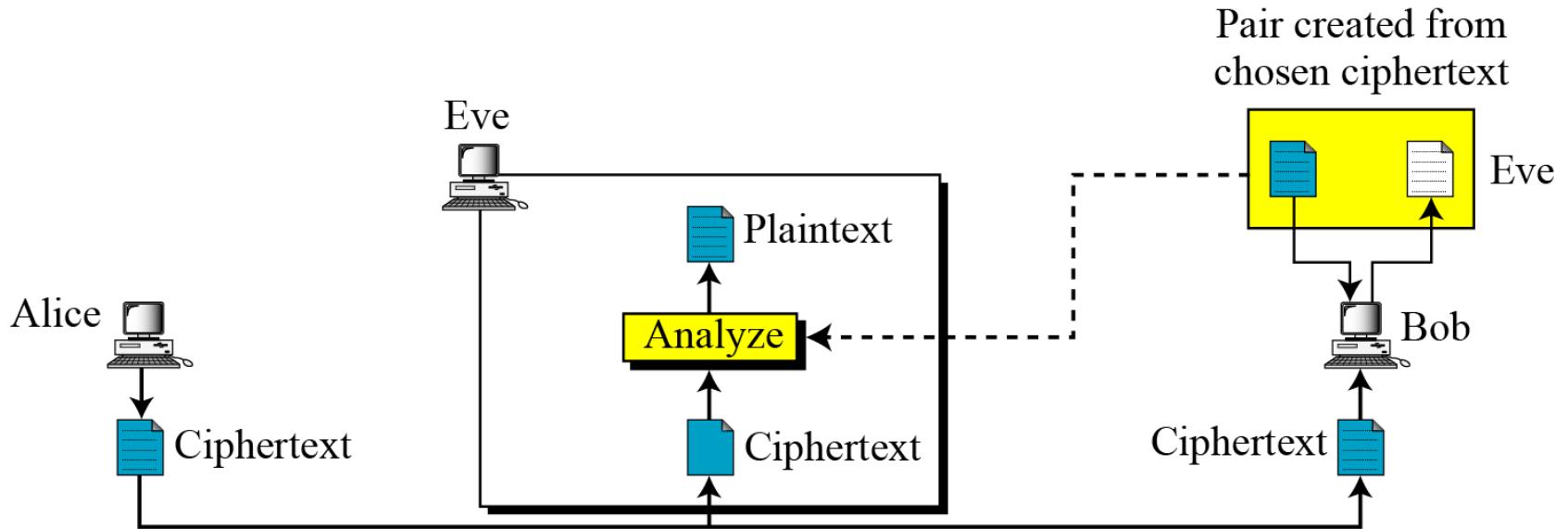
Chosen-Plaintext Attack

Pair created from chosen plaintext



Note: Refer/Read Text Book to get the description

Chosen-Ciphertext Attack



Note: Refer/Read Text Book to get the description

DES History

- In 1973, NBS (NIST) issues a public request for proposals for a national cipher standard, which must be
 - Secure
 - Public
 - Completely specified
 - Easy to understand
 - Available to all users
 - Economic and efficient in hardware
 - Able to be validated
 - Exportable
- IBM submitted LUCIFER (Feistel) (which was redesigned to become the DES)
- In 1977, adopted by NBS (NIST) as DES (Data Encryption Standard, Federal Information Processing Standard 46 (FIPS PUB 46))

DES History

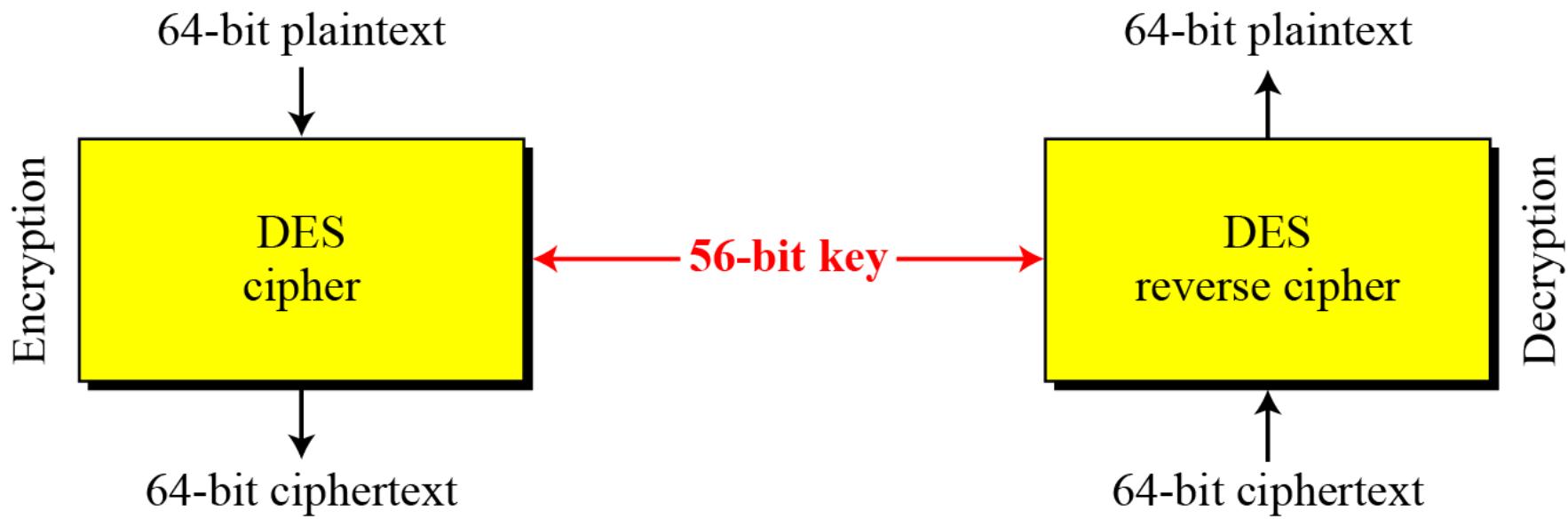
- Chronology
 - 1973: NBS publishes a first request for a standard encryption algorithm
 - 1974: NBS publishes a second request for encryption algorithms
 - 1975: DES is published in the Federal Register for comment
 - 1976: First and second workshop on DES
 - 1976: DES is approved as a standard
 - 1977: DES is published as a FIPS standard FIPS PUB 46
 - 1983: DES reaffirmed for the first time
 - 1986: Videocipher II, a TV satellite scrambling system based upon DES begins use by HBO
 - 1988: DES is reaffirmed for the second time as FIPS 46-1, superseding FIPS PUB 46
 - 1992: Biham and Shamir publish the first theoretical attack with less complexity than brute force: **differential cryptanalysis**. However, it requires an unrealistic 2^{47} chosen plaintexts
 - 1993: DES is reaffirmed for the third time as FIPS 46-2

DES History

- 1994: The first experimental cryptanalysis of DES is performed using linear cryptanalysis (Matsui, 1994)
- 1997: The DESCHALL Project breaks a message encrypted with DES for the first time in public
- 1998: The Electronic Frontier Foundation (EFF)'s DES cracker (Deep Crack) breaks a DES key in 56 hours
- 1999: Together, Deep Crack and distributed.net break a DES key in 22 hours and 15 minutes
- 1999: DES is reaffirmed for the fourth time as FIPS 46-3, which specifies the preferred use of Triple DES, with single DES permitted only in legacy systems
- 2001: The Advanced Encryption Standard is published in FIPS 197
- 2002: The AES standard becomes effective
- 2004: The withdrawal of FIPS 46-3 (and a couple of related standards) is proposed in the Federal Register
- 2005: NIST withdraws FIPS 46-3

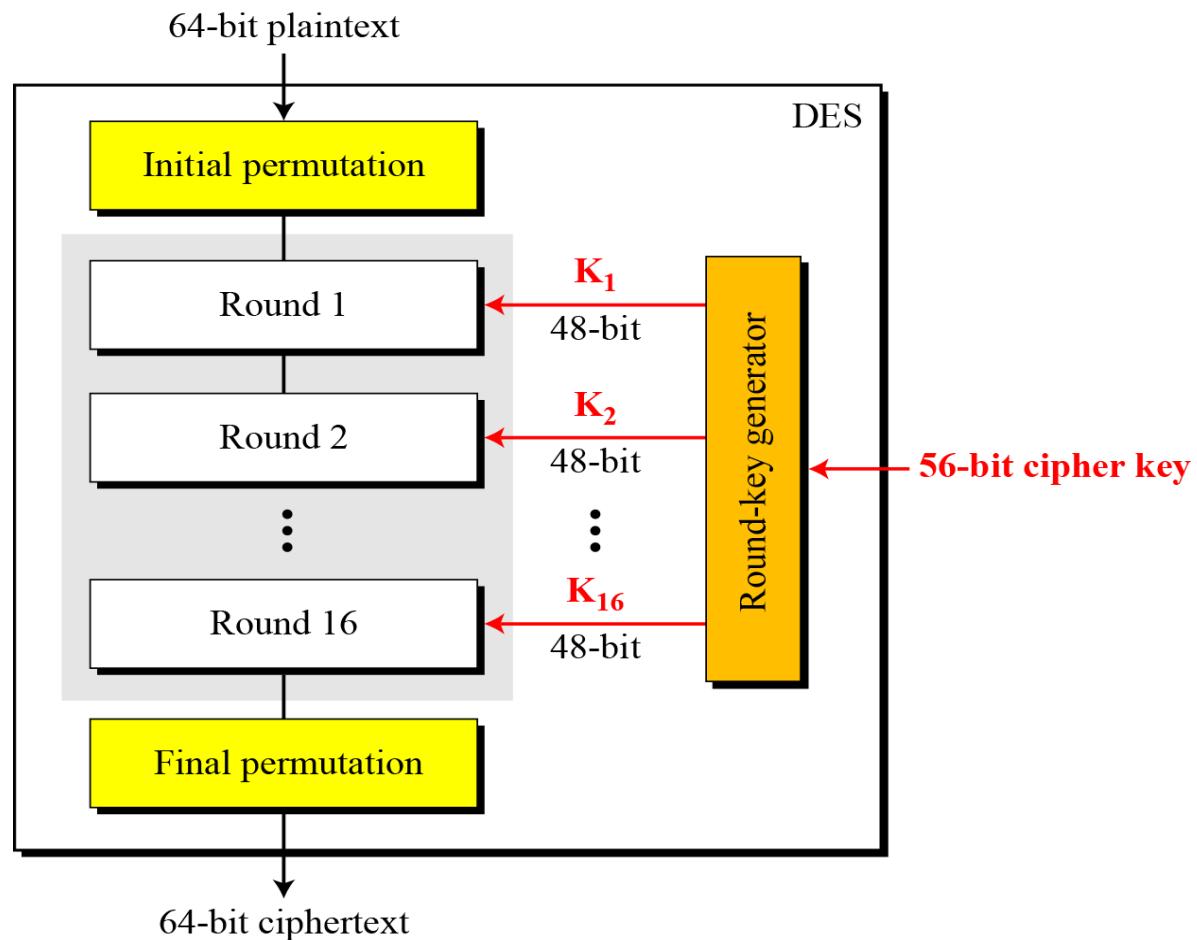
Encryption and Decryption with DES

DES is a block cipher, as shown in Figure

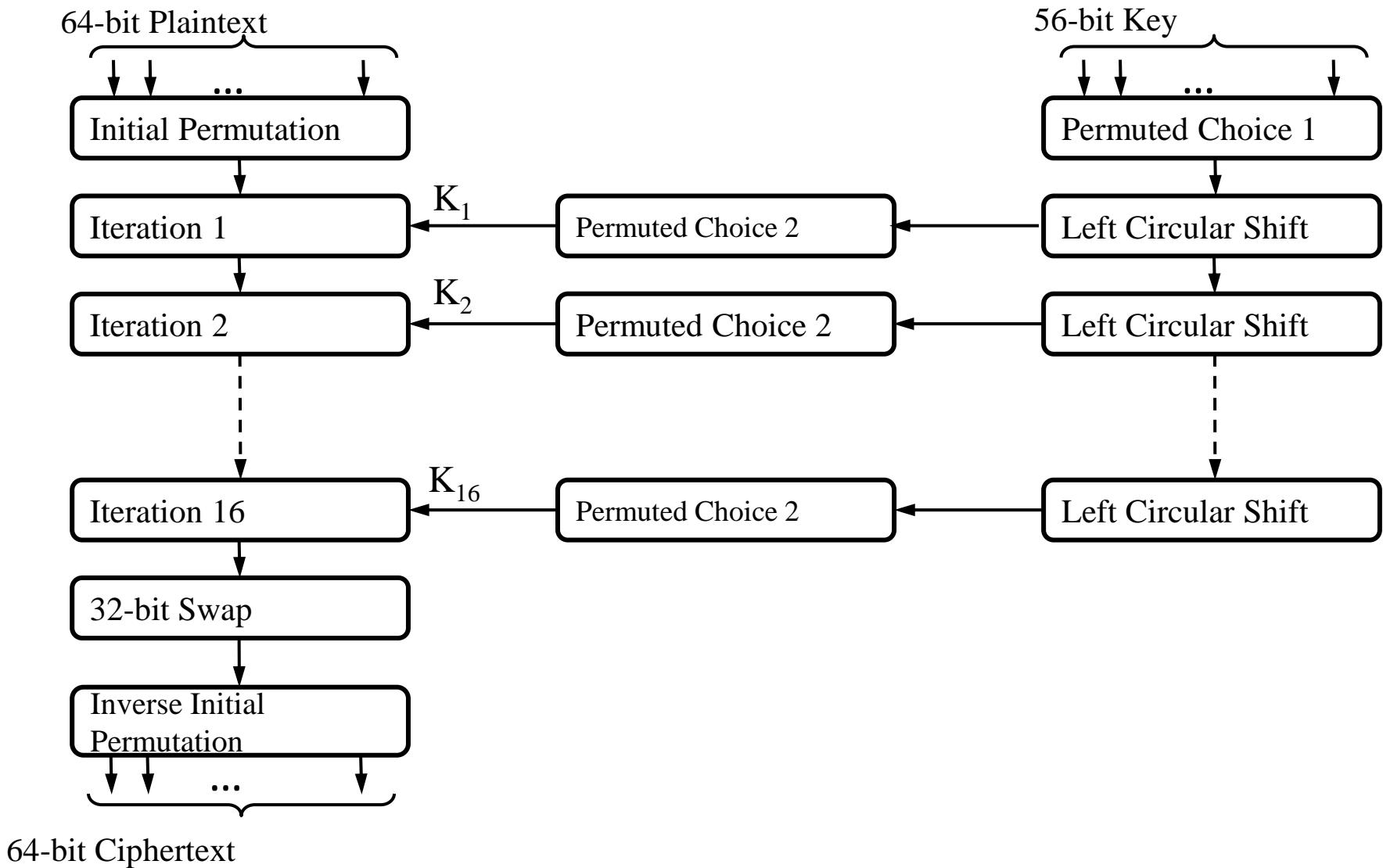


General Structure of DES

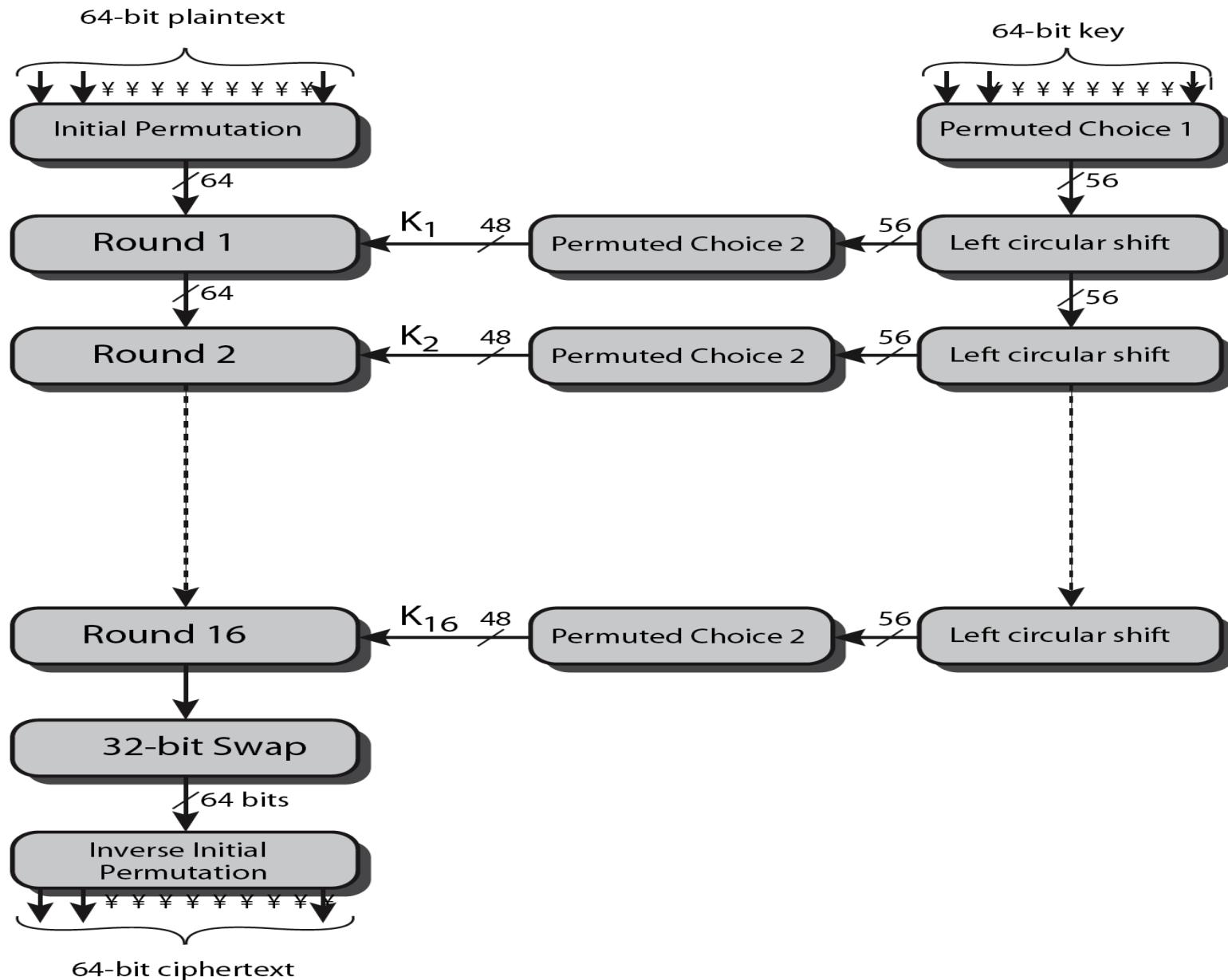
The encryption process is made of two permutations (P-boxes), which we call initial and final permutations, and sixteen Feistel rounds.



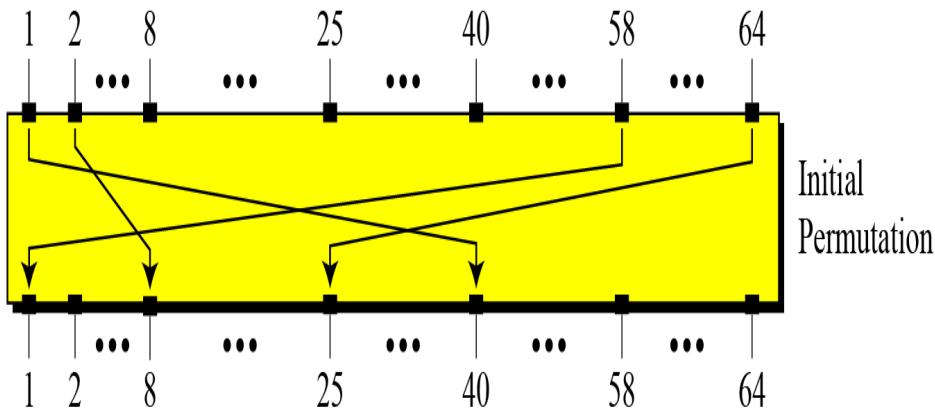
DES Encryption Overview



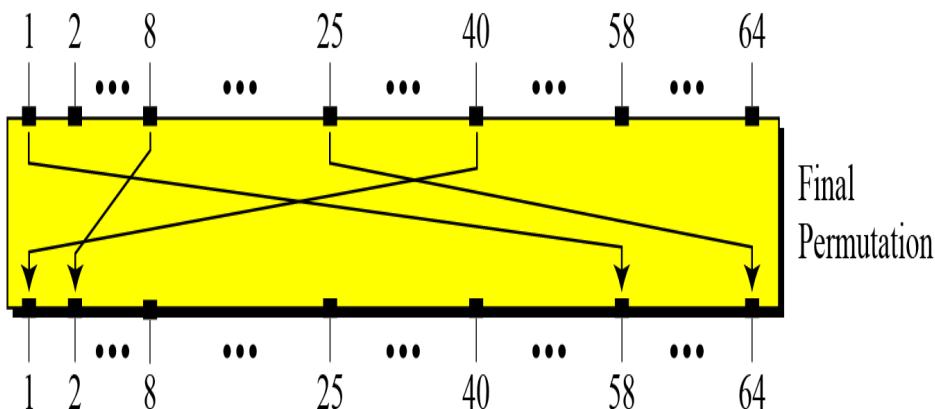
DES Encryption Overview



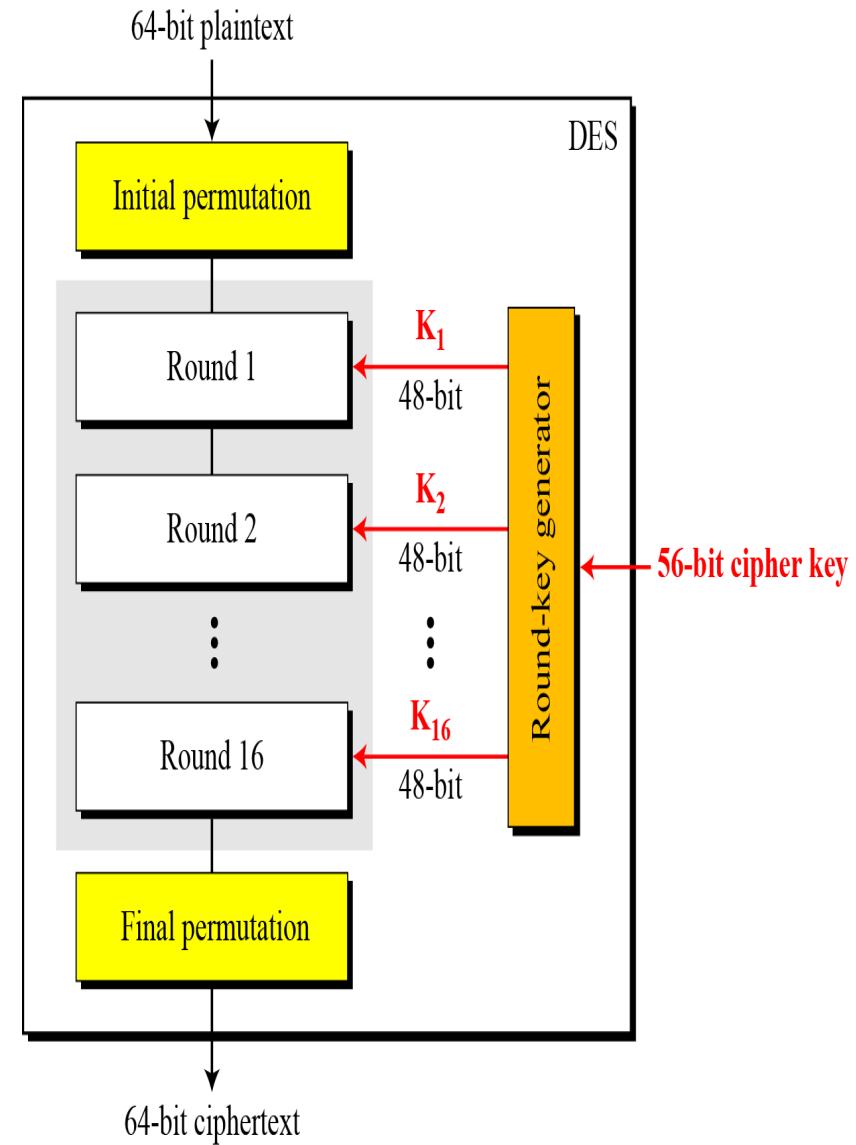
Initial and Final Permutation Steps in DES



Initial
Permutation



Final
Permutation



Initial and Final Permutation tables

<i>Initial Permutation</i>	<i>Final Permutation</i>
58 50 42 34 26 18 10 02	40 08 48 16 56 24 64 32
60 52 44 36 28 20 12 04	39 07 47 15 55 23 63 31
62 54 46 38 30 22 14 06	38 06 46 14 54 22 62 30
64 56 48 40 32 24 16 08	37 05 45 13 53 21 61 29
57 49 41 33 25 17 09 01	36 04 44 12 52 20 60 28
59 51 43 35 27 19 11 03	35 03 43 11 51 19 59 27
61 53 45 37 29 21 13 05	34 02 42 10 50 18 58 26
63 55 47 39 31 23 15 07	33 01 41 09 49 17 57 25

Problem No. 1

Find the output of the initial permutation box when the input is given in hexadecimal as:

0x0002 0000 0000 0001

Input has only two 1s (Bit 15 and bit 64): the output must also have only two 1s(the nature straight permutation).

The bit 15 in the input becomes bit 63 in the output. Bit 64 in the input becomes bit 25 in the output. So the output has only two 1s, bit 25 and bit 63.

0x0000 0080 0000 0002

Problem No. 2

<i>Initial Permutation</i>	<i>Final Permutation</i>
58 50 42 34 26 18 10 02	40 08 48 16 56 24 64 32
60 52 44 36 28 20 12 04	39 07 47 15 55 23 63 31
62 54 46 38 30 22 14 06	38 06 46 14 54 22 62 30
64 56 48 40 32 24 16 08	37 05 45 13 53 21 61 29
57 49 41 33 25 17 09 01	36 04 44 12 52 20 60 28
59 51 43 35 27 19 11 03	35 03 43 11 51 19 59 27
61 53 45 37 29 21 13 05	34 02 42 10 50 18 58 26
63 55 47 39 31 23 15 07	33 01 41 09 49 17 57 25

Problem No. 2

Prove that the initial and final permutations are the inverse of each other by finding the output of the final permutation if the input is

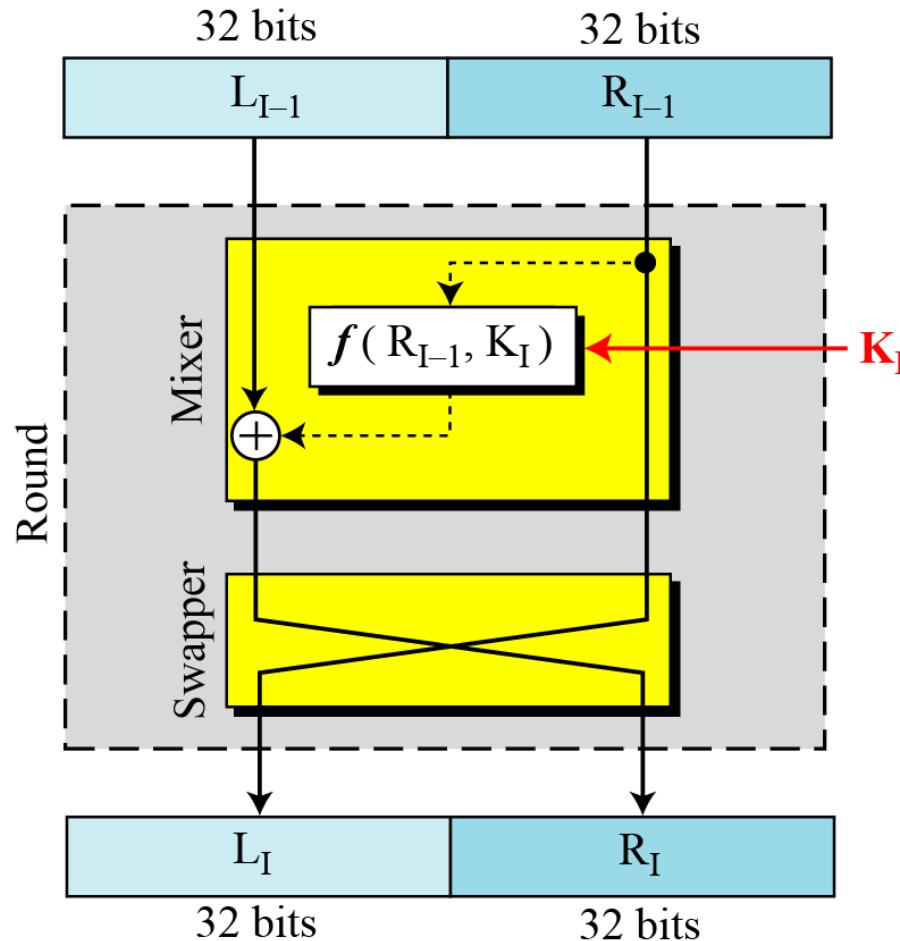
0x0000 0080 0000 0002

Solution: The bit 25 in the input becomes bit 64 in the output. Bit 63 in the input becomes bit 15 in the output. So the output has only two 1s, bit 15 and bit 64.

0x0002 0000 0000 0001

A round in DES (encryption site)

DES uses 16 rounds. Each round of DES is a Feistel cipher.



DES Contd.

- The computation consists of 16 iterations of a calculation
- The cipher function f operates on two blocks, one of 32 bits and one of 48 bits, and produces a block of 32 bits.
- The input block is then **LR**, 32 bit block **L** followed by a 32 bit block **R**.
- Let **K** be a block of 48 bits chosen from the 64-bit key. Then the output **L'R'** of an iteration with input **LR** is defined by:

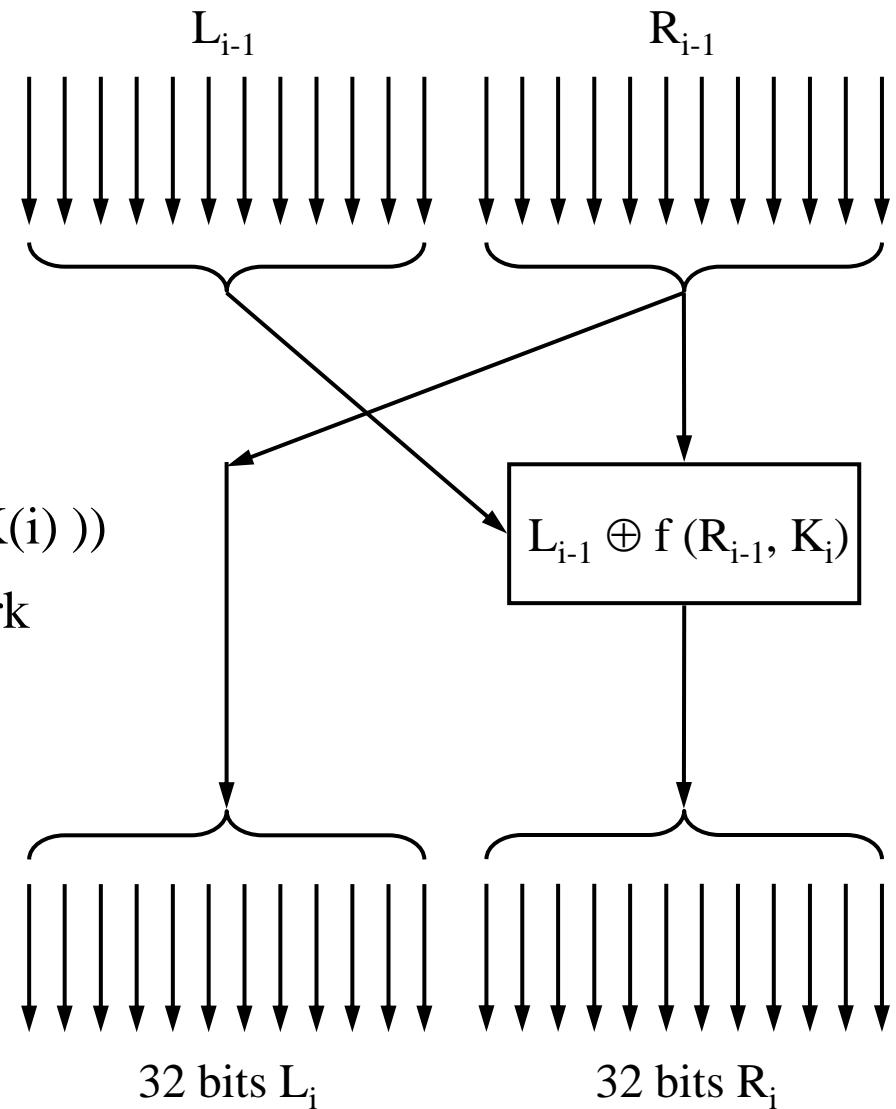
$$L' = R$$

$$R' = L (+) f(R, K)$$

- $L'R'$ is the output of the 16th iteration then $R'L'$ is the preoutput block.
- At each iteration a different block **K** of key bits is chosen from the 64-bit key designated by **KEY**.
- Let **KS** be a function which takes an integer n in the range from 1 to 16 and a 64-bit block **KEY** as input and yields as output a 48-bit block **K_n** which is a permuted selection of bits from **KEY**. That is

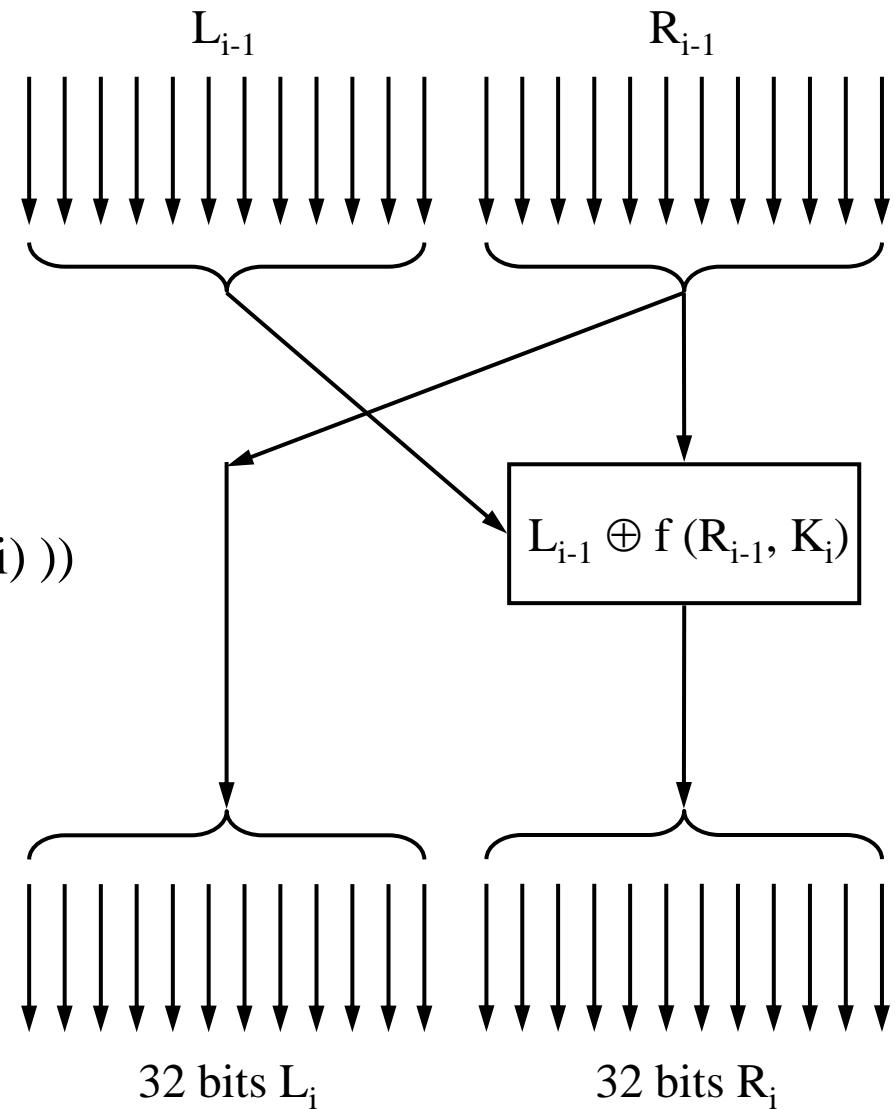
$$K_n = KS(n, KEY)$$

DES - Swapping of Left and Right Halves



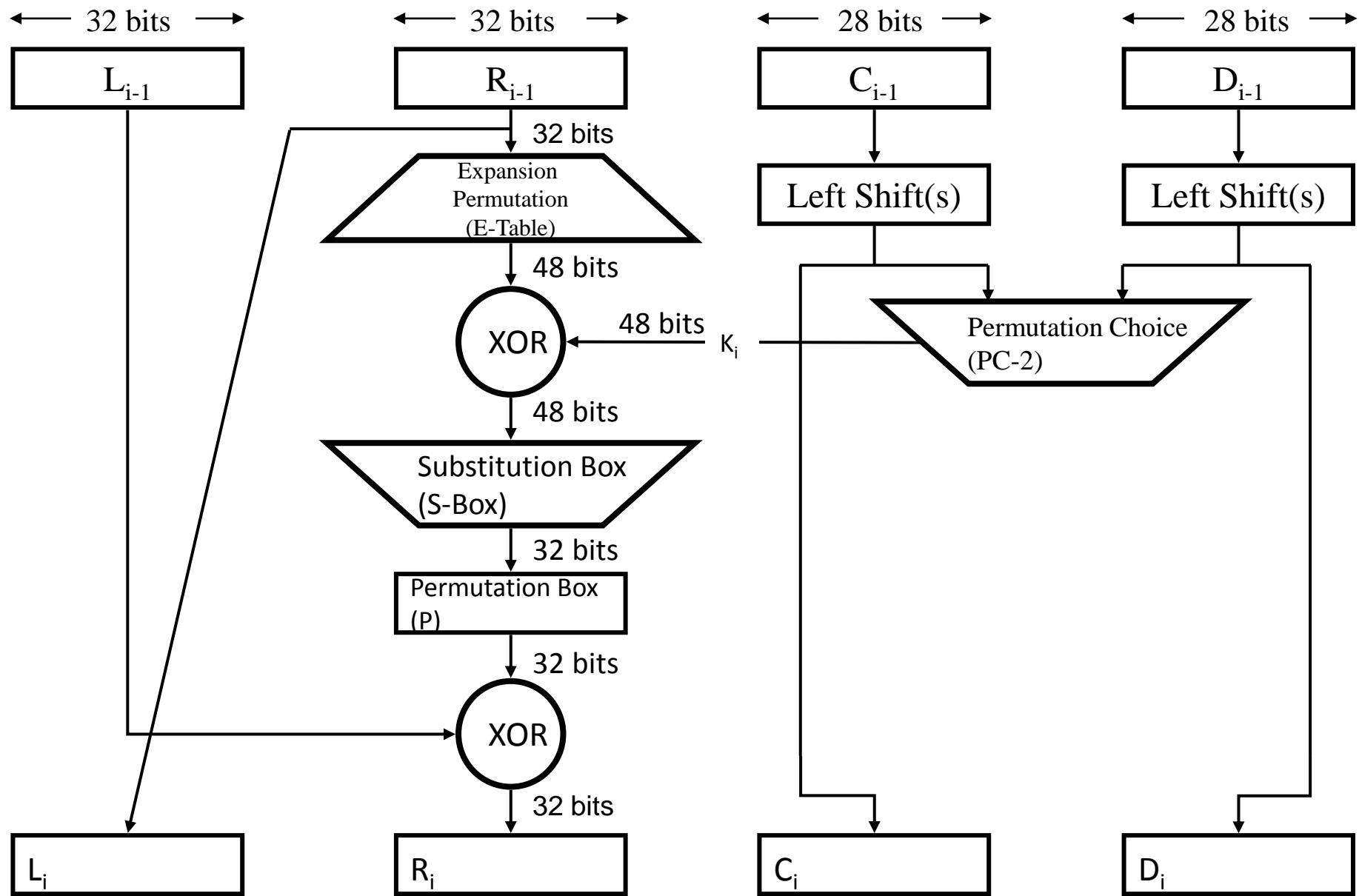
- This can be described functionally as:
 - $L(i) = R(i-1)$
 - $R(i) = L(i-1) \oplus P(S(E(R(i-1)) \oplus K(i)))$
- This forms one round in an S-P network

DES - Swapping of Left and Right Halves



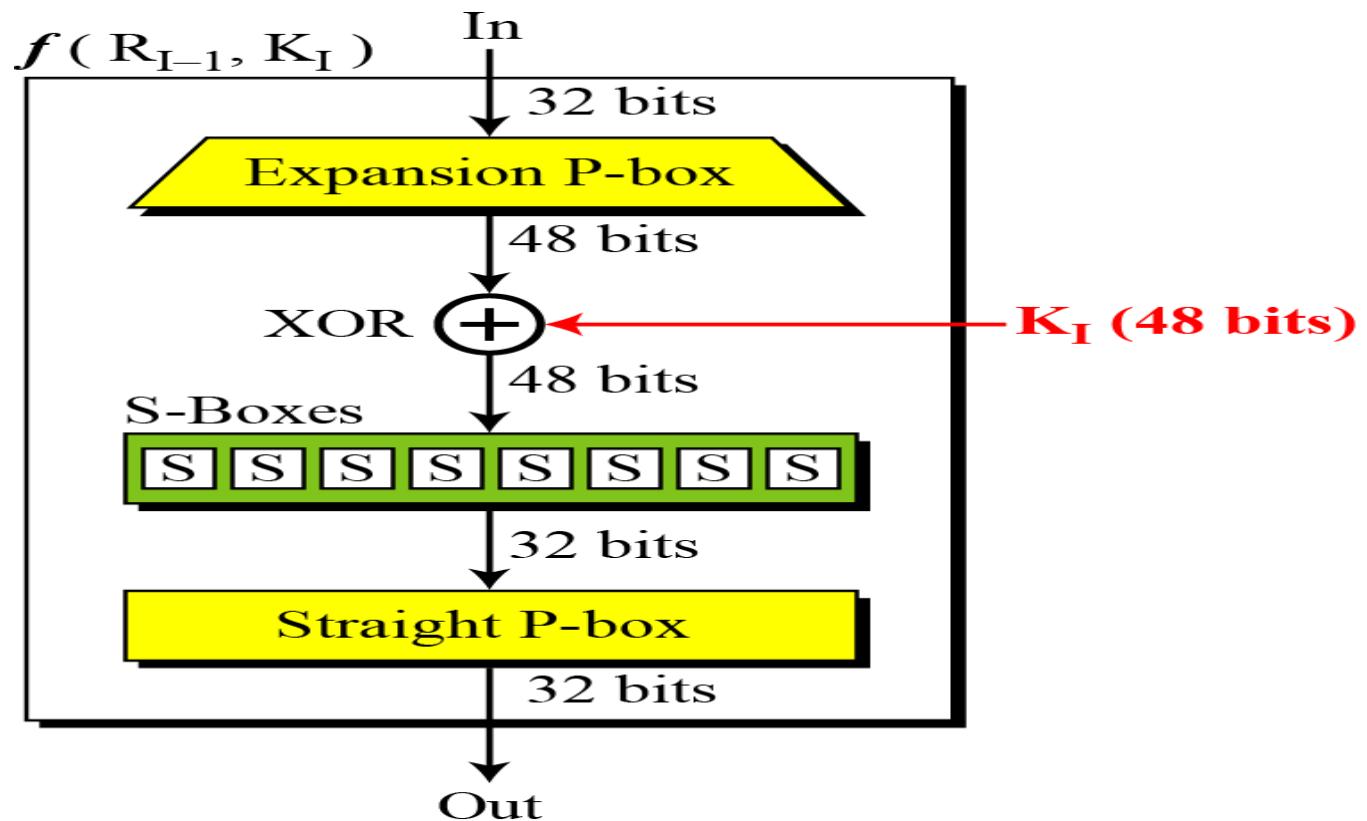
- This can be described functionally as:
 - $L(i) = R(i-1)$
 - $R(i) = L(i-1) \oplus P(S(E(R(i-1)) \oplus K(i)))$
- This forms one round in an S-P network

Details of Each Iteration



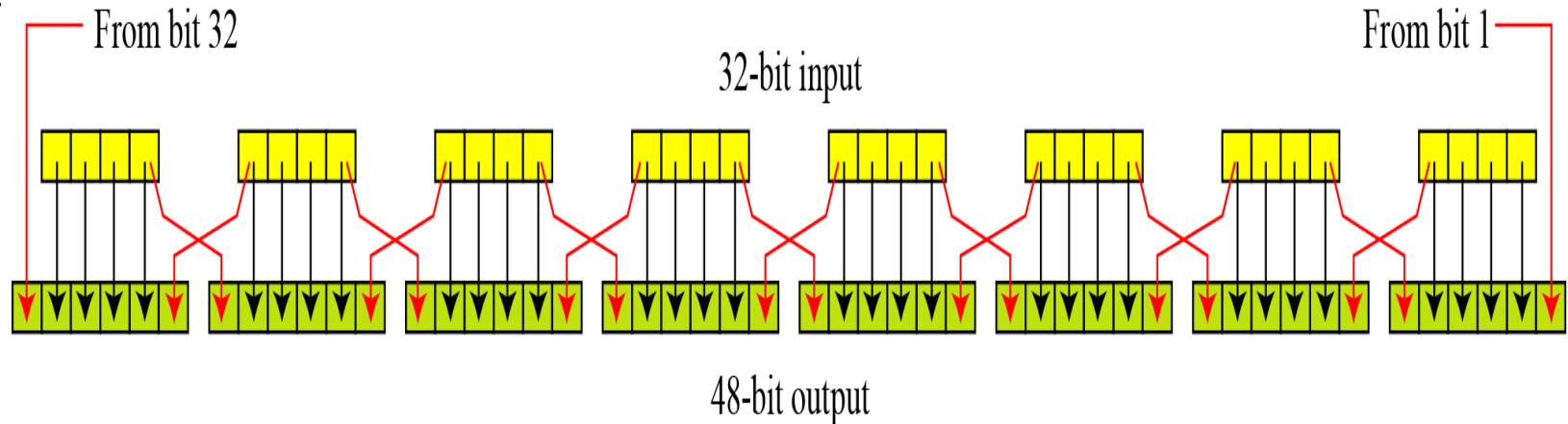
DES Function

The heart of DES is the DES function. The DES function applies a 48-bit key to the rightmost 32 bits to produce a 32-bit output.



Expansion P-box

Since R_{I-1} is a 32-bit input and K_I is a 48-bit key, we first need to expand R_{I-1} to 48 bits.



