

INFORMATION AND NETWORK SECURITY

MODULE 1

INTRODUCTION

Overview of Security

The requirements of **information security** within an organization have undergone two major changes in the last several decades. Before the widespread use of data processing equipment, the security of information felt to be valuable to an organization was provided primarily by physical and administrative means. An example of the former is the use of rugged filing cabinets with a combination lock for storing sensitive documents. An example of the latter is personnel screening procedures used during the hiring process. With the introduction of the computer, the need for automated tools for protecting files and other information stored on the computer became evident. This is especially the case for a shared system, such as a time-sharing system, and the need is even more acute for systems that can be accessed over a public telephone network, data network, or the Internet. The generic name for the collection of tools designed to protect data and to thwart hackers is **computer security**.

The second major change that affected security is the introduction of distributed systems and the use of networks and communications facilities for carrying data between terminal user and computer and between computer and computer. Network security measures are needed to protect data during their transmission. In fact, the term **network security** is somewhat misleading, because virtually all business, government, and academic organizations interconnect their data processing equipment with a collection of interconnected networks. Such a collection is often referred to as an internet, and the term **internet security** is used.

There are no clear boundaries between these two forms of security. For example, one of the most publicized types of attack on information systems is the computer virus. A virus may be introduced into a system physically when it arrives on a diskette or optical disk and is subsequently loaded onto a computer. Viruses may also arrive over an internet. In either case, once the virus is resident on a computer system, internal computer security tools are needed to detect and recover from the virus.

Consider the following examples of security violations:

1. User A transmits a file to user B. The file contains sensitive information (e.g., payroll records) that is to be protected from disclosure. User C, who is not authorized to read the file, is able to monitor the transmission and capture a copy of the file during its transmission.
2. A network manager, D, transmits a message to a computer, E, under its management. The message instructs computer E to update an authorization file to include the identities of a number of new users who are to be given access to that computer. User F intercepts the message, alters its contents to add or delete entries, and then forwards the message to E,

which accepts the message as coming from manager D and updates its authorization file accordingly.

3. Rather than intercept a message, user F constructs its own message with the desired entries and transmits that message to E as if it had come from manager D. Computer E accepts the message as coming from manager D and updates its authorization file accordingly.

4. An employee is fired without warning. The personnel manager sends a message to a server system to invalidate the employee's account. When the invalidation is accomplished, the server is to post a notice to the employee's file as confirmation of the action. The employee is able to intercept the message and delay it long enough to make a final access to the server to retrieve sensitive information. The message is then forwarded, the action taken, and the confirmation posted. The employee's action may go unnoticed for some considerable time.

5. A message is sent from a customer to a stockbroker with instructions for various transactions. Subsequently, the investments lose value and the customer denies sending the message.

Although this list by no means exhausts the possible types of security violations, it illustrates the range of concerns of network security.

1. Security involving communications and networks is not as simple as it might first appear to the novice. The requirements seem to be straightforward; indeed, most of the major requirements for security services can be given self-explanatory one-word labels: confidentiality, authentication, nonrepudiation, integrity. But the mechanisms used to meet those requirements can be quite complex, and understanding them may involve rather subtle reasoning.

2. In developing a particular security mechanism or algorithm, one must always consider potential attacks on those security features. In many cases, successful attacks are designed by looking at the problem in a completely different way, therefore exploiting an unexpected weakness in the mechanism.

3. Because of point 2, the procedures used to provide particular services are often counterintuitive:

It is not obvious from the statement of a particular requirement that such elaborate measures are needed. It is only when the various countermeasures are considered that the measures used make sense.

4. Having designed various security mechanisms, it is necessary to decide where to use them. This is true both in terms of physical placement (e.g., at what points in a network are certain security mechanisms needed) and in a logical sense [e.g., at what layer or layers of an architecture such as TCP/IP (Transmission Control Protocol/Internet Protocol) should mechanisms be placed].

5. Security mechanisms usually involve more than a particular algorithm or protocol. They usually also require that participants be in possession of some secret information (e.g., an encryption key), which raises questions about the creation, distribution, and protection of that secret information. There is also a reliance on communications protocols whose behavior may complicate the task of developing the security mechanism. For example, if the proper functioning of the security mechanism requires setting time limits on the transit

time of a message from sender to receiver, then any protocol or network that introduces variable, unpredictable delays may render such time limits meaningless.

Thus, there is much to consider. This increase in attacks coincides with an increased use of the Internet and with increases in the complexity of protocols, applications, and the Internet itself. Critical infrastructures increasingly rely on the Internet for operations. Individual users rely on the security of the Internet, email, the Web, and Web-based applications to a greater extent than ever. Thus, a wide range of technologies and tools are needed to counter the growing threat. At a basic level, cryptographic algorithms for confidentiality and authentication assume greater importance. As well, designers need to focus on Internet-based protocols and the vulnerabilities of attached operating systems and applications.

The OSI Security Architecture

To assess effectively the security needs of an organization and to evaluate and choose various security products and policies, the manager responsible for security needs some systematic way of defining the requirements for security and characterizing the approaches to satisfying those requirements. This is difficult enough in a centralized data processing environment; with the use of local and wide area networks, the problems are compounded.

ITU-T Recommendation X.800, *Security Architecture for OSI*, defines such a systematic approach.

The OSI security architecture is useful to managers as a way of organizing the task of providing security. Furthermore, because this architecture was developed as an international standard, computer and communications vendors have developed security features for their products and services that relate to this structured definition of services and mechanisms.

The OSI security architecture provides a useful, if abstract, overview of many of the concepts that this book deals with. The OSI security architecture focuses on security attacks, mechanisms, and services. These can be defined briefly as follows:

- **Security attack:** Any action that compromises the security of information owned by an organization.
- **Security mechanism:** A process (or a device incorporating such a process) that is designed to detect, prevent, or recover from a security attack.
- **Security service:** A processing or communication service that enhances the security of the data processing systems and the information transfers of an organization. The services are intended to counter security attacks, and they make use of one or more security mechanisms to provide the service.

Threat

A potential for violation of security, which exists when there is a circumstance, capability, action, or event that could breach security and cause harm. That is, a threat is a possible danger that might exploit a vulnerability.

Attack

An assault on system security that derives from an intelligent threat; that is, an intelligent act that is a deliberate attempt (especially in the sense of a method or technique) to evade security services and violate the security policy of a system.

Security Services

X.800 defines a security service as a service provided by a protocol layer of communicating open systems, which ensures adequate security of the systems or of data transfers. Perhaps a clearer definition is found in RFC 2828, which provides the following definition: a processing or communication service that is provided by a system to give a specific kind of protection to system resources; security services implement security policies and are implemented by security mechanisms.

Authentication

The assurance that the communicating entity is the one that it claims to be.

Peer Entity Authentication

Used in association with a logical connection to provide confidence in the identity of the entities connected.

Data Origin Authentication

In a connectionless transfer, provides assurance that the source of received data is as claimed.

Access Control

The prevention of unauthorized use of a resource (i.e., this service controls who can have access to a resource, under what conditions access can occur, and what those accessing the resource are allowed to do).

Data Confidentiality

The protection of data from unauthorized disclosure.

Connection Confidentiality

The protection of all user data on a connection.

Connectionless Confidentiality

The protection of all user data in a single data block

Selective-Field Confidentiality

The confidentiality of selected fields within the user data on a connection or in a single data block.

Traffic Flow Confidentiality

The protection of the information that might be derived from observation of traffic flows.

Data Integrity

The assurance that data received are exactly as sent by an authorized entity (i.e., contain no modification, insertion, deletion, or replay).

Connection Integrity with Recovery

Provides for the integrity of all user data on a connection and detects any modification, insertion, deletion, or replay of any data within an entire data sequence, with recovery attempted.

Connection Integrity without Recovery

As above, but provides only detection without recovery.

Selective-Field Connection Integrity

Provides for the integrity of selected fields within the user data of a data block transferred over a connection and takes the form of determination of whether the selected fields have been modified, inserted, deleted, or replayed.

Connectionless Integrity

Provides for the integrity of a single connectionless data block and may take the form of detection of data modification. Additionally, a limited form of replay detection may be provided.

Selective-Field Connectionless Integrity

Provides for the integrity of selected fields within a single connectionless data block; takes the form of determination of whether the selected fields have been modified.

Nonrepudiation

Provides protection against denial by one of the entities involved in a communication of having participated in all or part of the communication.

Nonrepudiation, Origin

Proof that the message was sent by the specified party.

Nonrepudiation, Destination

Proof that the message was received by the specified party.

Authentication

The authentication service is concerned with assuring that a communication is authentic. In the case of a single message, such as a warning or alarm signal, the function of the authentication service is to assure the recipient that the message is from the source that it claims to be from. In the case of an ongoing interaction, such as the connection of a terminal to a host, two aspects are involved. First, at the time of connection initiation, the service assures that the two entities are authentic, that is, that each is the entity that it claims to be. Second, the service must assure that the connection is not interfered with in such a way that a third party can masquerade as one of the two legitimate parties for the purposes of unauthorized transmission or reception.

Two specific authentication services are defined in X.800:

- **Peer entity authentication:** Provides for the corroboration of the identity of a peer entity in an association. It is provided for use at the establishment of, or at times during the data transfer phase of, a connection. It attempts to provide confidence that an entity is not performing either a masquerade or an unauthorized replay of a previous connection.
- **Data origin authentication:** Provides for the corroboration of the source of a data unit. It does not provide protection against the duplication or modification of data units. This type of service supports applications like electronic mail where there are no prior interactions between the communicating entities.

Access Control

In the context of network security, access control is the ability to limit and control the access to host systems and applications via communications links. To achieve this, each entity trying to gain access must first be identified, or authenticated, so that access rights can be tailored to the individual.

Data Confidentiality

Confidentiality is the protection of transmitted data from passive attacks. With respect to the content of a data transmission, several levels of protection can be identified. The broadest service protects all user data transmitted between two users over a period of time. For example, when a TCP connection is set up between two systems, this broad protection prevents the release of any user data transmitted over the TCP connection. Narrower forms of this service can also be defined, including the protection of a single message or even specific fields within a message. These refinements are less useful than the broad approach and may even be more complex and expensive to implement.

The other aspect of confidentiality is the protection of traffic flow from analysis. This requires that an attacker not be able to observe the source and destination, frequency, length, or other characteristics of the traffic on a communications facility.

Data Integrity

As with confidentiality, integrity can apply to a stream of messages, a single message, or selected fields. A connection-oriented integrity service, one that deals with a stream of messages, assures that messages are received as sent, with no duplication, insertion, modification, reordering, or replays. The destruction of data is also covered under this

service. Thus, the connection-oriented integrity service addresses both message stream modification and denial of service. On the other hand, a connectionless integrity service, one that deals with individual messages without regard to any larger context, generally provides protection against message modification only.

We can make a distinction between the service with and without recovery. Because the integrity service relates to active attacks, we are concerned with detection rather than prevention. If a violation of integrity is detected, then the service may simply report this violation, and some other portion of software or human intervention is required to recover from the violation. Alternatively, there are mechanisms available to recover from the loss of integrity of data, as we will review subsequently. The incorporation of automated recovery mechanisms is, in general, the more attractive alternative.

Nonrepudiation

Nonrepudiation prevents either sender or receiver from denying a transmitted message. Thus, when a message is sent, the receiver can prove that the alleged sender in fact sent the message. Similarly, when a message is received, the sender can prove that the alleged receiver in fact received the message.

Availability Service

Both X.800 and RFC 2828 define availability to be the property of a system or a system resource being accessible and usable upon demand by an authorized system entity, according to performance specifications for the system (i.e., a system is available if it provides services according to the system design whenever users request them). A variety of attacks can result in the loss of or reduction in availability. Some of these attacks are amenable to automated countermeasures, such as authentication and encryption, whereas others require some sort of physical action to prevent or recover from loss of availability of elements of a distributed system.

X.800 treats availability as a property to be associated with various security services. However, it makes sense to call out specifically an availability service. An availability service is one that protects a system to ensure its availability. This service addresses the security concerns raised by denial-of-service attacks. It depends on proper management and control of system resources and thus depends on access control service and other security services.

Security Mechanisms

The mechanisms are divided into those that are implemented in a specific protocol layer and those that are not specific to any particular protocol layer or security service. These mechanisms will be covered in the appropriate places in the book and so we do not elaborate now, except to comment on the definition of encipherment. X.800 distinguishes between reversible encipherment mechanisms and irreversible encipherment mechanisms. A reversible encipherment mechanism is simply an encryption algorithm that allows data to be encrypted and subsequently decrypted. Irreversible encipherment mechanisms

include hash algorithms and message authentication codes, which are used in digital signature and message authentication applications.