32	01	02	03	04	05
04	05	06	07	08	09
08	09	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	31	31	32	01

Expansion Permutation Table

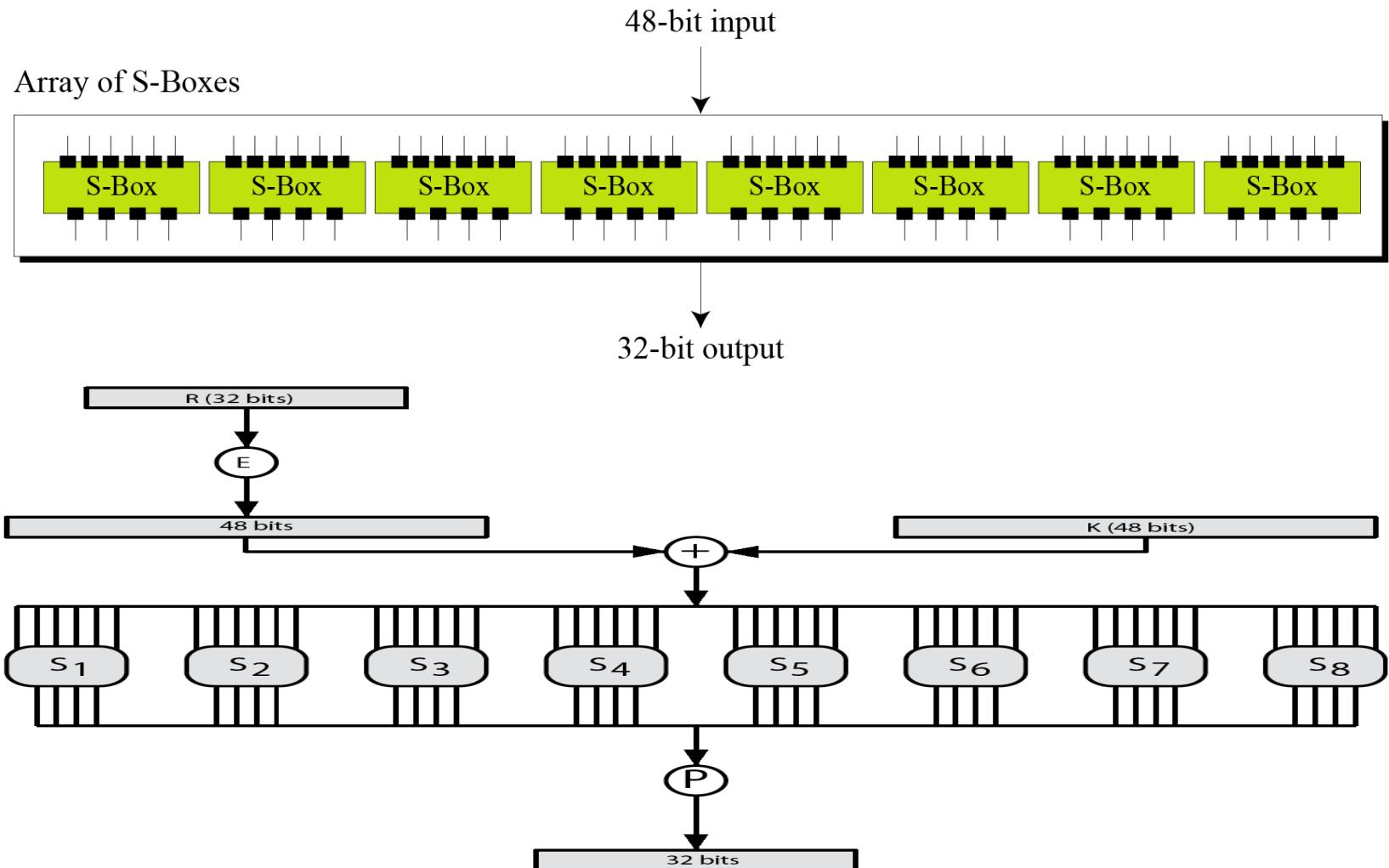
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

Whitener (XOR)

After the expansion permutation, DES uses the XOR operation on the expanded right section and the round key. Note that both the right section and the key are 48-bits in length. Also note that the round key is used only in this operation.

S-Boxes

The S-boxes do the real mixing (confusion). DES uses 8 S-boxes, each with a 6-bit input and a 4-bit output.



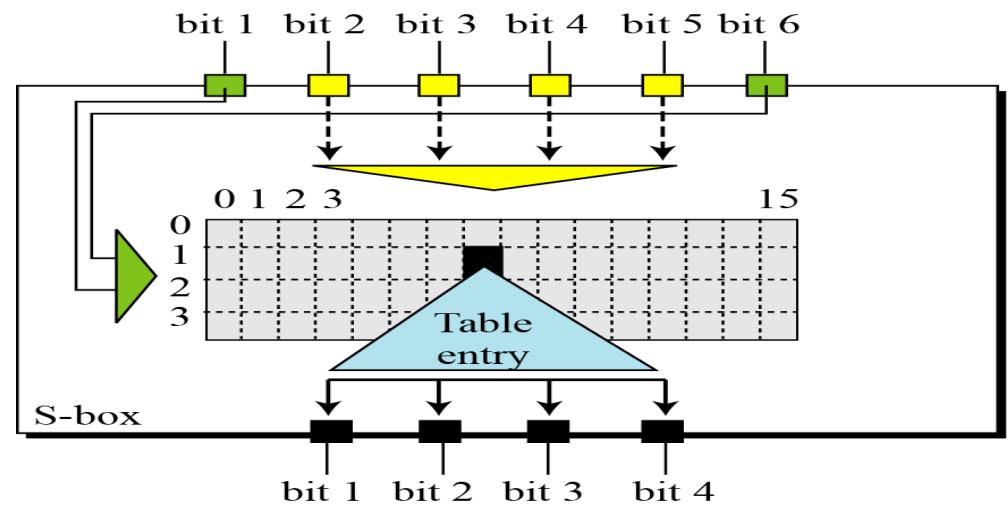
S-Box 1

Table shows the permutation for S-box 1. For the rest of the boxes see the textbook.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	04	13	01	02	15	11	08	03	10	06	12	05	09	00	07
1	00	15	07	04	14	02	13	10	03	06	12	11	09	05	03	08
2	04	01	14	08	13	06	02	11	15	12	09	07	03	10	05	00
3	15	12	08	02	04	09	01	07	05	11	03	14	10	00	06	13

Problem No. 3

The input to S-box 1 is 100011.
What is the output?



S-Box Structure

S1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
00(1)	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
01(2)	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
10(3)	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
11(4)	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

For example, for input 011011 the row is 01, that is row 1, and the column is determined by 1101, that is column 13. In row 1 column 13 appears 5 so that the output is 0101.

S-Box Structure

S_1	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S_2	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S-Box Structure

S_5	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S_6	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S_7	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S_8	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

The input to S-box 8 is 000000. What is the output?

Solution

If we write the first and the sixth bits together, we get 00 in binary, which is 0 in decimal. The remaining bits are 0000 in binary, which is 0 in decimal. We look for the value in row 0, column 0, in Table 6.10 (S-box 8). The result is 13 in decimal, which is 1101 in binary. So the input 000000 yields the output 1101.

Straight Permutation

16	07	20	21	29	12	28	17
01	15	23	26	05	18	31	10
02	08	24	14	32	27	03	09
19	13	30	06	22	11	04	25

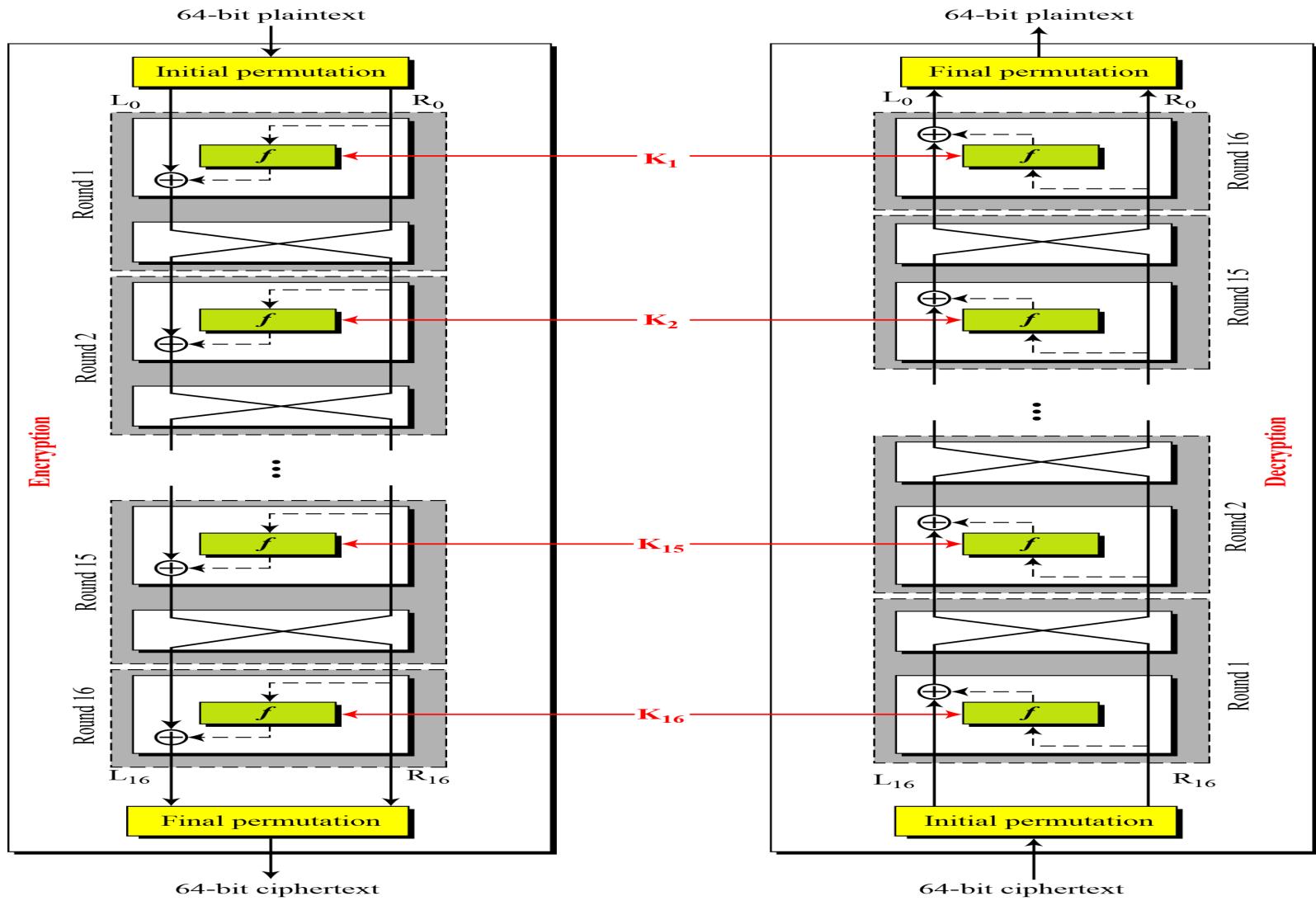
Cipher and Reverse Cipher

Using mixers and swappers, we can create the cipher and reverse cipher, each having 16 rounds.

First Approach

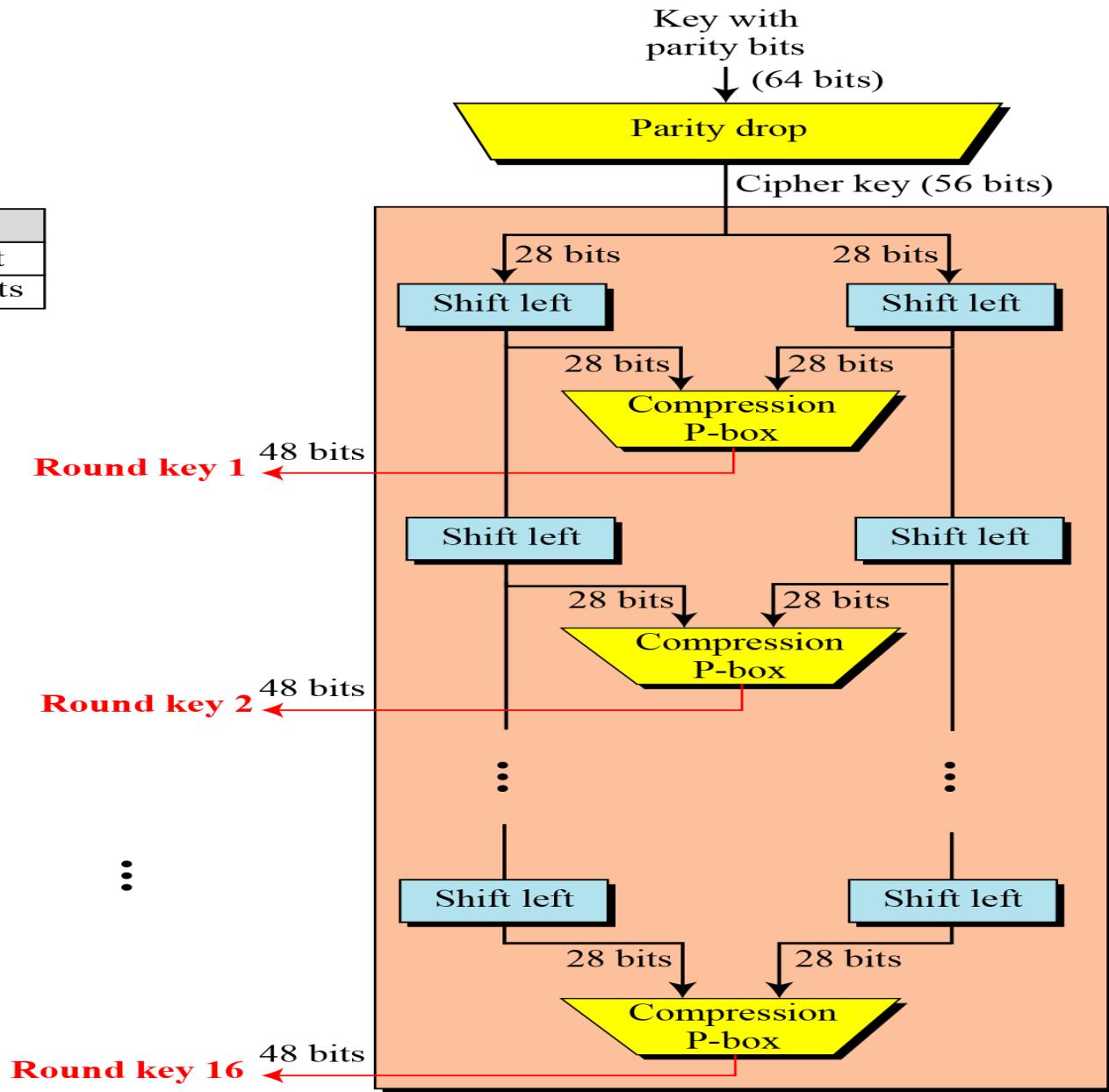
To achieve this goal, one approach is to make the last round (round 16) different from the others; it has only a mixer and no swapper.

DES Cipher and Reverse Cipher



Key Generation

Shifting	
Rounds	Shift
1, 2, 9, 16	one bit
Others	two bits



Round-Key Generator

Permutation 1

PC-1

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

PC-1

IP 58 50 42 34 26 18 10 2
 60 52 44 36 28 20 12 4
 62 54 46 38 30 22 14 6
 64 56 48 40 32 24 16 8
 57 49 41 33 25 17 9 1
 59 51 43 35 27 19 11 3
 61 53 45 37 29 21 13 5
 63 55 47 39 31 23 15 7

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Permutation Choice 2

PC-2

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Key Rotation Schedule

Round Number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Number of Left Shifts	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1
Total Number of Shifts	1	2	4	6	8	10	12	14	15	17	19	21	23	25	27	28

Iteration Corresponds to Left Shifts

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
1 1 2 2 2 2 2 1 2 2 2 2 2 2 2 1

Avalanche Effect

DES exhibits strong avalanche, where a change of **one** input or key bit results in changing approx **half** output bits.

Two desired properties of a block cipher are the avalanche effect and the completeness.

To check the avalanche effect in DES, let us encrypt two plaintext blocks (with the same key) that differ only in one bit and observe the differences in the number of bits in each round.

Plaintext: 0000000000000000

Key: 22234512987ABB23

Ciphertext: 4789FD476E82A5F1

Plaintext: 0000000000000001

Key: 22234512987ABB23

Ciphertext: 0A4ED5C15A63FEA3

Avalanche Effect Contd.

Although the two plaintext blocks differ only in the rightmost bit, the ciphertext blocks differ in 29 bits. This means that changing approximately 1.5 percent of the plaintext creates a change of approximately 45 percent in the ciphertext.

Rounds	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bit differences	1	6	20	29	30	33	32	29	32	39	33	28	30	31	30	29

Completeness Effect

Completeness effect means that each bit of the ciphertext needs to depend on many bits on the plaintext. The **Diffusion** and **Confusion** produced by P-boxes and S-boxes in DES shows a very strong completeness effect.

Design Criteria of S-boxes

The design provides confusion and diffusion of bits from each round to the next.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	04	13	01	02	15	11	08	03	10	06	12	05	09	00	07
1	00	15	07	04	14	02	13	10	03	06	12	11	09	05	03	08
2	04	01	14	08	13	06	02	11	15	12	09	07	03	10	05	00
3	15	12	08	02	04	09	01	07	05	11	03	14	10	00	06	13

- The entries of each row are permutation of values are between 0 and 15.
- If there is a single bit change in the input, two or more bits will be changed in the output.
- If two inputs to an S-box differ only in the two middle bits (bit 3 and bit 4), the output must differ in at least two bits.

Design Criteria of S-boxes Contd.

If two inputs to an S-box differ in the first two bits (bit 1 and bit 2) and are the same in the last two bits (5 and 6), the two outputs must be different.

There are only 32 6-bit input word pairs(X_i and X_j) in which $X_i \Theta X_j \neq (000000)_2$. These 32 input pairs create 32 4-bit output word pair. If we create the difference between the 32 output pair $d = Y_i \Theta Y_j$, no more than 8 of these d should be the same.

Design Criteria for P-boxes

Between two rounds of S-boxes, there are one straight P-box (32 bit to 32 bit) and one Expansion P-box (32 bit to 48 bit). These two P-boxes together provide diffusion of bits.

- Each S-box input comes from the output of a different S-box (in the previous round).
- The four outputs from each S-box go to six different S-boxes (in the next round).
- No two output bits from an S-box go the same S-boxes (in the next round).
- For each S-box, the two output bits go to the first or last two bits of an S-box in the next round. The other two output bits go the middle bits of an S-box in the next round.

Design Criteria for P-boxes Contd.

If we number the eight S-boxes, $S_1, S_2, S_3, \dots, S_8$

- An output of S_{j-2} goes to one of the first two bits of S_j (in the next round).
- An output bit from S_{j-1} goes to one of the last two bits of S_j (in the next round).
- An output of S_{j+1} goes to one of the two middle bits of S_j (in the next round).

DES Weaknesses

During the last few years critics have found some weakness in DES

Weakness in S-Boxes

- At least three weaknesses are mentioned in the literature for S-boxes.
- In S-box 4, the last three output bits can be derived in the same way as the first output bit by complementing some of the input bits.
- Two specifically chosen inputs to an S-box array can create the same output.
- It is possible to obtain the same output in a single round by changing bits in only three neighboring S-boxes.

Weakness in P-boxes

- It is not clear why the designer of DES used the initial and final permutation; these have no security benefits.
- In the expansion permutation (inside the function), the first and fourth bits of every 4-bit series are repeated.

Number of Rounds

DES uses sixteen rounds of Feistel ciphers. the ciphertext is thoroughly a random function of plaintext and ciphertext.

Weakness in Keys

Table 6.18 Weak keys

<i>Keys before parities drop (64 bits)</i>	<i>Actual key (56 bits)</i>
0101 0101 0101 0101	0000000 0000000
1F1F 1F1F 0E0E 0E0E	0000000 FFFFFFFF
E0E0 E0E0 F1F1 F1F1	FFFFFFF 0000000
FEFE FEFE FEFE FEFE	FFFFFFF FFFFFFFF

Let us try the first weak key in Table 6.18 to encrypt a block two times. After two encryptions with the same key the original plaintext block is created. Note that we have used the encryption algorithm two times, not one encryption followed by another decryption.

Key: 0x0101010101010101

Plaintext: 0x1234567887654321

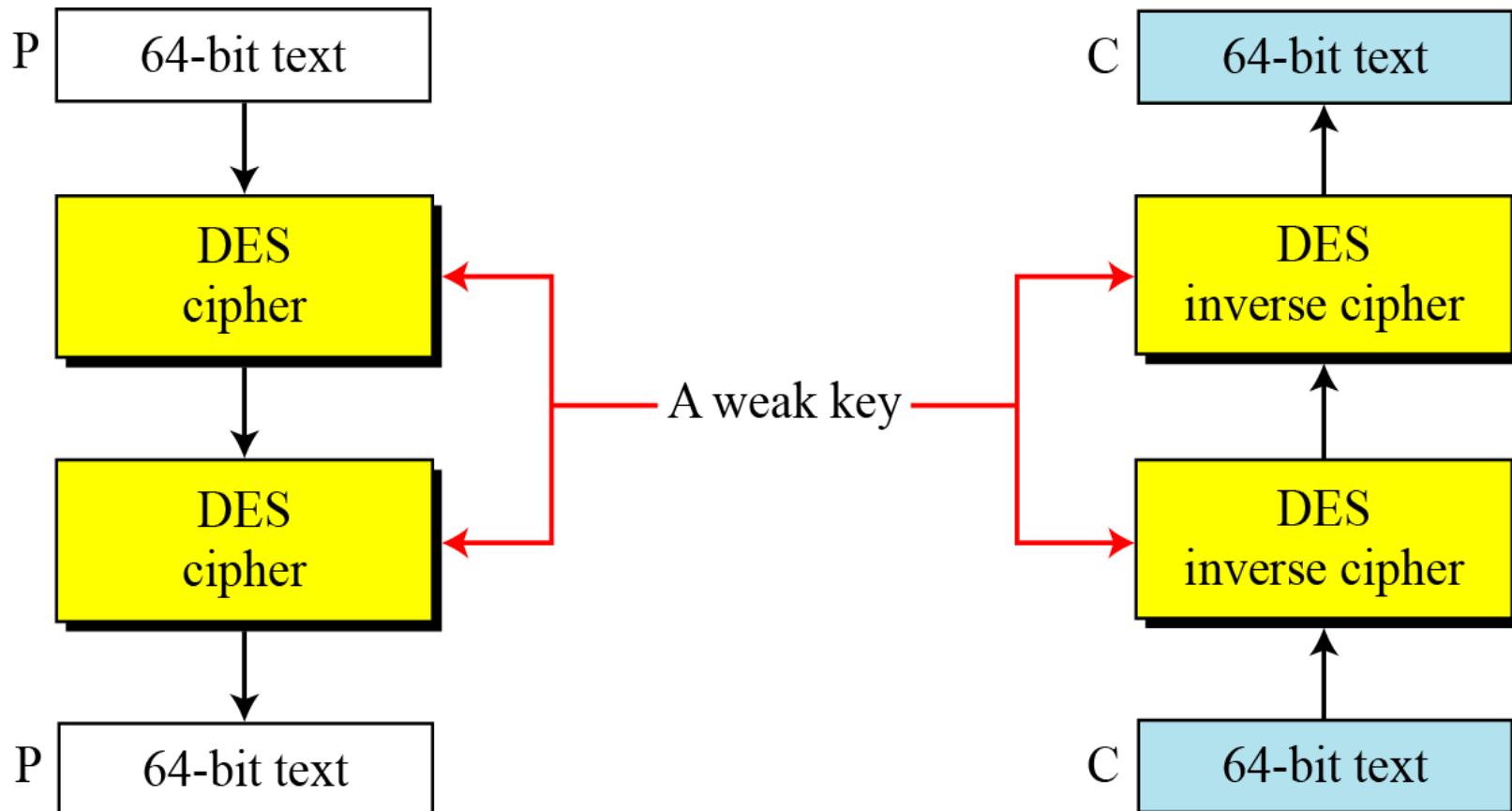
Ciphertext: 0x814FE938589154F7

Key: 0x0101010101010101

Plaintext: 0x814FE938589154F7

Ciphertext: 0x1234567887654321

Double Encryption and Decryption with a Weak Key



Semi Weak Keys

There are six key pairs that are called semi weak keys and they are shown in the below table. A semi weak key creates only two different round keys and each of them is repeated eight times. In addition, the round keys created from each pair are the same with different orders.

Table 6.19 *Semi-weak keys*

<i>First key in the pair</i>	<i>Second key in the pair</i>
01FE 01FE 01FE 01FE	FE01 FE01 FE01 FE01
1FE0 1FE0 0EF1 0EF1	E01F E01F F10E F10E
01E0 01E1 01F1 01F1	E001 E001 F101 F101
1FFE 1FFE 0EFE 0EFE	FE1F FE1F FE0E FE0E
011F 011F 010E 010E	1F01 1F01 0E01 0E01
EOF E0FE F1FE F1FE	FEE0 FEE0 FEF1 FEF1

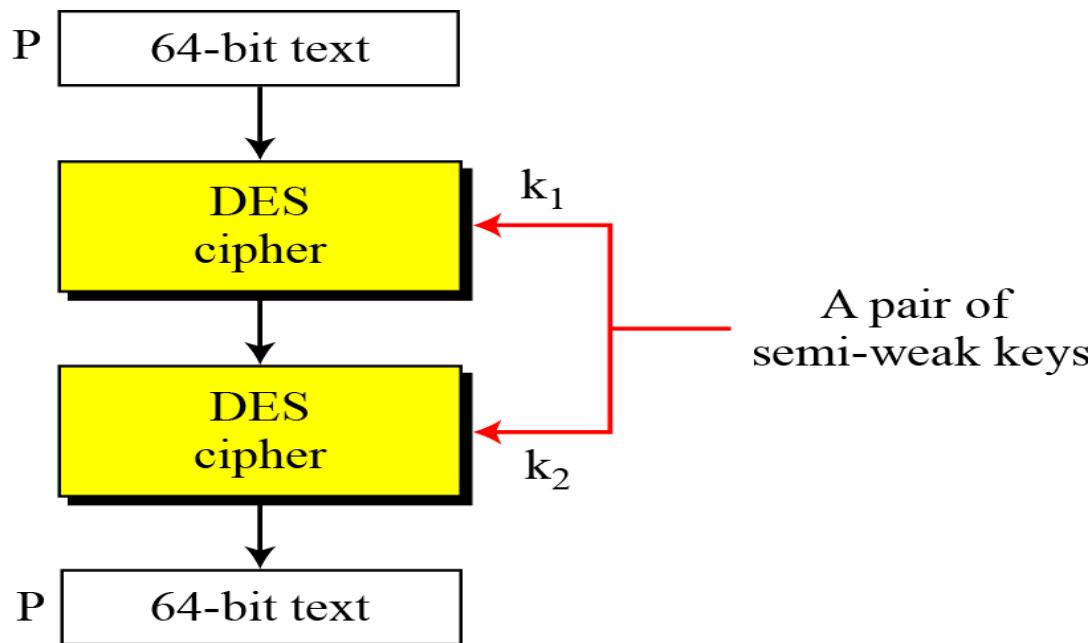
Semi Weak Keys

<i>Round key 1</i>	9153E54319BD	6EAC1ABCE642
<i>Round key 2</i>	6EAC1ABCE642	9153E54319BD
<i>Round key 3</i>	6EAC1ABCE642	9153E54319BD
<i>Round key 4</i>	6EAC1ABCE642	9153E54319BD
<i>Round key 5</i>	6EAC1ABCE642	9153E54319BD
<i>Round key 6</i>	6EAC1ABCE642	9153E54319BD
<i>Round key 7</i>	6EAC1ABCE642	9153E54319BD
<i>Round key 8</i>	6EAC1ABCE642	9153E54319BD
<i>Round key 9</i>	9153E54319BD	6EAC1ABCE642
<i>Round key 10</i>	9153E54319BD	6EAC1ABCE642
<i>Round key 11</i>	9153E54319BD	6EAC1ABCE642
<i>Round key 12</i>	9153E54319BD	6EAC1ABCE642
<i>Round key 13</i>	9153E54319BD	6EAC1ABCE642
<i>Round key 14</i>	9153E54319BD	6EAC1ABCE642
<i>Round key 15</i>	9153E54319BD	6EAC1ABCE642
<i>Round key 16</i>	6EAC1ABCE642	9153E54319BD

Possible Weak Keys There are also 48 keys that are called **possible weak keys**. A possible weak key is a key that creates only four distinct round keys; in other words, the sixteen round keys are divided into four groups and each group is made of four equal round keys.

Semi Weak Keys

A pair of semi-weak keys in Encryption and Decryption



What is the probability of randomly selecting a weak, a semi-weak, or a possible weak key?

Solution

DES has a key domain of 2^{56} . The total number of the above keys are 64 ($4 + 12 + 48$). The probability of choosing one of these keys is 8.8×10^{-16} , almost impossible.

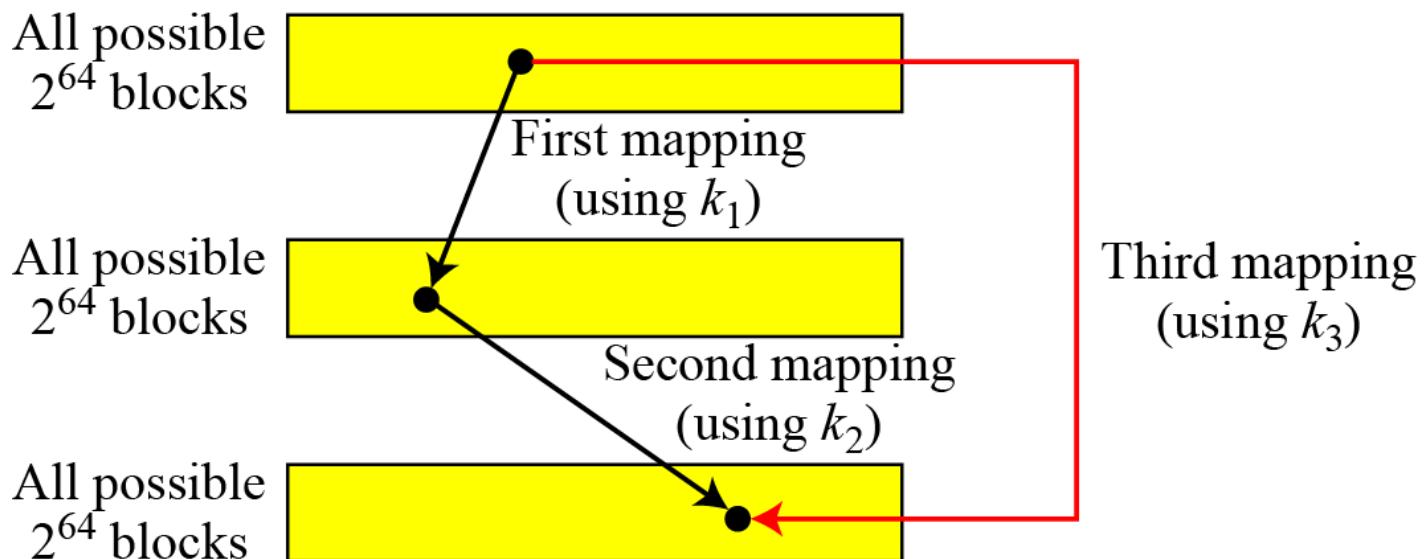
Strength of DES – Key Size

- 56-bit keys have $2^{56} = 7.2 \times 10^{16}$ values
- Brute Force search looks hard
- Recent advances have shown is possible
 - in 1997 on Internet in a few months
 - in 1998 on dedicated h/w (EFF) in a few days
 - in 1999 above combined in 22hrs!
- Still must be able to recognize plaintext
- Now considering alternatives to DES

Multiple DES

The major criticism of DES regards its key length. Fortunately DES is not a group. This means that we can use double or triple DES to increase the key size.

A substitution that maps every possible input to every possible output is a group.

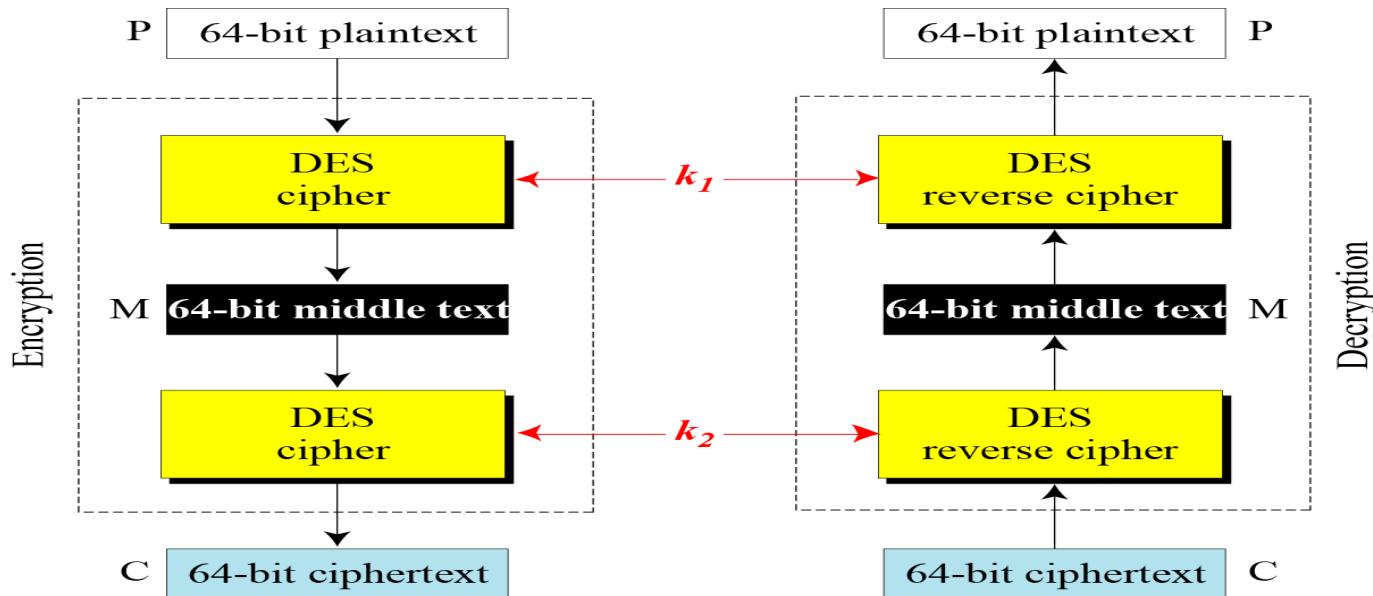


Double DES

The first approach is to use double DES (2DES).

Meet-in-the-Middle Attack

However, using a known-plaintext attack called [meet-in-the-middle attack](#) proves that double DES improves this vulnerability slightly (to 2^{57} tests), but not tremendously (to 2^{112}).



Meet-in-the-middle attack for Double DES

Double DES Contd.

Tables for meet-in-the-Middle attack

$$M = E_{k_1}(P)$$

M	k_1
●	

$$M = D_{k_2}(C)$$

M	k_2
●	

Find equal M's and record
corresponding k_1 and k_2

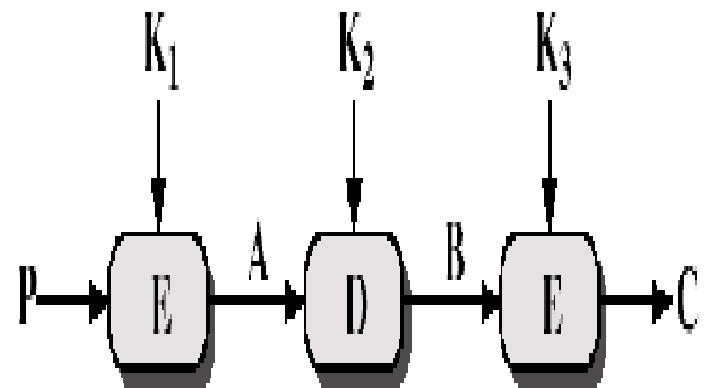
Triple DES Contd.

- Use three keys and three executions of the DES algorithm (encrypt-decrypt-encrypt)

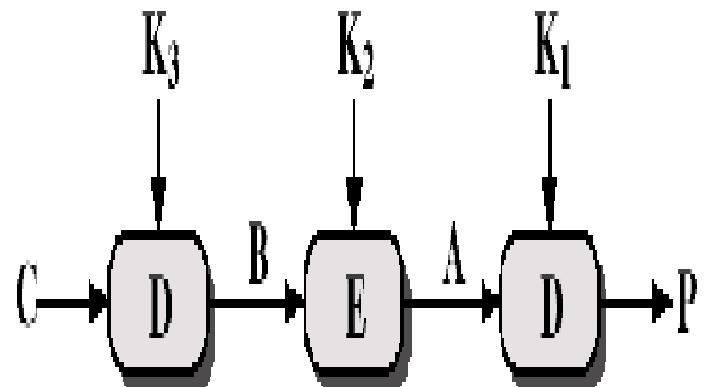
$$C = EK_3[DK_2[EK_1[P]]]$$

- C = Ciphertext
- P = Plaintext
- $EK[X]$ = encryption of X using key K
- $DK[Y]$ = decryption of Y using key K

- Effective key length of 168 bits



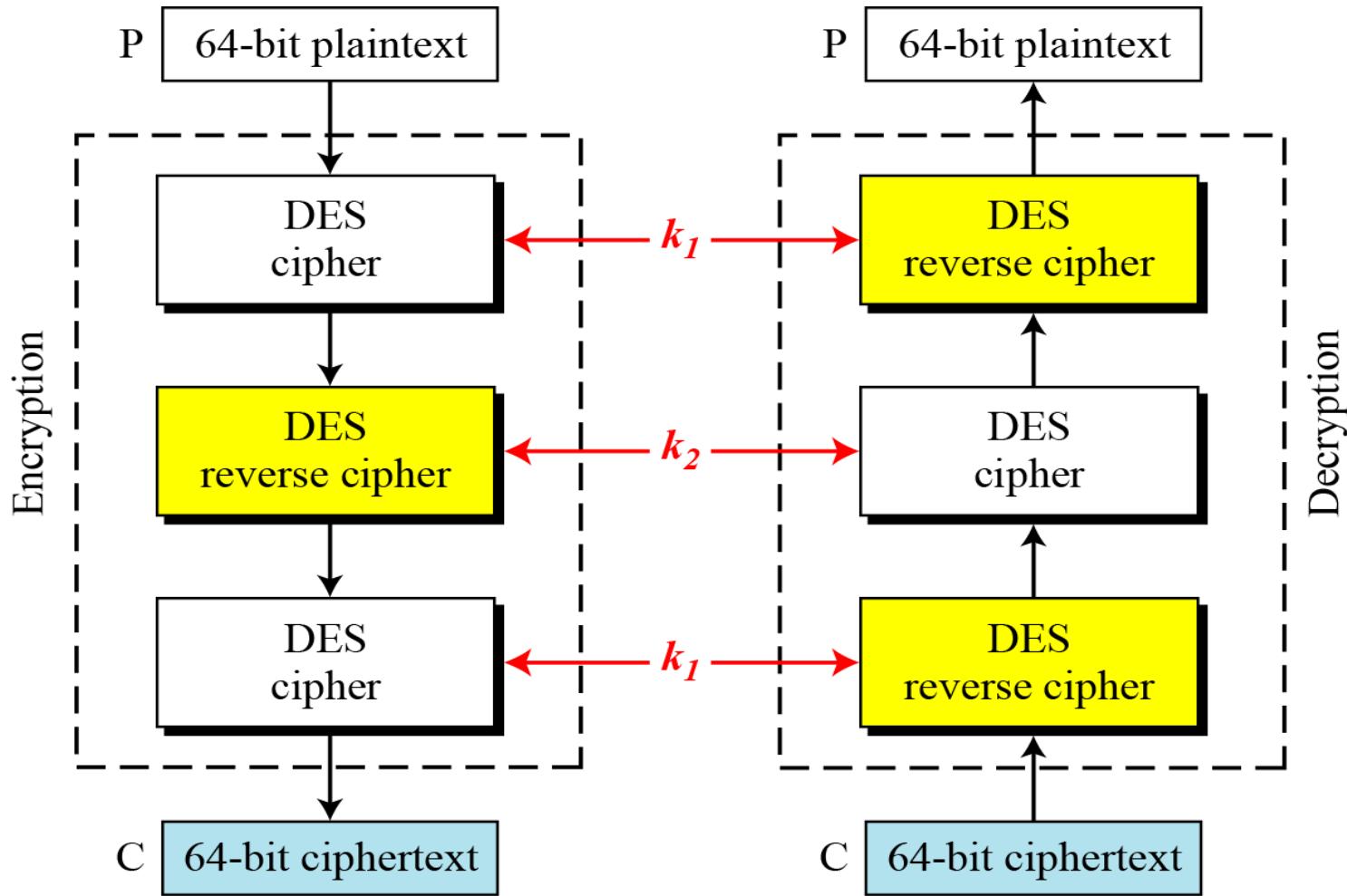
(a) Encryption



(b) Decryption

Triple DES

Triple DES with two keys



Triple DES

- Clearly a replacement for DES was needed
 - theoretical attacks that can break it
 - demonstrated exhaustive key search attacks
- AES is a new cipher alternative
- Prior to this alternative was to use multiple encryption with DES implementations
- Triple-DES is the chosen form

Why Triple-DES?

- Why not Double-DES?
 - NOT same as some other single-DES use, but have
- Meet-in-the-Middle attack
 - works whenever use a cipher twice
 - since $X = E_{K1}[P] = D_{K2}[C]$
 - attack by encrypting P with all keys and store
 - then decrypt C with keys and match X value
 - can show takes $O(2^{56})$ steps

Triple-DES with Two-Keys

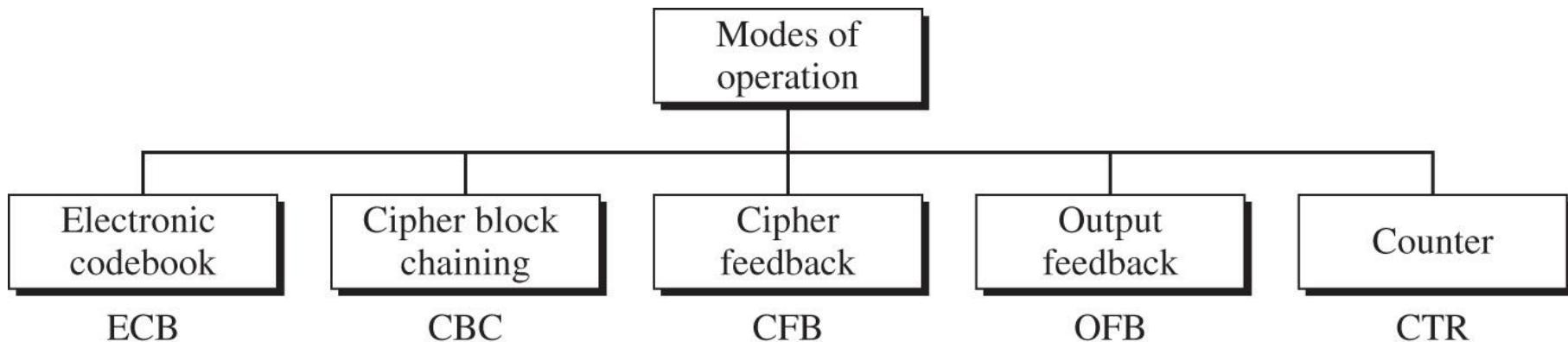
- Hence must use 3 encryptions
 - would seem to need 3 distinct keys
- but can use 2 keys with E-D-E sequence
 - $C = E_{K_1}[D_{K_2}[E_{K_1}[P]]]$
 - nb encrypt & decrypt equivalent in security
 - if $K_1=K_2$ then can work with single DES
- Standardized in ANSI X9.17 & ISO8732
- No current known practical attacks

Triple-DES with Three-Keys

- Although are no practical attacks on two-key Triple-DES have some indications
- Can use Triple-DES with Three-Keys to avoid even these
 - $C = E_{K3}[D_{K2}[E_{K1}[P]]]$
- has been adopted by some Internet applications, eg PGP, S/MIME

Cipher Block Modes

- Different ways to transmit data
- Ciphertext depend on something else (besides key) which is different each time
- Some designed to generate ciphertext one byte at a time
- Can be used with any block cipher (DES, AES...)



Electronic Codebook Book (ECB)

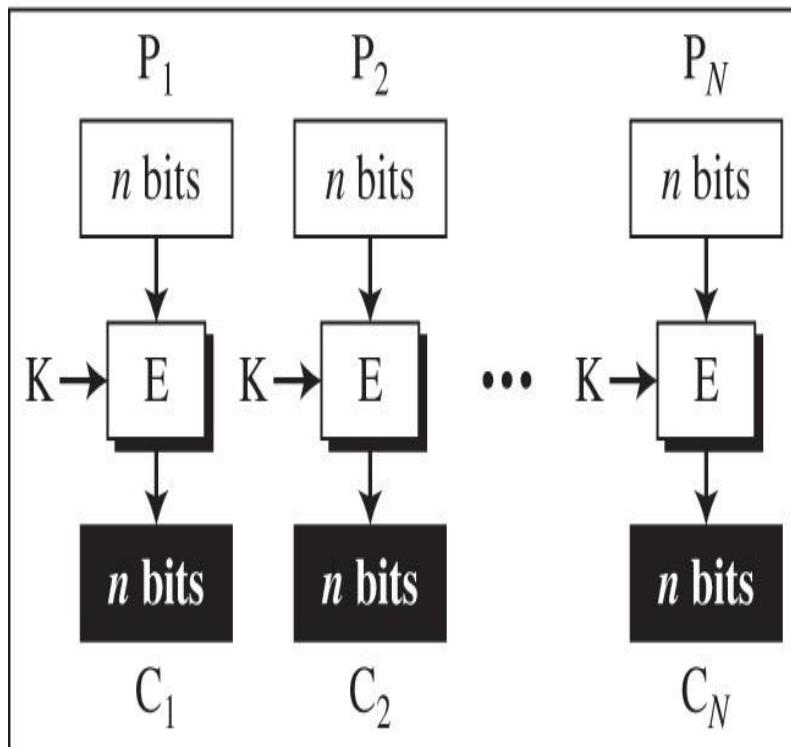
E: Encryption

D: Decryption

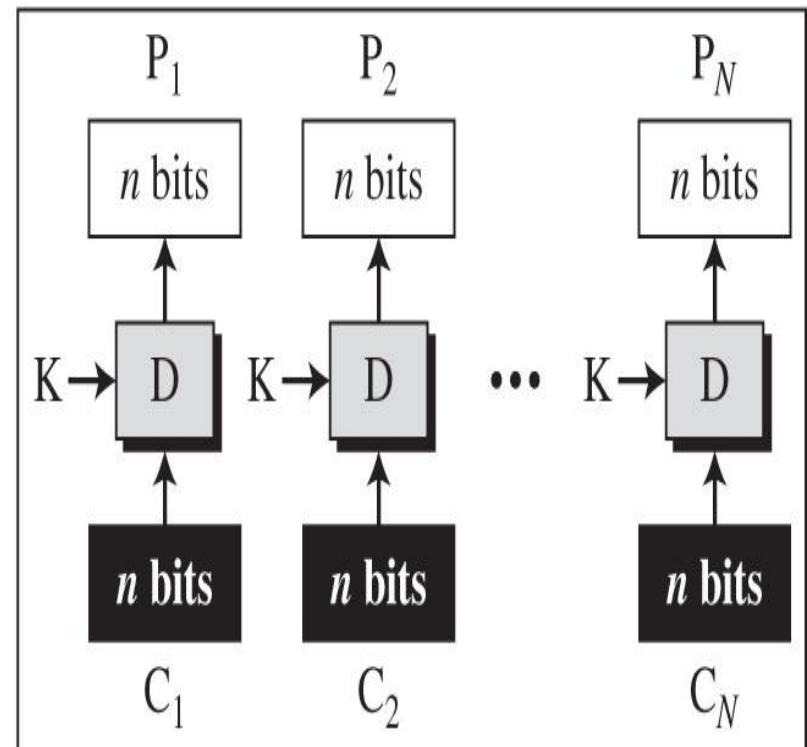
P_i : Plaintext block i

C_i : Ciphertext block i

K: Secret key



Encryption



Decryption

Electronic Codebook Book (ECB) Contd.

- It is the simplest mode of operation. Here, the incoming plain text message is divided into ‘N’ blocks of ‘n’ bits each. Each such block is then encrypted independently of the other blocks. For all blocks in a message, the same key is used for encryption.
- At the receiver’s end, the incoming data is divided into ‘N’ blocks of ‘n’ bits each, and by using the same key as was used for encryption, each block is decrypted to produce the corresponding plain text block.
- In ECB, since a single key is used for encrypting all the blocks of message, if a plain text block repeats in the original message, the corresponding cipher text block will also repeat in the encrypted message. Therefore ECB is suitable only for encrypting small message where the scope for repeating the same plain text blocks is quite less.

Electronic Codebook Mode (ECB) Contd.