SPECIFIC SECURITY MECHANISMS

May be incorporated into the appropriate protocol layer in order to provide some of the OSI security services.

Encipherment

The use of mathematical algorithms to transform data into a form that is not readily intelligible. The transformation and subsequent recovery of the data depend on an algorithm and zero or more encryption keys.

Digital Signature

Data appended to, or a cryptographic transformation of, a data unit that allows a recipient of the data unit to prove the source and integrity of the data unit and protect against forgery (e.g., by the recipient).

Access Control

A variety of mechanisms that enforce access rights to resources.

Data Integrity

A variety of mechanisms used to assure the integrity of a data unit or stream of data units.

Authentication Exchange

A mechanism intended to ensure the identity of an entity by means of information exchange.

Traffic Padding

The insertion of bits into gaps in a data stream to frustrate traffic analysis attempts.

Routing Control

Enables selection of particular physically secure routes for certain data and allows routing changes, especially when a breach of security is suspected.

Notarization

The use of a trusted third party to assure certain properties of a data exchange.

PERVASIVE SECURITY MECHANISMS

Mechanisms that are not specific to any particular OSI security service or protocol layer.

Trusted Functionality

That which is perceived to be correct with respect to some criteria (e.g., as established by a security policy).

Security Label

The marking bound to a resource (which may be a data unit) that names or designates the security attributes of that resource.

Event Detection

Detection of security-relevant events.

Security Audit Trail

Data collected and potentially used to facilitate a security audit, which is an independent review and examination of system records and activities.

Security Recovery

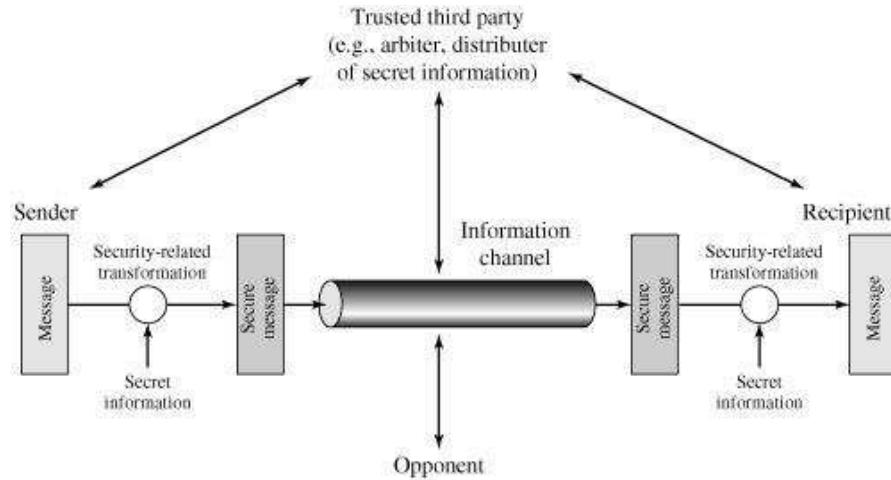
Deals with requests from mechanisms, such as event handling and management functions, and takes recovery actions.

Relationship between Security Services and Mechanisms

Mechanism									
Service	Encipherment	Digital Signature	Access Control	Data Integrity	Authentication Exchange	Traffic Padding	Routing Control	Notarization	
Peer entity authentication	Y	Y			Y				
Data origin authentication	Y	Y							
Access control			Y						
Confidentiality	Y						Y		
Traffic flow confidentiality	Y					Y	Y		
Data integrity	Y	Y		Y					
Nonrepudiation		Y		Y				Y	
Availability				Y	Y				

A Model for Network Security

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



A Model for Network Security

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

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

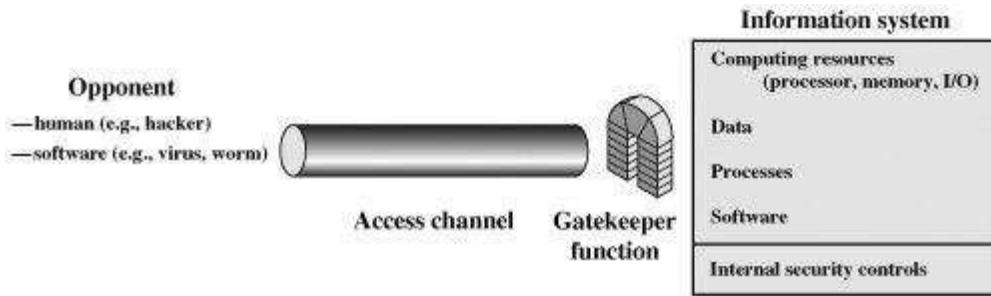
An example is an encryption key used in conjunction with the transformation to scramble the message before transmission and unscramble it on reception.

A trusted third party may be needed to achieve secure transmission. For example, a third party may be responsible for distributing the secret information to the two principals while keeping it from any opponent. Or a third party may be needed to arbitrate disputes between the two principals concerning the authenticity of a message transmission.

This general model shows that there are four basic tasks in designing a particular security service:

1. Design an algorithm for performing the security-related transformation. The algorithm should be such that an opponent cannot defeat its purpose.
2. Generate the secret information to be used with the algorithm.
3. Develop methods for the distribution and sharing of the secret information.
4. Specify a protocol to be used by the two principals that makes use of the security algorithm and the secret information to achieve a particular security service.

Network Access Security Model



Network Access Security Model

Another type of unwanted access is the placement in a computer system of logic that exploits vulnerabilities in the system and that can affect application programs as well as utility programs, such as editors and compilers.

Programs can present two kinds of threats:

- **Information access threats** intercept or modify data on behalf of users who should not have access to that data.
- **Service threats** exploit service flaws in computers to inhibit use by legitimate users.

Viruses and worms are two examples of software attacks. Such attacks can be introduced into a system by means of a disk that contains the unwanted logic concealed in otherwise useful software. They can also be inserted into a system across a network; this latter mechanism is of more concern in network security.

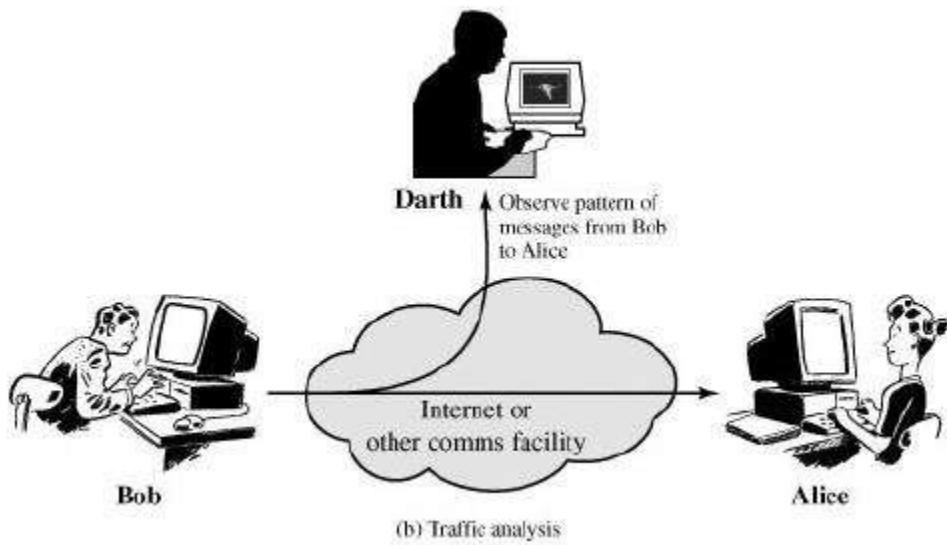
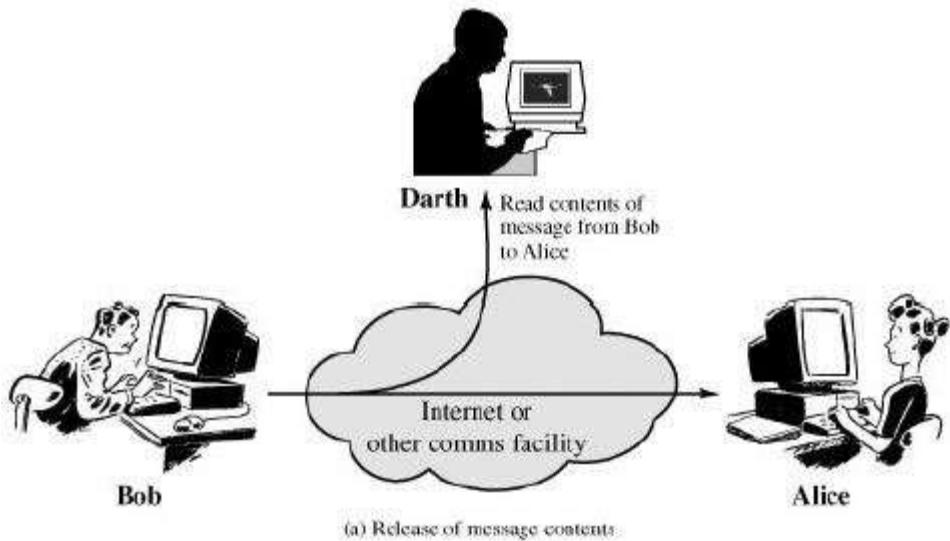
Security Attacks

A useful means of classifying security attacks, used both in X.800 and RFC 2828, is in terms of *passive attacks* and *active attacks*. A passive attack attempts to learn or make use of information from the system but does not affect system resources. An active attack attempts to alter system resources or affect their operation.

Passive Attacks

Passive attacks are in the nature of eavesdropping on, or monitoring of, transmissions. The goal of the opponent is to obtain information that is being transmitted. Two types of passive attacks are release of message contents and traffic analysis.

The **release of message contents** is easily understood. A telephone conversation, an electronic mail message, and a transferred file may contain sensitive or confidential information. We would like to prevent an opponent from learning the contents of these transmissions.



A second type of passive attack, **traffic analysis**, is subtler. Suppose that we had a way of masking the contents of messages or other information traffic so that opponents, even if they captured the message, could not extract the information from the message. The common technique for masking contents is encryption. If we had encryption protection in place, an opponent might still be able to observe the pattern of these messages. The opponent could determine the location and identity of communicating hosts and could observe the frequency and length of messages being exchanged. This information might be useful in guessing the nature of the communication that was taking place. Passive attacks are very difficult to detect because they do not involve any alteration of the data.

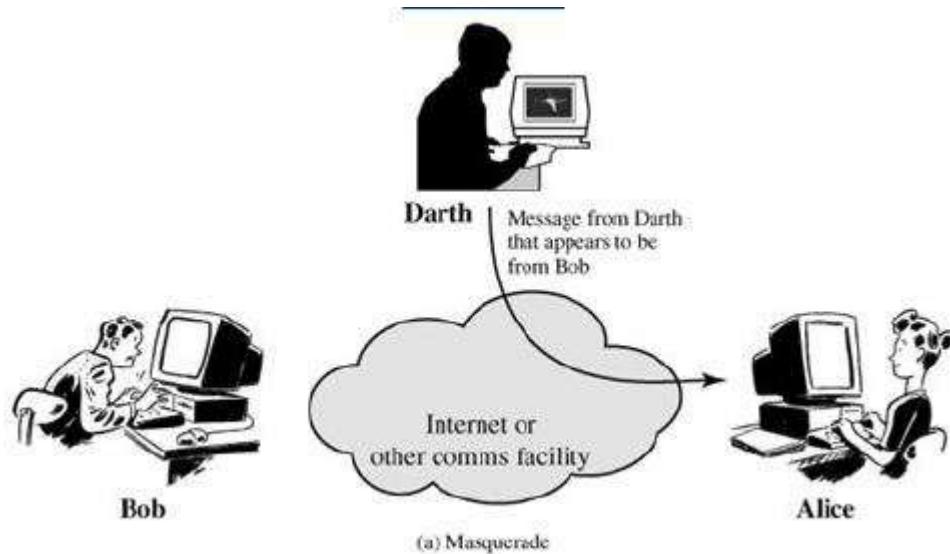
Typically, the message traffic is sent and received in an apparently normal fashion and neither the sender nor receiver is aware that a third party has read the messages or observed the traffic pattern.