Advantages

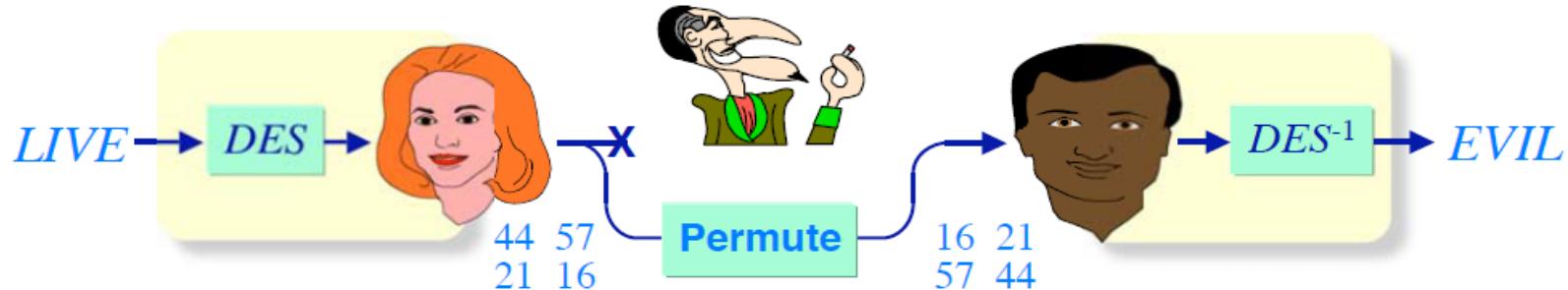
- Each block can be encrypted/decrypted in parallel
- Noise in one block affects no other block
- Simple

Disadvantage: vulnerable to cryptanalysis

- Repetitive information contained in the plaintext may show in the ciphertext, if aligned with blocks.
- If the same message (e.g., an SSN) is encrypted (with the same key)
- Typical application: secure transmission of short pieces of information (e.g. a temporary encryption key)

Electronic Codebook Mode (ECB) Contd.

Electronic Codebook (ECB) Mode Misordered blocks attacks



- ◆ Alice sends Bob a secret message
 - » Message is LIVE (11 08 21 04)
 - » Message is enciphered as: 44 57 21 16
- ◆ Stanley intercepts the message and rearranges the blocks
 - » Now the enciphered message is: 16 21 57 44
 - » (Stanley could also have deleted or replayed blocks)
- ◆ Bob receives the message, deciphers it as “EVIL”
 - » How can Bob know if the message is correct?

Cipher Block Chaining (CBC)

- Cipher Block Chaining mode ensures that even if a block of plain text repeats in the input, these two or more identical plain text blocks yield totally different cipher text blocks in the output. For this, a feedback mechanism is used.
- First step receives two inputs: the first block of plain text and a random block of text called as Initialization Vector (IV)
- Remember that the IV is used only in the first plain text block. However, the same key is for encryption of all plain text blocks.

Cipher Block Chaining (CBC) Contd.

- The plaintext is broken into blocks: P_1, P_2, P_3, \dots
- Each plaintext block is XORed (chained) with the previous ciphertext block before encryption (hence the name):

$$C_i = E_K(C_{i-1} \oplus P_i)$$

$$C_0 = \text{IV}$$

- Use an Initial Vector (IV) to start the process.
- Decryption : $P_i = C_{i-1} \oplus D_K(C_i)$
- Application : general block-oriented transmission.

Cipher Block Chaining (CBC)

E: Encryption

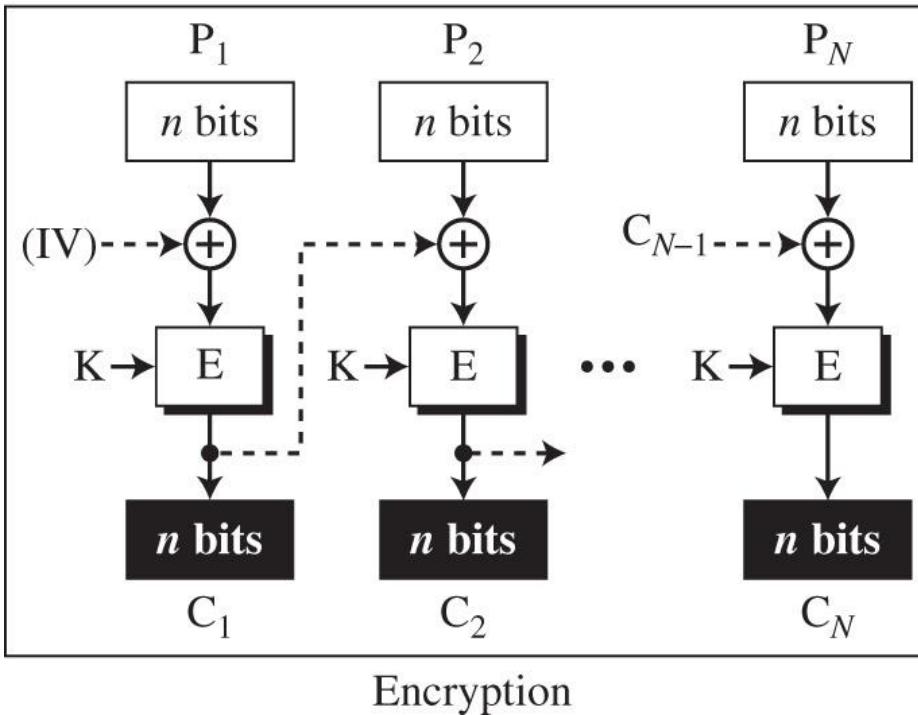
P_i : Plaintext block i

K: Secret key

D : Decryption

C_i : Ciphertext block i

IV: Initial vector (C_0)

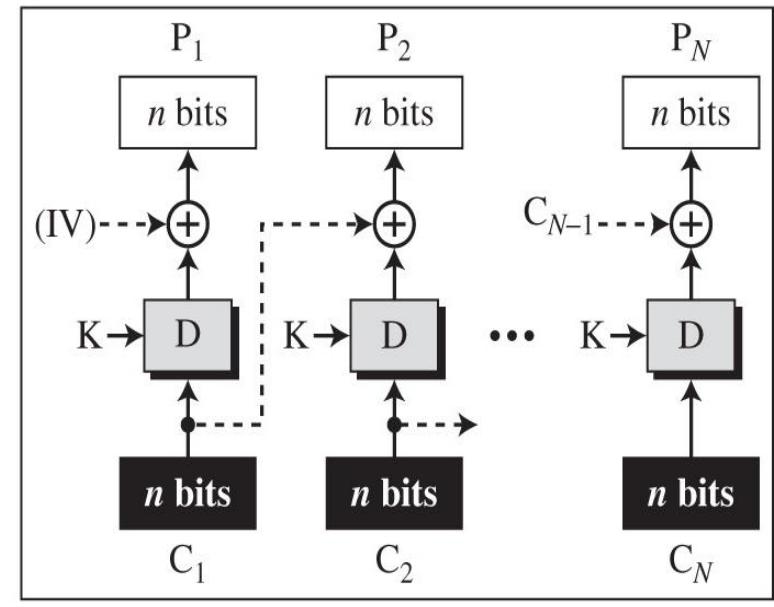


Encryption

Encryption

$$C0 = IV$$

$$Ci = E(K, Pi \oplus Ci-1)$$



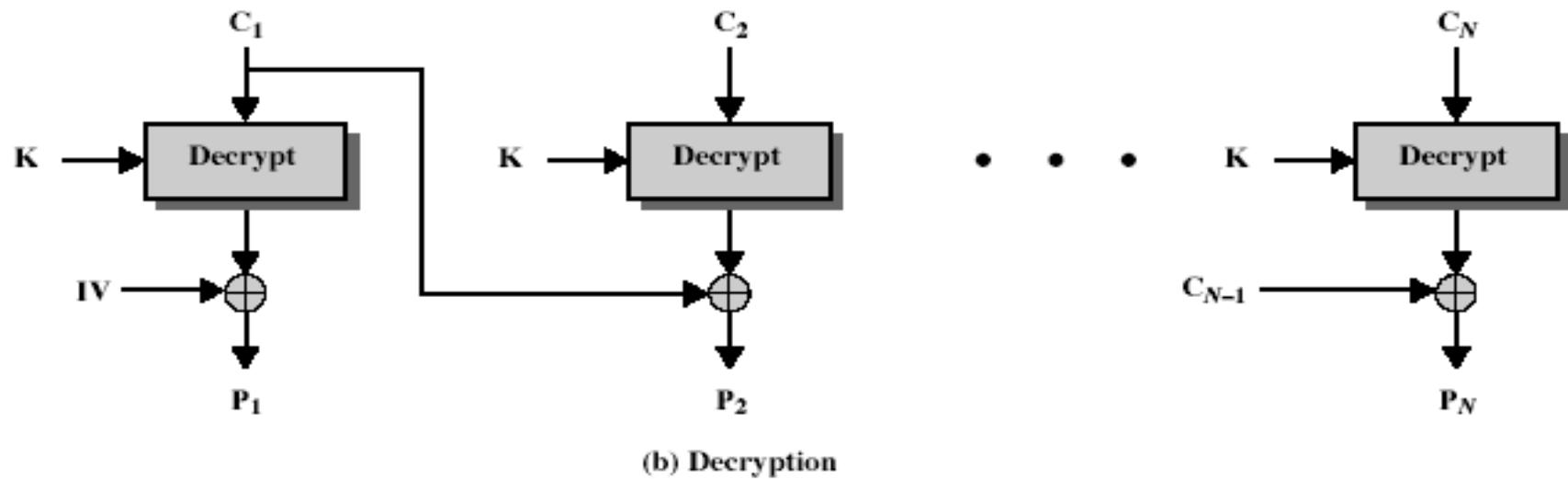
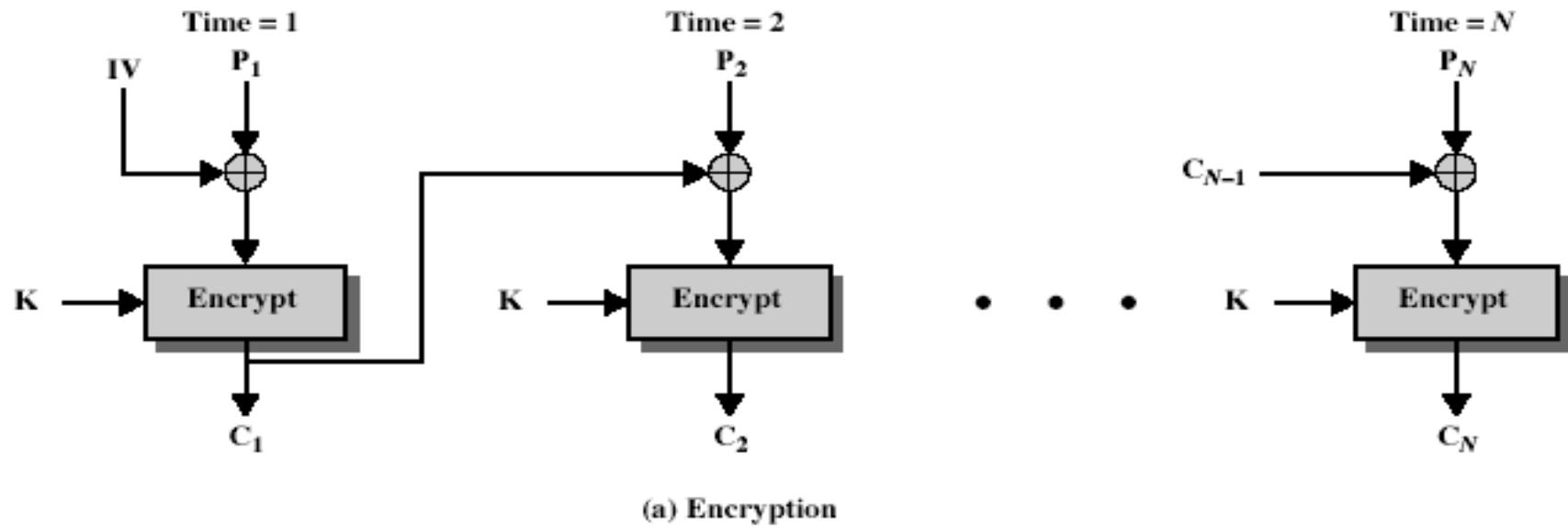
Decryption

Decryption

$$P0 = D(K, C0) \oplus IV$$

$$Pi = D(K, Ci) \oplus Ci-1$$

Cipher Block Chaining (CBC)



Advantages and Limitations of CBC

- So, repeated plaintext blocks are encrypted differently.
- A ciphertext block depends on **all** blocks before it

Need **Initialization Vector (IV)**.

- Which must be known to Sender & Receiver.
- If sent in clear, attacker can change bits of first block, and change IV to compensate.
- Hence IV must either be a fixed value OR must be sent encrypted in ECB mode before rest of message.

Problem No. 1

Consider a sensor X that periodically sends a 64-octet measurement to a receiver Y. One day the administrator decides that X should encrypt the measurement data using DES in CBC mode. How many octets does X now send for each measurement?

Solution to Problem No. 1

- DES takes a 8-octet (64-bit) plaintext block and yields a 8-octet cipherblock.
- CBC requires a 8-octet initialization vector (IV) to be sent along with the cipherblocks.
- So X now sends 64 octets of cipherblocks plus 8 octets of IV, for a total of 72 octets.

Problem No. 2

Recall that a **DES encryption operation takes a 64-bit plaintext block and a 56-bit key and produces a 64-bit ciphertext block.** Recall also that each DES encryption operation itself consists of a number of iterations, which we shall refer to as **basic iterations.**

For the DES encryption in CBC mode of a plaintext message of N 64-bit blocks, obtain the following (in terms of N)

- a. Total number of DES encryption operations.
- b. Size of the output. Explain briefly.
- c. Total number of basic iterations. Explain briefly.

Solution to Problem No. 2

Let the plaintext message be $[m_1, m_2, \dots, m_N]$.

Its CBC encryption is given by $C_j = \text{DES_Encrypt}(C_{j-1} \text{ XOR } m_j)$ for $j = 1, \dots, N$, where $C_0 = \text{IV}$.

- a. The DES-CBC Encryption involves N DES encryption operations.
- b. The output is $[\text{IV}, C_1, \dots, C_N]$, which is $(N+1)$ 64-bit blocks.
- c. Each DES encryption operation has
 - 16 iterations to transform the plaintext block into the ciphertext block. **So there are $16N$ of these iterations.**
 - 16 iterations to produce the 16 48-bit keys from the 56-bit key. But these 16 iterations need be done only once for the entire message. So the answer is **$16N + 16$ iterations**. **$16N$ and $32N$ are also acceptable.**

Problem No. 3

Sensor X periodically sends a 32-octet measurement to a receiver Y (1 octet = 8 bits). One day the administrator decides that X should protect the measurement data by adding a MAC obtained using DES in CBC mode (in the standard way). How many octets does X now send for each measurement?

Solution to Problem No. 3

DES operates on 8-octet (64-bit) data blocks.

CBC requires an IV of the encryption block size, so this too is 8 octets. The MAC consists of the IV and the residue (last cipher block) of DES-CBC encryption.

So X now sends 32-octet measurement plus 8-octet IV plus 8-octet residue: **48 octets total**

Cipher Feed Back (CFB)

- Not all applications can work with blocks of data. Security is also required in applications that are character-oriented. For instance, an operator can be typing keystrokes at terminal, which need to be immediately transmitted across the communications link in a secure manner. In such situations, stream cipher must be used.
- The Cipher Feedback(CFB) mode is useful in such cases. In this mode, data is encrypted in units that are smaller (e.g. they could be of size 8 bits, i.e. the size of a character typed by an operator) than a defined block size (which is usually 64 bits).

Cipher Feedback Mode Contd.

- Let us understand how CFB mode works, assuming that we are dealing with ‘J’ bits at a time. Since CFB is slightly more complicated as compared to the first two Cryptography Modes, we shall study CFB in a step-by-step fashion.

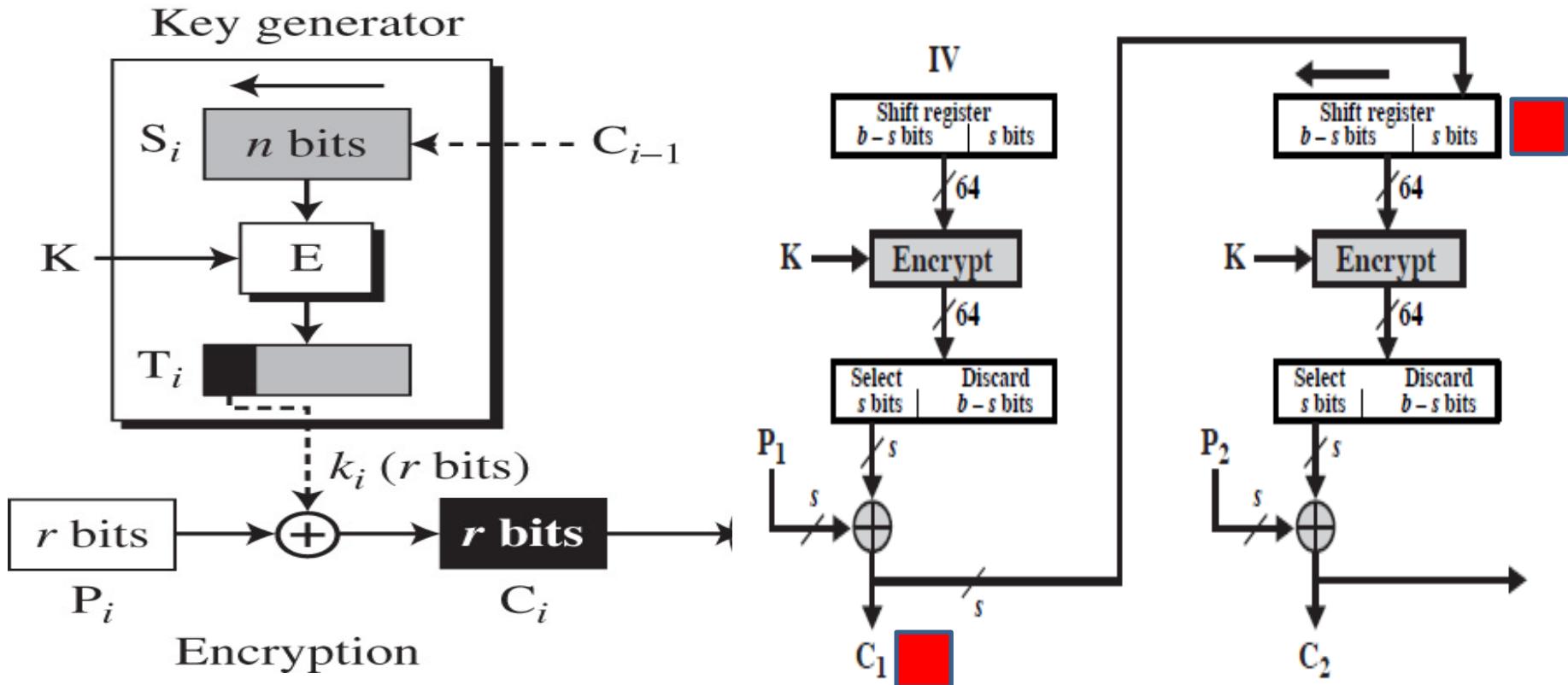
Step 1

- Like CBC, a 64 bit Initialization Vector (IV) is used in the case of CFB mode. The IV is kept in a shift register. It is encrypted in the first step to produce a corresponding 64-bit IV cipher text.

Cipher Feedback Mode Contd.

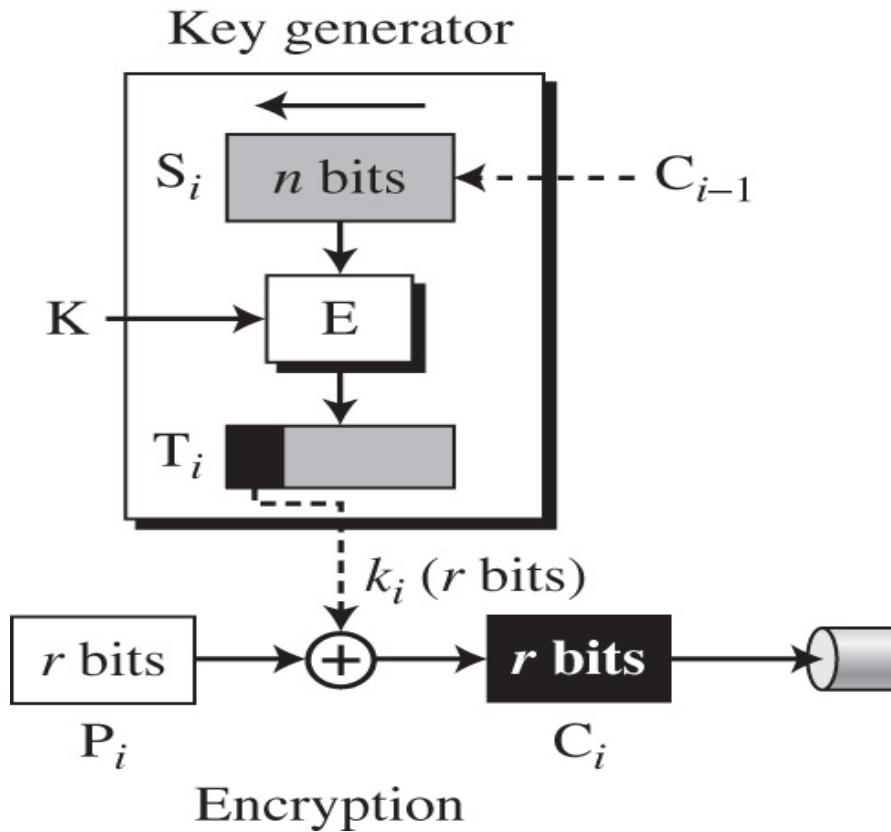
- The plaintext is a sequence of **segments** of s bits (where $s \leq$ block-size): $P_1, P_2, P_3, P_4, \dots$
- Encryption is used to generate a sequence of keys, each of s bits: $K_1, K_2, K_3, K_4, \dots$
- The ciphertext is $C_1, C_2, C_3, C_4, \dots$, where
$$C_i = P_i \oplus K_i$$
- How to generate the key stream?

Cipher Feedback Mode Contd.



- Previous ciphertexts used to create shift register S
- Shift register contents encrypted with key
- Results placed in “temporary register” T

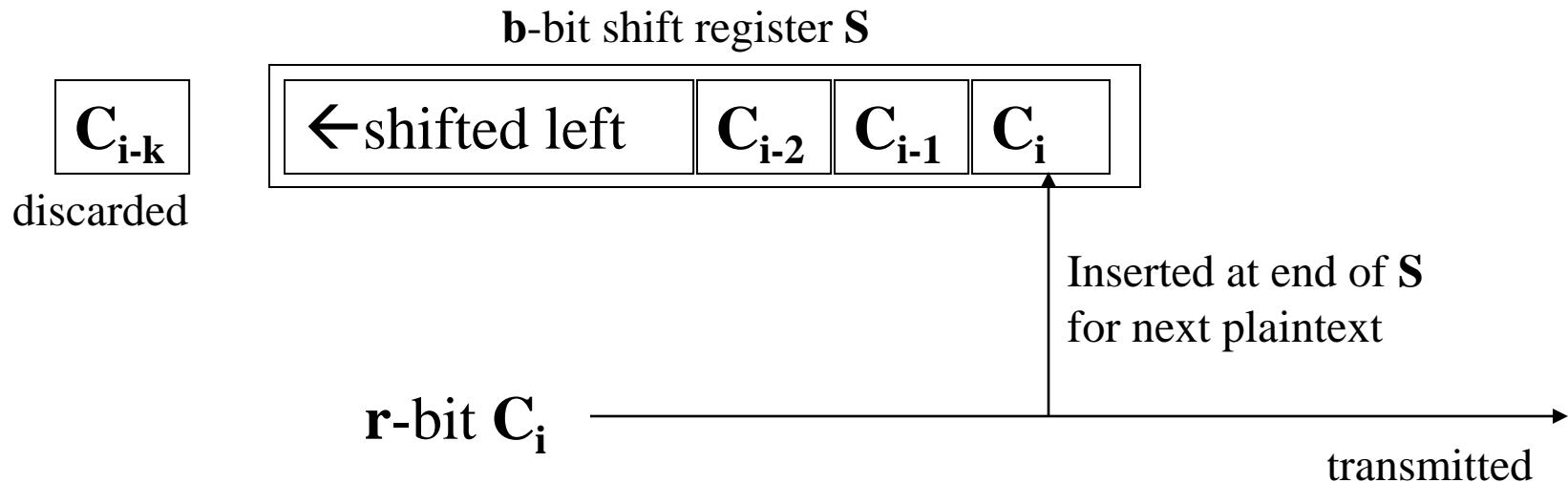
Cipher Feedback Mode Contd.



- First **r** bits of **T** used to create **byte key k_i**
- Byte key **XORed** with next **r** bits of plaintext to produces next **r** bits of ciphertext for transmission

Cipher Feedback Mode Contd.

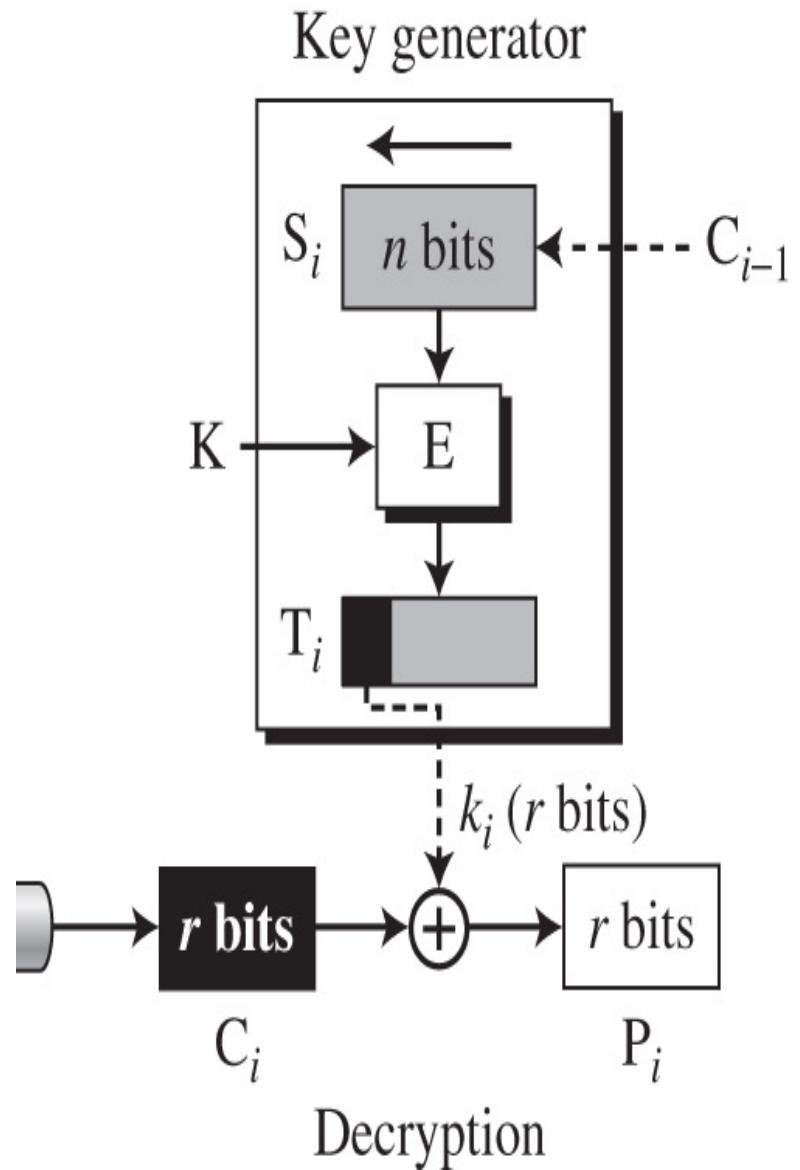
- Previous r bits of ciphertext added to end of shift register S
 - All other bits in S shifted left
 - First r bits discarded



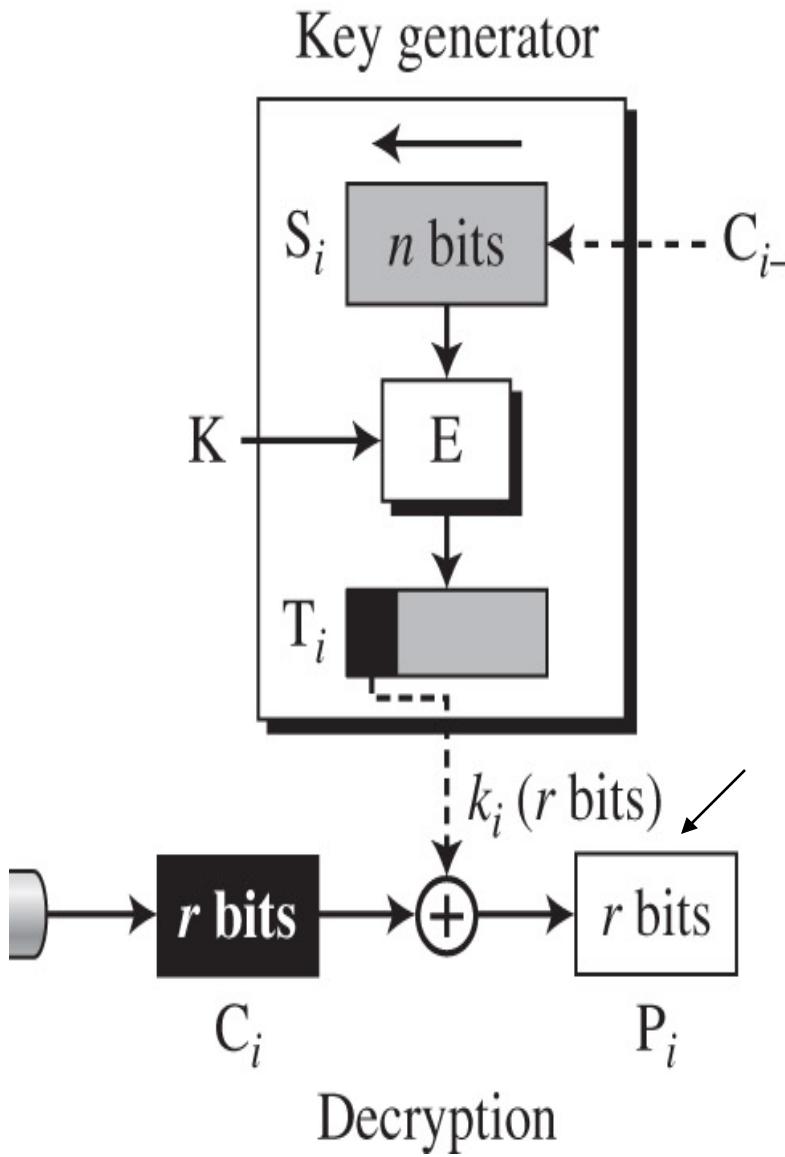
Cipher Feedback Mode Contd.

Decryption:

- Recipient uses previous ciphertext to create same shift register \mathbf{S}
 - Encrypted with key
 - First r bits taken to create byte key k_i
 - XORed with next r bits of ciphertext received to get next r bits of plaintext.



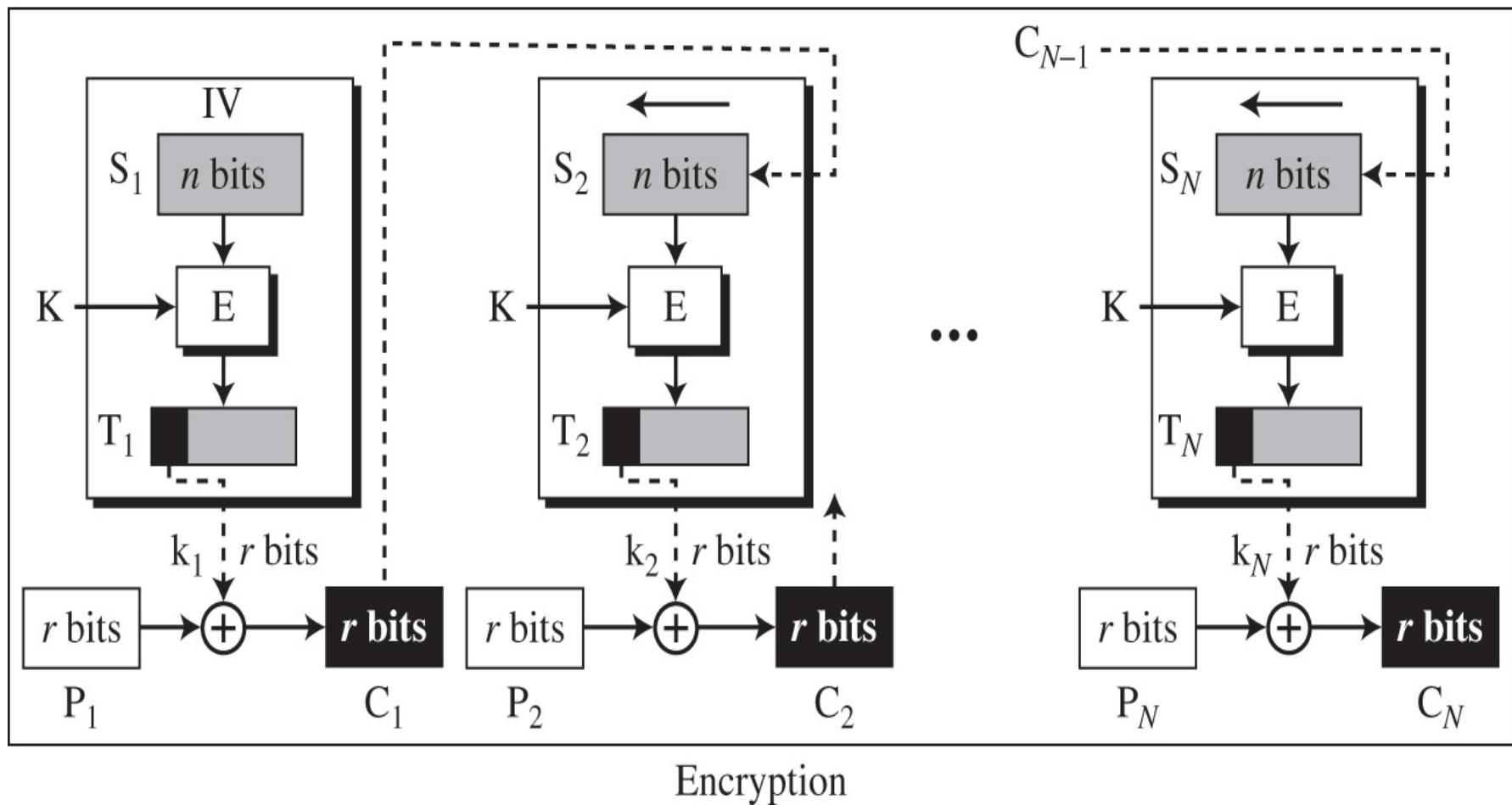
Cipher Feedback Mode Contd.



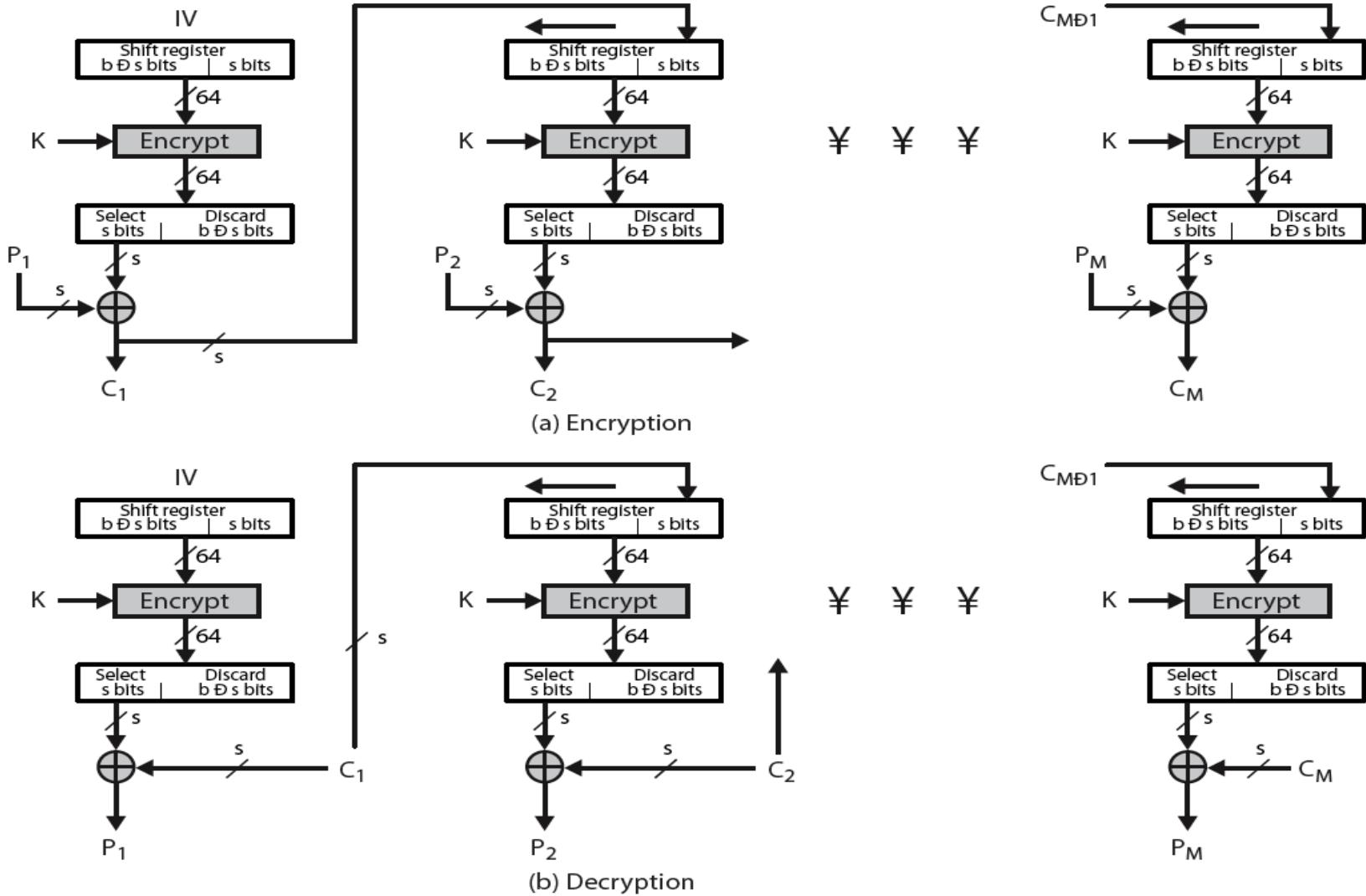
- Transmissions can be corrupted by noise
- In CFB one error corrupts many decrypted bytes (until error leaves shift register)
- Generally not a problem in modern networks which do error checking

Cipher Feedback Mode Contd.

- Initial contents of shift register **S** is **Initialization Vector (IV)**
- Rest of ciphertext depends on previous ciphertext



Cipher Feedback Mode Contd.



Advantages and Limitations of CFB

- Appropriate when data arrives in bits/bytes
- Limitation is need to stall while do block encryption after every n-bits
- CFB is a stream cipher as opposed to block cipher since it uses only 8-bit blocks
- CFB works like CBC by chaining all the preceding plaintexts
- CFB uses an initialization vector of size 64 bits
- CFB uses a shift register of 8 bits
- Note that the block cipher is used in **encryption** mode at **both** ends

Cipher Feedback Mode (CFB)

Problem

- CFB inherently sequential
 - Each block depends on previous block(s)
 - Cannot take advantage of parallel hardware to speed up encryption/decryption
 - Cannot generate byte keys in advance while waiting for rest of message

Solutions:

- Output Feedback Mode (OFB)
- Counter Mode (CTR)

Problem 1

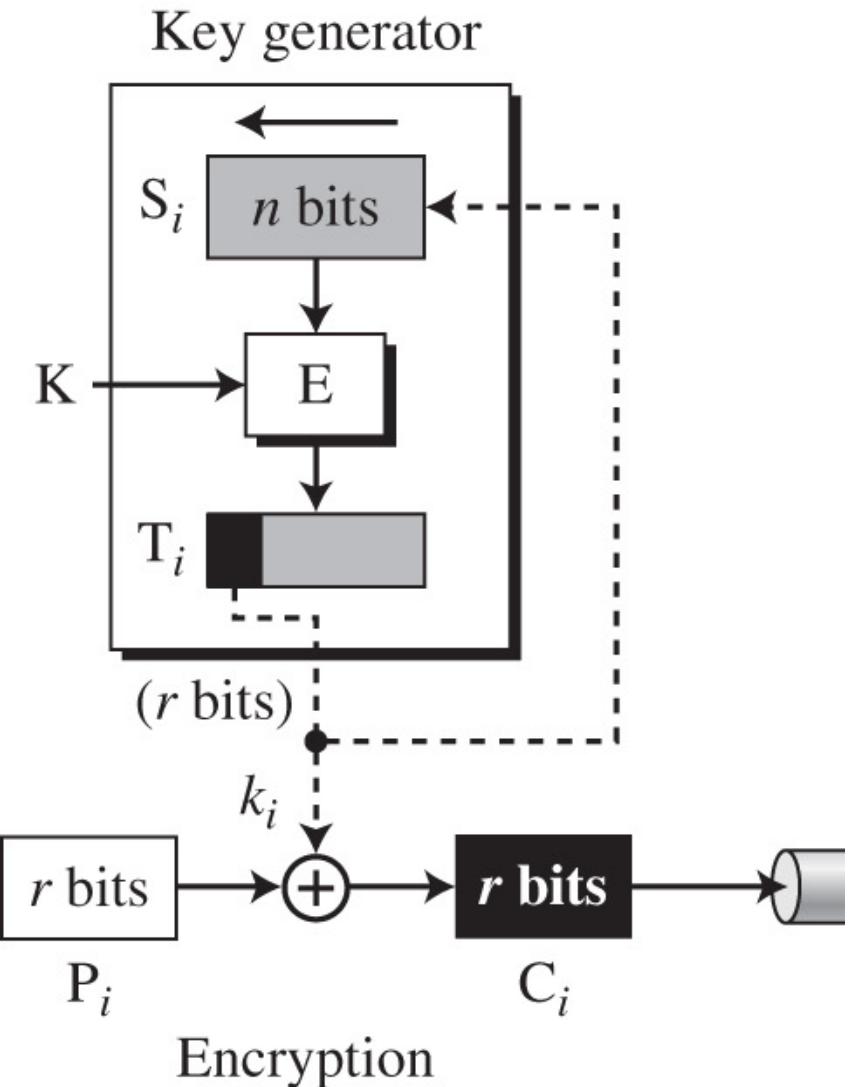
- With the ECB mode of DES, if there is an error in a block of the transmitted cipher text only the corresponding plaintext block is affected. However, in the CBC mode, this error propagates. For example, an error in the transmitted C_1 obviously corrupts in P_1 and P_2
 - Are any blocks beyond P_2 affected
 - Suppose that there is a bit error in the source version of P_1 . Through how many Ciphertext blocks is this error propagated? What is the effect at the receiver
- If a Bit error occurs in transmission of a cipher text character in 8bit CFB mode how far does the error propagates.

Solutions

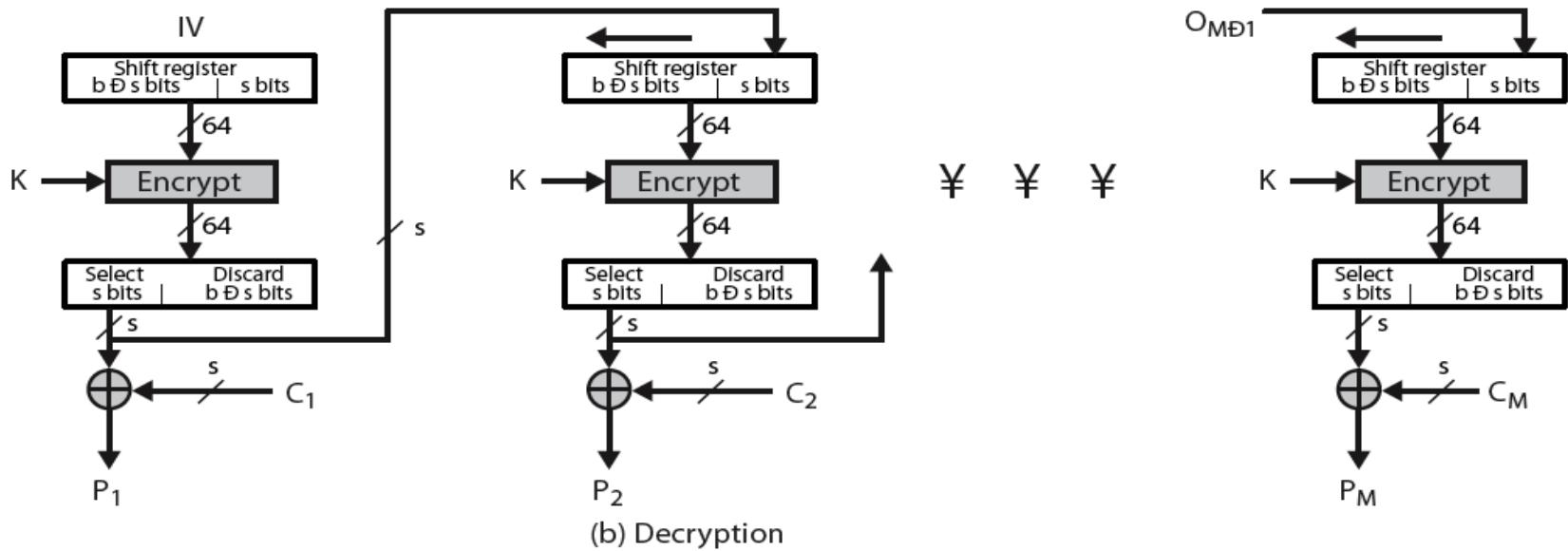
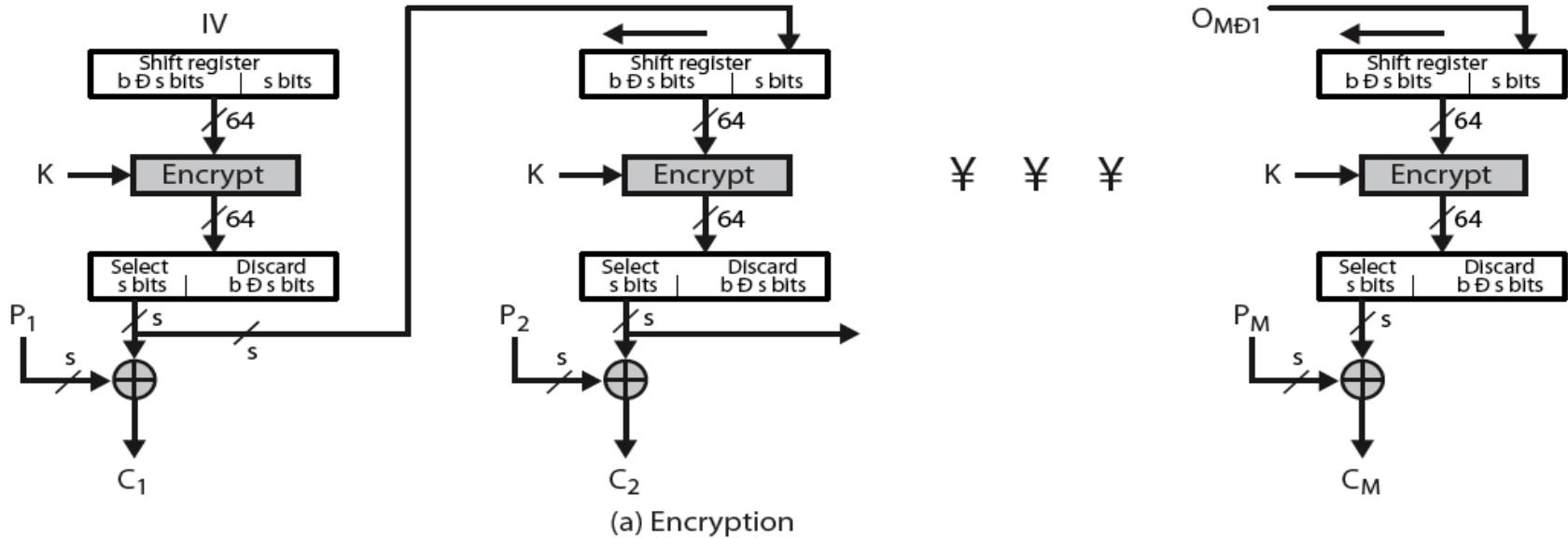
- A. No. For example, suppose C1 is corrupted. The output block P3 depends only on the input blocks C2 and C3.
 - B. An error in P1 affects C1. But since C1 is input to the calculation of C2, C2 is affected. This effect carries through indefinitely, so that all ciphertext blocks are affected. However, at the receiving end, the decryption algorithm restores the correct plaintext for blocks except the one in error. You can show this by writing out the equations for the decryption. Therefore, the error only effects the corresponding decrypted plaintext block.
2. Nine plaintext characters are affected. The plaintext character corresponding to the ciphertext character is obviously altered. In addition, the altered ciphertext character enters the shift register and is not removed until the next eight characters are processed.

Output Feedback Mode (OFB)

- Contents added to shift register taken directly from T
- Not dependent on the plaintext
- Could theoretically generate all byte keys in advance



Output Feedback Mode

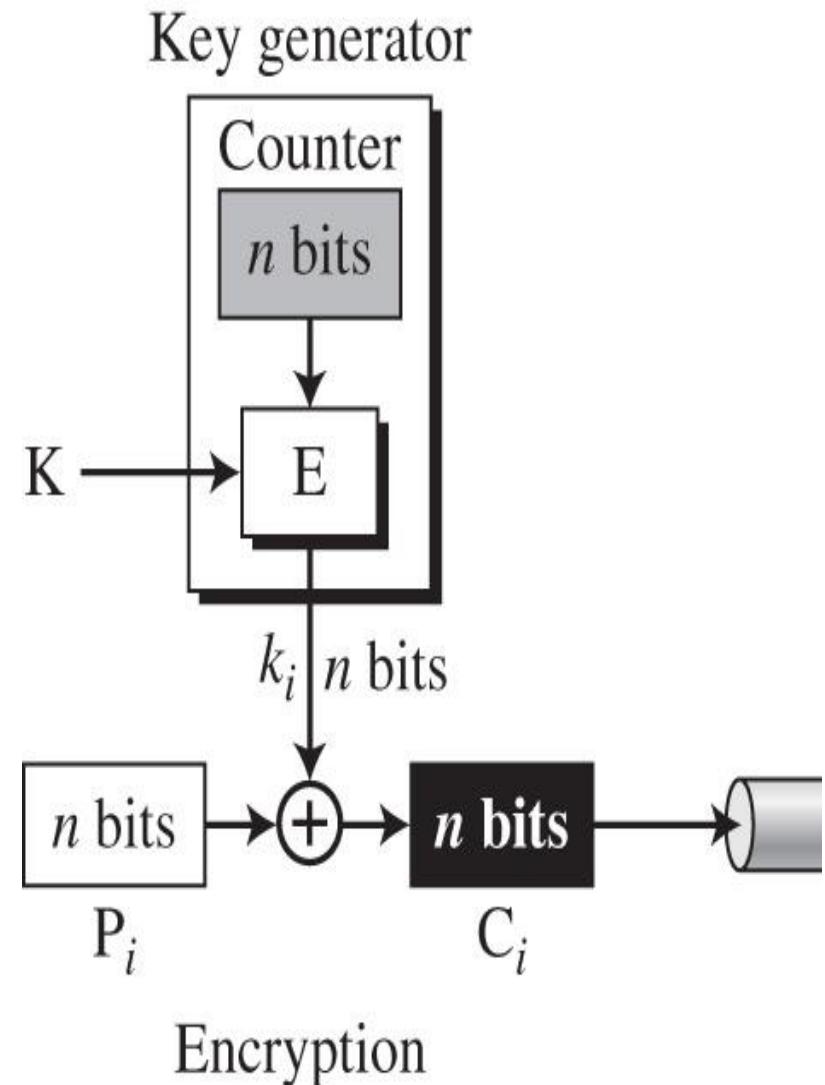


Counter Mode (CTR)

- Use a simple counter to generate next bytes of ciphertext
 - Counter increments each time → different ciphertext generated
 - Know all counter values in advance → Know all byte keys k_i in advance → Can encrypt/decrypt in parallel

Counter Mode (CTR)

- Counter generates next n bits used in key generator
 - Encrypted with key
 - XORed with plaintext
 - Can select first r bits of result for stream transmission



Counter Mode (CTR)

- Sender and recipient must know initial counter value **IV**
 - Can be transmitted via ECB mode

E : Encryption

P_i : Plaintext block i

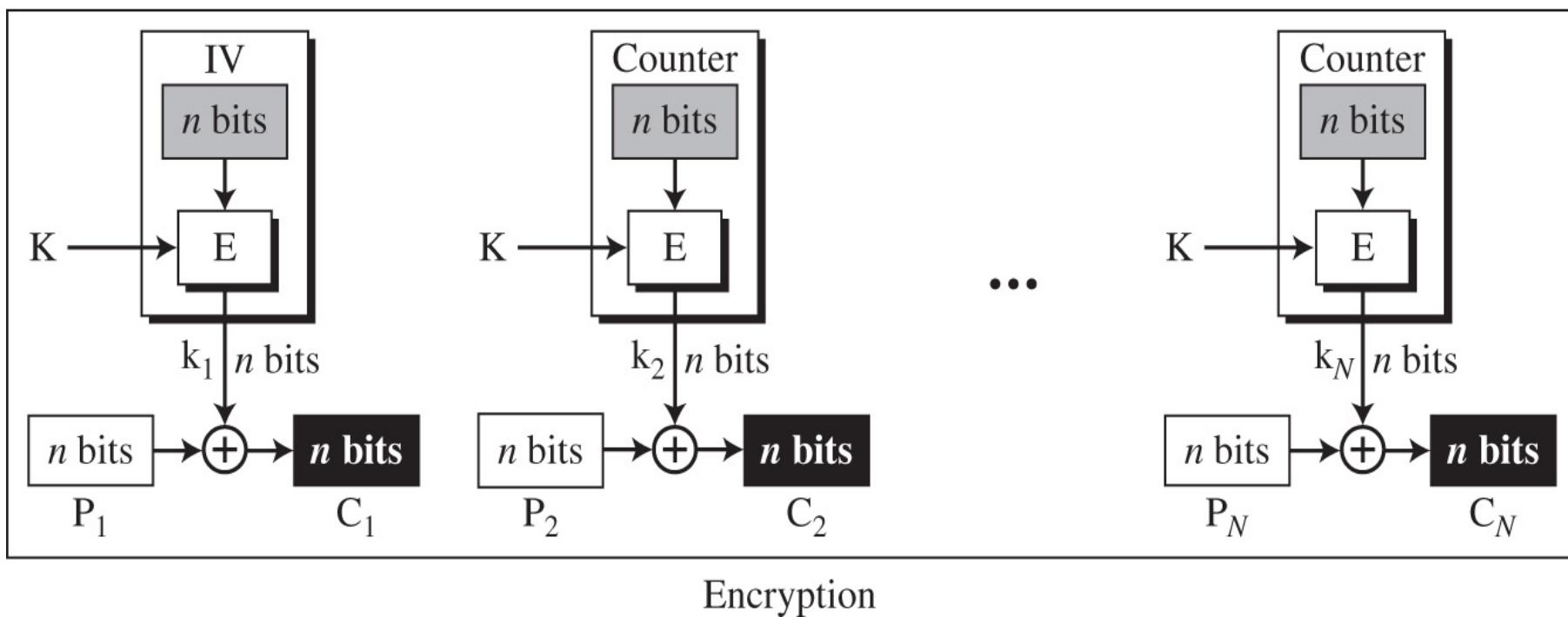
K : Secret key

IV: Initialization vector

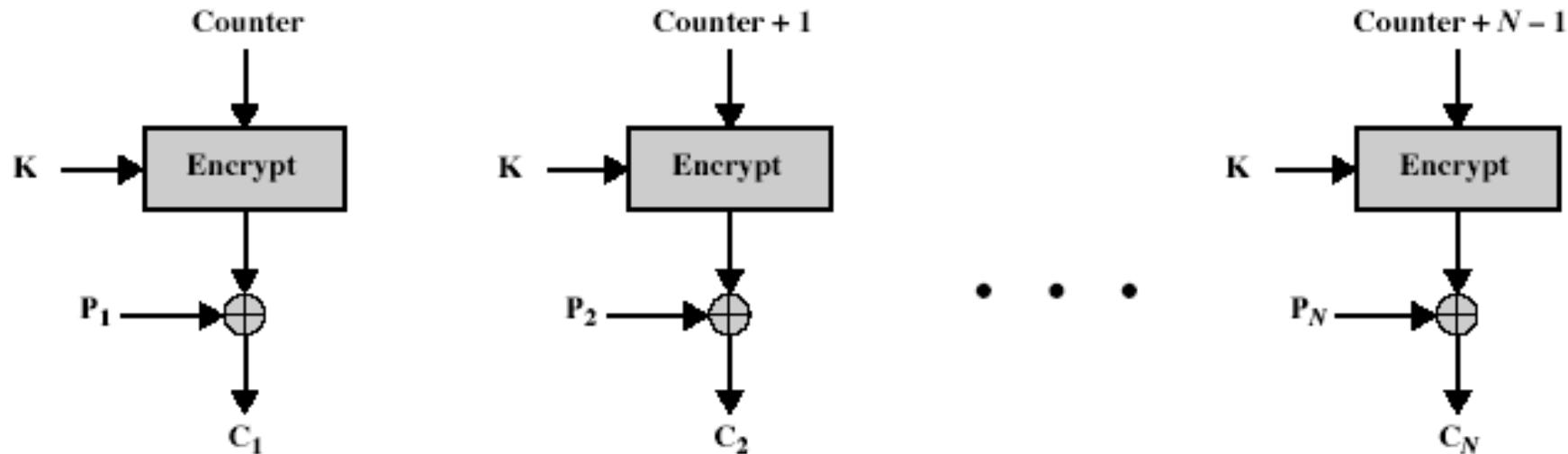
C_i : Ciphertext block i

k_i : Encryption key i

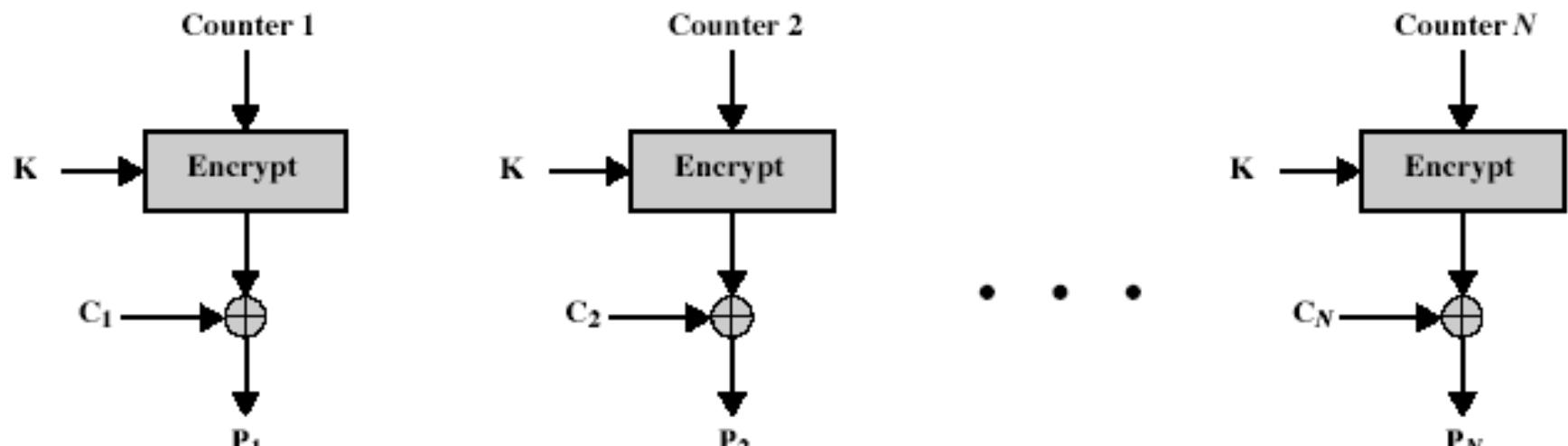
The counter is incremented for each block.



Counter Mode

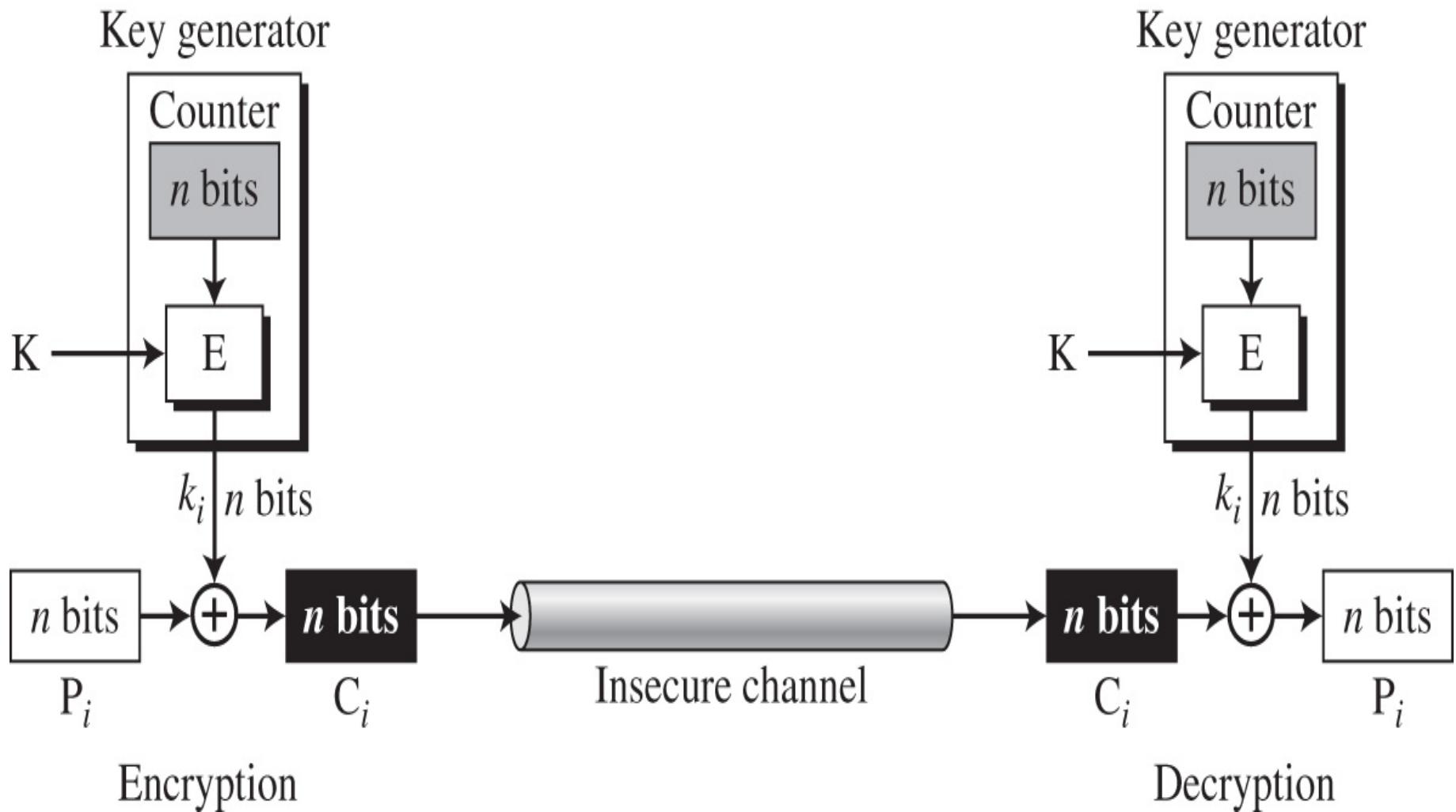


(a) Encryption



(b) Decryption

Counter Mode (CTR)



Comparison of Different Modes

Operation Mode	Description	Type of Results	Data Unit Size
ECB	Each b-bit block is encrypted independently with the same Cipher Key.	Block Cipher	n
CBC	Same as ECB, but each block is first exclusive ORed with the previous Cipher Text.	Block Cipher	n
CFB	Each r-bit block is exclusive-ORed with an r-bit key, which is part of previous cipher text.	Stream Cipher	$r \leq n$
OFB	Same as CFB, but the shift register is updated by the previous r-bit key.	Stream Cipher	$r \leq n$
CTR	Same OFB, but a counter is used instead of a shift register.	Stream Cipher	n

What are SSL and TLS?

- SSL – Secure Socket Layer
- TLS – Transport Layer Security
- both provide a secure transport connection between applications (e.g., a web server and a browser)
- SSL was developed by Netscape
- SSL version 3.0 has been implemented in many web browsers (e.g., Netscape Navigator and MS Internet Explorer) and web servers and widely used on the Internet
- SSL v3.0 was specified in an Internet Draft (1996)
- it evolved into TLS specified in RFC 2246
- TLS can be viewed as SSL v3.1

The Secure Socket Layer (SSL) Protocol

- SSL was originally designed to primarily protect HTTP sessions:
 - In the early 1990's there was a similar protocol called S-HTTP
 - However, as S-HTTP capable browsers were not free of charge and SSL version 2.0 was included in browsers of Netscape Communications, it quickly became predominant
 - SSL v.2 contained some flaws and so Microsoft Corporation developed a competing protocol called Private Communication Technology (PCT)
 - Netscape improved the protocol and SSL v.3 became the de-facto standard protocol for securing HTTP traffic
 - Nevertheless, SSL can be deployed to secure arbitrary applications that run over TCP
 - In 1996 the IETF decided to specify a generic *Transport Layer Security (TLS)* protocol that is based on SSL

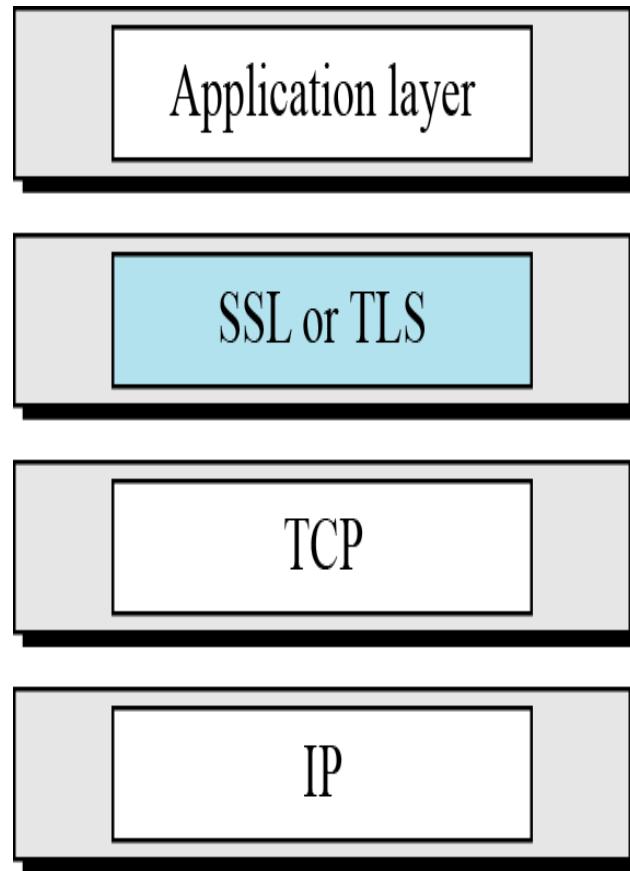
SSL (Secure Socket Layer)

- 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

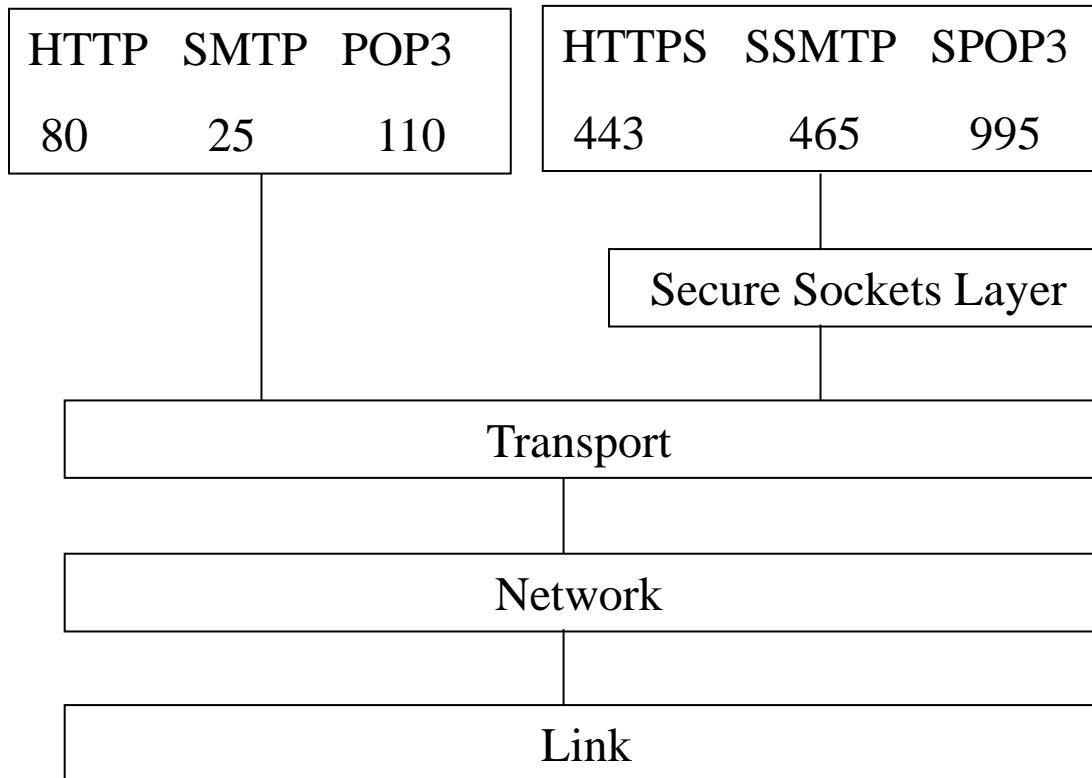
Broad overview

- SSL runs on top of TCP
 - Provides an API similar to that of TCP
- Technically, SSL runs in the application layer
 - Advantage: does not require changes to TCP
- From the programmer's point of view, it is in the transport layer
 - Same API as for TCP
 - Runs only with TCP, not UDP
- Primarily used for HTTP traffic

Location of SSL and TLS in the Internet model



Where SSL Fits



Services Provided by SSL

- SSL encrypts data so that no one who intercepts is able to read it.
- SSL can assure a client that they are dealing with the real server they intended to connect to.
- SSL can prevent any unauthorized clients from connecting to the server.

Services

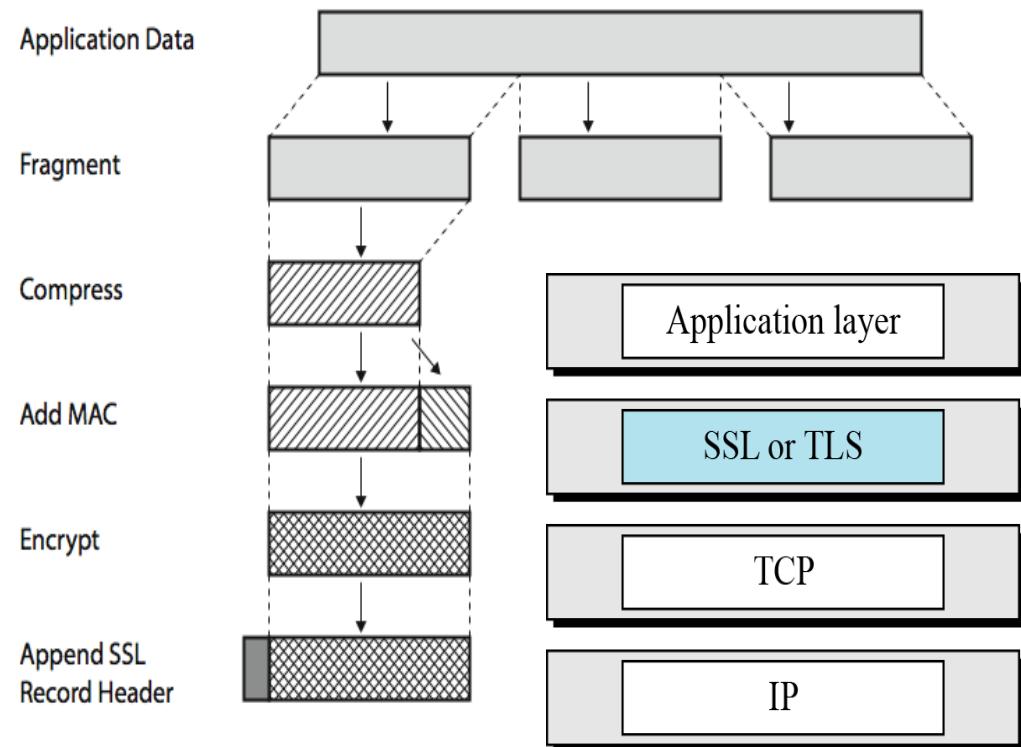
Fragmentation

Compression

Message Integrity

Confidentiality

Framing



SSL architecture

SSL
Handshake
Protocol

SSL Change
Cipher Spec
Protocol

SSL
Alert
Protocol

applications
(e.g., HTTP)

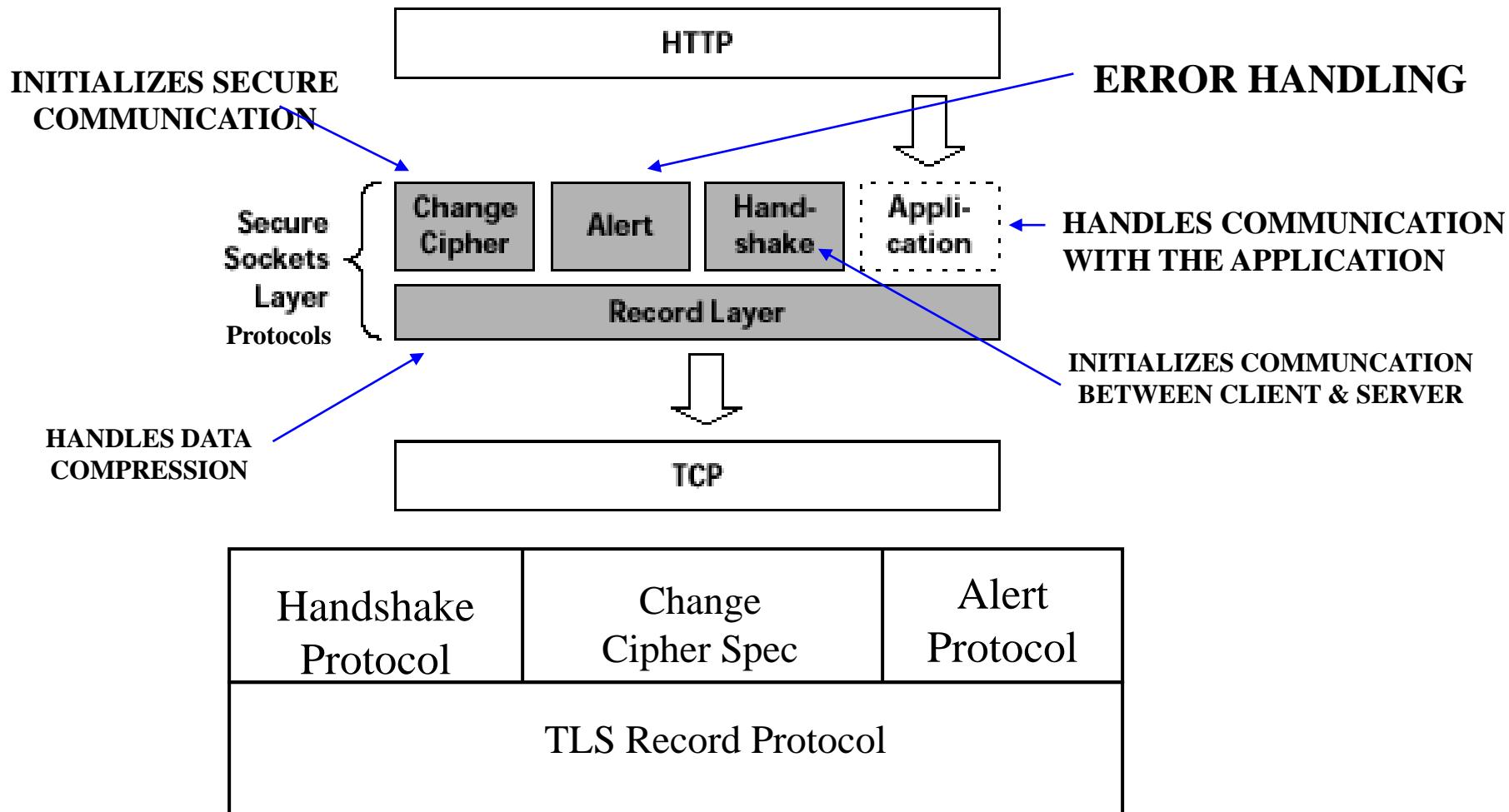
SSL Record Protocol

TCP

IP

Architecture

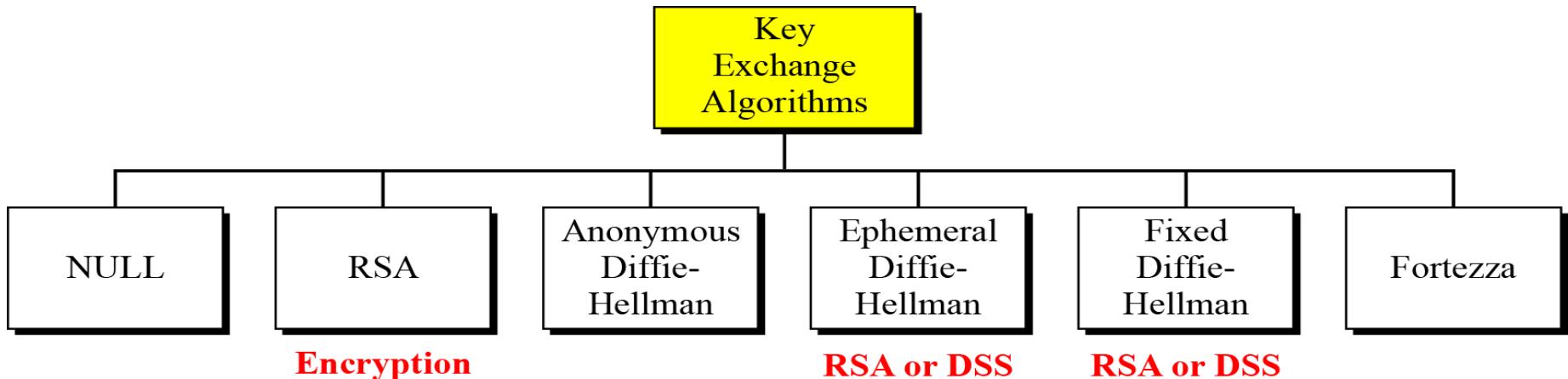
- Record Protocol to transfer application and TLS information
- A session is established using a Handshake Protocol



SSL Components

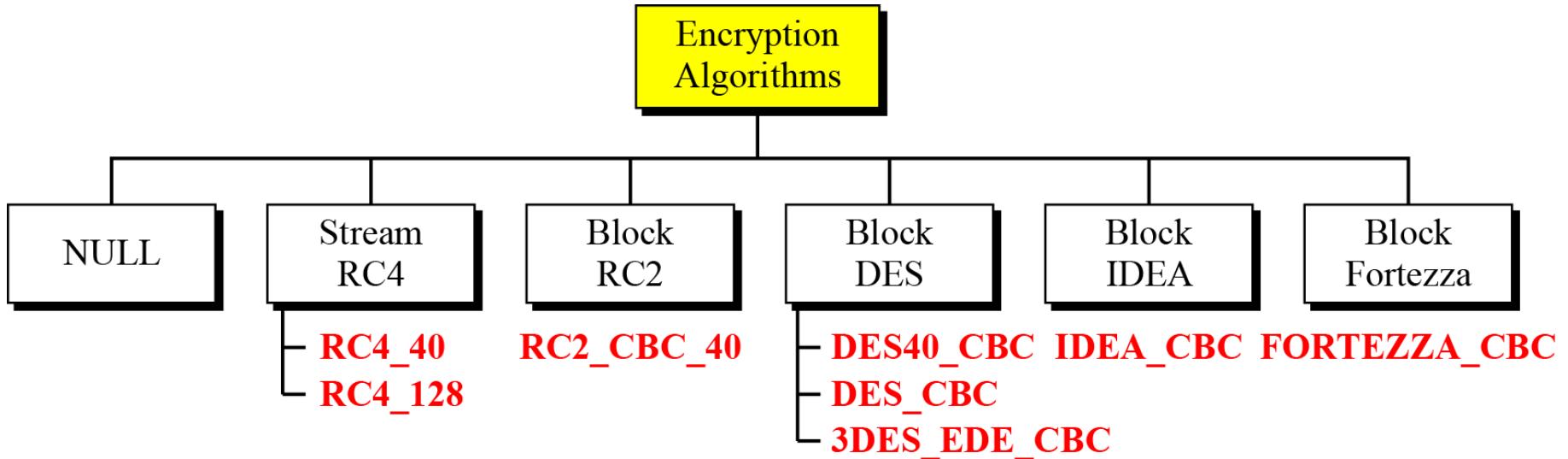
- SSL Handshake Protocol
 - negotiation of security algorithms and parameters
 - key exchange
 - server authentication and optionally client authentication
- SSL Record Protocol
 - fragmentation
 - compression
 - message authentication and integrity protection
 - encryption
- SSL Alert Protocol
 - error messages (fatal alerts and warnings)
- SSL Change Cipher Spec Protocol
 - a single message that indicates the end of the SSL handshake

Key Exchange Algorithms



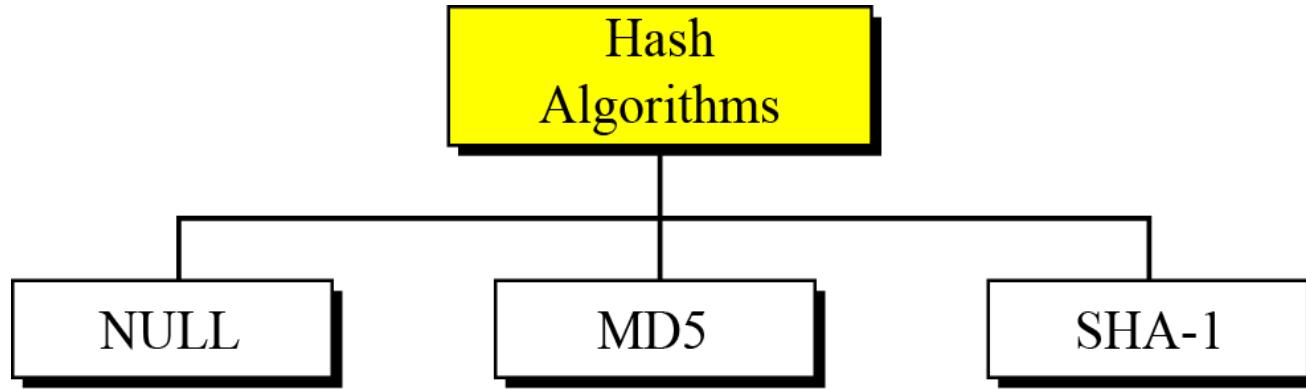
- To Exchange an authenticated and confidential message, the client and server each need six cryptographic secrets (Four keys and two initialization vector).
- To create these secrets, one pre-master secret must be established between the two parties.
- SSL defines six key-exchange methods to establish this pre-master secret

Encryption/Decryption Algorithms



- There are several choices for the Encryption/Decryption Algorithm. These are divided into six group as shown in figure.
- All block protocols use an 8-byte initialization vector(IV) except for Fortezza which used 20-byte IV.

Hash Algorithms for Message Integrity



NULL

The two parties may decline to use an algorithm. In this case, there is no hash function and the message is not authenticated.

MD5

The two parties may choose MD5 as the hash algorithm. In this case, a 128-bit MD5 hash algorithm is used.

SHA-1

The two parties may choose SHA as the hash algorithm. In this case, a 160-bit SHA-1 hash algorithm is used.

Cipher Suite

- The combination of key exchange, hash, and encryption algorithms defines a cipher suite for each SSL session.
- Each suite starts with the term SSL followed by the key exchange algorithm.
- The word “WITH” separates the key exchange algorithm from the encryption and hash algorithm.

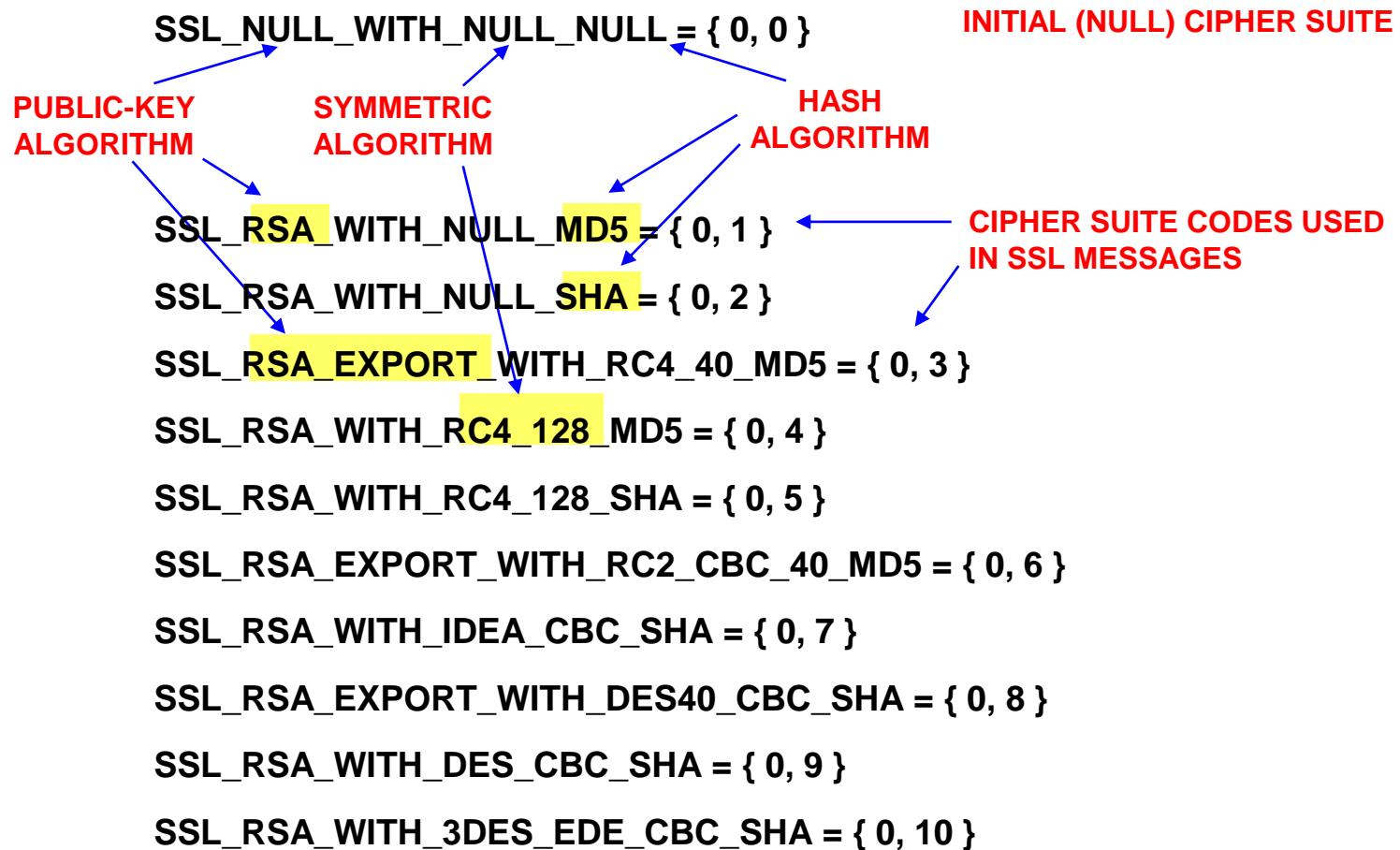
SSL_DHE_RSA_WITH_DES_CBC_SHA

- DHE-RSA (Ephemeral Diffie-Hellman with RSA digital Signature.)
- DH is fixed Diffie-Hellman
- DHE is Ephemeral Diffie-Hellman
- DH-anon is anonymous Diffie-Hellman

SSL Cipher Suite List

<i>Cipher suite</i>	<i>Key Exchange</i>	<i>Encryption</i>	<i>Hash</i>
SSL_NULL_WITH_NULL_NULL	NULL	NULL	NULL
SSL_RSA_WITH_NULL_MD5	RSA	NULL	MD5
SSL_RSA_WITH_NULL_SHA	RSA	NULL	SHA-1
SSL_RSA_WITH_RC4_128_MD5	RSA	RC4	MD5
SSL_RSA_WITH_RC4_128_SHA	RSA	RC4	SHA-1
SSL_RSA_WITH_IDEA_CBC_SHA	RSA	IDEA	SHA-1
SSL_RSA_WITH DES_CBC_SHA	RSA	DES	SHA-1
SSL_RSA_WITH_3DES_EDE_CBC_SHA	RSA	3DES	SHA-1
SSL_DH_anon_WITH_RC4_128_MD5	DH_anon	RC4	MD5
SSL_DH_anon_WITH DES_CBC_SHA	DH_anon	DES	SHA-1
SSL_DH_anon_WITH_3DES_EDE_CBC_SHA	DH_anon	3DES	SHA-1
SSL_DHE_RSA_WITH DES_CBC_SHA	DHE_RSA	DES	SHA-1
SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA	DHE_RSA	3DES	SHA-1
SSL_DHE_DSS_WITH DES_CBC_SHA	DHE_DSS	DES	SHA-1
SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA	DHE_DSS	3DES	SHA-1
SSL_DH_RSA_WITH DES_CBC_SHA	DH_RSA	DES	SHA-1
SSL_DH_RSA_WITH_3DES_EDE_CBC_SHA	DH_RSA	3DES	SHA-1
SSL_DH_DSS_WITH DES_CBC_SHA	DH_DSS	DES	SHA-1
SSL_DH_DSS_WITH_3DES_EDE_CBC_SHA	DH_DSS	3DES	SHA-1
SSL_FORTEZZA_DMS_WITH NULL_SHA	Fortezza	NULL	SHA-1
SSL_FORTEZZA_DMS_WITH_FORTEZZA_CBC_SHA	Fortezza	Fortezza	SHA-1
SSL_FORTEZZA_DMS_WITH_RC4_128_SHA	Fortezza	RC4	SHA-1

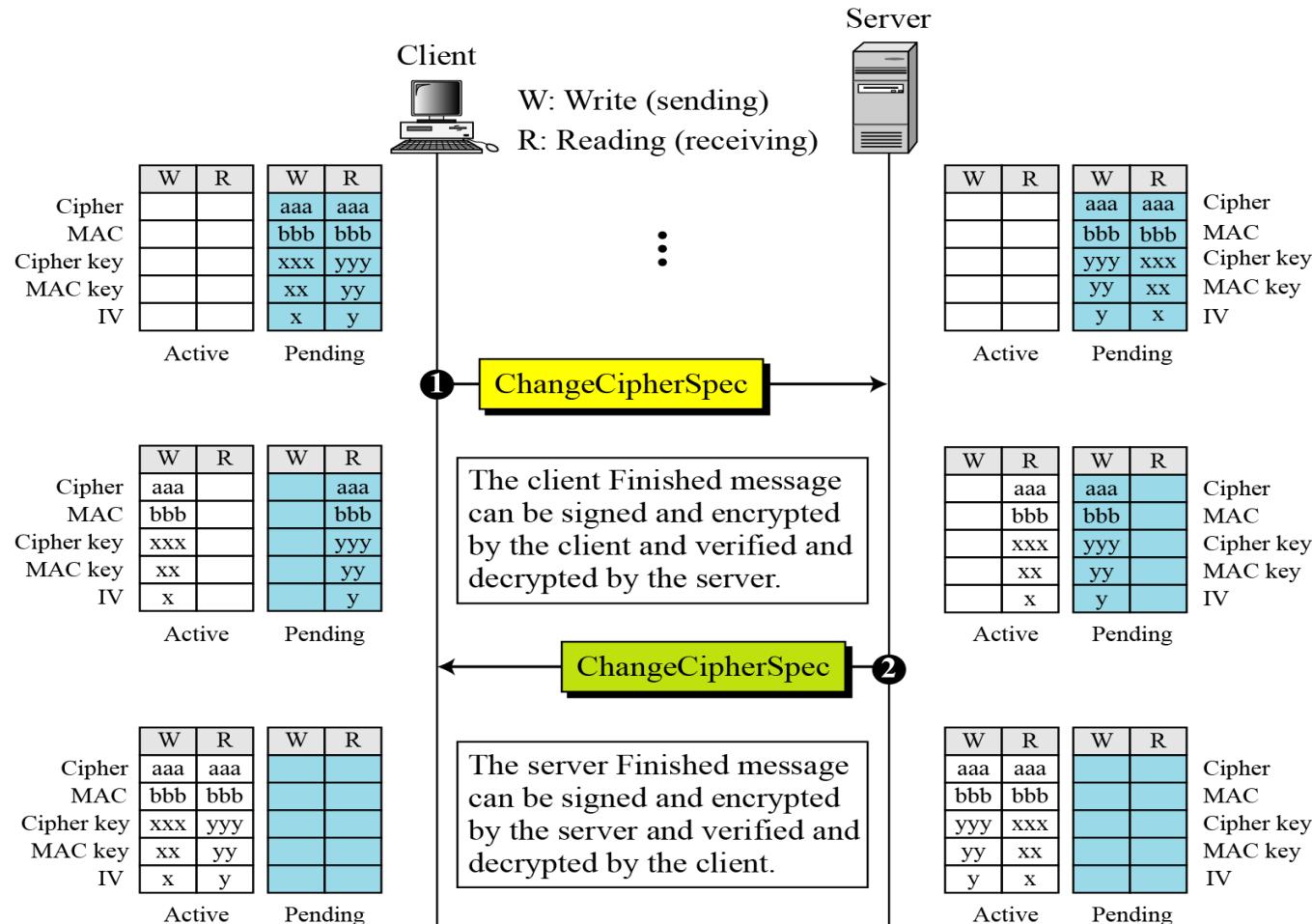
Client Hello - Cipher Suites



Compression Algorithms

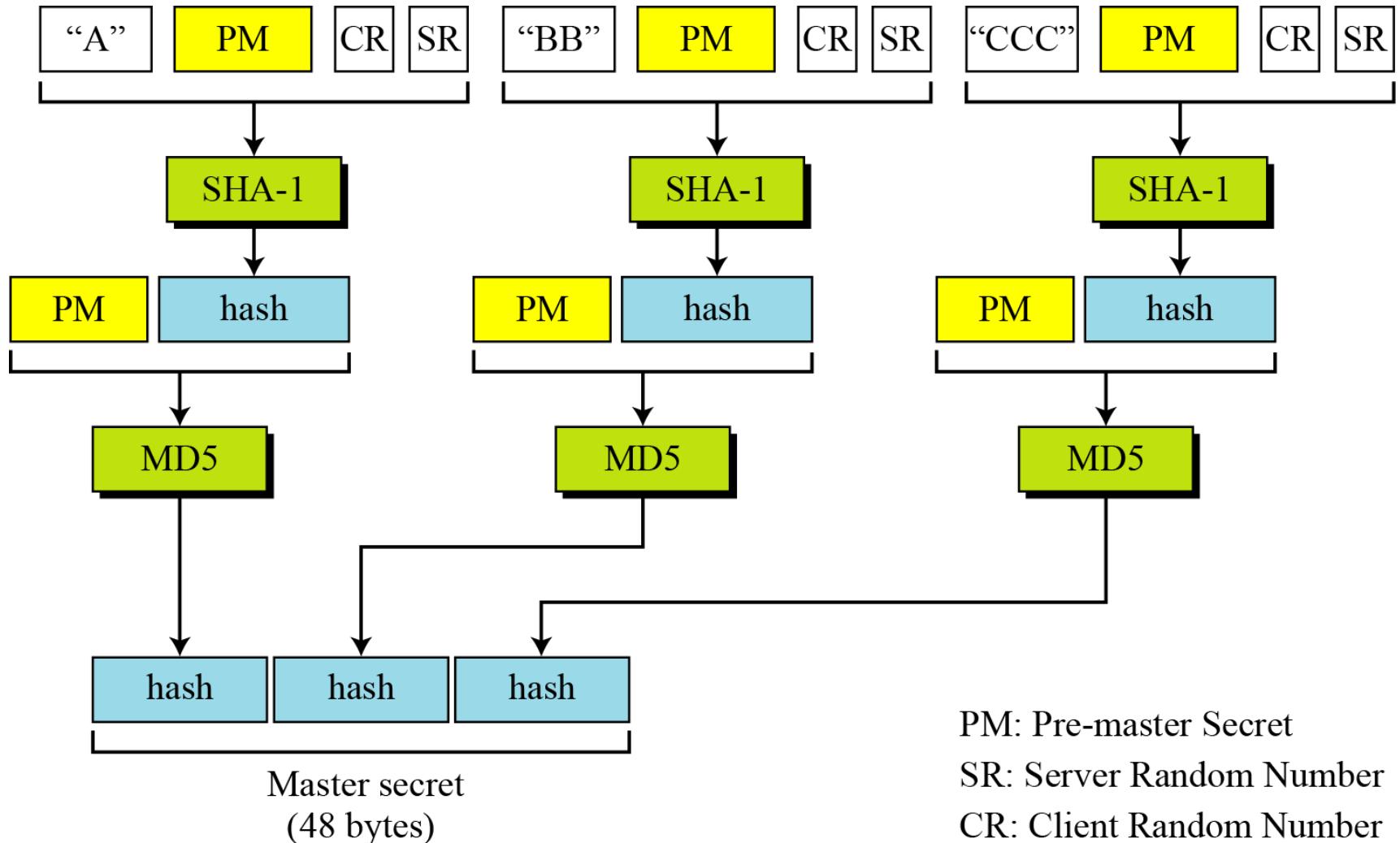
Compression is optional in SSLv3. No specific compression algorithm is defined for SSLv3. Therefore, the default compression method is NULL.