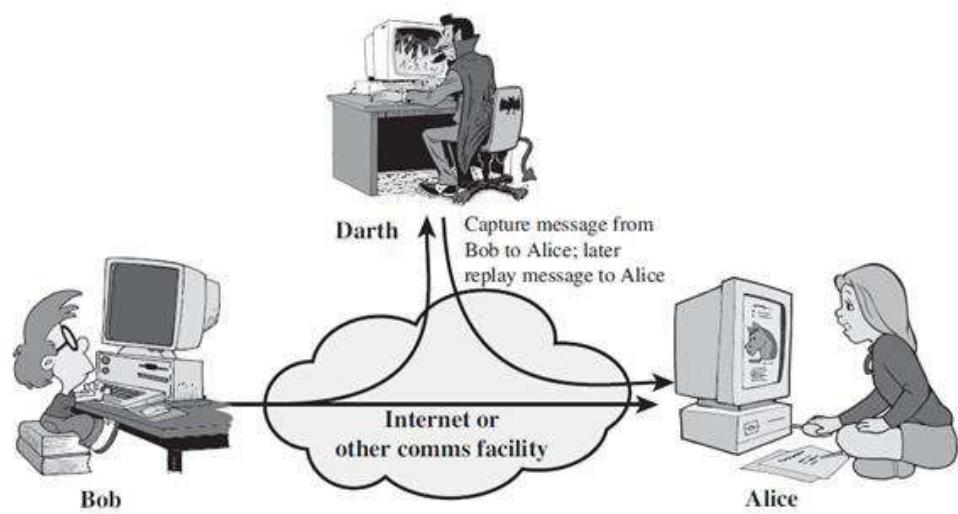
However, it is feasible to prevent the success of these attacks, usually by means of encryption. Thus, the emphasis in dealing with passive attacks is on prevention rather than detection.

Active Attacks

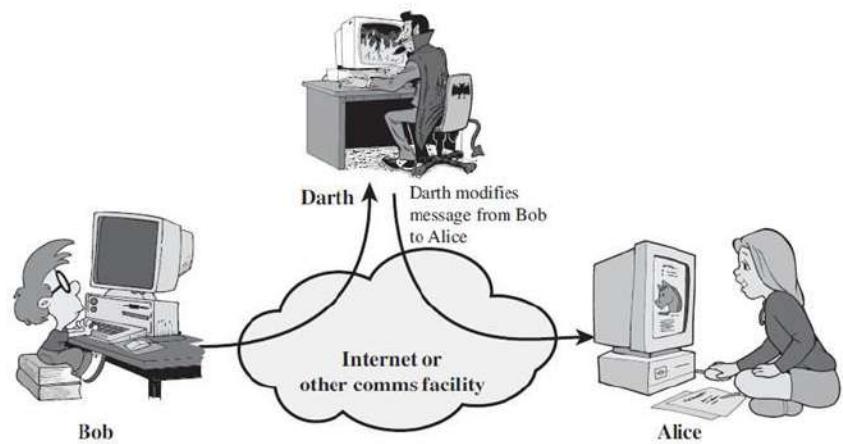
Active attacks involve some modification of the data stream or the creation of a false stream and can be subdivided into four categories: masquerade, replay, modification of messages, and denial of service.

A **masquerade** takes place when one entity pretends to be a different entity. A masquerade attack usually includes one of the other forms of active attack. For example, authentication sequences can be captured and replayed after a valid authentication sequence has taken place, thus enabling an authorized entity with few privileges to obtain extra privileges by impersonating an entity that has those privileges.

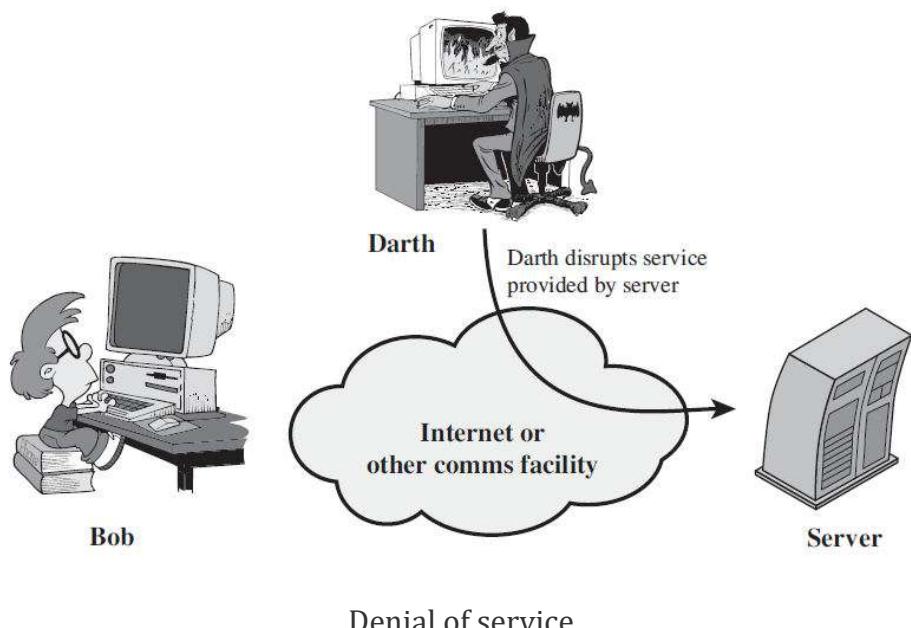




Replay



Modification of messages



Replay involves the passive capture of a data unit and its subsequent retransmission to produce an unauthorized effect

Modification of messages simply means that some portion of a legitimate message is altered, or that messages are delayed or reordered, to produce an unauthorized effect. For example, a message meaning "Allow John Smith to read confidential file *accounts*" is modified to mean "Allow Fred Brown to read confidential file *accounts*."

The **denial of service** prevents or inhibits the normal use or management of communications facilities. This attack may have a specific target; for example, an entity may suppress all messages directed to a particular destination (e.g., the security audit service). Another form of service denial is the disruption of an entire network, either by disabling the network or by overloading it with messages so as to degrade performance. Active attacks present the opposite characteristics of passive attacks. Whereas passive attacks are difficult to detect, measures are available to prevent their success. On the other hand, it is quite difficult to prevent active attacks absolutely, because of the wide variety of potential physical, software, and network vulnerabilities. Instead, the goal is to detect active attacks and to recover from any disruption or delays caused by them. If the detection has a deterrent effect, it may also contribute to prevention.

Differences between active and passive attacks

Sr. No.	Active Attack	Passive Attack
1.	In active attack, Modification in information take place.	While in passive attack, Modification in the information does not take place.
2.	Active Attack is danger for Integrity as well as availability .	Passive Attack is danger for Confidentiality .
3.	In active attack attention is on detection.	While in passive attack attention is on prevention.
4.	Due to active attack system is always damaged.	While due to passive attack, there is no any harm to the system.
5.	In active attack, Victim gets informed about the attack.	While in passive attack, Victim does not get informed about the attack.
6.	In active attack, System resources can be changed.	While in passive attack, System resources are not change.