Movement of parameters from pending state to active state

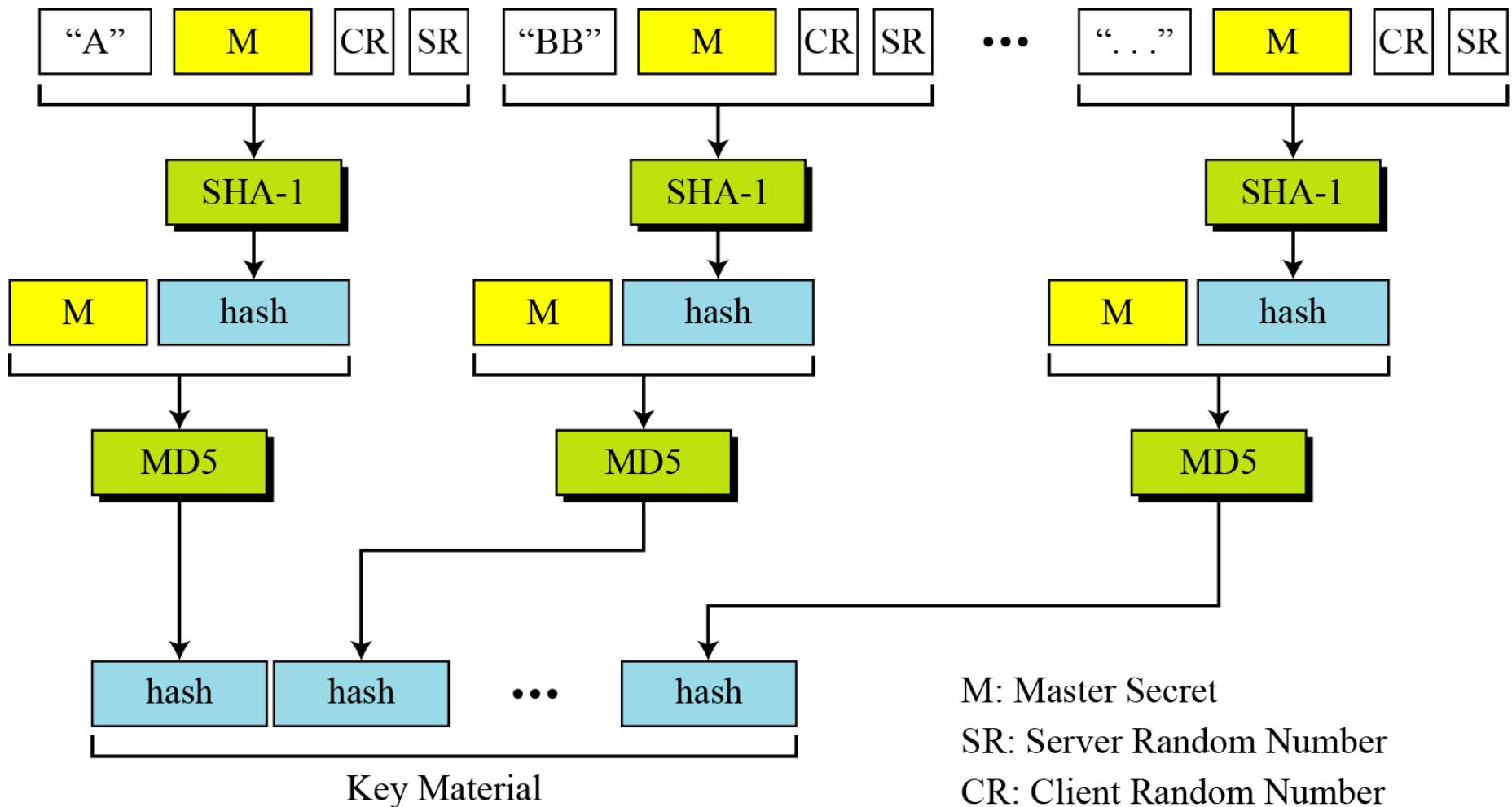


Cryptographic Parameter Generation

Calculation of master secret from pre-master secret



Calculation of key Material from Master Secret

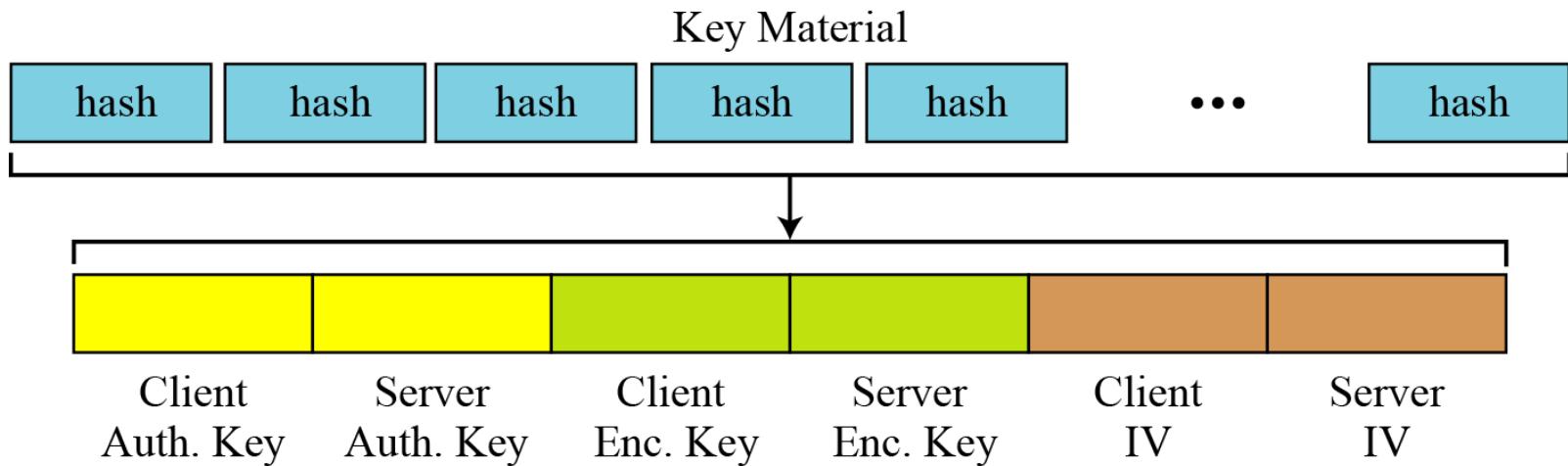


Extractions of Cryptographic Secrets from Key Material

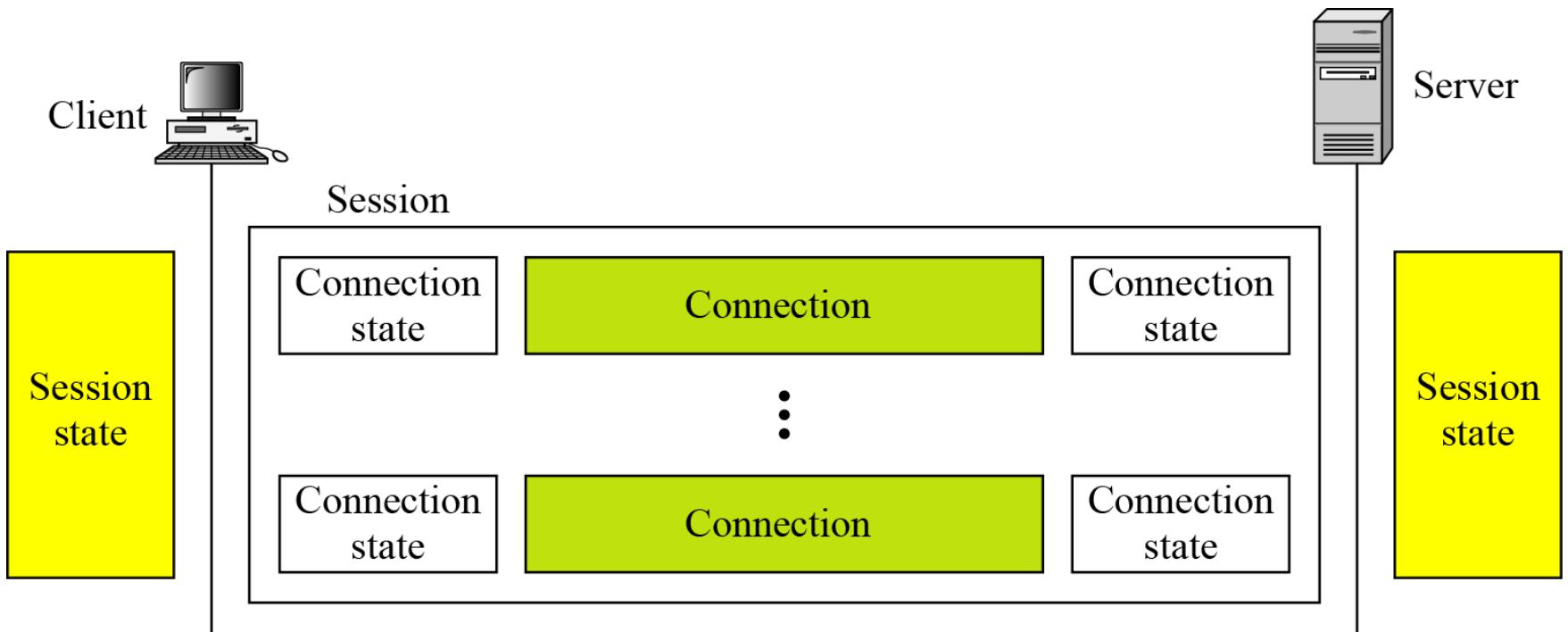
Auth. Key: Authentication Key

Enc. Key: Encryption Key

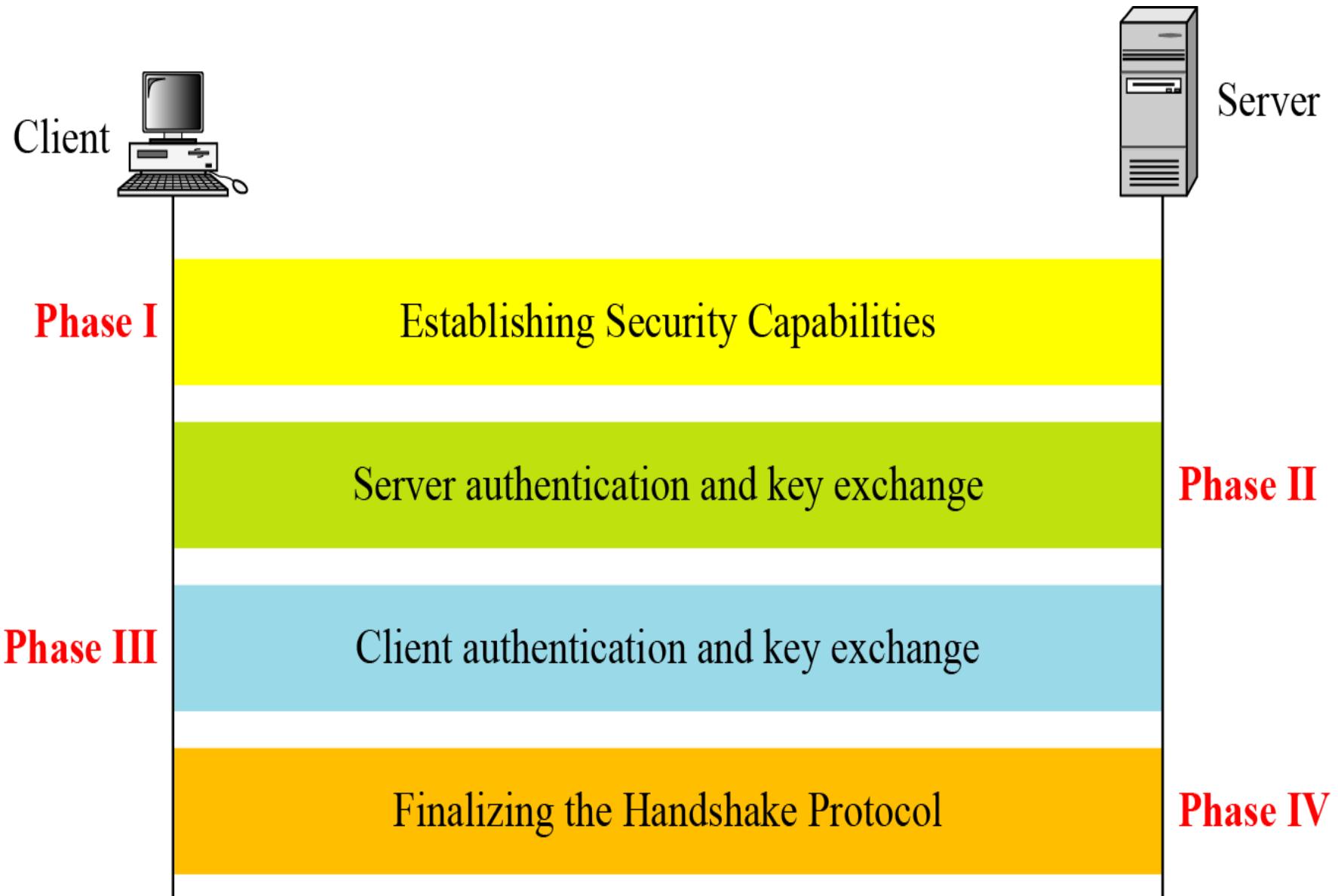
IV: Initialization Vector



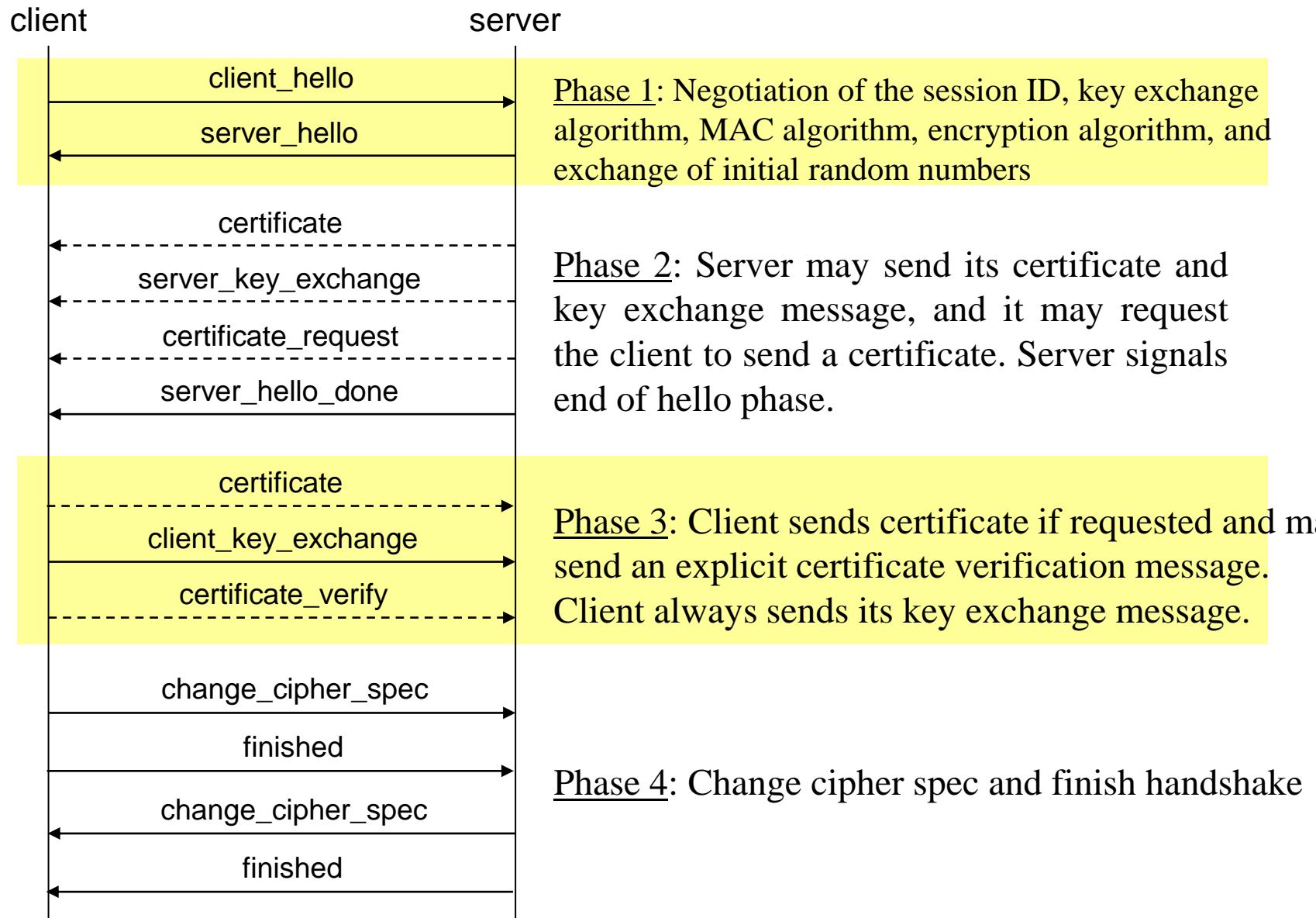
A Session and Connections



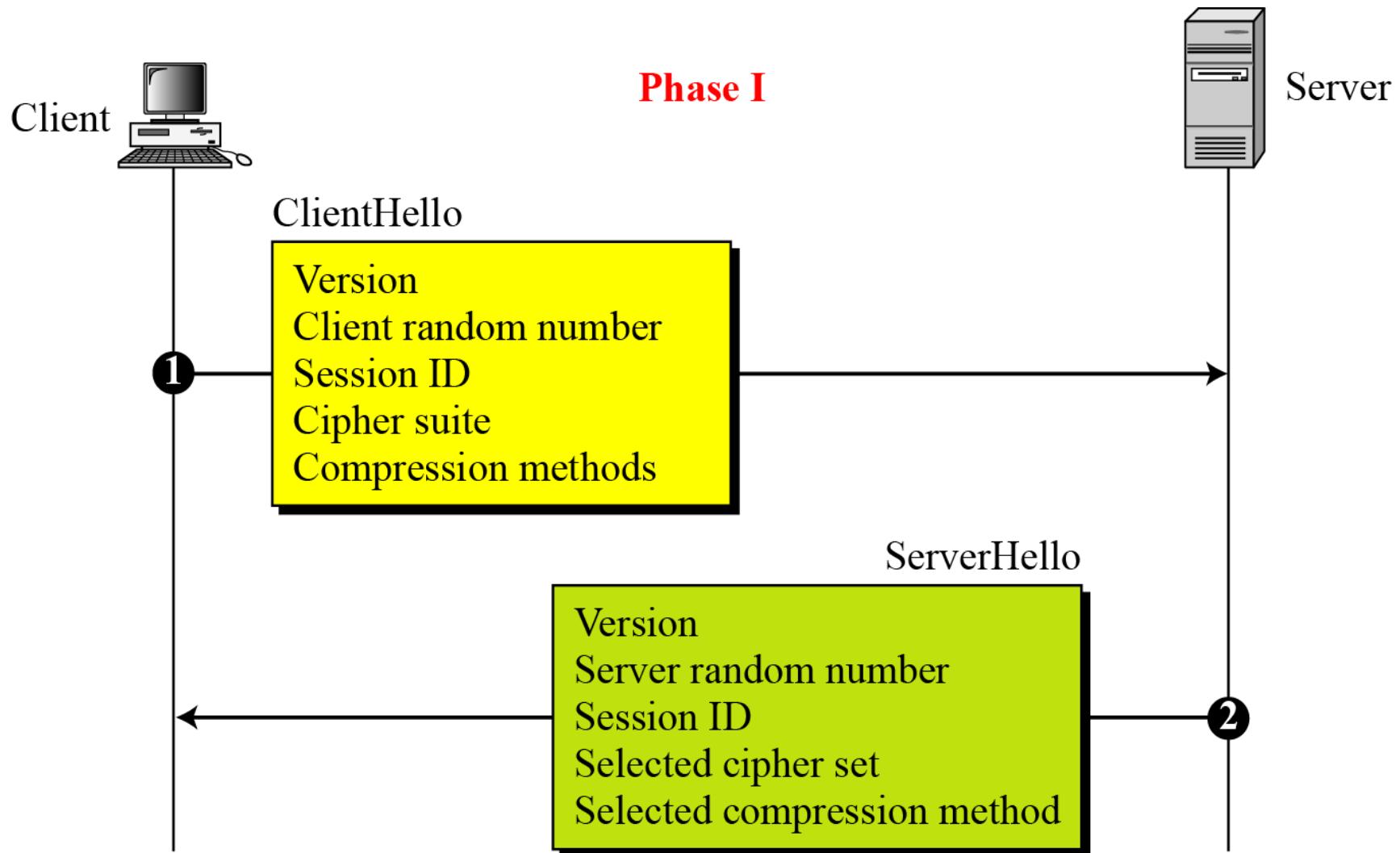
Handshake Protocol



SSL Handshake Protocol – overview



Phase I of Handshake Protocol



Phase I of Handshake Protocol

Client Hello : The client sends the Client Hello Message. It contains the following

- The Highest SSL version number the Client Support.
- A 32-byte random number (from Client) that will be used for Master Secret Generation
- A session ID that defines the session.
- A cipher suite that define the list of algorithm that the client can support.
- A list of compression methods that the client can support.

Phase I of Handshake Protocol

Server Hello : Server responds to the client with a ServerHello message. It contains the following

- SSL version number. This number is the lower of two version numbers. The highest supported by the Client and the highest supported by the Server.
- A 32-byte random number (from Server) that will be used for Master Secret Generation
- A session ID that defines the session.
- Selected cipher set from the client list.
- Selected compression methods from the client list.

After Phase I of Handshake Protocol

After Phase I, the Client and Server know the following

- The version of SSL
- The algorithm for key exchange, Message Authentication and Encryption
- Compression Method
- The two random number for key Generation

Client Hello

- Protocol version
 - SSLv3(major=3, minor=0)
 - TLS (major=3, minor=1)
- Random Number
 - 32 bytes
 - First 4 bytes, time of the day in seconds, other 28 bytes random
 - Prevents replay attack
- Session ID
 - 32 bytes – indicates the use of previous cryptographic material
- Compression algorithm

Handshake Phase 1 – Establish Security Capabilities

- Client Hello (a list of client's preferences)
 - version: highest SSL version supported by client
 - client's random
 - also includes a timestamp
 - against replay attacks
 - session ID
 - nonzero means client wants to use an existing session state for a new connection state (abbreviated handshake);
 - zero means new connection on a new session
 - compression methods supported by client
 - Cipher Suite
 - a list that contains the combination of crypto algorithms supported by the client in the order of preference
 - each entry is a key exchange algorithm and a cipher spec

Handshake Phase 1 – Establish Security Capabilities

- Server Hello (response to client's requests)
 - version: version proposed by client if also supported by server; otherwise highest supported by server
 - server's random
 - same structure as client's, but independent of client's random
 - session ID
 - if client offered one and it is also supported by server, then the same ID (abbreviated handshake)
 - otherwise a new ID assigned by server
 - compression methods chosen from the client's list
 - Cipher Suite selected from the client's list

After Phase I of Handshake Protocol

After Phase I, the Client and Server know the following

- The version of SSL
- The algorithm for key exchange, Message Authentication and Encryption
- Compression Method
- The two random number for key Generation

Handshake Protocol

- The most complex part of SSL
- Allows server and client:
 - to authenticate each other
 - to negotiate encryption and MAC algorithms
 - to create cryptographic keys to be used
 - in general, to establish a *session* and then a *connection*
- handshake is done before any data is transmitted
 - so cannot assume a secure record protocol

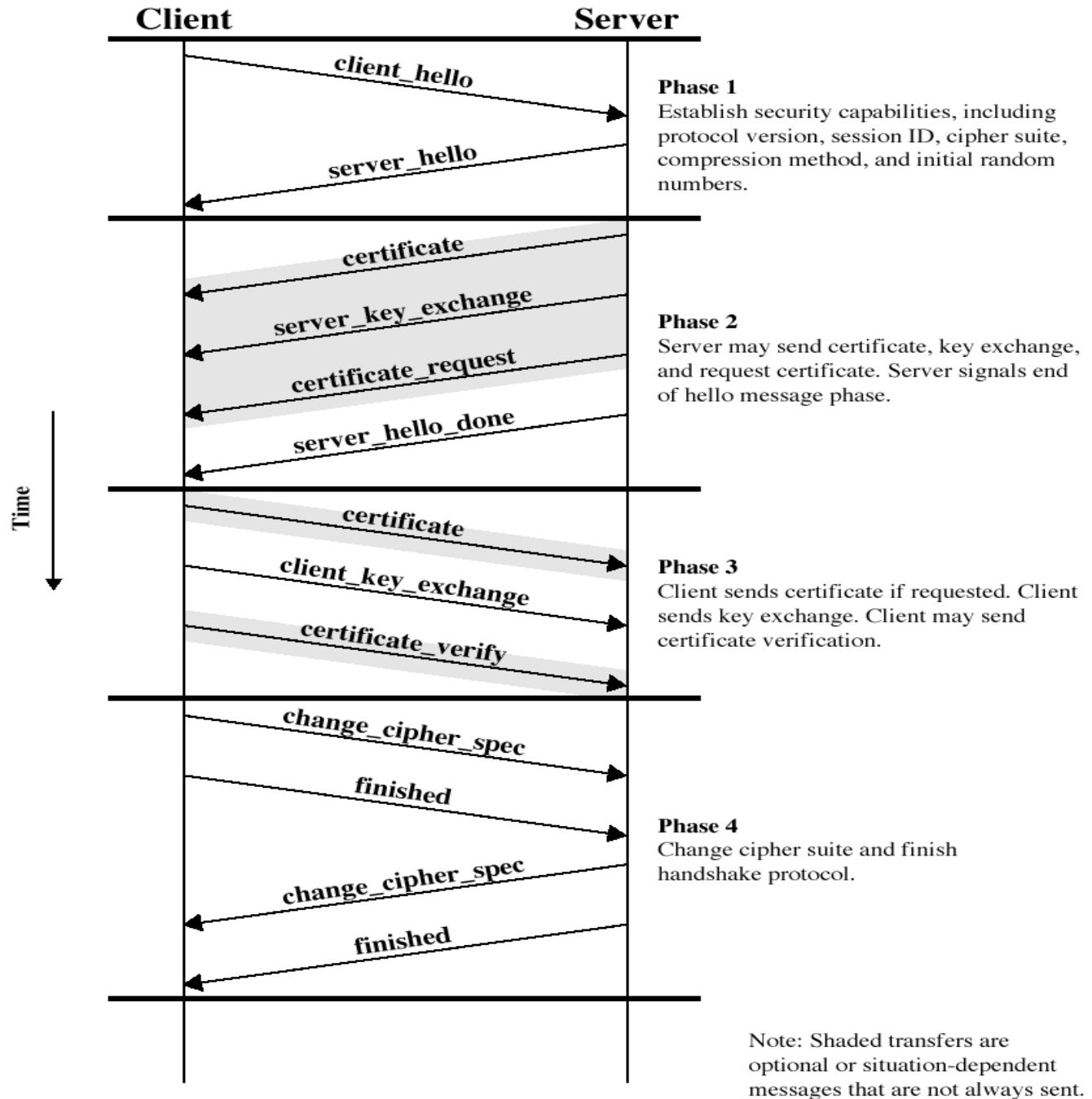
Handshake Protocol

- A series of messages in 4 phases
 1. Establish Security Capabilities
 2. Server Authentication and Key Exchange
 3. Client Authentication and Key Exchange
 4. Finish
- Handshake message format
- Message types

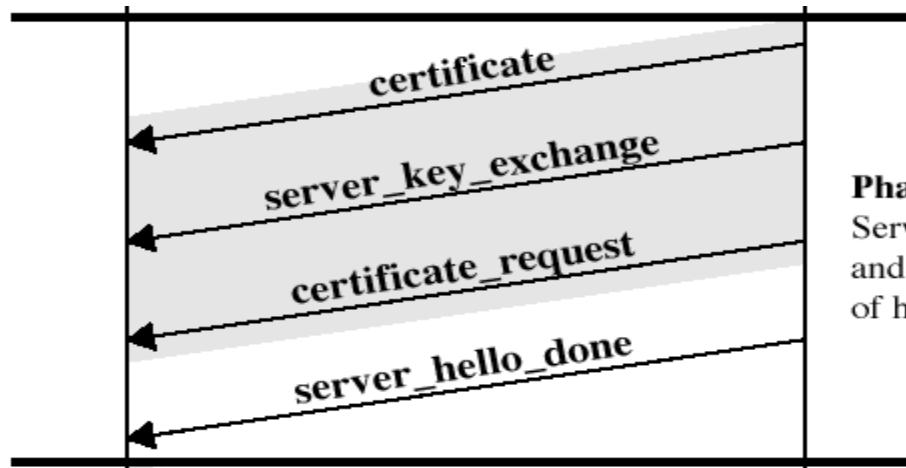
1 byte	3 bytes	≥ 0 bytes
Type	Length	Content

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

Handshake Protocol



Handshake Phase 2: Server Auth. and Key Exchange

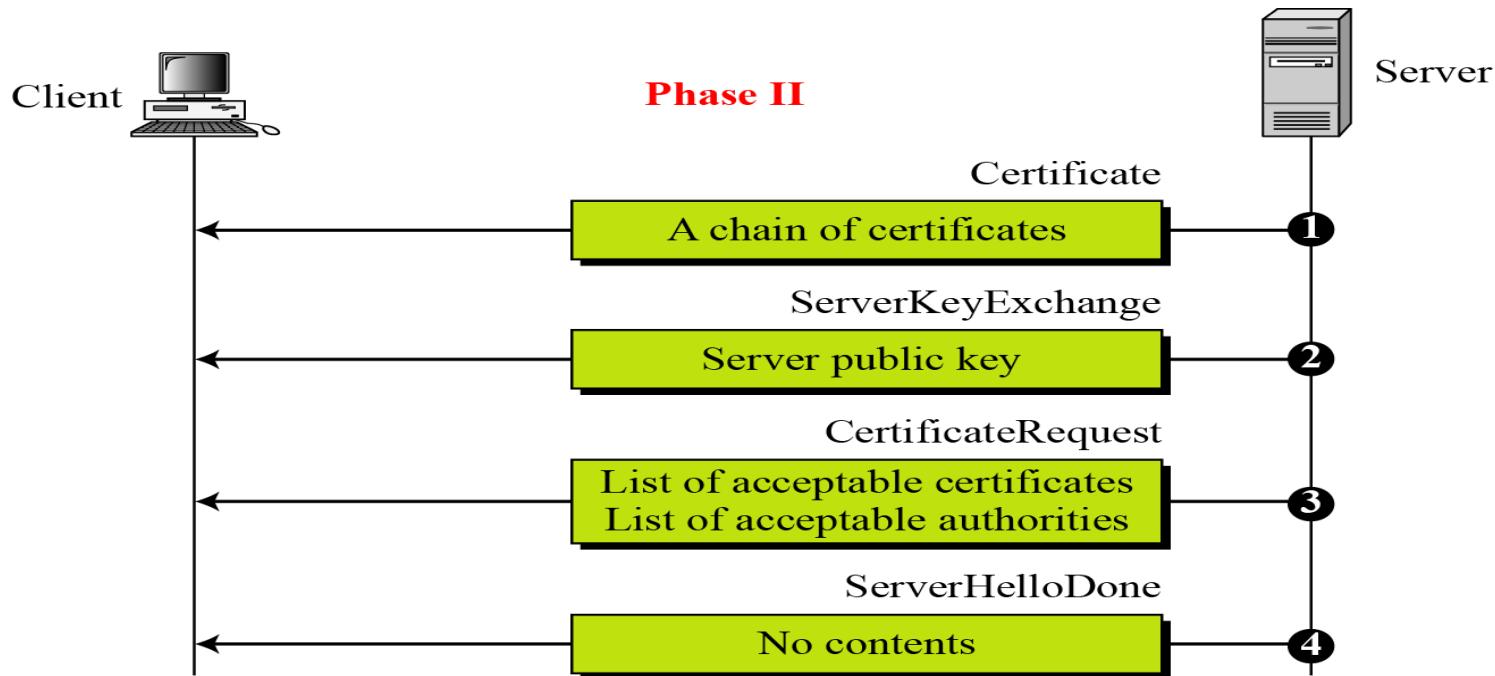


Phase 2

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

- Certificate is needed except for anon-DH (anon-DH is not used most of the time)
 - Certificate is the basis for server authentication
 - if fixed DH, then certificate contains enough information for key exchange (so server key exchange message is not needed)

Phase II of Handshake Protocol



In Phase-II, the server authenticates itself if needed.

- Sender may send its certificate, its public key and may also request certificates from the client.
- At the end server announced that the server hello process is done.

Handshake Phase 2: Server Auth. and Key Exchange

Certificate :

- If it is required, the server sends a Certificate message to authenticate itself.
- The message includes a list of certificates of type X.509.
- The certificate is not needed if the key-exchange algorithm is anonymous Diffie-Hellman.

ServerKeyExchange:

- After the Certificate message, the server sends a ServerKeyExchange message that includes its contribution to the pre-master secret.
- If this message is not required if the key-exchange method is RSA or fixed Diffie-Helman.

Handshake Phase 2: Server Auth. and Key Exchange

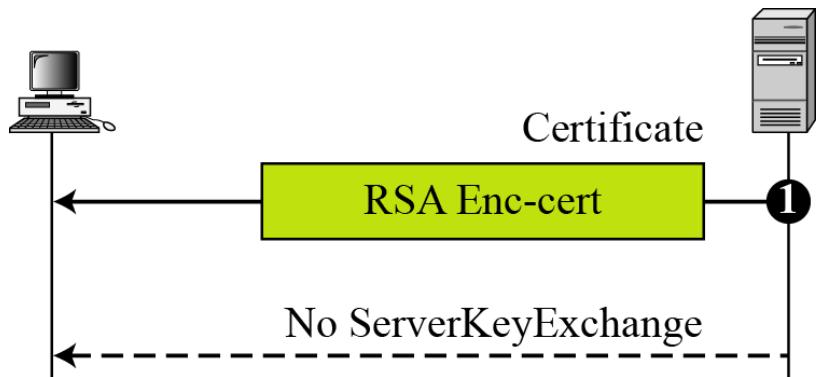
Certificate Request:

- Server may require the client to authenticate itself.
- In this case, the server sends a CertificateRequest message in PhaseII that asks for certification in Phase III from the client.
- The server cannot request a certificate from the client if it is anonymous Diffie-Hellman.

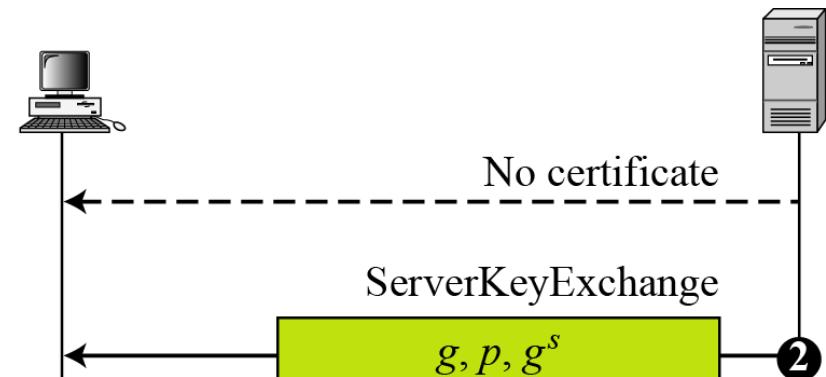
ServerKeyExchange:

- The last message in Phase-II is the ServerHellDone message which is a signal to the client that Phase-II is over and that the client needs to start Phase III.

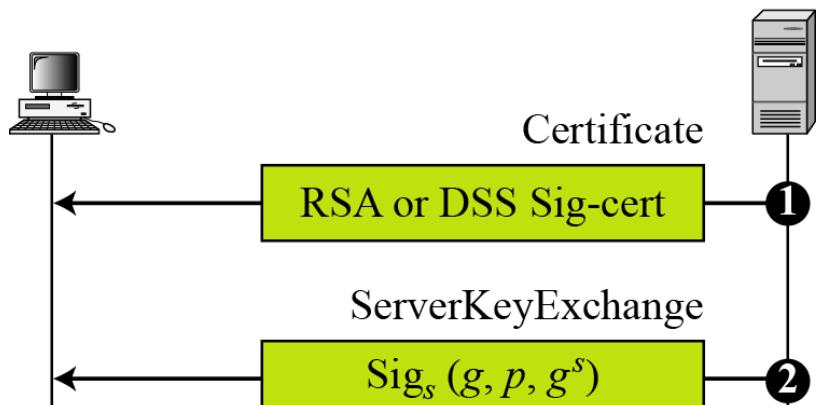
Four cases in Phase II



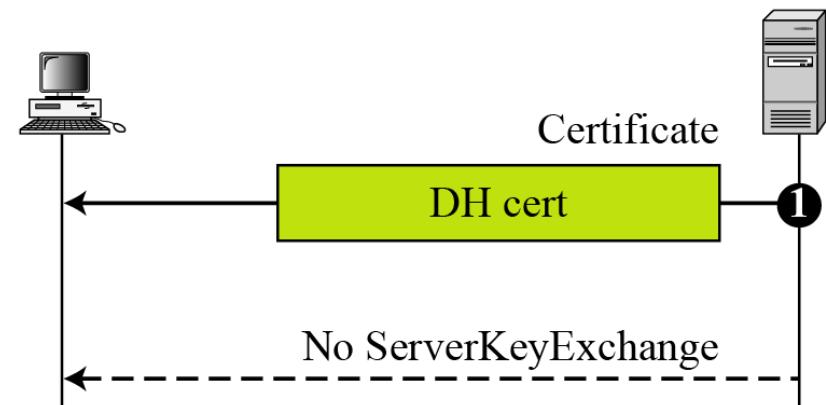
a. RSA



b. Anonymous DH



c. Ephemeral DH



d. Fixed DH

Handshake Phase 2: Server Auth. and Key Exchange

- Server Key Exchange
 - not needed for
 - fixed DH and RSA key exchange
 - message content depends on the key exchange method agreed
 - Anon-DH
 - message contains public DH parameters and server's DH public key
 - Ephemeral DH
 - same as anon-DH plus a signature on them
 - Signatures contain random values (that are exchanged in hello phase) to resist against replay attacks

Handshake Phase 2: Server Auth. and Key Exchange

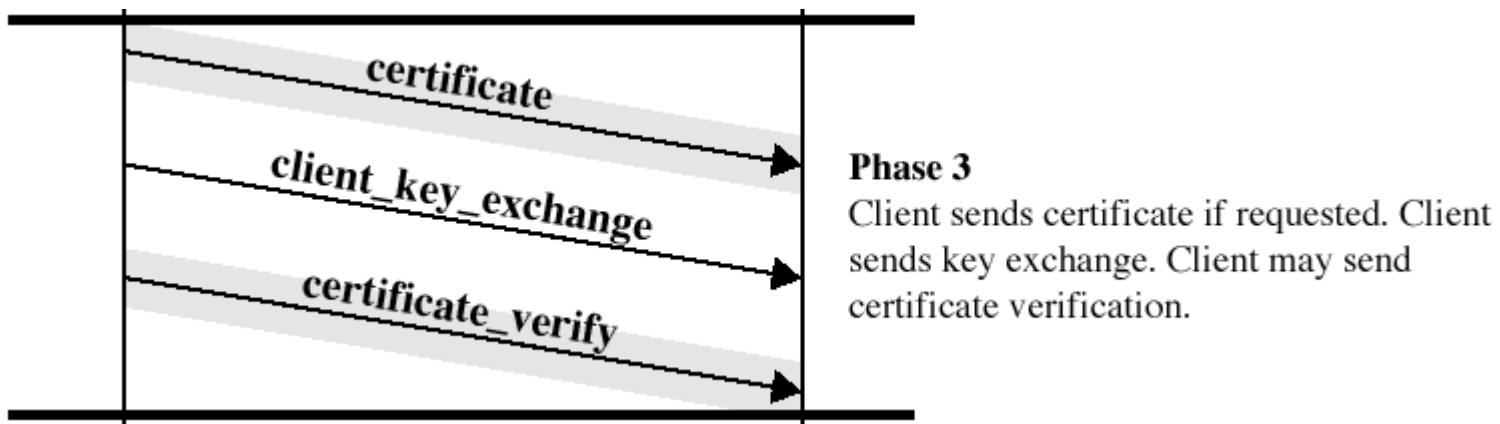
- Server Key Exchange
 - not needed for
 - fixed DH and RSA key exchange
 - message content depends on the key exchange method agreed
 - Anon-DH
 - message contains public DH parameters and server's DH public key
 - Ephemeral DH
 - same as anon-DH plus a signature on them
 - Signatures contain random values (that are exchanged in hello phase) to resist against replay attacks

Handshake Phase 2: Server Auth. and Key Exchange

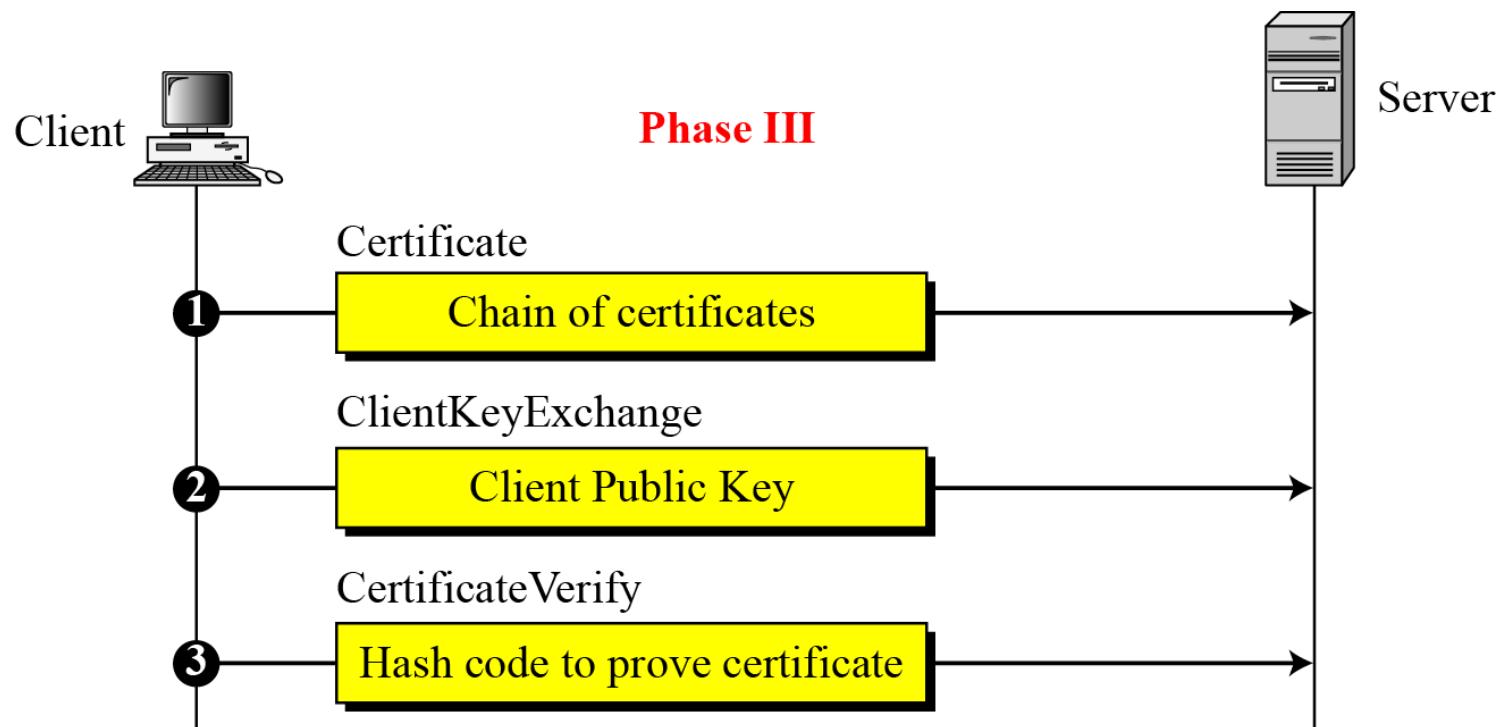
- Certificate Request Message
 - although not common in practice, server may request client to send its certificate
 - to authenticate the client
 - two fields: certificate type and acceptable CAs
 - a list of them
 - Certificate types
 - fixed DH (certificate may be signed with RSA or DSS)
 - ephemeral DH (certificate may contain RSA or DSS key)
 - signature only (not used for key exchange but for auth.)
 - Certificate may contain RSA or DSS key
- Server Hello Done message
 - server is finished and will wait for client's response

Handshake Phase 3: Client Auth. and Key Exchange

- Upon receipt of server hello done
 - client checks the server certificate and server hello parameters
 - after that client starts sending its own messages
- Client's Certificate
 - is sent if requested and available



Phase III of Handshake Protocol

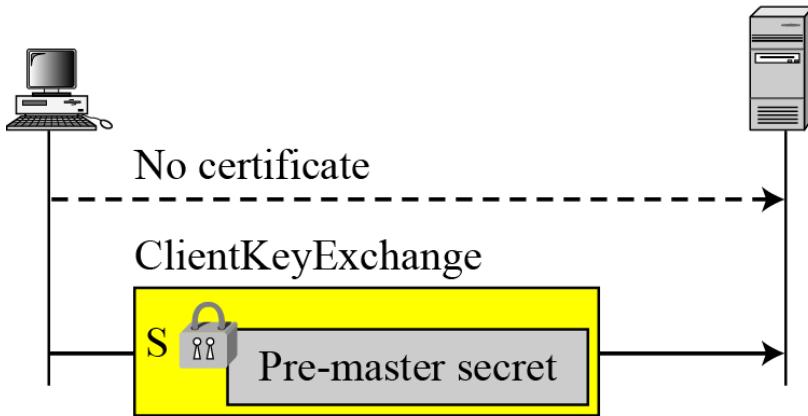


Handshake Phase 3: Client Auth. and Key Exchange

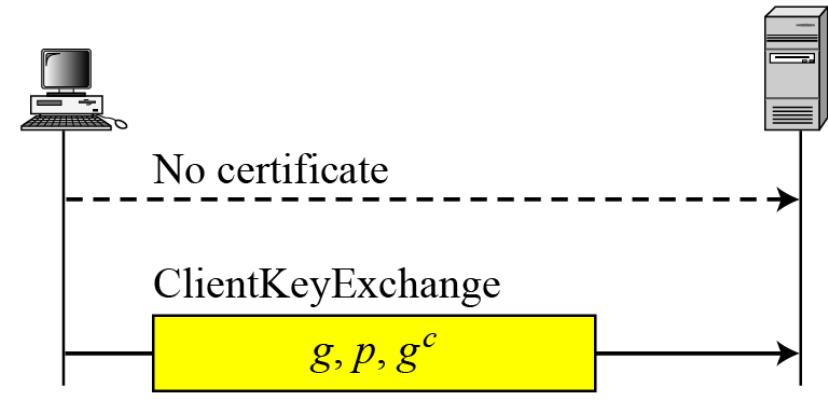
- Client Key exchange message
 - content depends on the key exchange method agreed
 - RSA
 - 48-byte pre-master secret is encrypted using server's RSA key (obtained at phase 2)
 - fixed-DH
 - client DH parameters are in client certificate, so key exchange message is null
 - Anon or ephemeral DH
 - Client DH parameters and DH public key are sent
 - no signature even for ephemeral DH
 - no client authentication and authenticated key exchange so far (only key exchange)

Four cases in Phase III

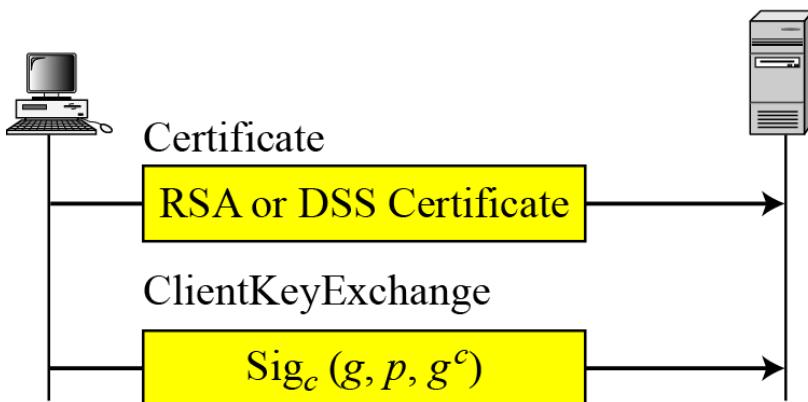
S encrypted with server's public key
 Sig_c : Signed with client's public key



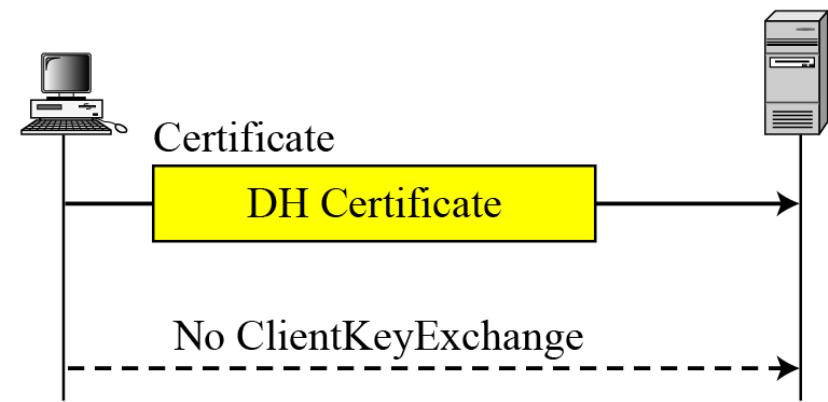
a. RSA



b. Anonymous DH



c. Ephemeral DH



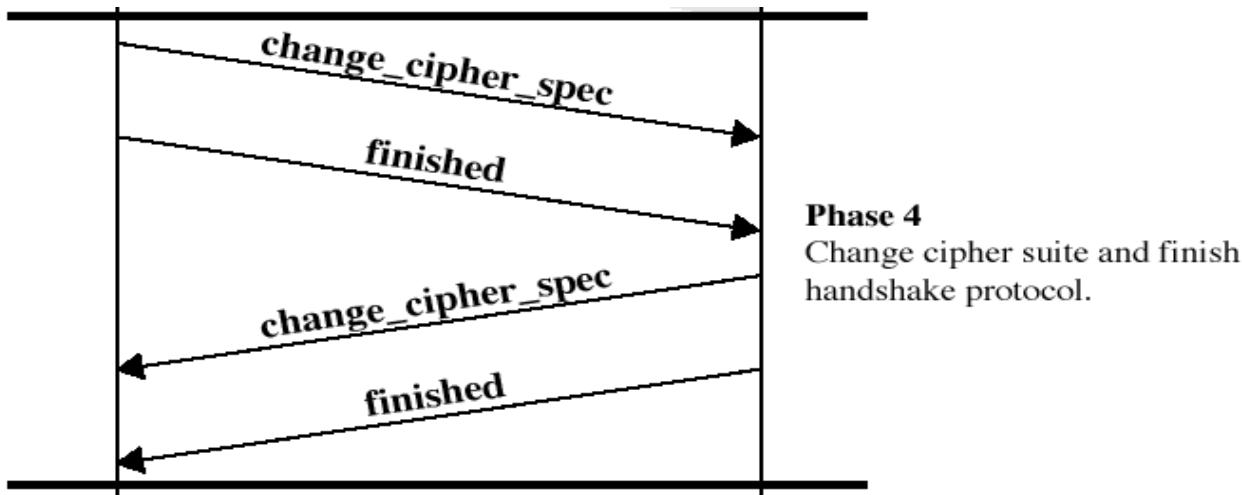
d. Fixed DH

Handshake Phase 3: Client Auth. and Key Exchange

- Certificate Verify message
 - in client key exchange message, the client is not authenticated
 - anyone could send the key exchange message
 - a method for authentication is the certificate verify message
 - client shows ownership of private key that corresponds to the public key in client certificate by signing a hash that contains the master secret and handshake messages
 - except for fixed DH that does not contain a signature key
 - what about authentication for fixed DH case?
 - no real authentication but the attacker cannot produce the pre-master and master secrets since it does not know the DH private key (so there is a kind of an implied auth.)

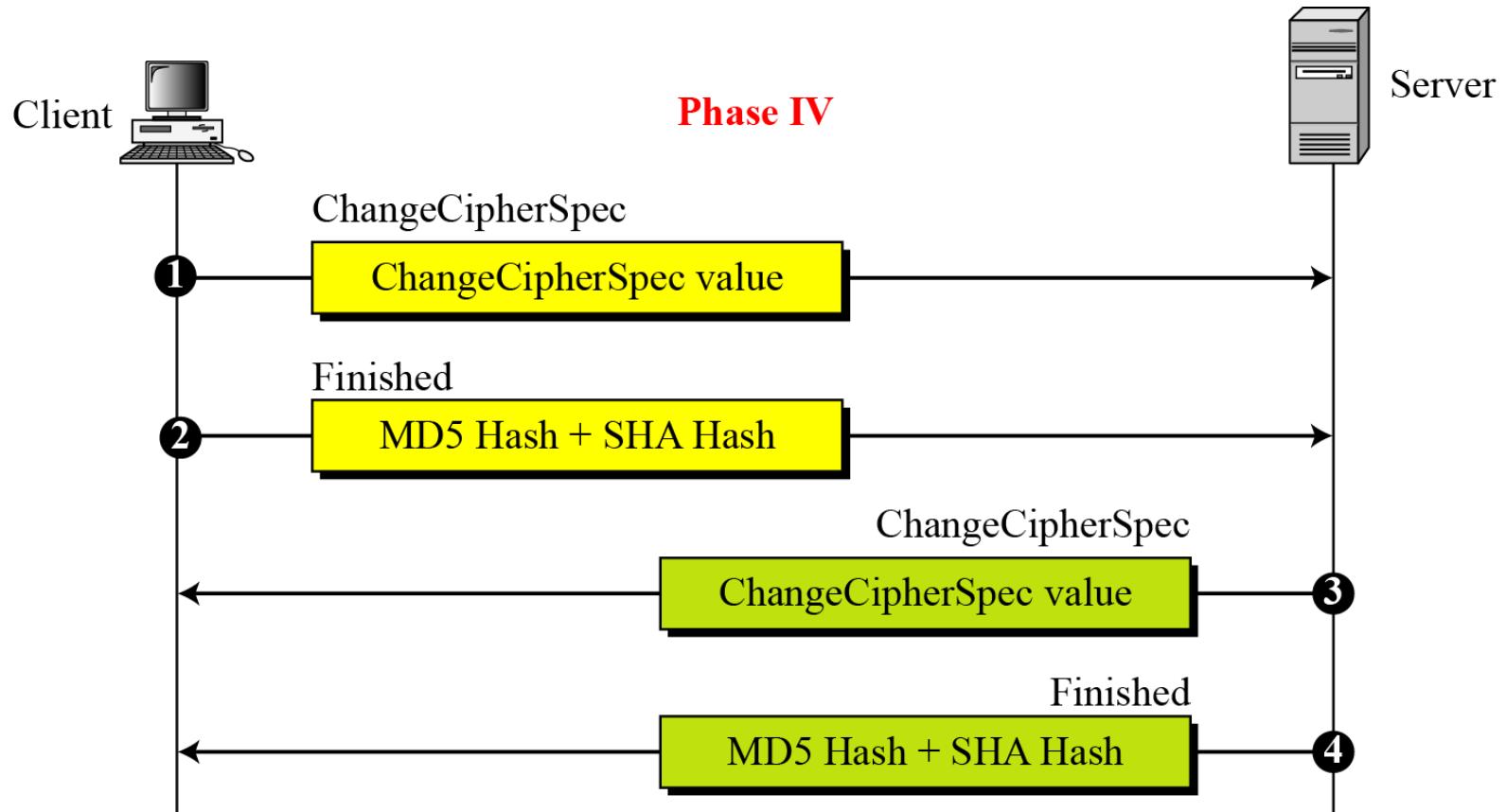
Handshake Phase 4: Finish

- At the end of Phase 3, both client and server can calculate keys
 - Phase 4 is wrap-up phase



- Change cipher spec messages
 - to make the pending cipher spec the current one

Phase IV of Handshake Protocol



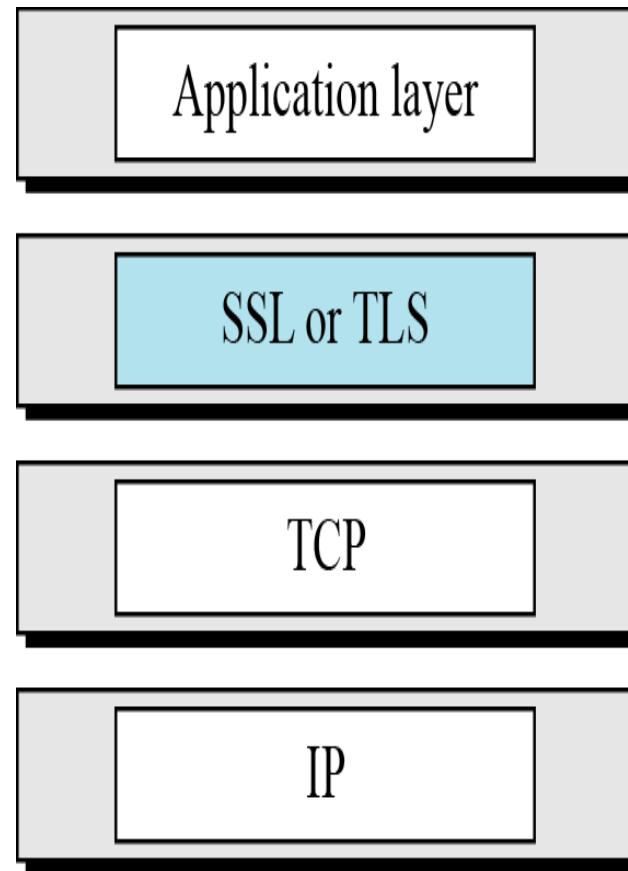
Handshake Phase 4: Finish

- ChangeCipherSpec** : The client sends a ChangeCipherSpec message to show that it has moved all of the cipher suite set and the parameters from pending state to the active state. This message is actually part of the ChangeCipherSpec Protocol.
- Finished** : The next message is also sent by the client. It is finished message that announces the end of the handshaking protocol by the client.

Handshake Phase 4: Finish

- Finished message
 - a MAC on exchanged handshake messages using the master secret
 - to verify that handshake is successful and both parties have the same master secret
 - client's *finished* is verified by server and vice versa
 - the connection state of the record protocol that encrypts and MACs the finished message is the new one
 - so this is also verification of all the keys that are recently created

Location of SSL and TLS in the Internet model



TLS (Transport Layer Security)

- TLS is a proposed Internet Standard (RFC 5246)
 - similar to SSL v3, some differences are given here
- Version number
 - record format is the same, but the major version 3, minor version 3 (v3.3)
- MAC
 - TLS uses HMAC with pads XORed (unlike SSL where pads are appended)
- additional alert codes

TLS (Transport Layer Security)

- Same cipher suites of SSL except Fortezza
 - actually it is not common in SSL v3 either
- No ephemeral client certificates in TLS
 - since signature-only certificates are used for that purpose
- some changes in certificate verify and finished message calculations
- a different Pseudorandom function (PRF) based on HMAC
 - master secret and key block calculations use PRF in TLS

TLS: Version

The first difference is the version number (major and minor). The current version of SSL is 3.0; the current version of TLS is 1.0. In other words, SSLv3.0 is compatible with TLSv1.0.

TLS : Cipher Suite

Another minor difference between SSL and TLS is the lack of support for the Fortezza method. **TLS does not support Fortezza for key exchange or for encryption/decryption.** Below Tables shows the cipher suite list for TLS and SSL.

<i>Cipher suite</i>	<i>Key Exchange</i>	<i>Encryption</i>	<i>Hash</i>
TLS_NULL_WITH_NULL_NULL	NULL	NULL	NULL
TLS_RSA_WITH_NULL_MD5	RSA	NULL	MD5
TLS_RSA_WITH_NULL_SHA	RSA	NULL	SHA-1
TLS_RSA_WITH_RC4_128_MD5	RSA	RC4	MD5
TLS_RSA_WITH_RC4_128_SHA	RSA	RC4	SHA-1
TLS_RSA_WITH_IDEA_CBC_SHA	RSA	IDEA	SHA-1
TLS_RSA_WITH_DES_CBC_SHA	RSA	DES	SHA-1
TLS_RSA_WITH_3DES_EDE_CBC_SHA	RSA	3DES	SHA-1
TLS_DH_anon_WITH_RC4_128_MD5	DH_anon	RC4	MD5
TLS_DH_anon_WITH_DES_CBC_SHA	DH_anon	DES	SHA-1
TLS_DH_anon_WITH_3DES_EDE_CBC_SHA	DH_anon	3DES	SHA-1
TLS_DHE_RSA_WITH_DES_CBC_SHA	DHE_RSA	DES	SHA-1
TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA	DHE_RSA	3DES	SHA-1
TLS_DHE_DSS_WITH_DES_CBC_SHA	DHE_DSS	DES	SHA-1
TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA	DHE_DSS	3DES	SHA-1
TLS_DH_RSA_WITH_DES_CBC_SHA	DH_RSA	DES	SHA-1
TLS_DH_RSA_WITH_3DES_EDE_CBC_SHA	DH_RSA	3DES	SHA-1
TLS_DH_DSS_WITH_DES_CBC_SHA	DH_DSS	DES	SHA-1
TLS_DH_DSS_WITH_3DES_EDE_CBC_SHA	DH_DSS	3DES	SHA-1

<i>Cipher suite</i>	<i>Key Exchange</i>	<i>Encryption</i>	<i>Hash</i>
SSL_NULL_WITH_NULL_NULL	NULL	NULL	NULL
SSL_RSA_WITH_NULL_MD5	RSA	NULL	MD5
SSL_RSA_WITH_NULL_SHA	RSA	NULL	SHA-1
SSL_RSA_WITH_RC4_128_MD5	RSA	RC4	MD5
SSL_RSA_WITH_RC4_128_SHA	RSA	RC4	SHA-1
SSL_RSA_WITH_IDEA_CBC_SHA	RSA	IDEA	SHA-1
SSL_RSA_WITH_DES_CBC_SHA	RSA	DES	SHA-1
SSL_RSA_WITH_3DES_EDE_CBC_SHA	RSA	3DES	SHA-1
SSL_DH_anon_WITH_RC4_128_MD5	DH_anon	RC4	MD5
SSL_DH_anon_WITH_DES_CBC_SHA	DH_anon	DES	SHA-1
SSL_DH_anon_WITH_3DES_EDE_CBC_SHA	DH_anon	3DES	SHA-1
SSL_DHE_RSA_WITH_DES_CBC_SHA	DHE_RSA	DES	SHA-1
SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA	DHE_RSA	3DES	SHA-1
SSL_DHE_DSS_WITH_DES_CBC_SHA	DHE_DSS	DES	SHA-1
SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA	DHE_DSS	3DES	SHA-1
SSL_DH_RSA_WITH_DES_CBC_SHA	DH_RSA	DES	SHA-1
SSL_DH_RSA_WITH_3DES_EDE_CBC_SHA	DH_RSA	3DES	SHA-1
SSL_DH_DSS_WITH_DES_CBC_SHA	DH_DSS	DES	SHA-1
SSL_DH_DSS_WITH_3DES_EDE_CBC_SHA	DH_DSS	3DES	SHA-1
SSL_FORTEZZA_DMS_WITH_NULL_SHA	Fortezza	NULL	SHA-1
SSL_FORTEZZA_DMS_WITH_FORTEZZA_CBC_SHA	Fortezza	Forzezza	SHA-1
SSL_FORTEZZA_DMS_WITH_RC4_128_SHA	Fortezza	RC4	SHA-1

Generation of Cryptographic Secrets

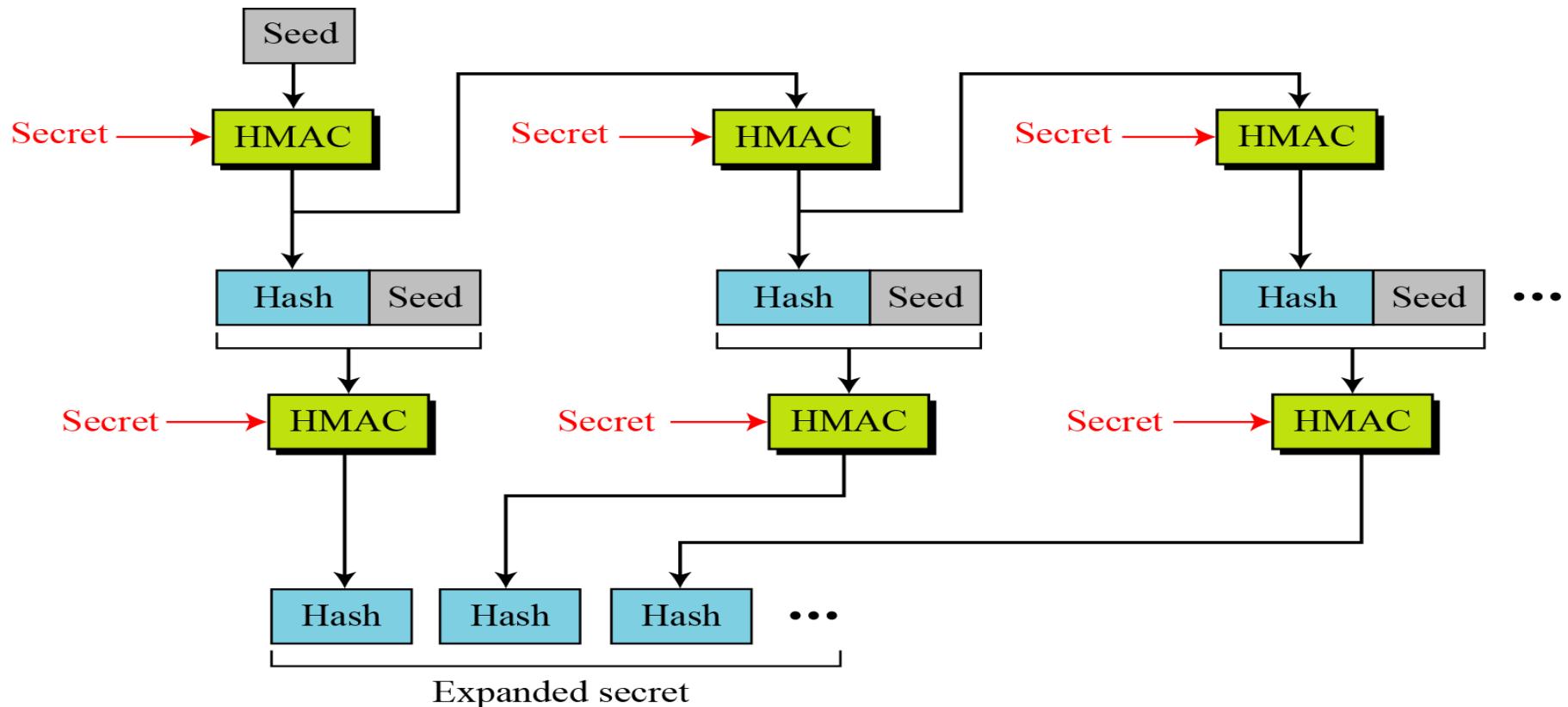
The generation of cryptographic secret is more complex in TLS than in SSL.

The TLS defines two functions

- Data Expansion Function
- Pseudorandom Function

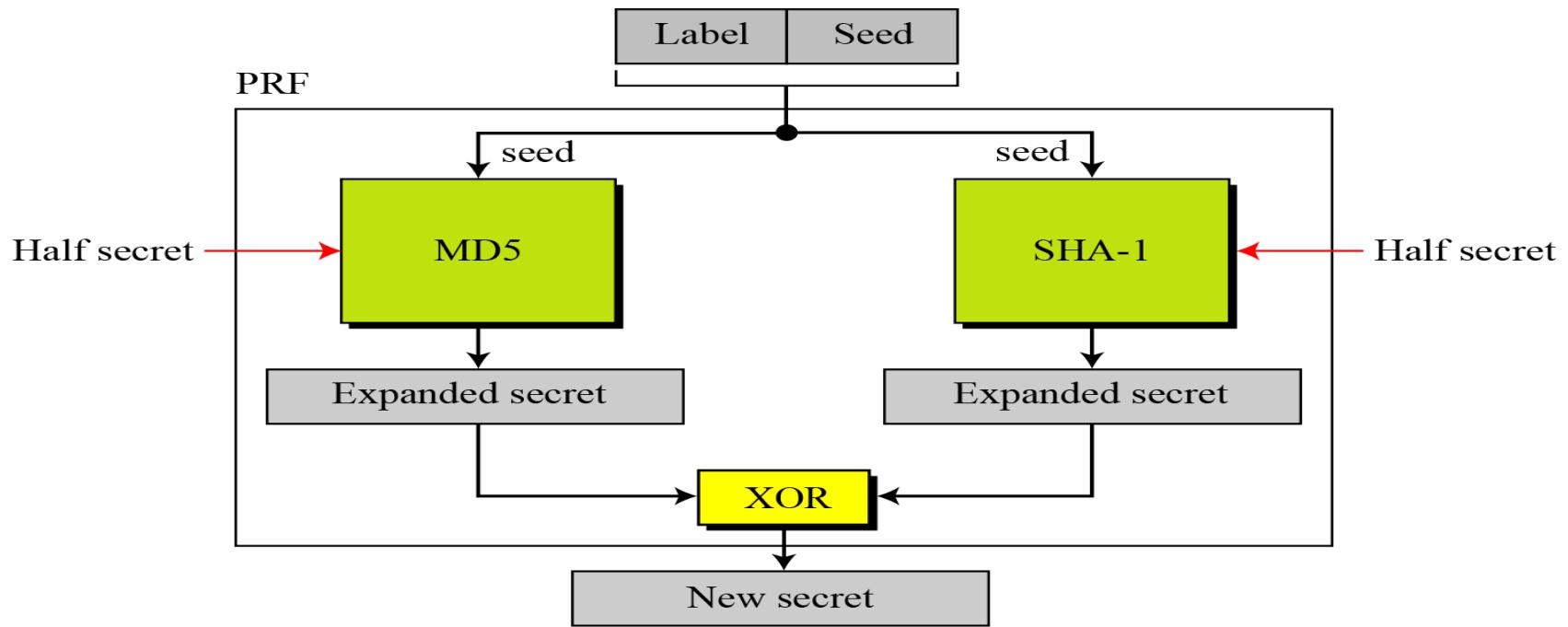
Data-Expansion Function

- Data-Expansion Function uses a predefined HMAC (either MD5 or SHA1) to expand a secret into larger one.
- This function can be considered a multiple section function, where each section creates one hash value.

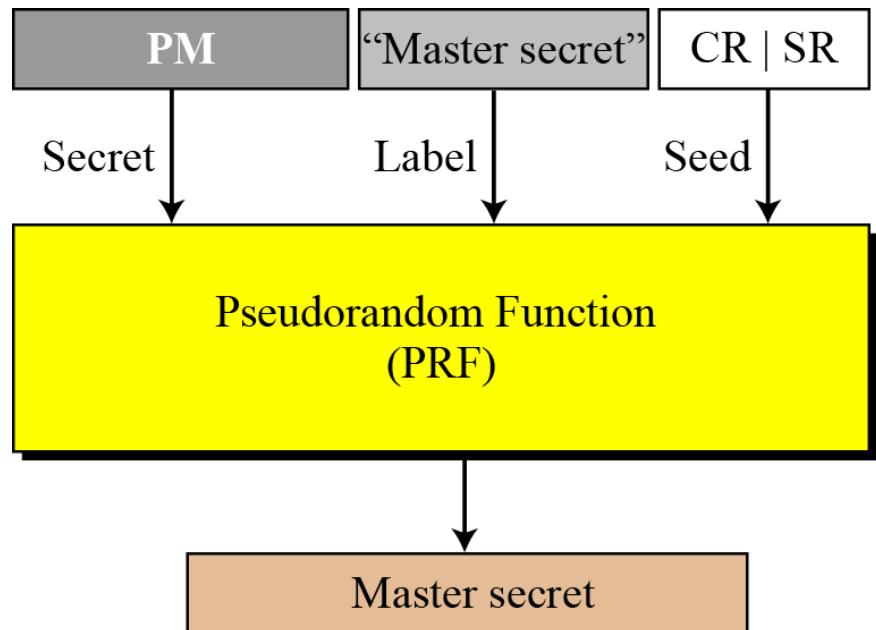


Pseudorandom Function

- TLS defines a Pseudorandom Function(PRF) to be combination of two data-expansion functions, one using MD5 and other SHA-1.



Master Secret Generation in TLS



PM: Pre-master Secret

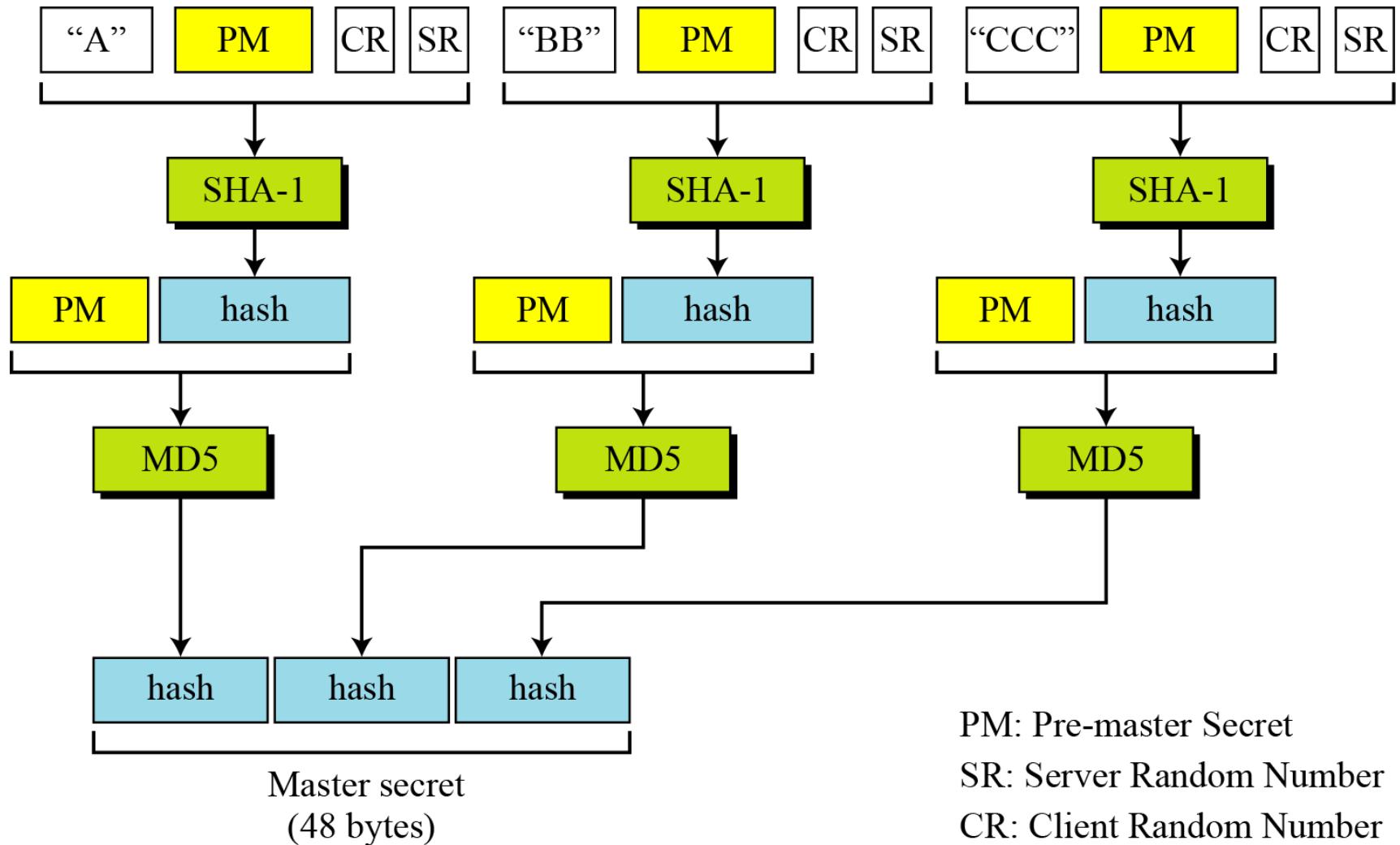
CR: Client Random Number

SR: Server Random Number

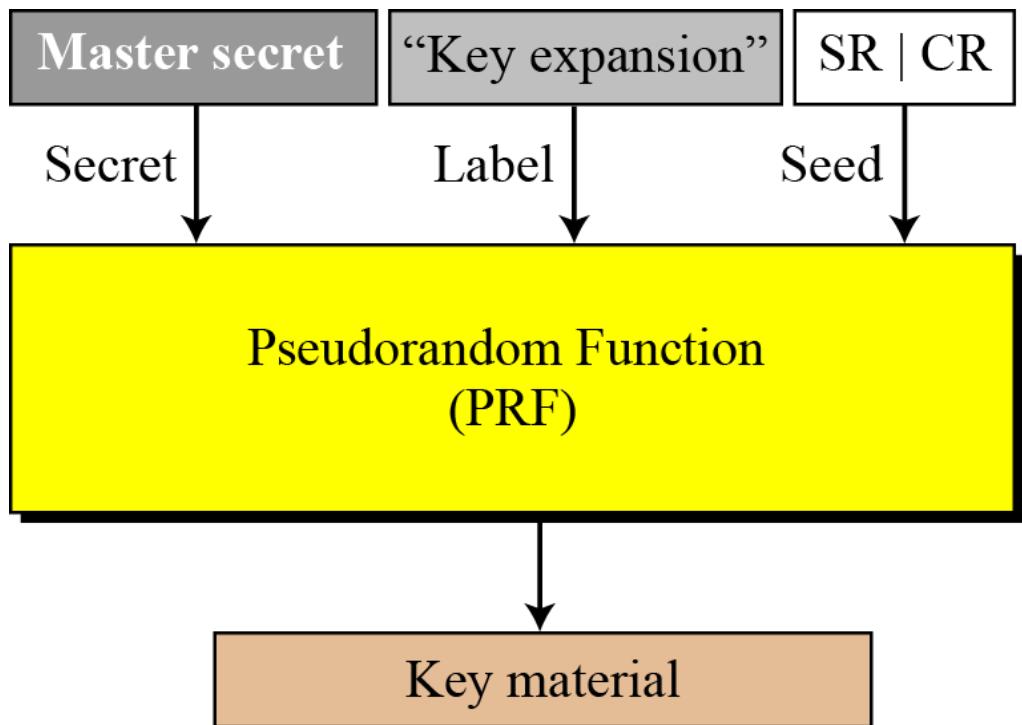
|: Concatenation

Master Secret Generation in SSL

Calculation of master secret from pre-master secret



Key Material Generation TLS

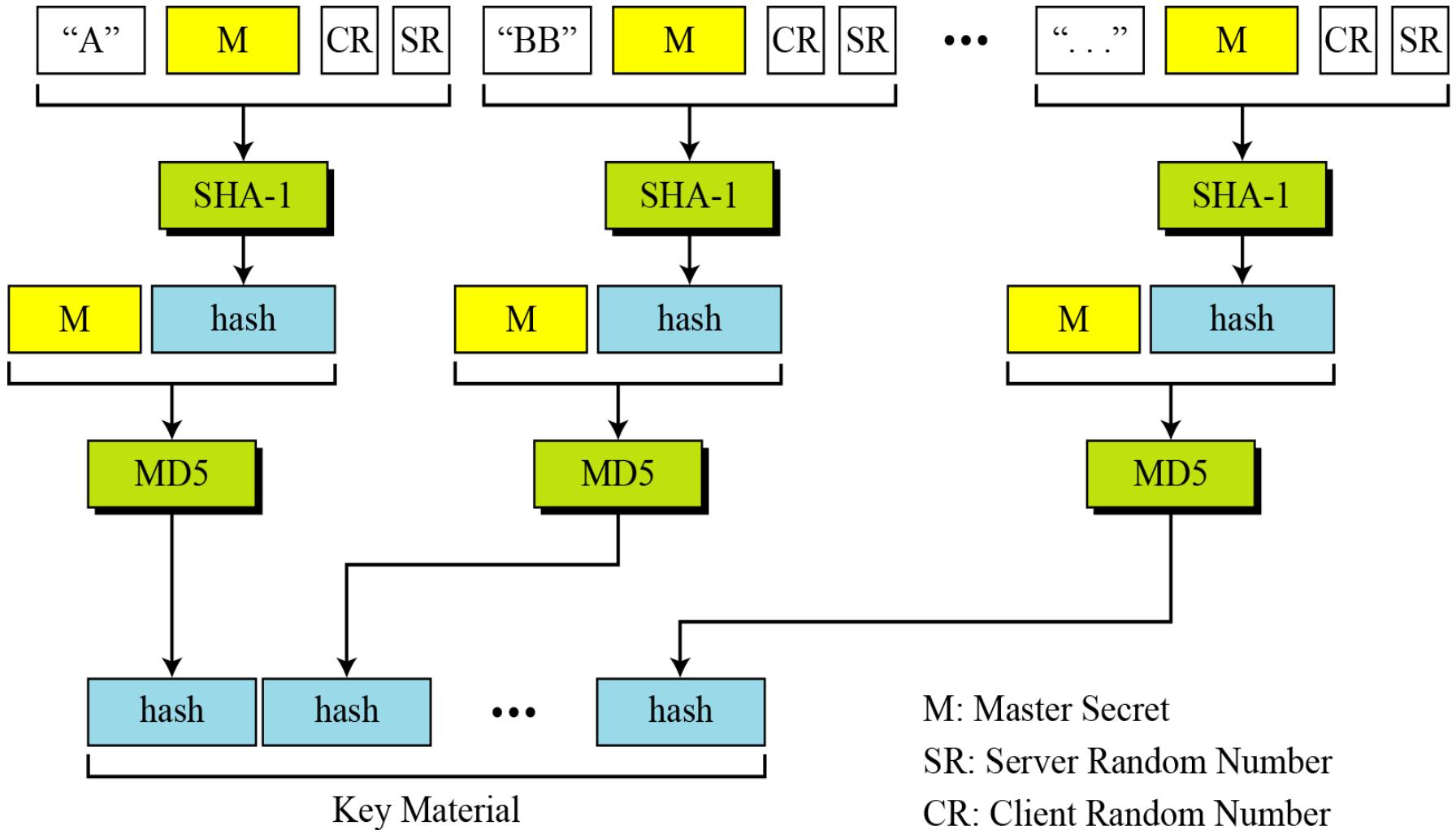


CR: Client Random Number

SR: Server Random Number

|: Concatenation

Calculation of Key Material in SSL



Alert Protocol

TLS supports all of the alerts defined in SSL except for **NoCertificate**. TLS also adds some new ones to the list. Table 17.7 shows the full list of alerts supported by TLS.

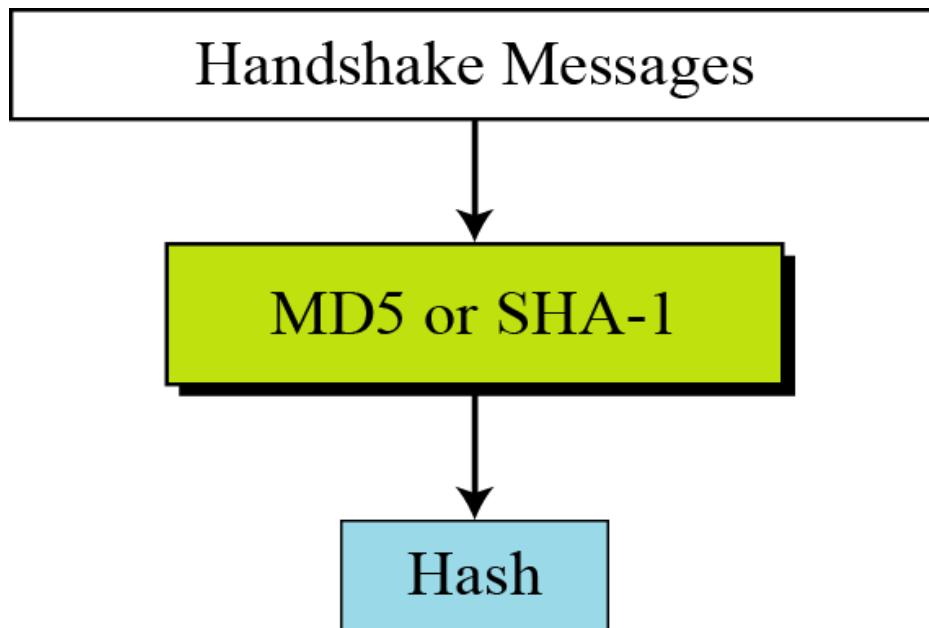
<i>Value</i>	<i>Description</i>	<i>Meaning</i>
0	<i>CloseNotify</i>	Sender will not send any more messages.
10	<i>UnexpectedMessage</i>	An inappropriate message received.
20	<i>BadRecordMAC</i>	An incorrect MAC received.
21	<i>DecryptionFailed</i>	Decrypted message is invalid.
22	<i>RecordOverflow</i>	Message size is more than $2^{14} + 2048$.
30	<i>DecompressionFailure</i>	Unable to decompress appropriately.
40	<i>HandshakeFailure</i>	Sender unable to finalize the handshake.
42	<i>BadCertificate</i>	Received certificate corrupted.
43	<i>UnsupportedCertificate</i>	Type of received certificate is not supported.
44	<i>CertificateRevoked</i>	Signer has revoked the certificate.
45	<i>CertificateExpired</i>	Certificate has expired.
46	<i>CertificateUnknown</i>	Certificate unknown.
47	<i>IllegalParameter</i>	A field out of range or inconsistent with others.
48	<i>UnknownCA</i>	CA could not be identified.

Handshake Protocol in TLS

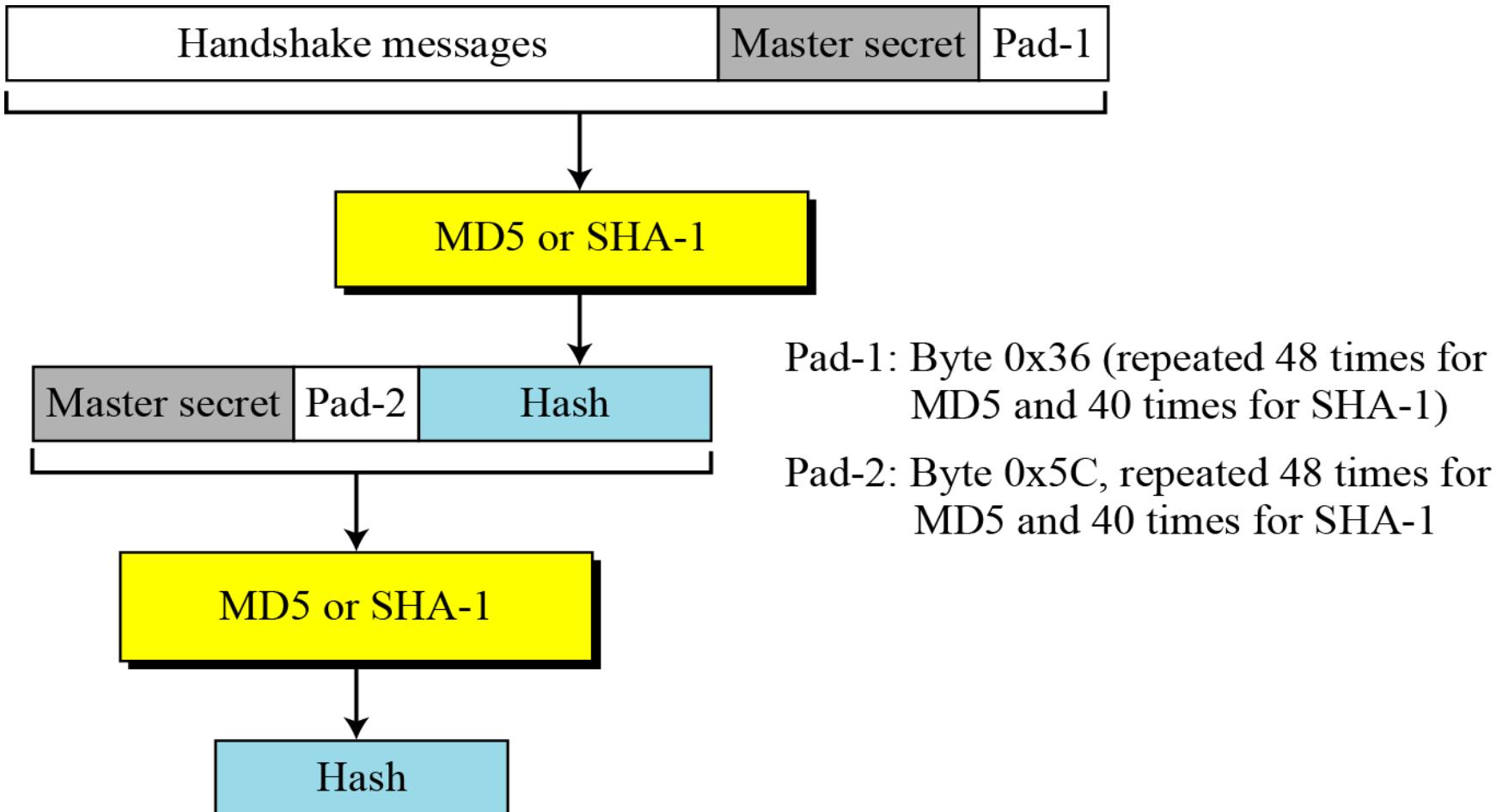
TLS has made some changes in Handshake Protocol. Specifically, in Certificate Verify message and Finished message.

Certificate Verify Message in TLS

In SSL, the hash used in the Certificate Verify message is the two step of the hand shake message plus a pad and the master secret.

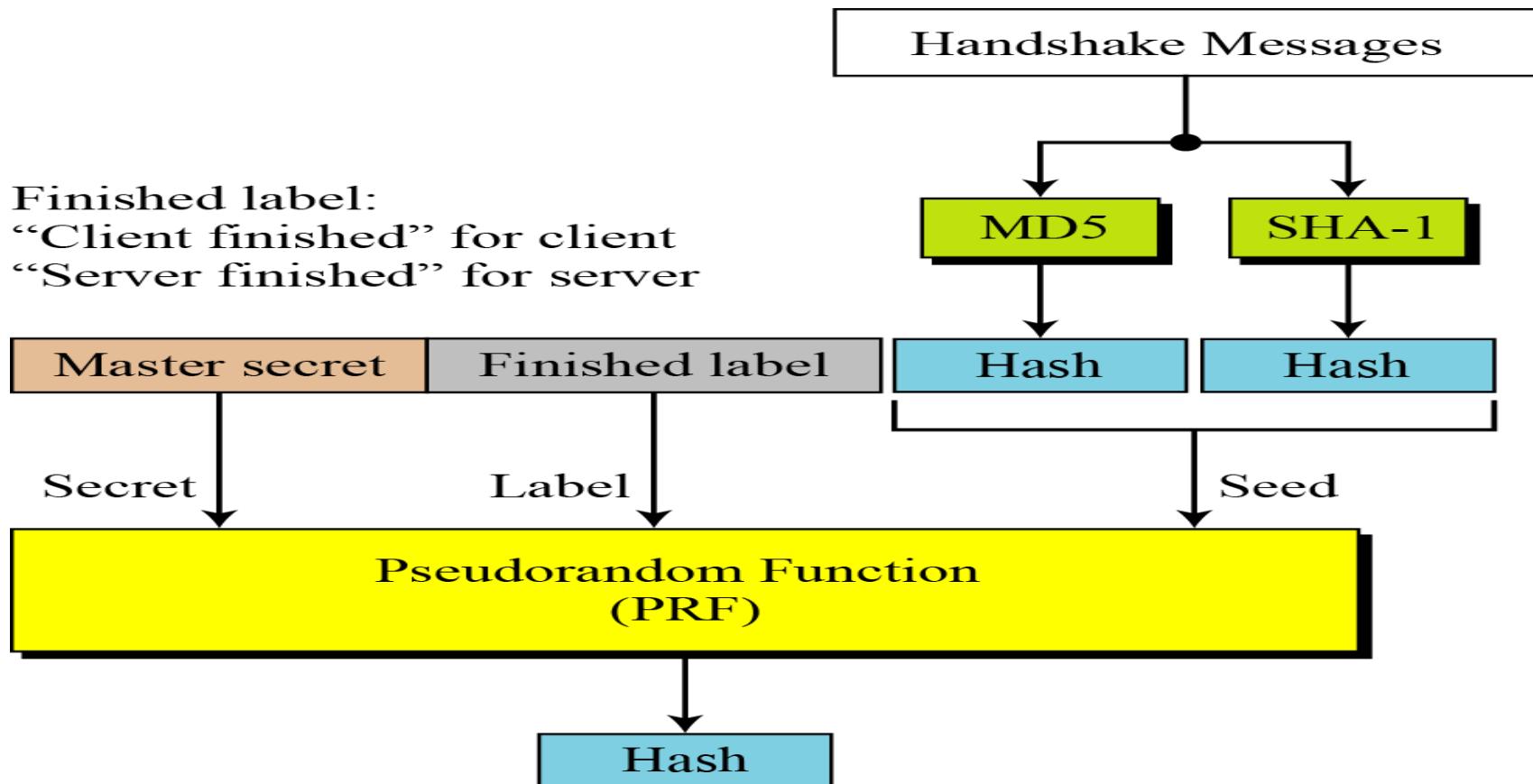


Hash Calculation for Certificate Verify Message

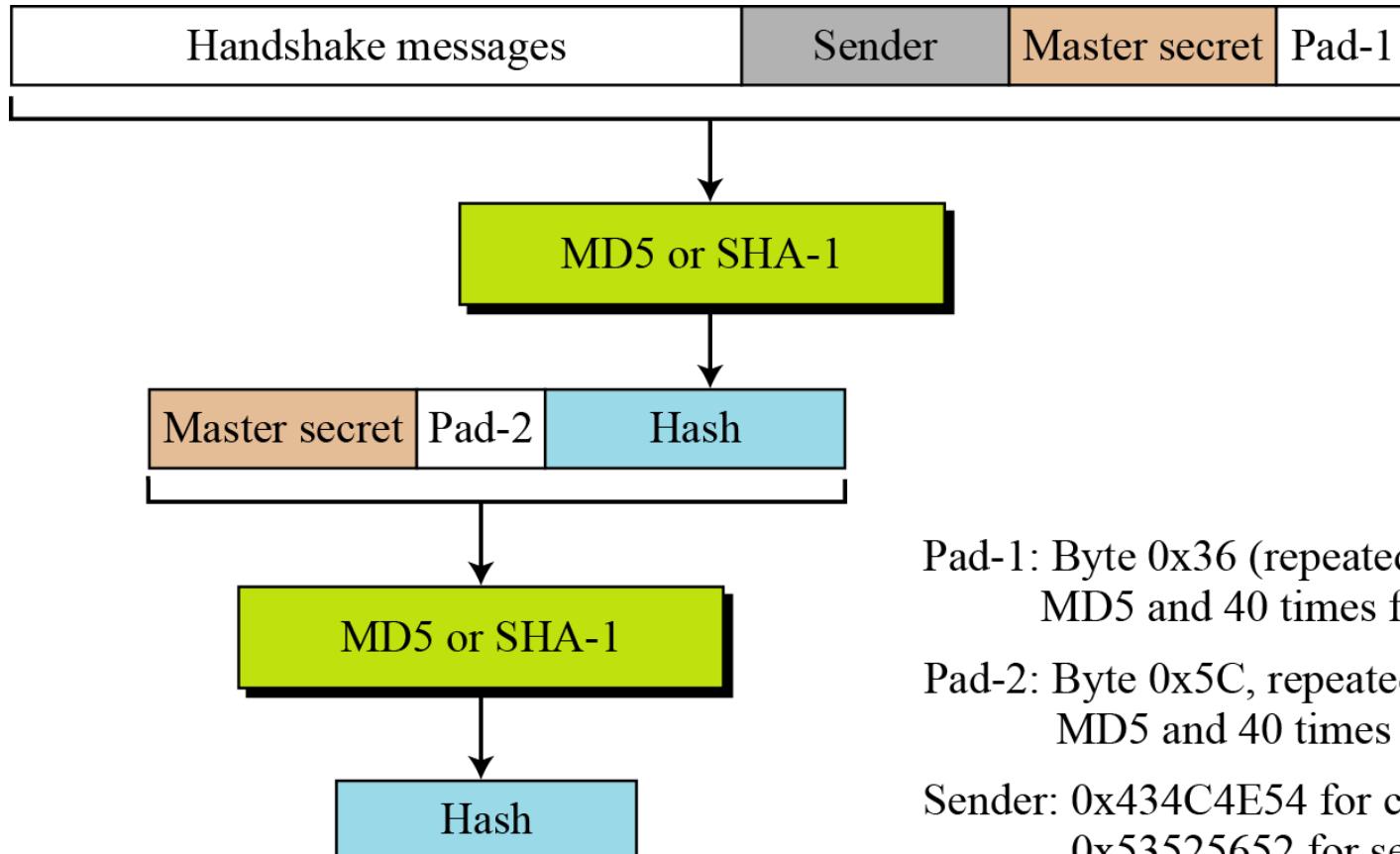


Finished Message in TLS

The calculation of the hash for the finished message has also been changed. TLS uses the Pseudorandom Function to calculate two hashes used for the finished message.