Access Control Matrix

Protection State:

- The **state** of a system is the collection of the current values of all memory locations, all secondary storage, and all registers and other components of the system.
- The subset of this collection that deals with protection is the **protection state** of the system.

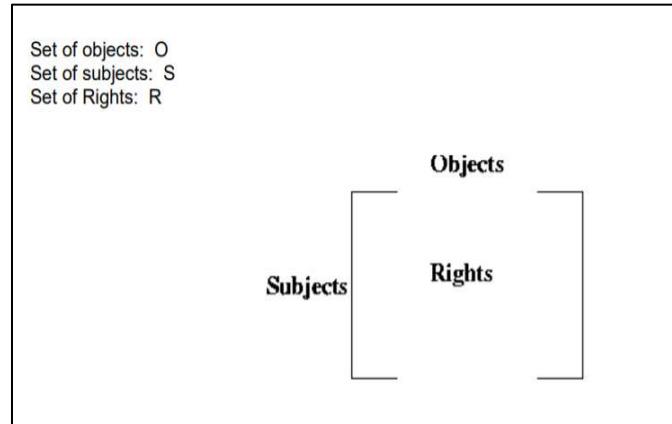
An **access control matrix** is one tool that can describe the current protection state

- Consider the set of possible protection states P .
- Some subset Q of P consists of exactly those states in which the system is authorized to reside.
- So, whenever the system state is in Q , the system is secure.
- When the current state is in $P - Q$, the system is not secure.
- Characterizing the states in Q is the function of a **security policy**.
- Preventing the system from entering a state in $P - Q$ is the function of a **security mechanism**.
- The **access control matrix model** is the most precise model used to describe a protection state.
- It characterizes the rights of each **subject** (active entity, such as a process) with respect to every other entity.

Access Control Matrix Model

- Access Control Matrix** or **Access Matrix** is an abstract, formal security model of protection state in computer systems that characterize the rights of each subject with respect to every object in the system.
- The set of all protected entities (ie., entities that are relevant to the protection state of the system) is called the set of **objects** O .

- The set of *subjects* S is the set of active objects, such as processes and users.
- In the access control matrix model, the relationship between these entities is captured by a matrix A with *rights* drawn from a set of rights R in each entry $a[s, o]$, where $s \in S$, $o \in O$, and $a[s, o] \subseteq R$.
- The subject s has the set of rights $a[s, o]$ over the object o .
- The set of protection *states* of the system is represented by the triple (S, O, A) .



Example:

	file 1	file 2	process 1	process 2
process 1	read, write, own	read	read, write, execute, own	write
process 2	append	read, own	read	read, write, execute, own

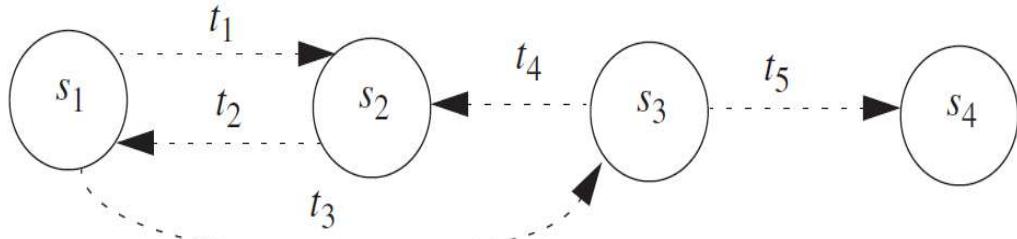
An access control matrix: The system has two processes and two files. The set of rights is {read, write, execute, append, own}.

Security Policies

Consider a computer system to be finite-state automation with a set of transition functions that change state. Then:

- A **security policy** is a statement that partitions the states of the system into a set of *authorized*, or *secure*, states and a set of *unauthorized*, or *non-secure*, states.
- A **secure system** is a system that starts in an authorized state and cannot enter an unauthorized state.
- Consider the finite-state machine.
- It consists of four states and five transitions.

- The security policy partitions the states into a set of authorized states $A = \{ s1, s2 \}$ and a set of unauthorized states $UA = \{ s3, s4 \}$.



A simple finite-state machine: In this example, the authorized states are $s1$ and $s2$.

- This system is not secure, because regardless of which authorized state it starts in, it can enter an unauthorized state.
- However, if the edge from $s1$ to $s3$ were not present, the system would be secure, because it could not enter an unauthorized state from an authorized state.
- A **breach of security** occurs when a system enters an unauthorized state.
- Let X be a set of entities and let I be some information. Then I has the property of **confidentiality** with respect to X if no member of X can obtain information about I .
- Let X be a set of entities and let I be some information or a resource. Then I has the property of **integrity** with respect to X if all members of X trust I .
- Let X be a set of entities and let I be a resource. Then I has the property of **availability** with respect to X if all members of X can access I .
- A **security mechanism** is an entity or procedure that enforces some part of the security policy.
- A **security model** is a model that represents a particular policy or set of policies.

Types of Security Policies

1. A **military security policy** (also called a *governmental security policy*) is a security policy developed primarily to provide confidentiality.
- The name comes from the military's need to keep information, such as the date that a troop ship will sail, secret.
2. A **commercial security policy** is a security policy developed primarily to provide integrity.
- The name comes from the need of commercial firms to prevent tampering with their data, because they could not survive such compromises.
3. A **confidentiality policy** is a security policy dealing only with confidentiality.
4. An **integrity policy** is a security policy dealing only with integrity.

Types of Access Control

- A security policy may use two types of access controls, alone or in combination.
- In one, access control is left to the discretion of the owner.
- In the other, the operating system controls access, and the owner cannot override the controls.
- If an individual user can set an access control mechanism to allow or deny access to an object, that mechanism is a **discretionary access control (DAC)**, also called an **identity-based access control (IBAC)**.
- When a system mechanism controls access to an object and an individual user cannot alter that access, the control is a **mandatory access control (MAC)**, occasionally called a **rule-based access control**.
- An **originator controlled access control (ORCON or ORGCON)** bases access on the creator of an object (or the information it contains).

Confidentiality Policies

Goals of Confidentiality Policies:

- A confidentiality policy, also called an *information flow policy*, prevents the unauthorized disclosure of information.
- Unauthorized alteration of information is secondary.
- Example: Privacy Act requires that certain personal data be kept confidential. E.g., income tax returns are legally confidential & are only available to Internal Revenue Service or to legal authority with court order. It limits the distribution of documents/information.

The Bell-LaPadula Model

- The Bell-LaPadula Model corresponds to military-style classifications.
- The **Bell-LaPadula Model (BLP)** is a state machine model used for enforcing access control in government and military applications.
- The Bell-LaPadula model is a security method of keeping files confidential.
- The US government uses **classification levels**, which are rated lowest to highest: Unclassified, Confidential, Secret and Top Secret to a file.
- It describes a set of access control rules which use security labels on objects and clearances for subjects.
- The simplest type of confidentiality classification is a set of *security clearances* arranged in a linear (total) ordering.

TOP SECRET (TS)	Tamara, Thomas	Personnel Files
SECRET (S)	Sally, Samuel	Electronic Mail Files
CONFIDENTIAL (C)	Claire, Clarence	Activity Log Files
UNCLASSIFIED (UC)	Ulaley, Ursula	Telephone List Files

At the left is the basic confidentiality classification system. The four security levels are arranged with the most sensitive at the top and the least sensitive at the bottom. In the middle are individuals grouped by their security clearances, and at the right is a set of documents grouped by their security levels.

- These clearances represent sensitivity levels.
- The higher the security clearance, the more sensitive the information (and the greater the need to keep it confidential).
- A subject has a *security clearance*.
- The goal of the Bell-LaPadula security model is to prevent read access to objects at a security classification higher than the subject's clearance.
- The Bell-LaPadula security model combines mandatory and discretionary access controls.
- The set of access rights given to a subject are the following:
- **Read-Only:** The subject can only read the object.
- **Append:** The subject can only write to the object but it cannot read.
- **Execute:** The subject can execute the object but can neither read nor write.
- **Read-Write:** The subject has both read and write permissions to the object

Restrictions imposed by the Bell-Lapadula Model:

The following restrictions are imposed by the model:

1. **reading down:** A subject has only read access to objects whose security level is below the subject's current clearance level. This prevents a subject from getting access to information available in security levels higher than its current clearance level.
 2. **writing up:** A subject has append access to objects whose security level is higher than its current clearance level. This prevents a subject from passing information to levels lower than its current level.
- Let $L(S) = ls$ be the security clearance of subject S , and
 - Let $L(O) = lo$ be the security classification of object O .
 - For all security classifications li , $i = 0, \dots, k - 1$, $li < li+1$.
 - **Simple Security Condition, Preliminary Version:**
 S can read O if and only if $lo \leq ls$ and
 S has discretionary read access to O .
 - **Property (Star property) Preliminary Version:**
 S can write O if and only if $ls \leq lo$ and
 S has discretionary write access to O .
 - TS (Top secret) cannot write documents lower than TS. → Prevent classified information leak.

Basic Security Theorem, Preliminary Version

- Let Σ be a system with secure initial state σ_0
- Let T be the set of state transformations.
- If every element of T preserves the simple security condition, preliminary version, and the $*$ -property, preliminary version, then every state σ_i , $i \geq 0$, is secure.

Integrity Policies

- **Problem area:** systems require data to be changed accurately and follow the rules. Disclosure is not a major concern.
- Lipner identifies five requirements for preserving data integrity:
 1. Users will not write their own programs, but will use existing production programs and databases.
 2. Programmers will develop and test programs on a nonproduction system; if they need access to actual data, they will be given production data via a special process, but will use it on their development system.
 3. A special process must be followed to install a program from the development system onto the production system.
 4. The special process in requirement 3 must be controlled and audited.
 5. The managers and auditors must have access to both the system state and the system logs that are generated.

The requirements suggest 3 principles of operation:

1. Separation of duty
2. Separation of function
3. Auditing

Separation of duty

- The principle of separation of duty states that if two or more steps are required to perform a critical function, at least two different people should perform the steps.
- Moving a program from the development system to the production system is an example of a critical function.

Separation of function

- Developers do not develop new programs on production systems because of the potential threat to production data.
- Similarly, the developers do not process production data on the development systems.
- Depending on the sensitivity of the data, the developers and testers may receive sanitized production data.
- The development environment must be as similar as possible to the actual production environment.

Auditing

- Commercial systems emphasize recovery and accountability.

- Auditing is the process of analyzing systems to determine what actions took place and who performed them.

Biba Integrity Model

- In his model, a system consists of a set S of subjects, a set O of objects, and a set I of integrity levels.
- The levels are ordered.
- Data at a higher level is more accurate and/or reliable (with respect to some metric) than data at a lower level.
- Integrity labels, in general, are not also security labels.
- They are assigned and maintained separately, because the reasons behind the labels are different.
- Security labels primarily limit the flow of information; integrity labels primarily inhibit the modification of information.
- Biba's model is the dual of the Bell-LaPadula Model. Its rules are as follows.

1. $s \in S$ can read $o \in O$ if and only if $i(s) \leq i(o)$.
2. $s \in S$ can write to $o \in O$ if and only if $i(o) \leq i(s)$.
3. $s_1 \in S$ can execute $s_2 \in S$ if and only if $i(s_2) \leq i(s_1)$.

Note: rules 1 and 2 imply that if both read and write are allowed, $i(s) = i(o)$.

Clark-Wilson Integrity Model

- This model uses transactions as the basic operation, which models many commercial systems more realistically than previous models.
- One main concern of a commercial environment is the integrity of the data in the system and of the actions performed on that data.
- The data is said to be in a consistent state (or consistent) if it satisfies given properties.
- For example, let D be the amount of money deposited so far today, W the amount of money withdrawn so far today, YB the amount of money in all accounts at the end of yesterday, and TB the amount of money in all accounts so far today.
- Then the consistency property is: $D + YB - W = TB$
- Before and after each action, the consistency conditions must hold.
- A well-formed transaction is a series of operations that transition the system from one consistent state to another consistent state.
- For example, if a depositor transfer's money from one account to another, the transaction is the transfer; two operations, the deduction from the first account and the addition to the second account, make up this transaction. Each operation may leave the data in an inconsistent state, but the well-formed transaction must preserve consistency.
- The second feature of a commercial environment relevant to an integrity policy is the integrity of the transactions themselves.
- Someone must certify that the transactions are implemented correctly.