Hash Calculation for Finished Message in SSL



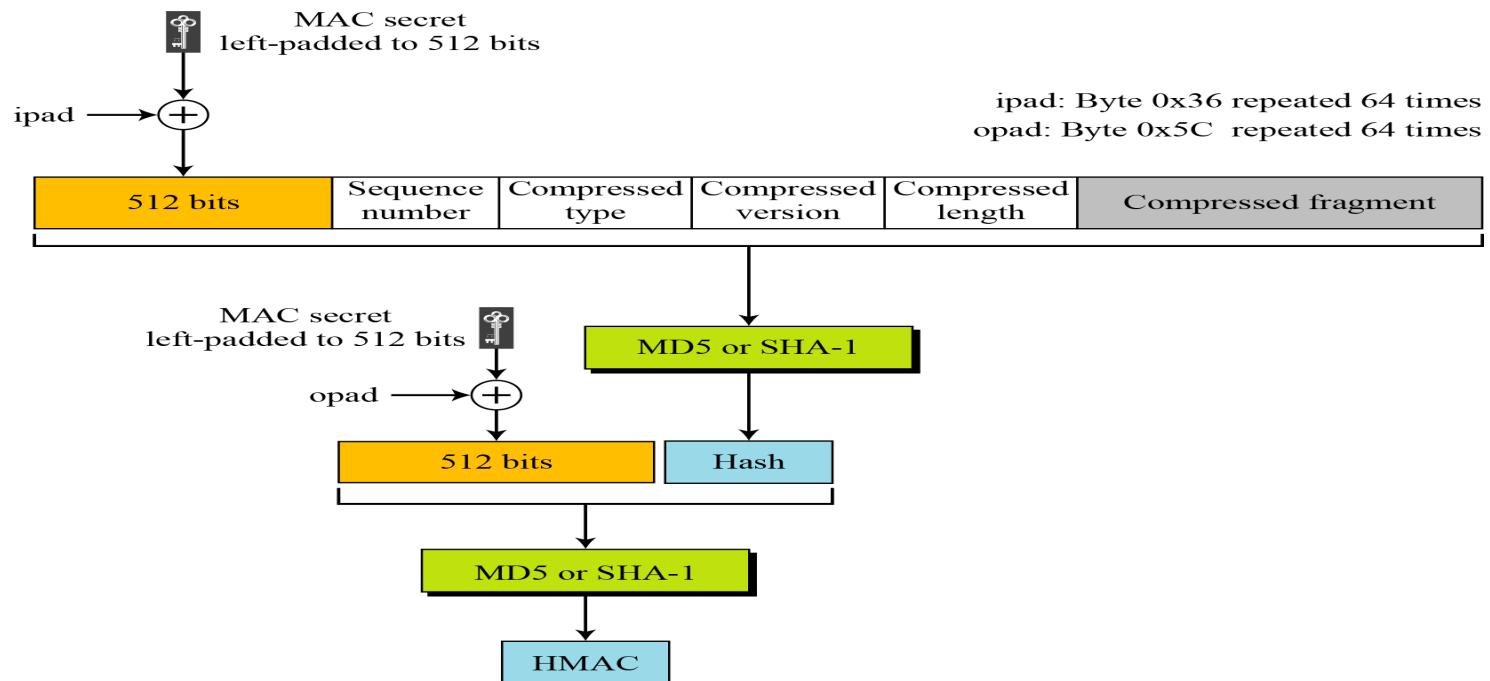
Pad-1: Byte 0x36 (repeated 48 times for MD5 and 40 times for SHA-1)

Pad-2: Byte 0x5C, repeated 48 times for MD5 and 40 times for SHA-1

Sender: 0x434C4E54 for client;
0x53525652 for server

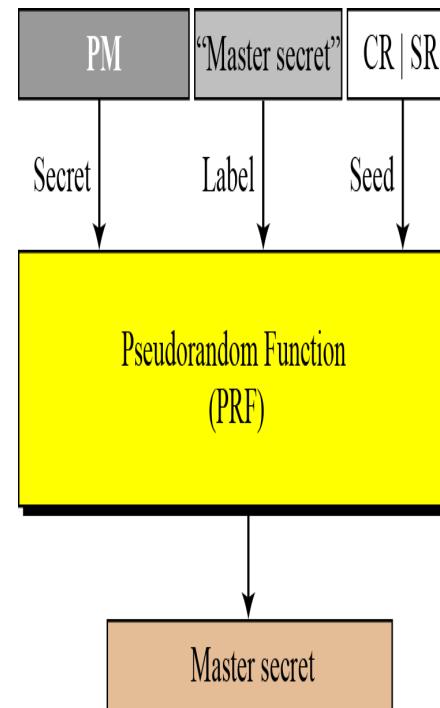
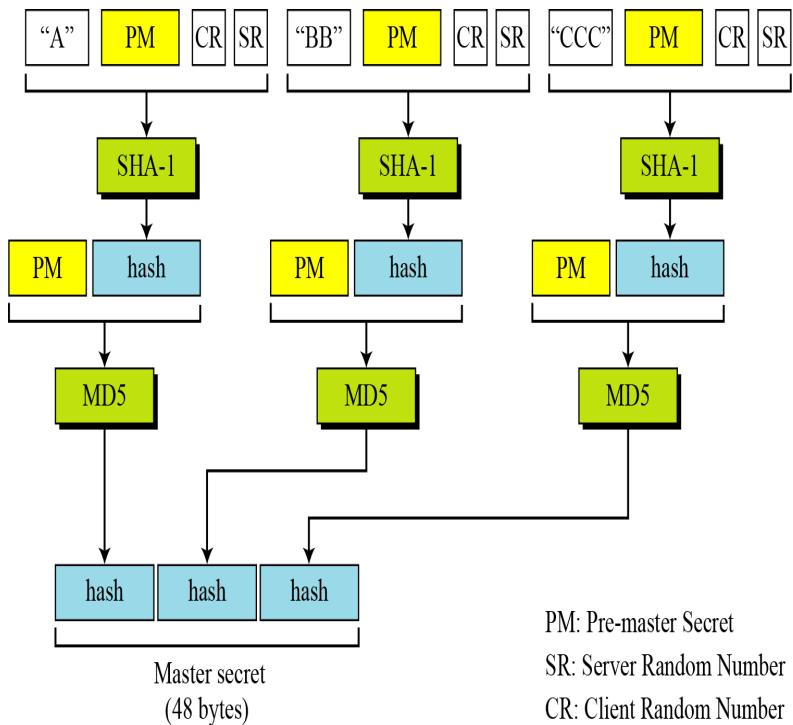
TLS: Record Protocol

The only change in the record protocol is the use of HMAC for signing the message



Question No. 1

Compare the calculation of the master secret in SSL with that in TLS. In SSL, the pre-master is included three times in the calculation, in TLS only Once. Which calculation is more efficient in terms of Space and Time.



Solution to Question No. 1

At first glance, it looks that TLS uses the premaster secret only once to create the Master Secret, but if we look more carefully at the data expansion function and the PRF function, it shows that this calculation in TLS is more complex than the corresponding calculation in SSL. Hence, it is believed that, calculation in TLS is less efficient than the one in SSL.

Question No. 2

The calculation of key material in SSL requires several iterations, the one for TLS does not. How can TLS calculate key material of variable length.

Solution to Question No. 2

Although TLS uses only one PRF function, the PRF function is made of two data expansion function and each expansion function is an iteration of two-stage HMAC calculation. Therefore, TLS also uses iteration to create variable-size key materials although it is not as explicit as the SSL in this issue.

Question No. 3

When the session is resumed, which of the following cryptographic secrets need to be recalculated.

Pre-master secret

- Master secret
- Authentication keys
- Encryption keys
- IVs

Solution to Question No. 3

Authentication keys, encryption keys, and IV's need to be created. The premaster and master secret do not need to be created again.

Question No. 4

How SSL or TLS react to a brute-force attack. Can an intruder use an exhaustive computer search to find the encryption key in SSL or TLS? Which protocol is more secure in this respect SSL or TLS?

Solution to Question No. 4

The key size in SSL or TLS depends on the algorithm used for encryption. If an encryption algorithm with small key-size (such as single DES) is used, the protocol is less immune to brute-force attack. If an encryption algorithm with a large key size is used (such as 3DES), the protocol is more immune to brute-force attack.

Question No. 5

Is SSL or TLS more secure to Man-in-the-Middle attack? Can an intruder create key material between the client and intruder and between intruder and server?

Solution to Question No. 5

The two protocol are equally immune to the man-in-the-middle attack. The immunity depends on the type of algorithm used for key exchange as discussed.

Symmetric-Key Cryptography

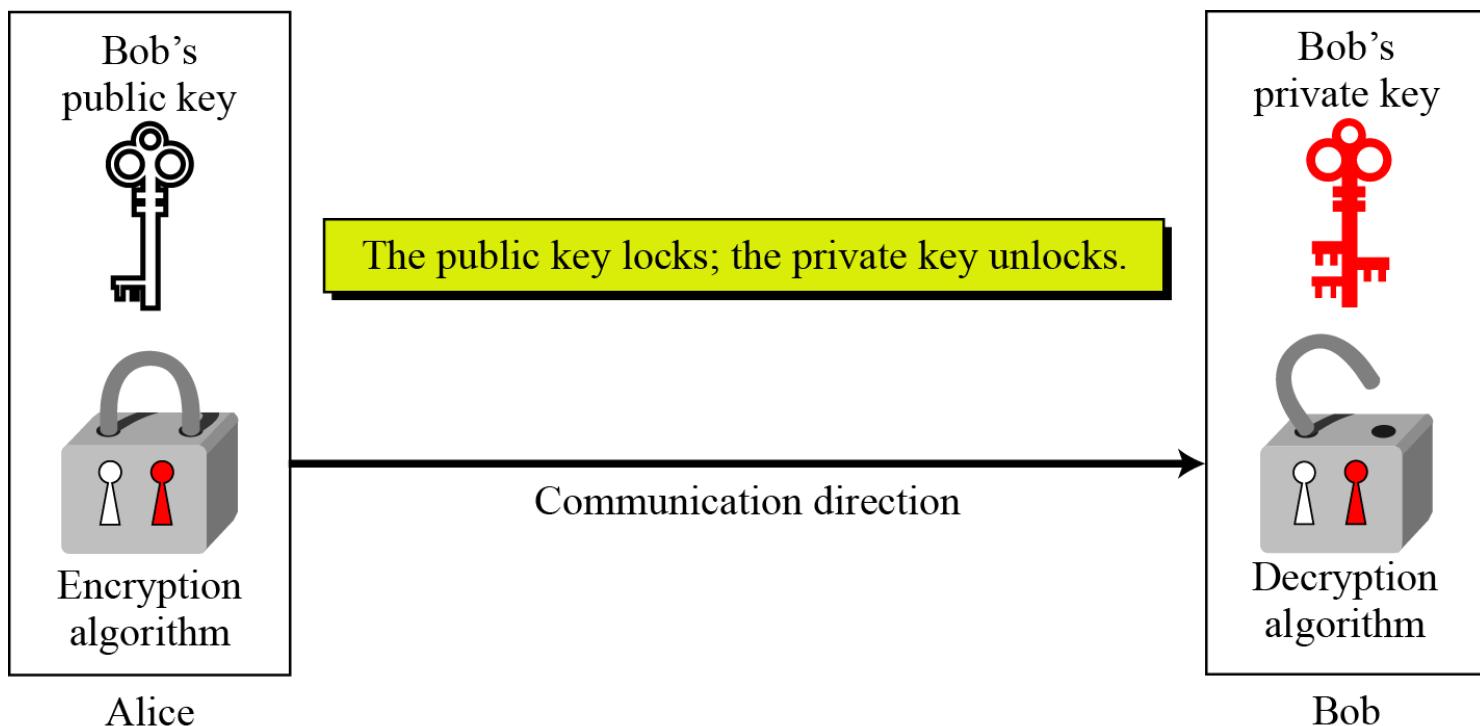
INTRODUCTION

Symmetric and asymmetric-key cryptography will exist in parallel and continue to serve the community. We actually believe that they are complements of each other; the advantages of one can compensate for the disadvantages of the other.

**Symmetric-key cryptography is based on sharing secrecy;
asymmetric-key cryptography is based on personal secrecy.**

Asymmetric key cryptography uses two separate keys: one private and one public.

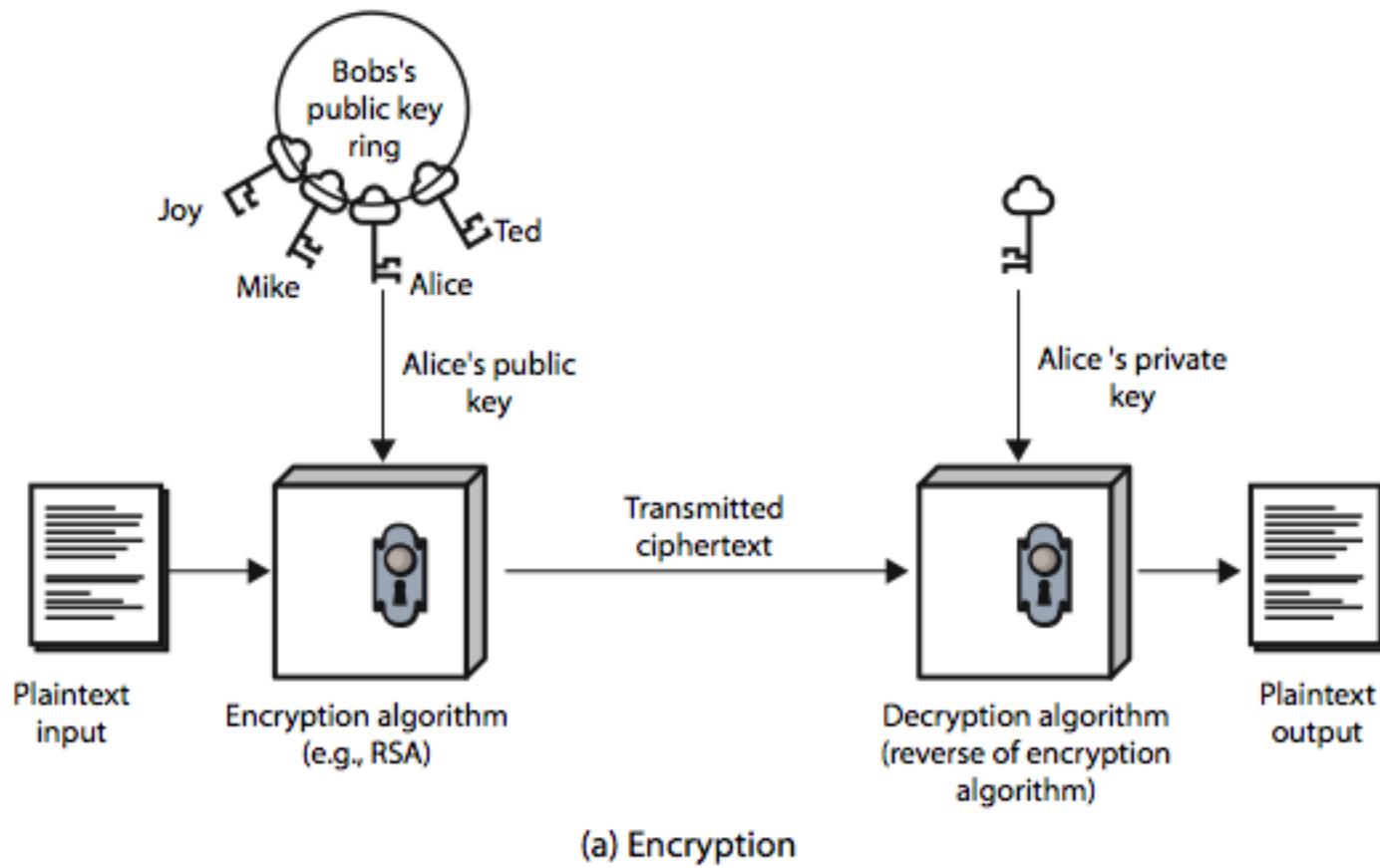
Locking and unlocking in asymmetric-key cryptosystem



Public-Key Cryptography

- **public-key/two-key/asymmetric** cryptography involves the use of **two** keys:
 - a **public-key**, which may be known by anybody, and can be used to **encrypt messages**, and **verify signatures**
 - a **private-key**, known only to the recipient, used to **decrypt messages**, and **sign (create) signatures**
- is **asymmetric** because
 - those who encrypt messages or verify signatures **cannot** decrypt messages or create signatures

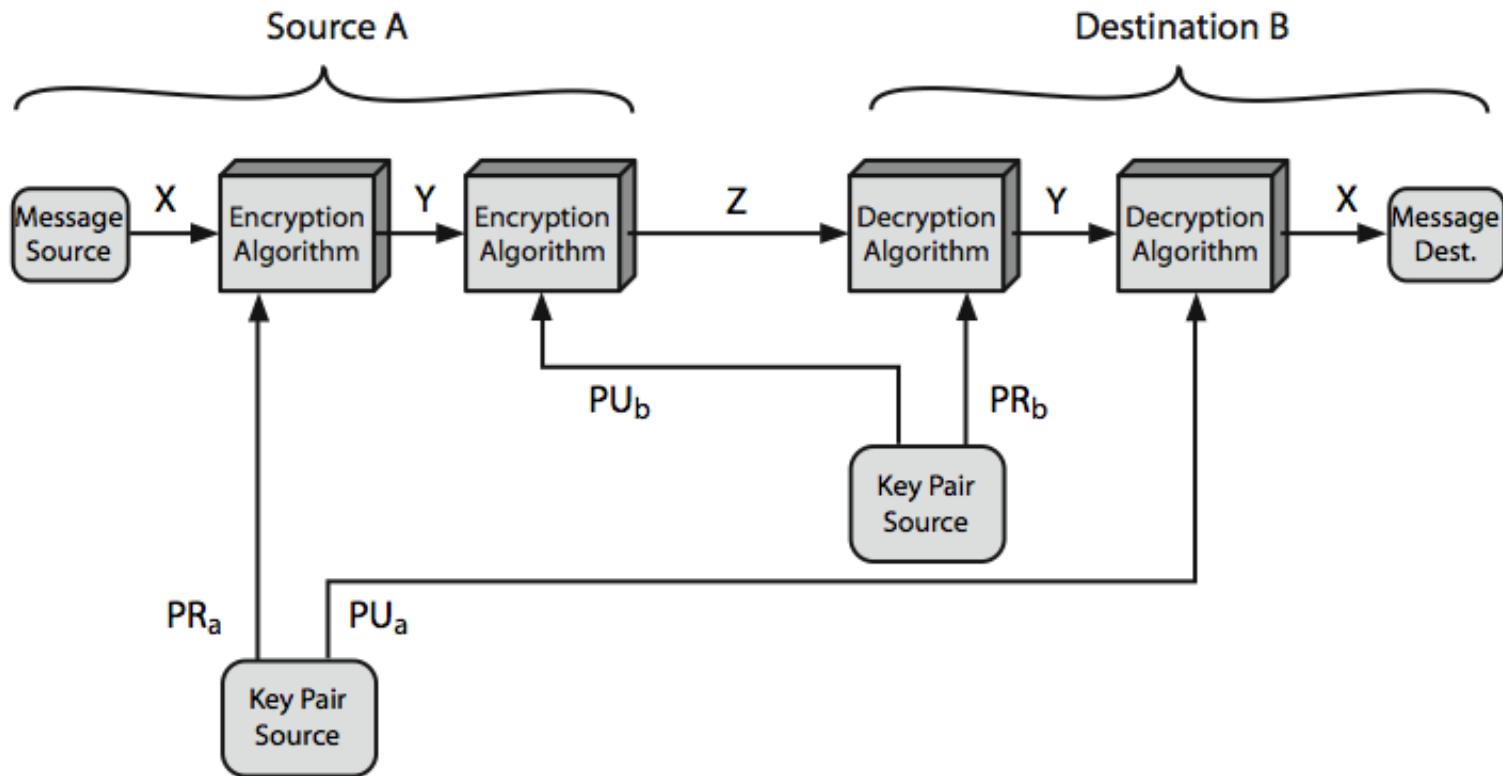
Public-Key Cryptography



Public-Key Characteristics

- Public-Key algorithms rely on two keys where:
 - it is computationally infeasible to find decryption key knowing only algorithm & encryption key
 - it is computationally easy to en/decrypt messages when the relevant (en/decrypt) key is known
 - either of the two related keys can be used for encryption, with the other used for decryption (for some algorithms)

Public-Key Cryptosystems



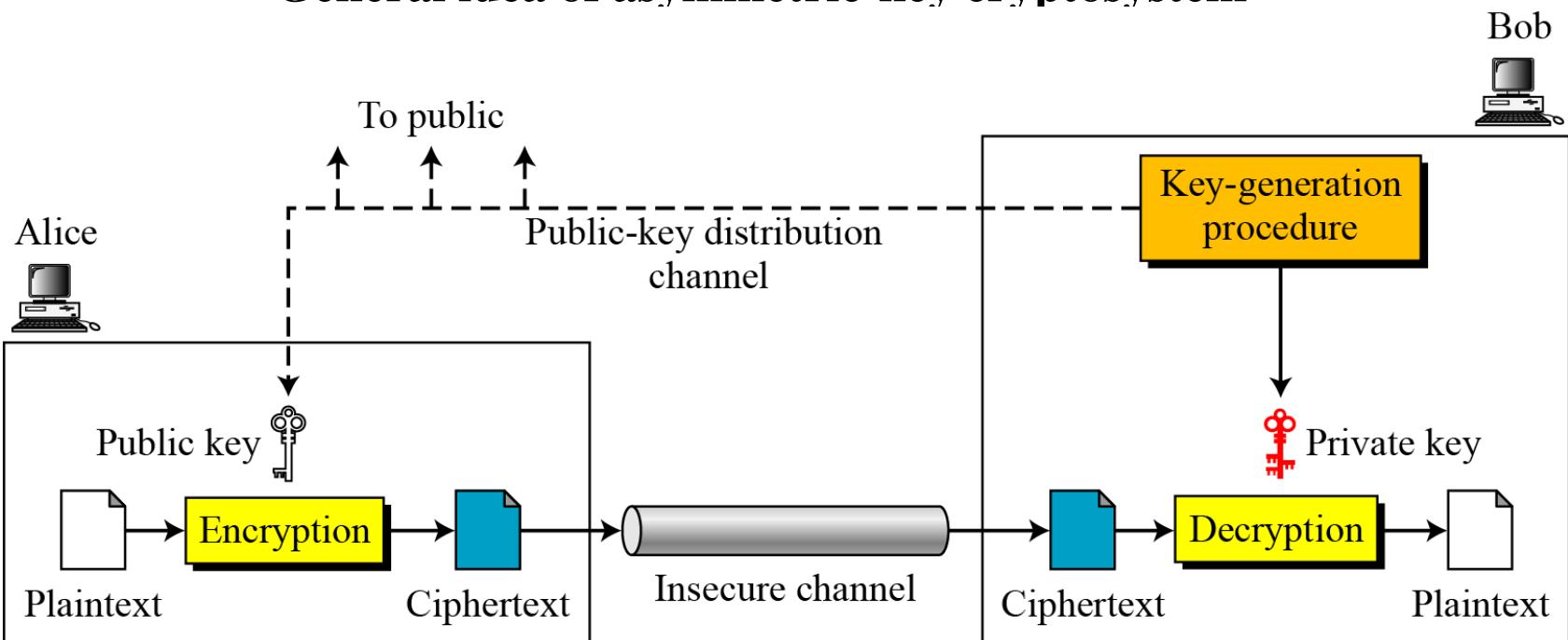
Public-Key Applications

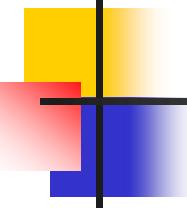
- can classify uses into 3 categories:
 - **encryption/decryption** (provide secrecy)
 - **digital signatures** (provide authentication)
 - **key exchange** (of session keys)
- some algorithms are suitable for all uses, others are specific to one

Security of Public Key Schemes

- like private key schemes brute force **exhaustive search** attack is always theoretically possible
- but keys used are too large (>512bits)
- security relies on a **large enough** difference in difficulty between **easy** (en/decrypt) and **hard** (cryptanalyse) problems
- more generally the **hard** problem is known, but is made hard enough to be impractical to break
- requires the use of **very large numbers**
- hence is **slow** compared to private key schemes

General idea of asymmetric-key cryptosystem





Plaintext/Ciphertext

Unlike in symmetric-key cryptography, plaintext and ciphertext are treated as integers in asymmetric-key cryptography.

Encryption/Decryption

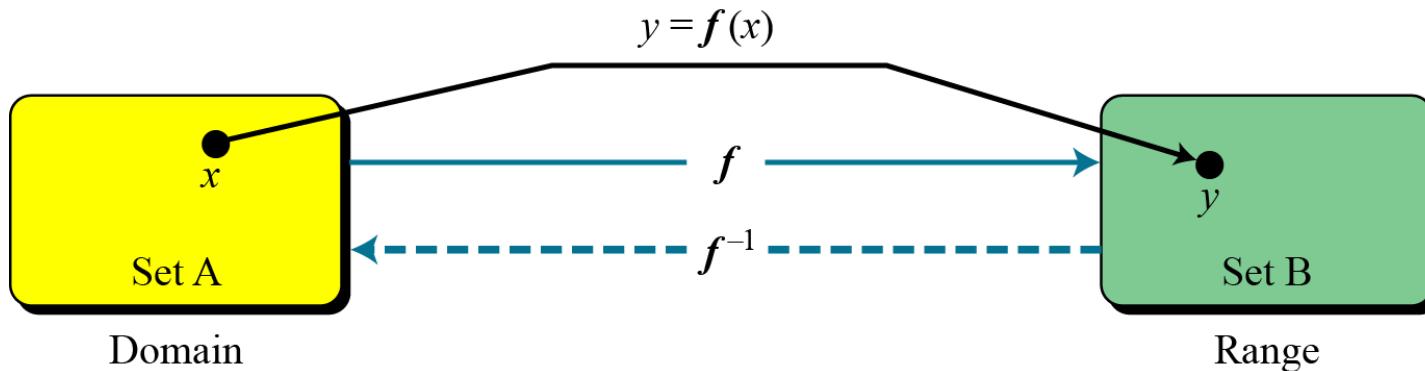
$$C = f(K_{public}, P) \quad P = g(K_{private}, C)$$

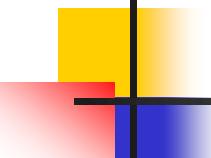
Trapdoor One-Way Function

The main idea behind asymmetric-key cryptography is the concept of the trapdoor one-way function.

Functions

A function as rule mapping a domain to a range





One-Way Function (OWF)

- 1. f is easy to compute.*
 - 2. f^{-1} is difficult to compute.*

Trapdoor One-Way Function (TOWF)

- 3. Given y and a trapdoor, x can be computed easily.*

Example 10. 1

When n is large, $n = p \times q$ is a one-way function. Given p and q , it is always easy to calculate n ; given n , it is very difficult to compute p and q . This is the factorization problem.

Example 10. 2

When n is large, the function $y = x^k \bmod n$ is a trapdoor one-way function. Given x , k , and n , it is easy to calculate y . Given y , k , and n , it is very difficult to calculate x . This is the discrete logarithm problem. However, if we know the trapdoor, k' such that $k \times k' \equiv 1 \pmod{\phi(n)}$, we can use $x = y^{k'} \bmod n$ to find x .

Knapsack Cryptosystem

Definition

$a = [a_1, a_2, \dots, a_k]$ and $x = [x_1, x_2, \dots, x_k]$.

$$s = knapsackSum(a, x) = x_1a_1 + x_2a_2 + \dots + x_ka_k$$

Given a and x , it is easy to calculate s . However, given s and a it is difficult to find x .

Superincreasing Tuple

$$a_i \geq a_1 + a_2 + \dots + a_{i-1}$$

Algorithm 10.1 *knapsacksum* and *inv_knapsackSum* for a superincreasing k-tuple

knapsackSum ($x [1 \dots k], a [1 \dots k]$)

{

$s \leftarrow 0$

for ($i = 1$ to k)

{

$s \leftarrow s + a_i \times x_i$

}

return s

}

inv_knapsackSum ($s, a [1 \dots k]$)

{

for ($i = k$ down to 1)

{

if $s \geq a_i$

{

$x_i \leftarrow 1$

$s \leftarrow s - a_i$

}

else $x_i \leftarrow 0$

}

return $x [1 \dots k]$

}

Example 10. 3

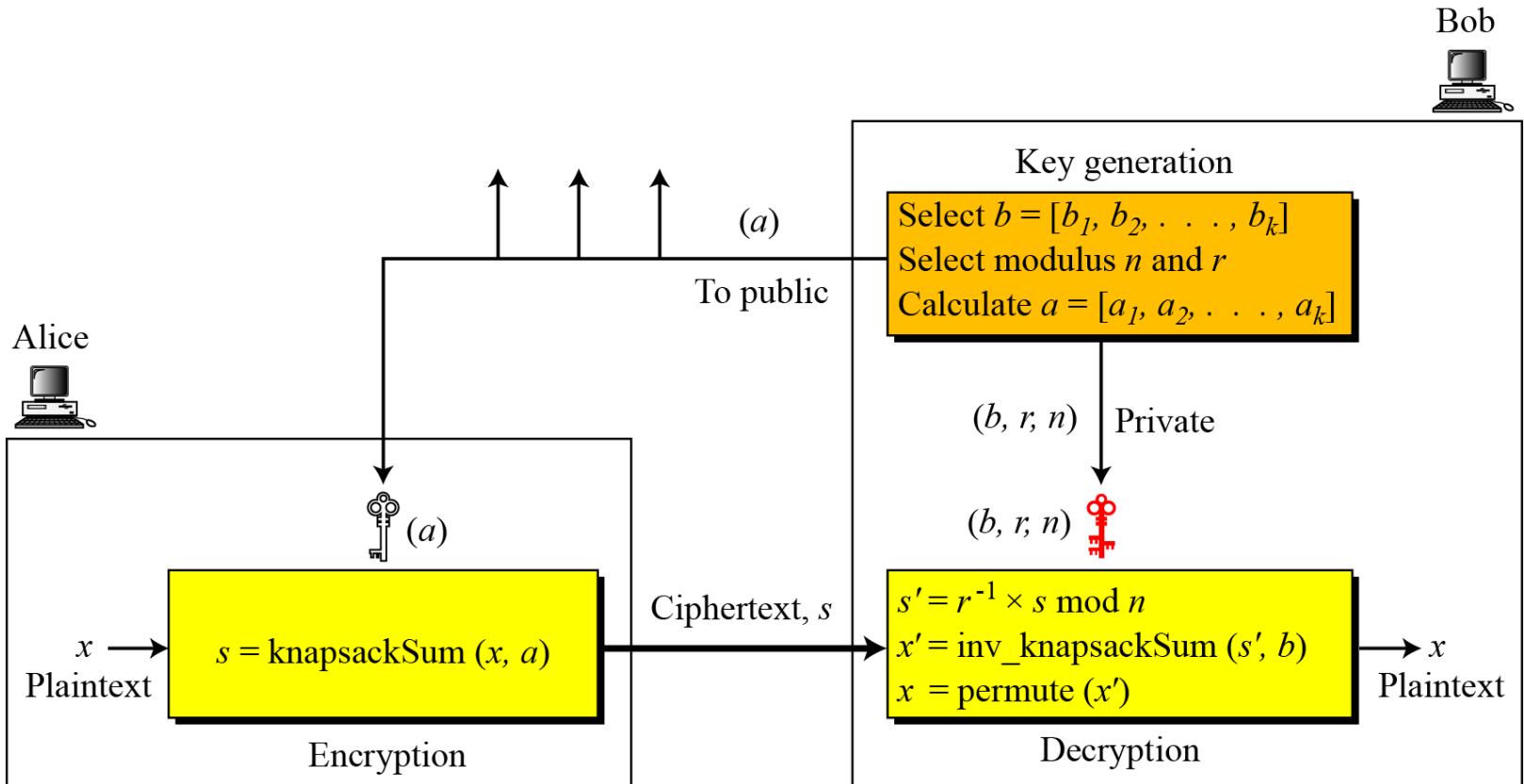
As a very trivial example, assume that $a = [17, 25, 46, 94, 201, 400]$ and $s = 272$ are given. Table 10.1 shows how the tuple x is found using `inv_knapsackSum` routine in Algorithm 10.1. In this case $x = [0, 1, 1, 0, 1, 0]$, which means that 25, 46, and 201 are in the knapsack.

Table 10.1 Values of i , a_i , s , and x_i in Example 10.3

i	a_i	s	$s \geq a_i$	x_i	$s \leftarrow s - a_i \times x_i$
6	400	272	false	$x_6 = 0$	272
5	201	272	true	$x_5 = 1$	71
4	94	71	false	$x_4 = 0$	71
3	46	71	true	$x_3 = 1$	25
2	25	25	true	$x_2 = 1$	0
1	17	0	false	$x_1 = 0$	0

Secret Communication with Knapsacks.

Secret communication with knapsack cryptosystem



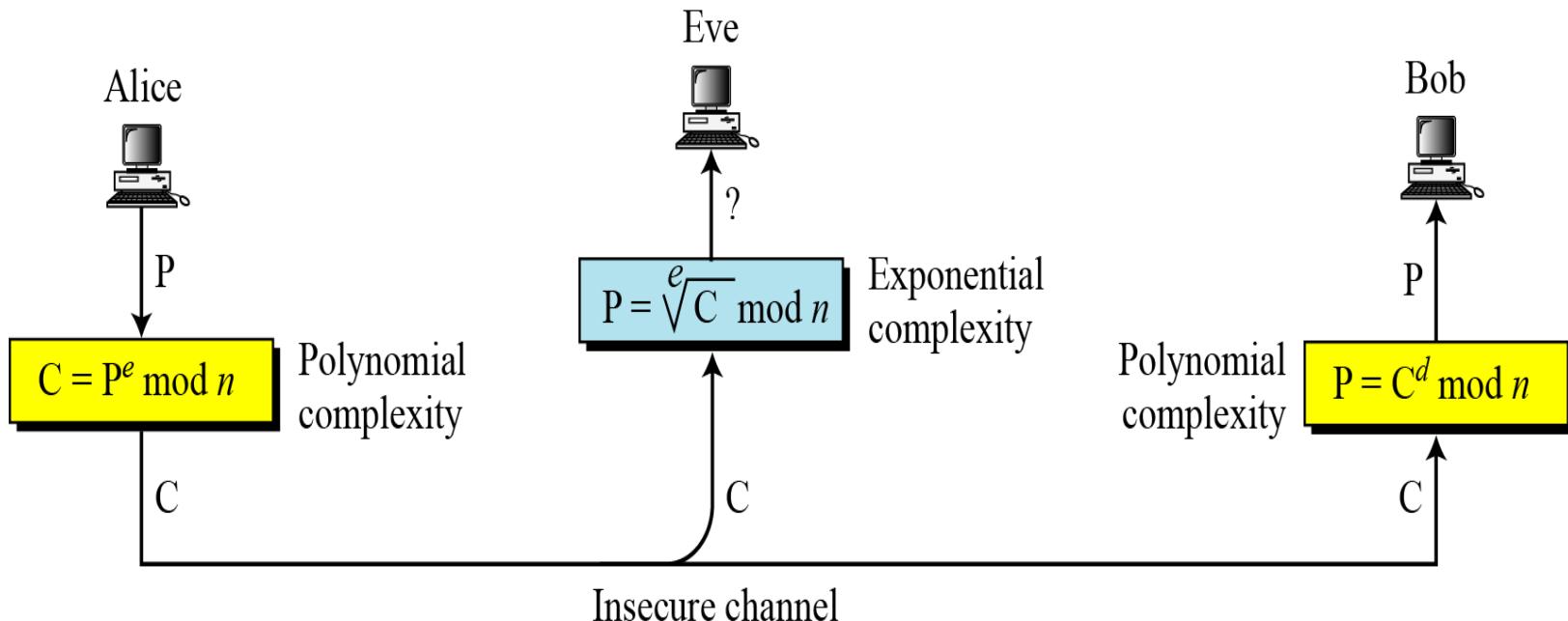
Example 10.4

This is a trivial (very insecure) example just to show the procedure.

1. Key generation:
 - a. Bob creates the superincreasing tuple $b = [7, 11, 19, 39, 79, 157, 313]$.
 - b. Bob chooses the modulus $n = 900$ and $r = 37$, and $[4 \ 2 \ 5 \ 3 \ 1 \ 7 \ 6]$ as permutation table.
 - c. Bob now calculates the tuple $t = [259, 407, 703, 543, 223, 409, 781]$.
 - d. Bob calculates the tuple $a = \text{permute}(t) = [543, 407, 223, 703, 259, 781, 409]$.
 - e. Bob publicly announces a ; he keeps n, r , and b secret.
2. Suppose Alice wants to send a single character “g” to Bob.
 - a. She uses the 7-bit ASCII representation of “g”, $(1100111)_2$, and creates the tuple $x = [1, 1, 0, 0, 1, 1, 1]$. This is the plaintext.
 - b. Alice calculates $s = \text{knapsackSum}(a, x) = 2165$. This is the ciphertext sent to Bob.
3. Bob can decrypt the ciphertext, $s = 2165$.
 - a. Bob calculates $s' = s \times r^{-1} \pmod{n} = 2165 \times 37^{-1} \pmod{900} = 527$.
 - b. Bob calculates $x' = \text{Inv_knapsackSum}(s', b) = [1, 1, 0, 1, 0, 1, 1]$.
 - c. Bob calculates $x = \text{permute}(x') = [1, 1, 0, 0, 1, 1, 1]$. He interprets the string $(1100111)_2$ as the character “g”.

RSA CRYPTOSYSTEM

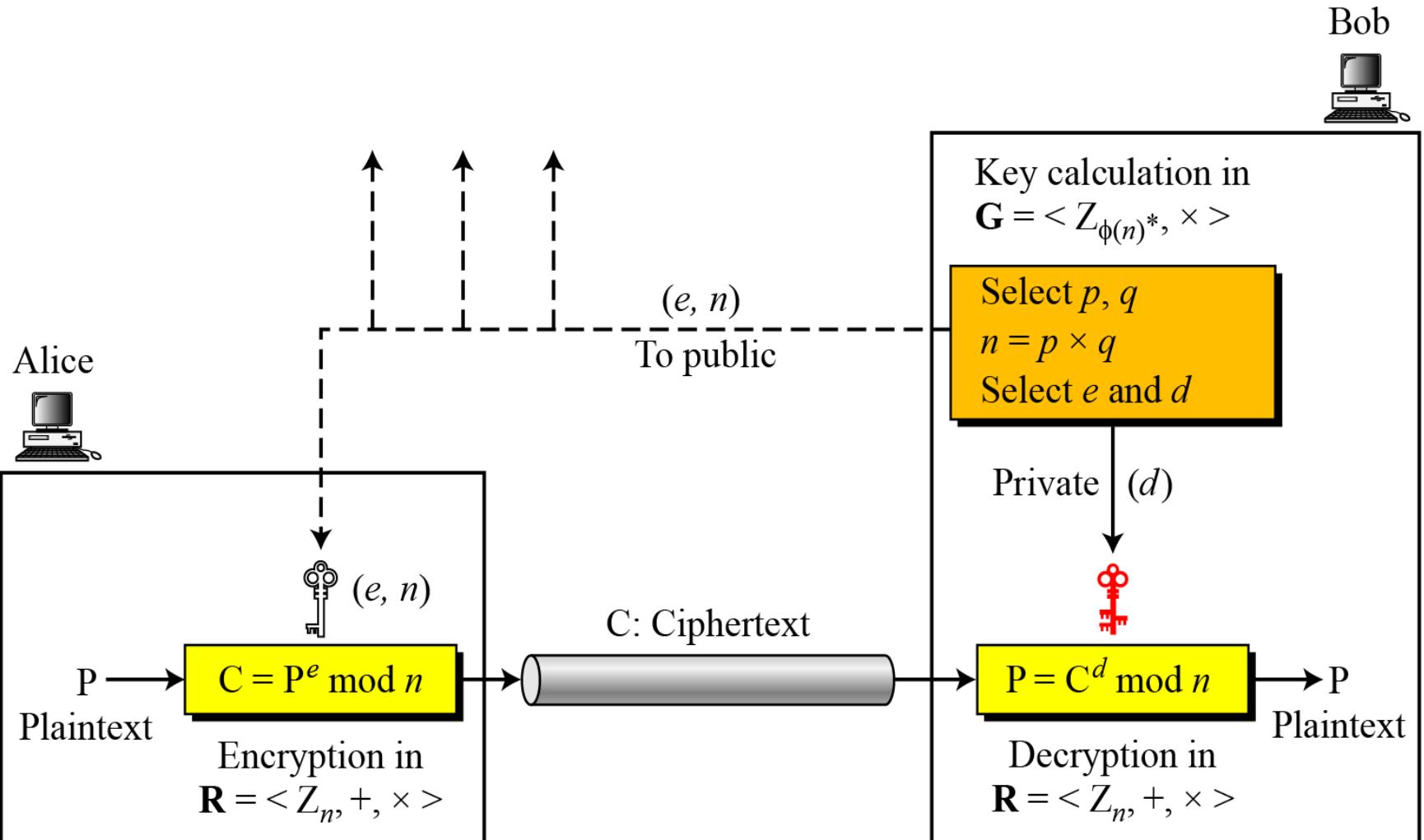
The most common public-key algorithm is the RSA cryptosystem, named for its inventors (Rivest, Shamir, and Adleman).



**RSA uses modular exponentiation for encryption/decryption;
To attack it, Eve needs to calculate $\sqrt[e]{C} \text{ mod } n$.**

Procedure

Encryption, Decryption, and key generation in RSA



Algorithm 10.2 RSA Key Generation

RSA_Key_Generation

{

Select two large primes p and q such that $p \neq q$.

$n \leftarrow p \times q$

$\phi(n) \leftarrow (p - 1) \times (q - 1)$

Select e such that $1 < e < \phi(n)$ and e is coprime to $\phi(n)$

$d \leftarrow e^{-1} \bmod \phi(n)$ // d is inverse of e modulo $\phi(n)$

Public_key $\leftarrow (e, n)$ // To be announced publicly

Private_key $\leftarrow d$ // To be kept secret

return Public_key and Private_key

}

Proof of RSA

If $n = p \times q$, $a < n$, and k is an integer, then $a^{k \times \phi(n) + 1} \equiv a \pmod{n}$.

$$P_1 = C^d \pmod{n} = (P^e \pmod{n})^d \pmod{n} = P^{ed} \pmod{n}$$

$$ed = k\phi(n) + 1 \qquad \qquad \qquad // d \text{ and } e \text{ are inverses modulo } \phi(n)$$

$$P_1 = P^{ed} \pmod{n} \rightarrow P_1 = P^{k\phi(n)+1} \pmod{n}$$

$$P_1 = P^{k\phi(n)+1} \pmod{n} = P \pmod{n} \qquad \qquad \qquad // \text{Euler's theorem (second version)}$$

Why RSA Works

- because of Euler's Theorem:
 - $a^{\phi(n)} \text{mod } n = 1$ where $\gcd(a,n)=1$
- in RSA have:
 - $n=p \cdot q$
 - $\phi(n)=(p-1)(q-1)$
 - carefully chose e & d to be inverses mod $\phi(n)$
 - hence $e \cdot d = 1 + k \cdot \phi(n)$ for some k
- hence :
$$\begin{aligned} C^d &= M^{e \cdot d} = M^{1+k \cdot \phi(n)} = M^1 \cdot (M^{\phi(n)})^k \\ &= M^1 \cdot (1)^k = M^1 = M \text{ mod } n \end{aligned}$$

RSA Key Setup

- each user generates a public/private key pair by:
- selecting two large primes at random - p, q
- computing their system modulus $n=p \cdot q$
 - note $\phi(n) = (p-1)(q-1)$
- selecting at random the encryption key e
 - where $1 < e < \phi(n)$, $\gcd(e, \phi(n)) = 1$
- solve following equation to find decryption key d
 - $e \cdot d \equiv 1 \pmod{\phi(n)}$ and $0 \leq d \leq n$
- publish their public encryption key: $PU = \{e, n\}$
- keep secret private decryption key: $PR = \{d, n\}$

RSA Use

- to encrypt a message M the sender:
 - obtains **public key** of recipient $PU=\{e,n\}$
 - computes: $C = M^e \text{ mod } n$, where $0 \leq M < n$
- to decrypt the ciphertext C the owner:
 - uses their private key $PR=\{d,n\}$
 - computes: $M = C^d \text{ mod } n$
- note that the message M must be smaller than the modulus n (block if needed)

RSA Example - Key Setup

1. Select primes: $p=17$ & $q=11$
2. Compute $n = pq = 17 \times 11 = 187$
3. Compute $\phi(n) = (p-1)(q-1) = 16 \times 10 = 160$
4. Select e: $\gcd(e, 160) = 1$; choose $e=7$
5. Determine d: $de \equiv 1 \pmod{160}$ and $d < 160$ Value is $d=23$ since $23 \times 7 = 161 = 10 \times 160 + 1$
6. Publish public key PU={7,187}
7. Keep secret private key PR={23,187}

Some Trivial Examples

Example 10. 5

Bob chooses 7 and 11 as p and q and calculates $n = 77$. The value of $f(n) = (7 - 1)(11 - 1)$ or 60. Now he chooses two exponents, e and d , from Z_{60}^* . If he chooses e to be 13, then d is 37. Note that $e \times d \bmod 60 = 1$ (they are inverses of each other). Now imagine that Alice wants to send the plaintext 5 to Bob. She uses the public exponent 13 to encrypt 5.

Plaintext: 5

$$C = 5^{13} = 26 \bmod 77$$

Ciphertext: 26

Bob receives the ciphertext 26 and uses the private key 37 to decipher the ciphertext:

Ciphertext: 26

$$P = 26^{37} = 5 \bmod 77$$

Plaintext: 5

Some Trivial Examples

Example 10.6

Now assume that another person, John, wants to send a message to Bob. John can use the same public key announced by Bob (probably on his website), 13; John's plaintext is 63. John calculates the following:

Plaintext: 63

$$C = 63^{13} \equiv 28 \pmod{77}$$

Ciphertext: 28

Bob receives the ciphertext 28 and uses his private key 37 to decipher the ciphertext:

Ciphertext: 28

$$P = 28^{37} \equiv 63 \pmod{77}$$

Plaintext: 63

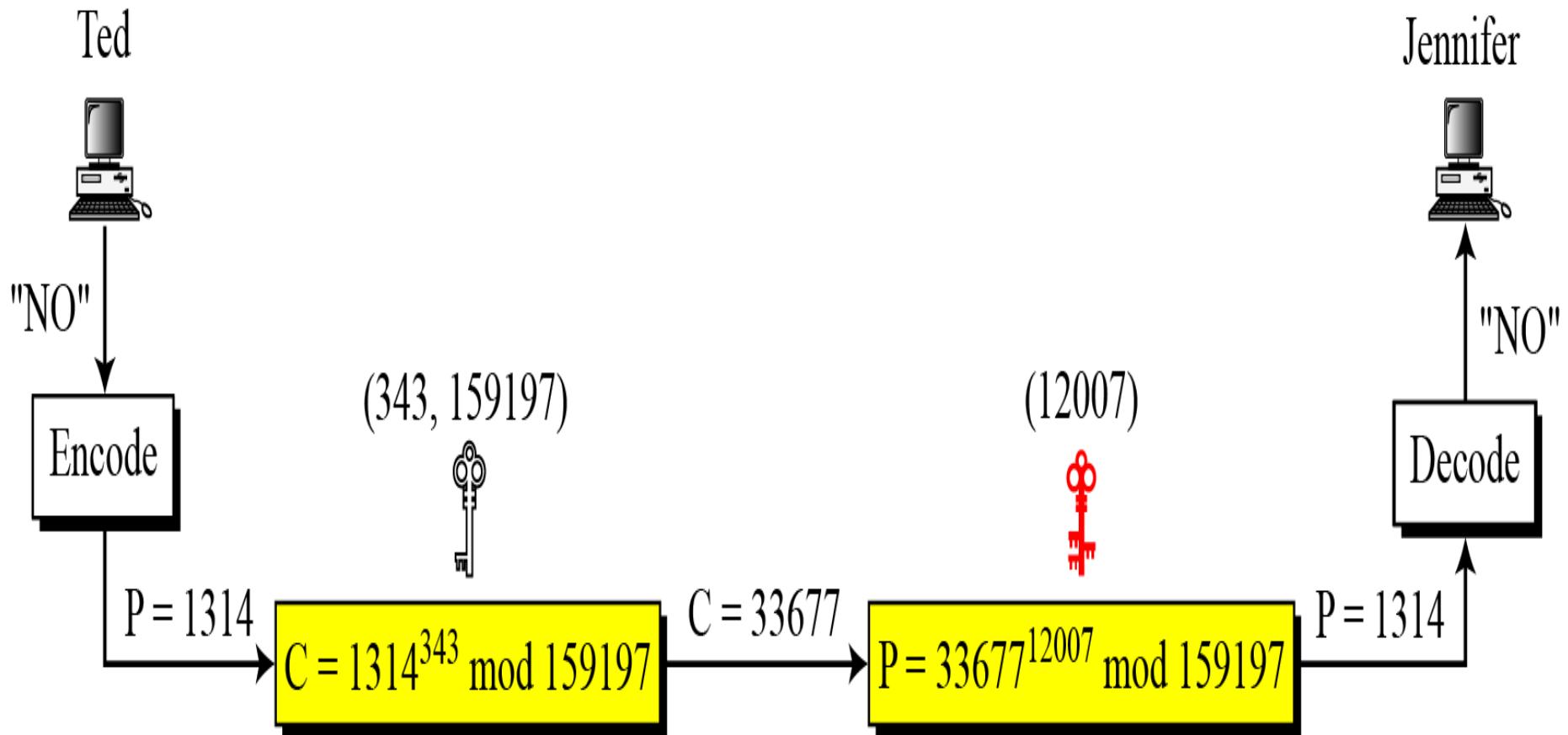
Some Trivial Examples

Example 10.7

Jennifer creates a pair of keys for herself. She chooses $p = 397$ and $q = 401$. She calculates $n = 159197$. She then calculates $f(n) = 158400$. She then chooses $e = 343$ and $d = 12007$. Show how Ted can send a message to Jennifer if he knows e and n .

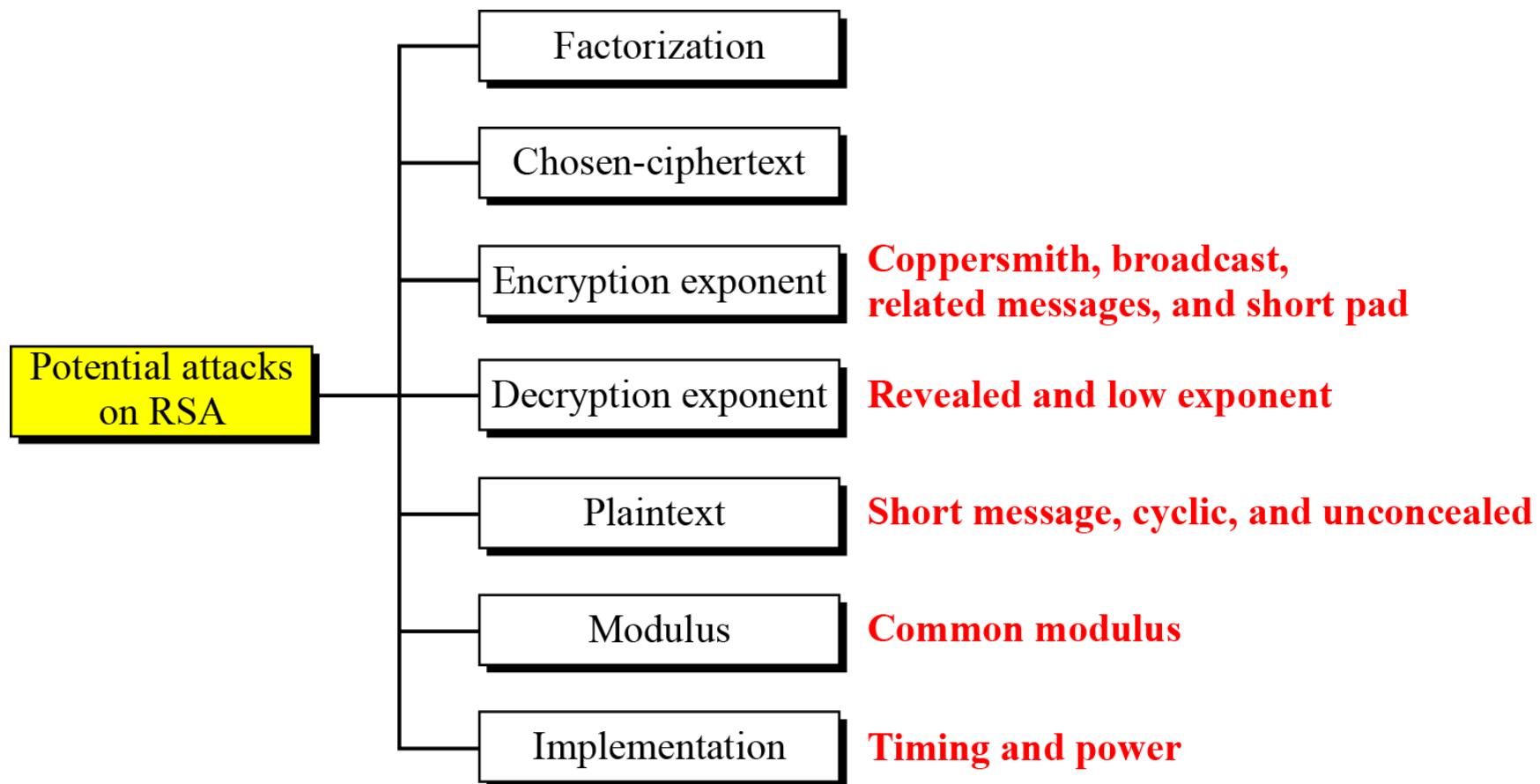
Suppose Ted wants to send the message “NO” to Jennifer. He changes each character to a number (from 00 to 25), with each character coded as two digits. He then concatenates the two coded characters and gets a four-digit number. The plaintext is 1314. Figure 10.7 shows the process.

Encryption and Decryption in Example



Attacks on RSA

Taxonomy of potential attacks on RSA



Digital Signature

Let us begin by looking at the differences between conventional signatures and digital signatures.

Inclusion

A conventional signature is included in the document; it is part of the document. But when we sign a document digitally, we send the signature as a separate document.

Verification Method

For a conventional signature, when the recipient receives a document, she compares the signature on the document with the signature on file. For a digital signature, the recipient receives the message and the signature. The recipient needs to apply a verification technique to the combination of the message and the signature to verify the authenticity.

Relationship

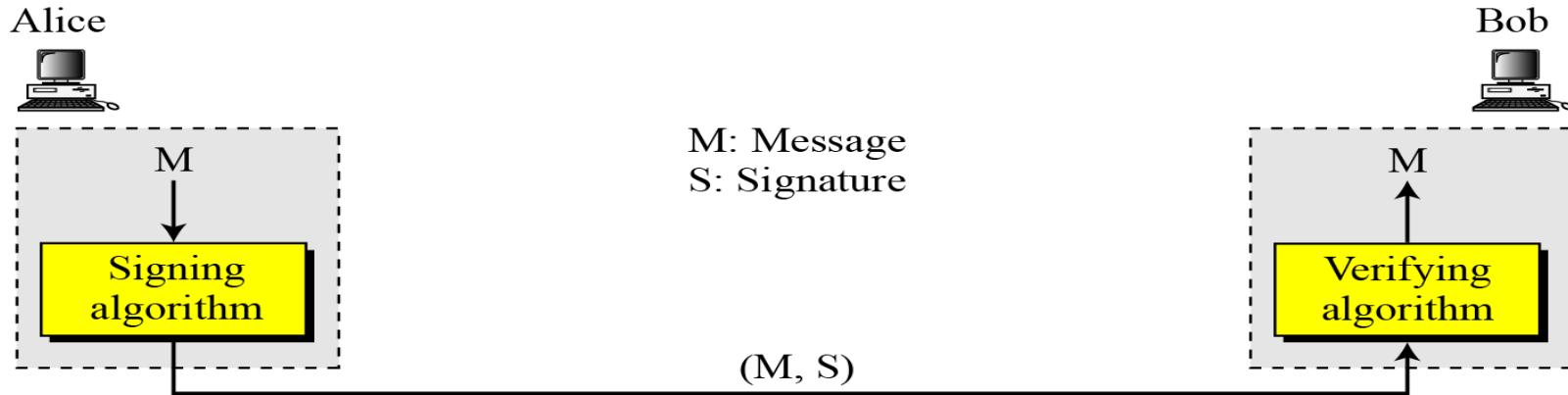
For a conventional signature, there is normally a one-to-many relationship between a signature and documents. For a digital signature, there is a one-to-one relationship between a signature and a message.

Duplicity

In conventional signature, a copy of the signed document can be distinguished from the original one on file. In digital signature, there is no such distinction unless there is a factor of time on the document.

PROCESS

Figure shows the digital signature process. The sender uses a signing algorithm to sign the message. The message and the signature are sent to the receiver. The receiver receives the message and the signature and applies the verifying algorithm to the combination. If the result is true, the message is accepted; otherwise, it is rejected.

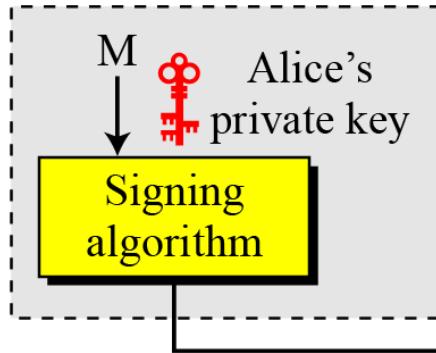


Digital Signature Process

Need for Keys

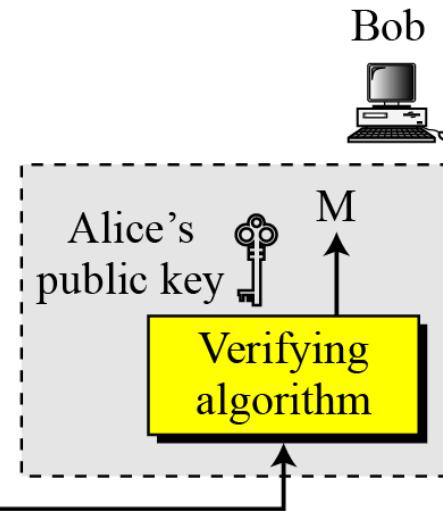
Adding key to the digital signature process

Alice



M: Message
S: Signature

Bob

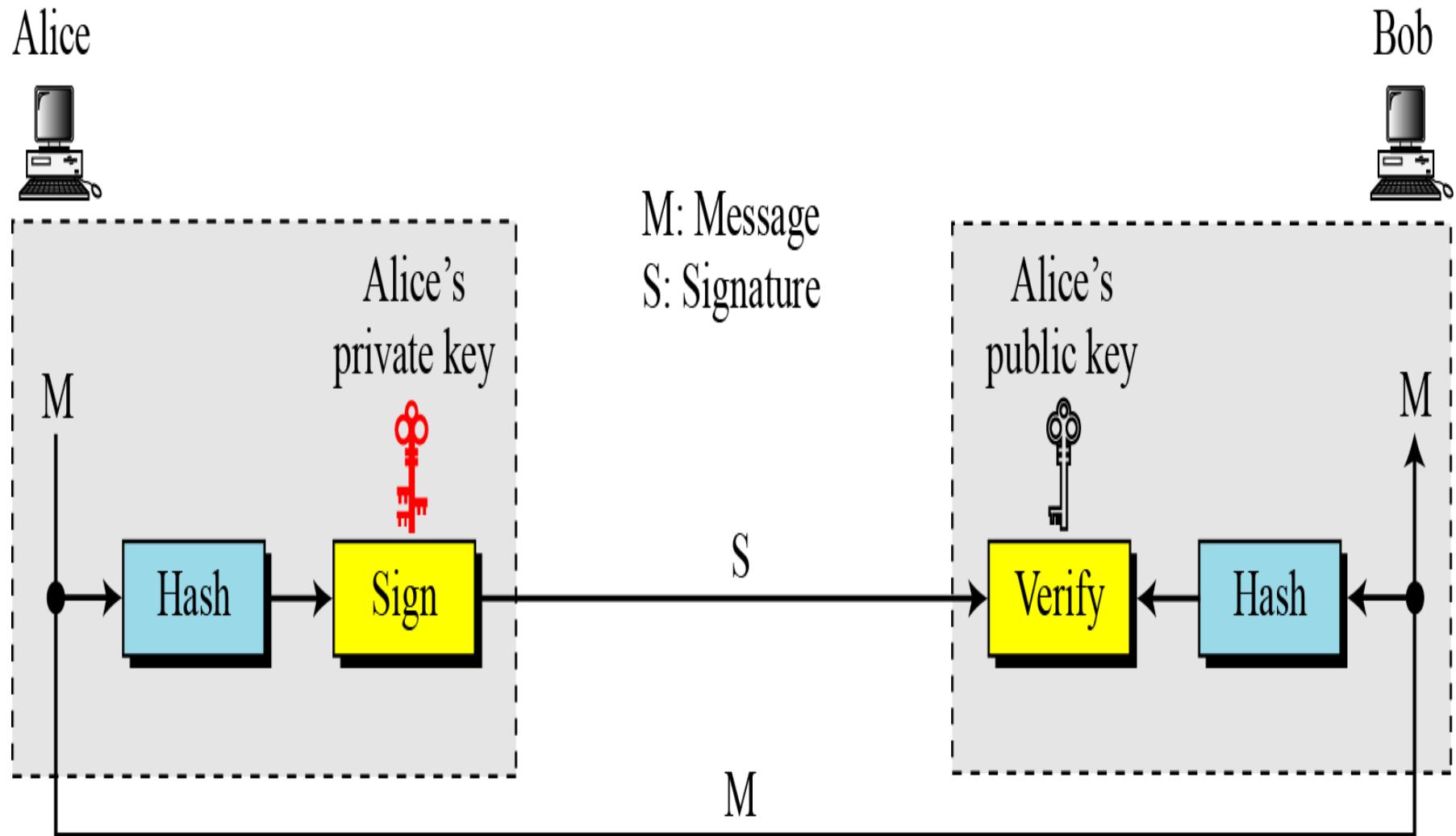


**A digital signature needs a public-key system.
The signer signs with her private key; the verifier
verifies with the signer's public key.**

Continued

A cryptosystem uses the private and public keys of the receiver: a digital signature uses the private and public keys of the sender.

Signing the Digest



SERVICES

A digital signature can directly provide message authentication, message integrity, and nonrepudiation

Message Authentication

A secure digital signature scheme, like a secure conventional signature can provide message authentication.

A digital signature provides message authentication.

Message Integrity

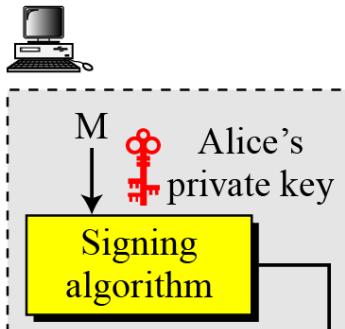
The integrity of the message is preserved even if we sign the whole message because we cannot get the same signature if the message is changed.

A digital signature provides message integrity.

Nonrepudiation

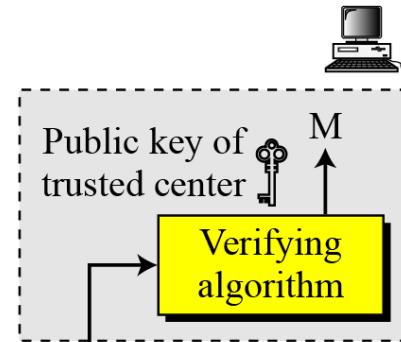
Using a trusted center for nonrepudiation

Alice



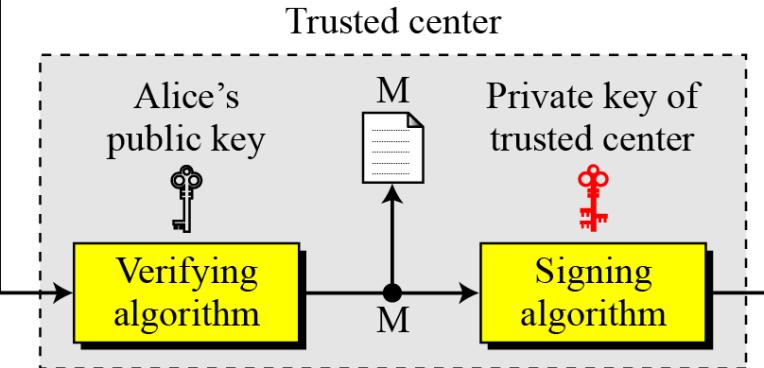
M: Message
S_A: Alice's signature
S_T: Signature of trusted center

Bob



(M, S_A)

Trusted center



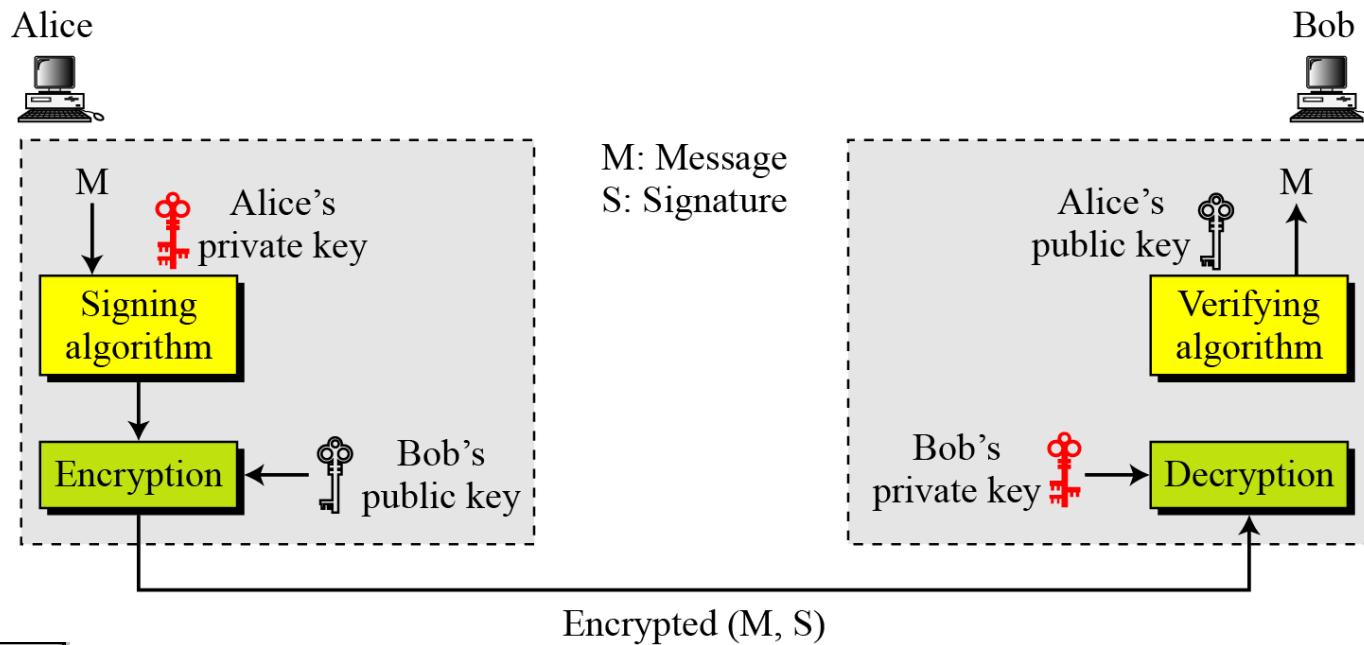
(M, S_T)

Note

Nonrepudiation can be provided using a trusted party.

Confidentiality

Adding confidentiality to a digital signature scheme



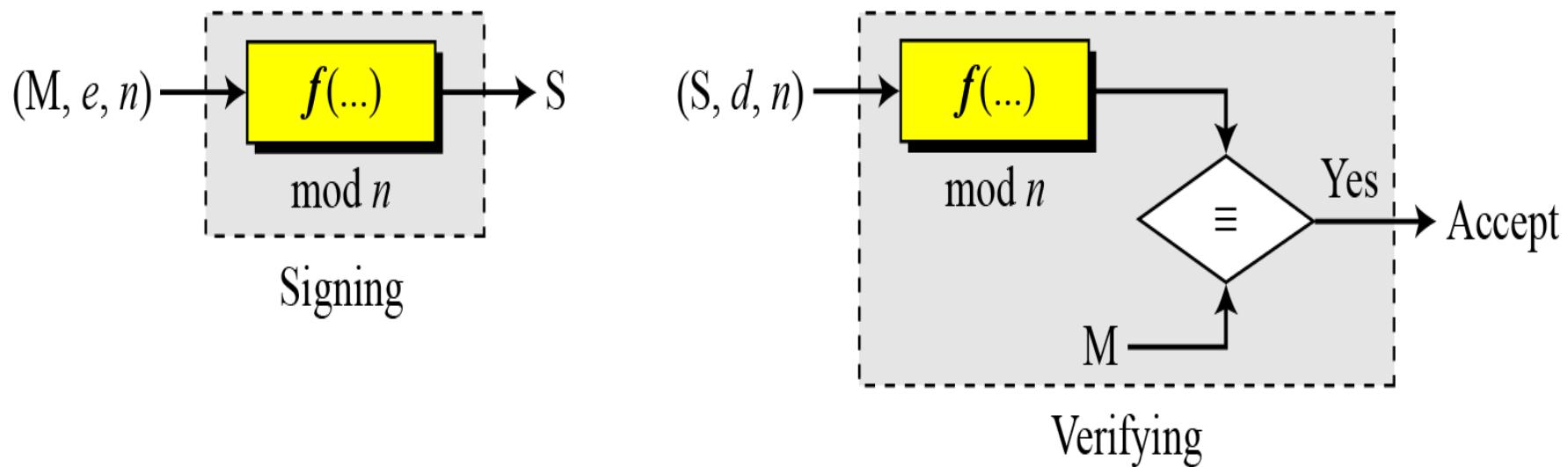
Note

A digital signature does not provide privacy. If there is a need for privacy, another layer of encryption/decryption must be applied.

RSA Digital Signature Scheme

General idea behind the RSA digital signature scheme

M: Message (e, n) : Alice's public key
S: Signature d : Alice's private key



Continued

Key Generation

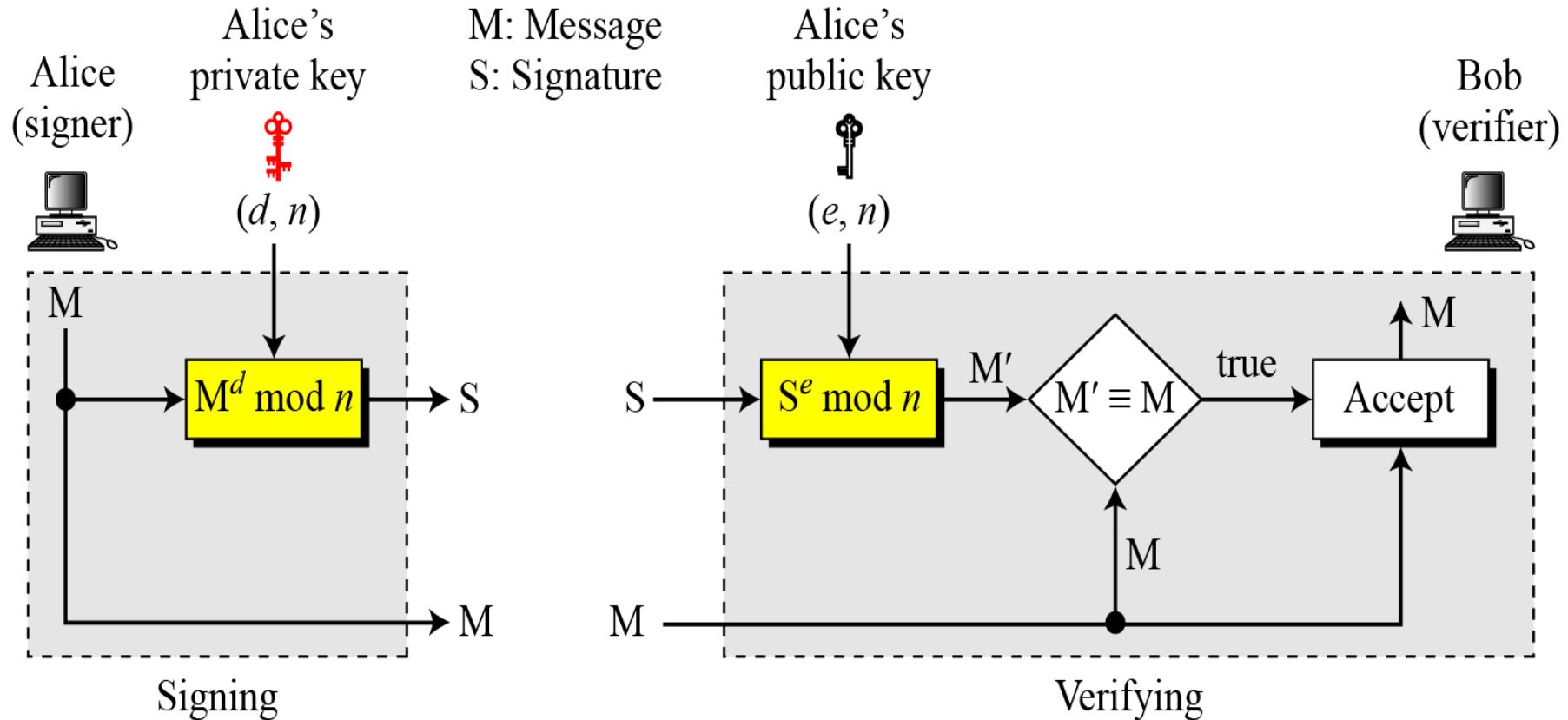
Key generation in the RSA digital signature scheme is exactly the same as key generation in the RSA

Note

In the RSA digital signature scheme, d is private; e and n are public.

Signing and Verifying

RSA digital Signature Scheme



Example

As a trivial example, suppose that Alice chooses $p = 823$ and $q = 953$, and calculates $n = 784319$. The value of $f(n)$ is 782544. Now she chooses $e = 313$ and calculates $d = 160009$. At this point key generation is complete. Now imagine that Alice wants to send a message with the value of $M = 19070$ to Bob. She uses her private exponent, 160009, to sign the message:

$$M: 19070 \rightarrow S = (19070^{160009}) \bmod 784319 = 210625 \bmod 784319$$

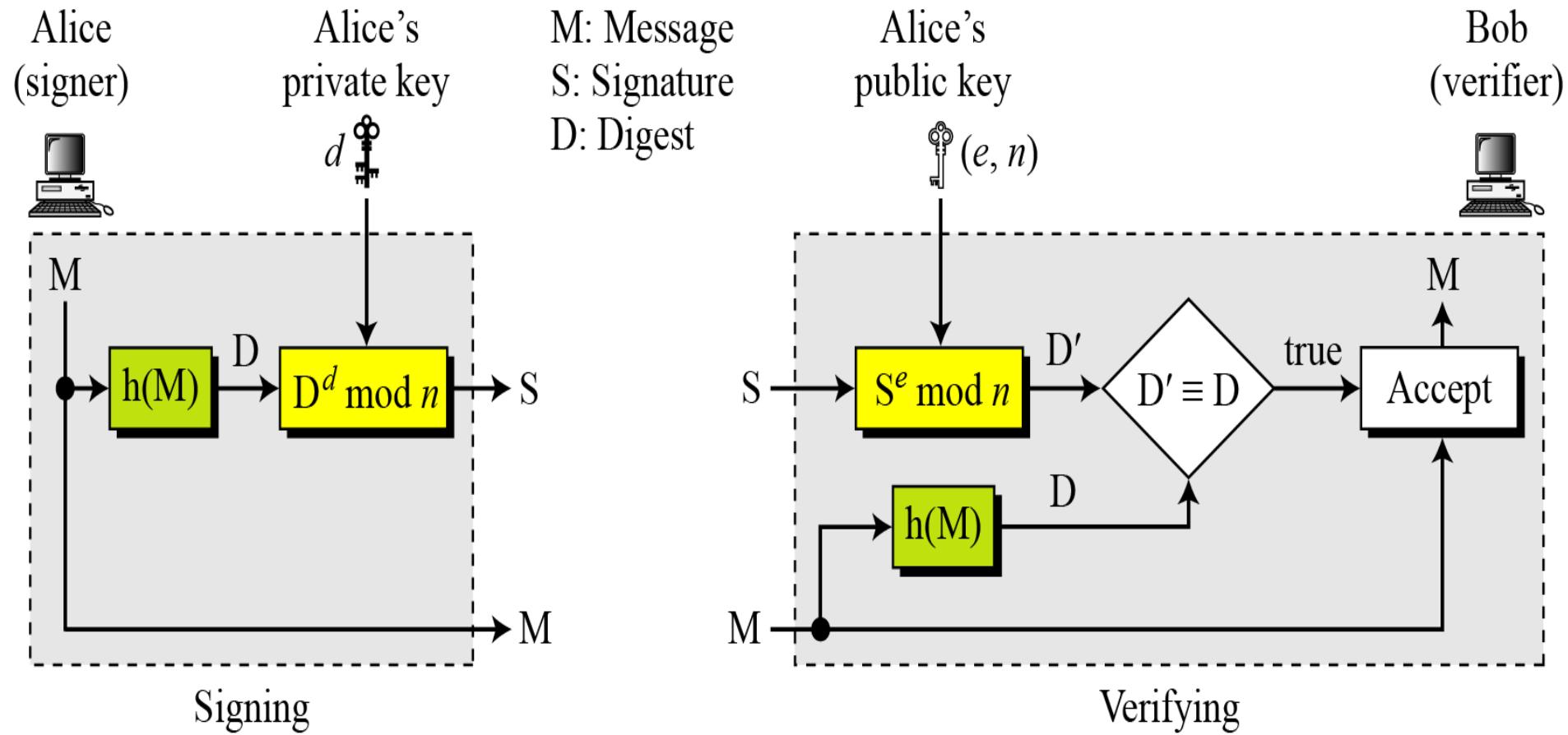
Alice sends the message and the signature to Bob. Bob receives the message and the signature. He calculates

$$M' = 210625^{313} \bmod 784319 = 19070 \bmod 784319 \rightarrow M \equiv M' \bmod n$$

Bob accepts the message because he has verified Alice's signature.

RSA Signature on the Message Digest

The RSA Signature on the Message Digest



When the digest is signed instead of the message itself, the susceptibility of the RSA digital signature scheme depends on the strength of the hash algorithm.

ElGamal Digital Signature Scheme

General idea behind the ElGamal digital signature scheme

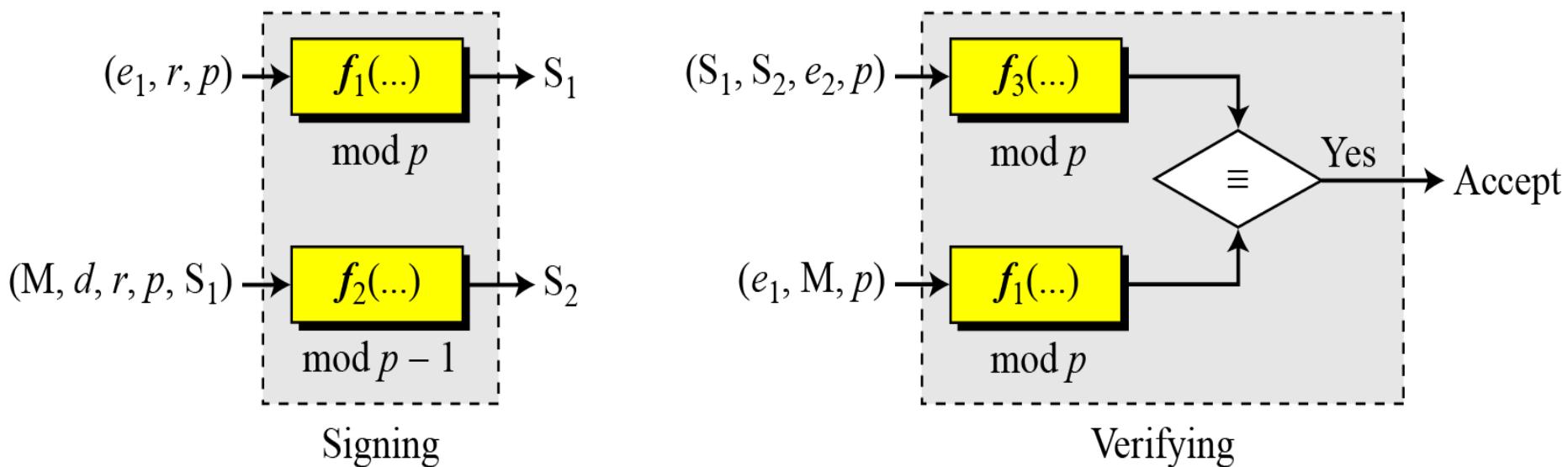
S_1, S_2 : Signatures

M: Message

(e_1, e_2, p) : Alice's public key

d : Alice's private key

r : Random secret



Continued

Key Generation

The key generation procedure here is exactly the same as the one used in the cryptosystem.

**In ElGamal digital signature scheme,
public key is (e_1, e_2, p)
private key is d**

Verifying and Signing

ElGamal Digital Signature Scheme

M: Message

r : Random secret

S_1, S_2 : Signatures

d : Alice's private key

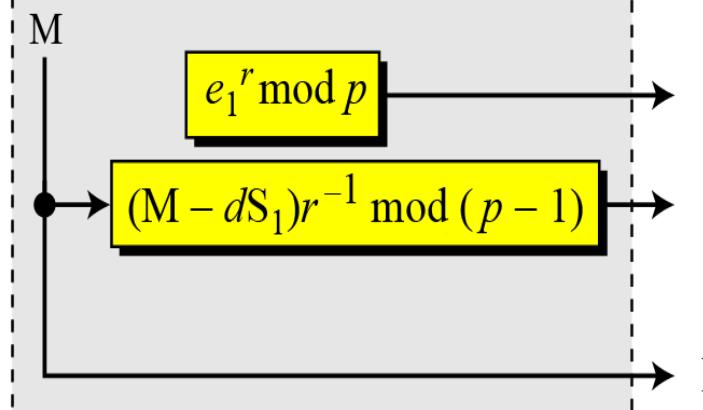
V_1, V_2 : Verifications

(e_1, e_2, p) : Alice's public key

Alice
(signer)

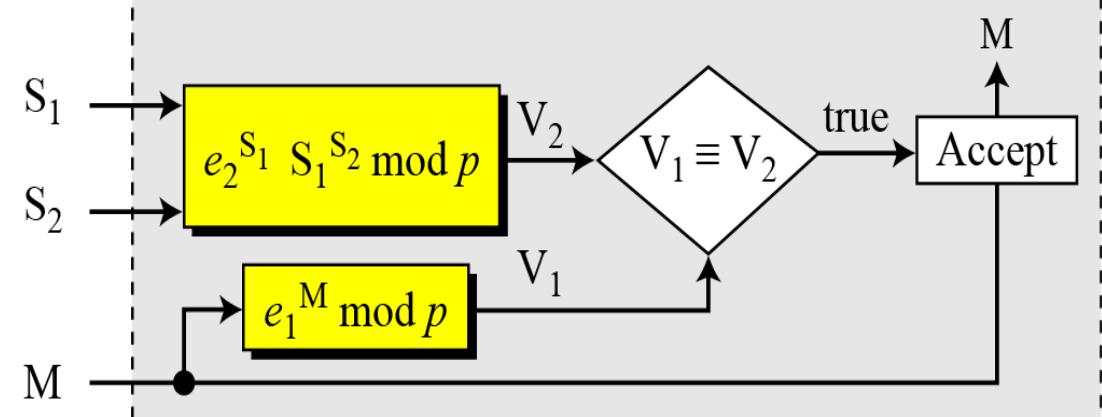
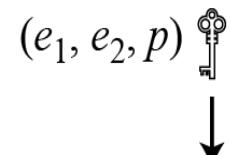


r



Signing

Bob
(verifier)



Verifying

Example 13.2

Here is a trivial example. Alice chooses $p = 3119$, $e_1 = 2$, $d = 127$ and calculates $e_2 = 2^{127} \bmod 3119 = 1702$. She also chooses r to be 307. She announces e_1 , e_2 , and p publicly; she keeps d secret. The following shows how Alice can sign a message.

$$M = 320$$

$$S_1 = e_1^r = 2^{307} = 2083 \bmod 3119$$

$$S_2 = (M - d \times S_1) \times r^{-1} = (320 - 127 \times 2083) \times 307^{-1} = 2105 \bmod 3118$$

Alice sends M , S_1 , and S_2 to Bob. Bob uses the public key to calculate V_1 and V_2 .

$$V_1 = e_1^M = 2^{320} = 3006 \bmod 3119$$

$$V_2 = d^{S_1} \times S_1^{S_2} = 1702^{2083} \times 2083^{2105} = 3006 \bmod 3119$$

Continued

Now imagine that Alice wants to send another message, $M = 3000$, to Ted. She chooses a new r , 107. Alice sends M , S_1 , and S_2 to Ted. Ted uses the public keys to calculate V_1 and V_2 .

$$M = 3000$$

$$S_1 = e_1^r = 2^{107} = 2732 \bmod 3119$$

$$S_2 = (M - d \times S_1) r^{-1} = (3000 - 127 \times 2083) \times 107^{-1} = 2526 \bmod 3118$$

$$V_1 = e_1^M = 2^{3000} = 704 \bmod 3119$$

$$V_2 = d^{S_1} \times S_1^S = 1702^{2732} \times 2083^{2526} = 704 \bmod 3119$$

Schnorr Digital Signature Scheme

General idea behind the Schnorr digital signature scheme

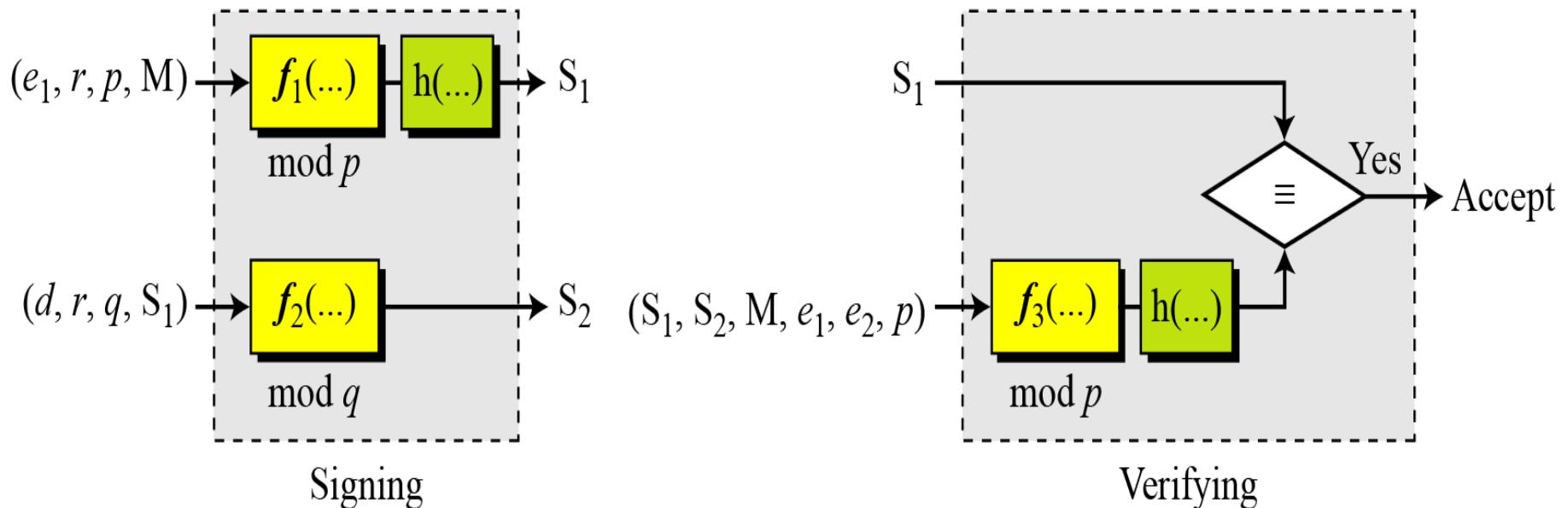
S_1, S_2 : Signatures

(d) : Alice's private key

M : Message

r : Random secret

(e_1, e_2, p, q) : Alice's public key



Key Generation

- 1) Alice selects a prime p , which is usually 1024 bits in length.
- 2) Alice selects another prime q .
- 3) Alice chooses e_1 to be the q th root of 1 modulo p .
- 4) Alice chooses an integer, d , as her private key.
- 5) Alice calculates $e_2 = e_1^d \text{ mod } p$.
- 6) Alice's public key is (e_1, e_2, p, q) ; her private key is (d) .

**In the Schnorr digital signature scheme,
public key is (e_1, e_2, p, q)
private key (d) .**

Signing and Verifying

Schnorr Digital Signature Scheme

M: Message

r : Random secret

| : Concatenation

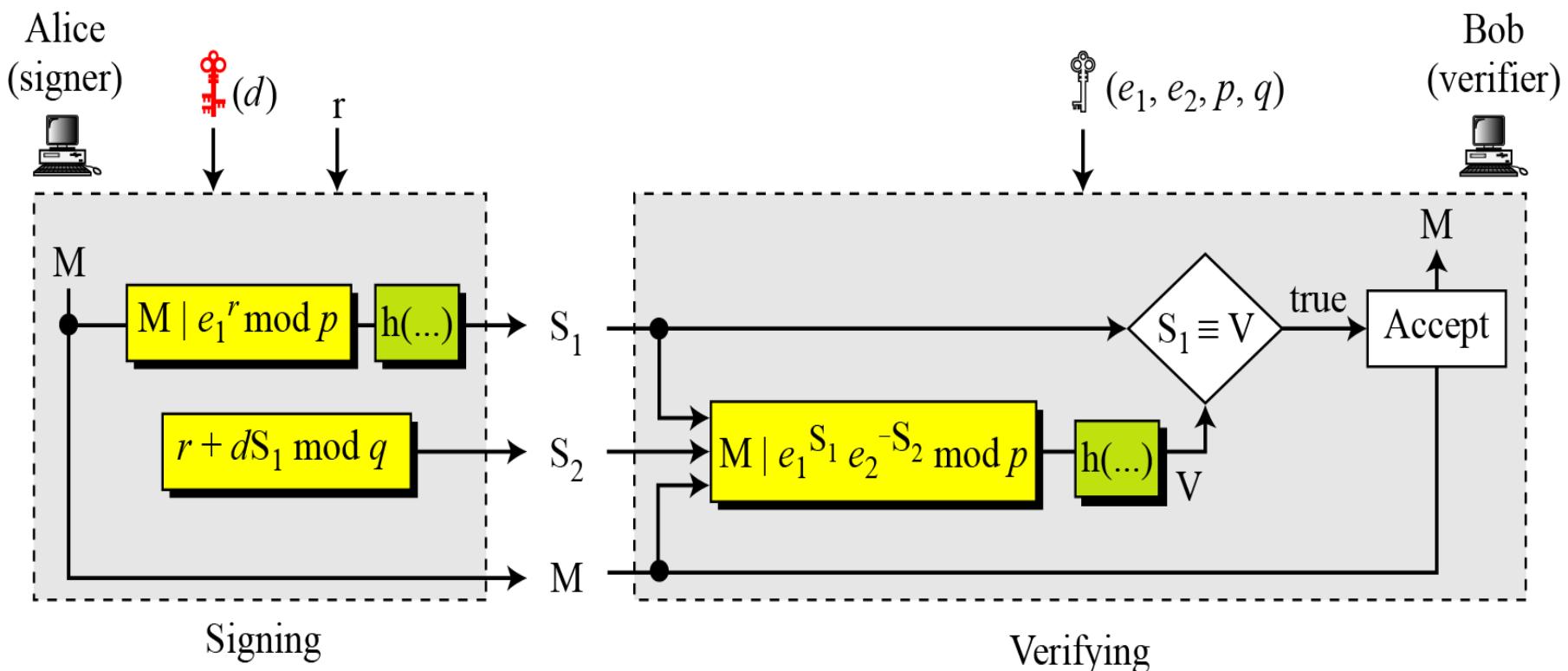
S_1, S_2 : Signatures

(d): Alice's private key

$h(\dots)$: Hash algorithm

V: Verification

(e_1, e_2, p, q): Alice's public key



Signing

1. Alice chooses a random number r .
2. Alice calculates $S_1 = h(M|e_1^r \bmod p)$.
3. Alice calculates $S_2 = r + d \times S_1 \bmod q$.
4. Alice sends M , S_1 , and S_2 .

Verifying Message

1. Bob calculates $V = h(M | e_1^{S_2} e_2^{-S_1} \bmod p)$.
2. If S_1 is congruent to V modulo p , the message is accepted;

Example 13.4

Here is a trivial example. Suppose we choose $q = 103$ and $p = 2267$. Note that $p = 22 \times q + 1$. We choose $e_0 = 2$, which is a primitive in \mathbb{Z}_{2267}^* . Then $(p - 1) / q = 22$, so we have $e_1 = 2^{22} \bmod 2267 = 354$. We choose $d = 30$, so $e_2 = 354^{30} \bmod 2267 = 1206$. Alice's private key is now (d) ; her public key is (e_1, e_2, p, q) .

Alice wants to send a message M . She chooses $r = 11$ and calculates $e_2^r = 354^{11} = 630 \bmod 2267$. Assume that the message is 1000 and concatenation means 1000630. Also assume that the hash of this value gives the digest $h(1000630) = 200$. This means $S_1 = 200$. Alice calculates $S_2 = r + d \times S_1 \bmod q = 11 + 1026 \times 200 \bmod 103 = 35$. Alice sends the message $M = 1000$, $S_1 = 200$, and $S_2 = 35$. The verification is left as an exercise.