- The principle of separation of duty requires that the certifier and the implementers be different people.
- In order for the transaction to corrupt the data (either by illicitly changing the data or by leaving the data in an inconsistent state), two different people must either make similar mistakes or collude to certify the well-formed transaction as correct.

The Model:

- The Clark-Wilson model defines data subject to its integrity controls as constrained data items, or CDIs.
- Data not subject to the integrity controls are called unconstrained data items, or UDIs.
- For example, in a bank, the balances of accounts are CDIs since their integrity is crucial to the operation of the bank, whereas the gifts selected by the account holders when their accounts were opened would be UDIs, because their integrity is not crucial to the operation of the bank.
- The set of CDIs and the set of UDIs partition the set of all data in the system being modeled.
- The model also defines two sets of procedures.
- ***Integrity verification procedures***, or IVPs, test that the CDIs conform to the integrity constraints at the time the IVPs are run. In this case, the system is said to be in a *valid state*.
- ***Transformation procedures***, or TPs, change the state of the data in the system from one valid state to another; TPs implement well-formed transactions.
- **Example:** The balances in the accounts are CDIs; checking that the accounts are balanced, as described above, is an IVP. Depositing money, withdrawing money, and transferring money between accounts are TPs.

Certification Rules:

- **Certification rule 1 (CR1):** When any IVP is run, it must ensure that all CDIs are in a valid state.
- **Certification rule 2 (CR2):** For some associated set of CDIs, a TP must transform those CDIs in a valid state into a (possibly different) valid state.
- CR2 defines as *certified* a relation that associates a set of CDIs with a particular TP.
- Let C be the certified relation. Then, in the bank example,
 $(\text{balance}, \text{account1}), (\text{balance}, \text{account2}), \dots, (\text{balance}, \text{accountn}) \in C$.
- CR2 implies that a TP may corrupt a CDI if it is not certified to work on that CDI.
- **For example**, the TP that invests money in the bank's stock portfolio would corrupt account balances even if the TP were certified to work on the portfolio, because the actions of the TP make no sense on the bank accounts.
 - Hence, the system must prevent TPs from operating on CDIs for which they have not been certified.

This leads to the following *enforcement rule*:

- **Enforcement rule 1 (ER1):** The system must maintain the *certified* relations, and must ensure that only TPs certified to run on a CDI manipulate that CDI.
- **Enforcement rule 2 (ER2):** The system must associate a user with each TP and set of CDIs.

- The TP may access those CDIs on behalf of the associated user.
- If the user is not associated with a particular TP and CDI, then the TP cannot access that CDI on behalf of that user.
- Certification rule 3 (CR3):** The *allowed* relations must meet the requirements imposed by the principle of separation of duty.
- Enforcement rule 3 (ER3):** The system must authenticate each user attempting to execute a TP.
- Certification rule 4 (CR4):** All TPs must append enough information to reconstruct the operation to an append-only CDI.
- Certification rule 5 (CR5):** Any TP that takes as input a UDI may perform only valid transformations, or no transformations, for all possible values of the UDI. The transformation either rejects the UDI or transforms it into a CDI.
- Enforcement rule 4 (ER4):** Only the certifier of a TP may change the list of entities associated with that TP. No certifier of a TP, or of an entity associated with that TP, may ever have execute permission with respect to that entity.

Hybrid Policies

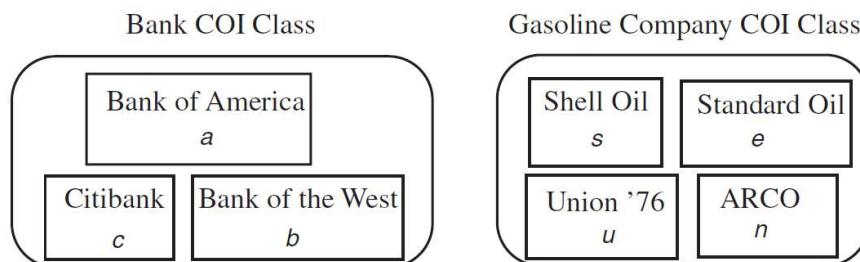
Chinese Wall Model:

- Security policy that refers equally to confidentiality and integrity.
- It describes policies that involve a conflict of interest in business.
- Problem:**
 - Consultant advises Bank1 and Bank2 about investments.
 - Conflict of interest: advice for either bank would affect advice to the other bank
- Solution:**
 - Consultant can only access objects on his/her side of the wall

Definitions:

- Objects:** items of information related to a company.
- Company dataset (CD):** collection of objects related to a single company.
- Conflict of interest class (COI):** collection of datasets of companies in competition
 - Let $COI(O)$ represent the COI class that contains object O , and let $CD(O)$ be the company dataset that contains object O .
 - The model assumes that each object belongs to exactly one COI class.

Example:



The Chinese wall model database: It has two COI classes. The one for banks contains three CDs. The other one, for gasoline companies, contains four CDs. Each (COI, CD) pair is represented by a lowercase letter (for example, (Bank COI, Citibank) is c). Susan may have access to no more than one CD in each COI, so she could access Citibank's CD and ARCO's CD, but not Citibank's CD and Bank of America's CD.

Temporal Element:

- Rights depend on access history.
 - Initially, a subject can read any object in any CD of any COI.
 - If a subject reads an object in a CD in a COI, he can never read an object in another CD in the same COI.
 - Possible that information learned earlier may allow him to make decisions later
- $PR(S)$ denotes the set of objects that a subject S has already read.
- ***CW-Simple Security Condition, Preliminary Version:*** S can read O if and only if either of the following is true.
 1. There is an object O' such that S has accessed O' and $CD(O') = CD(O)$.
 2. For all objects $O', O' \in PR(S) \Rightarrow COI(O') \neq COI(O)$.

Subject affects:

1. Once a subject reads any object in a COI class, the only other objects in that COI class that the subject can read are in the same CD as the read object.
2. The minimum number of subjects needed to access every object in a COI class is the same as the number of CDs in that COI class. Since most companies have information that is available to all subjects, the model distinguishes between sanitized and un-sanitized data by adding condition 3.
3. O is a sanitized object.

Sanitization:

- Public information may belong to a CD.
- As is publicly available, no conflicts of interest arise.
- So, should not affect ability of subject to read.
- Typically, all sensitive data removed from such information before it is released publicly (called sanitization)

CW-Simple Security Condition: S can read O if and only if any of the following holds.

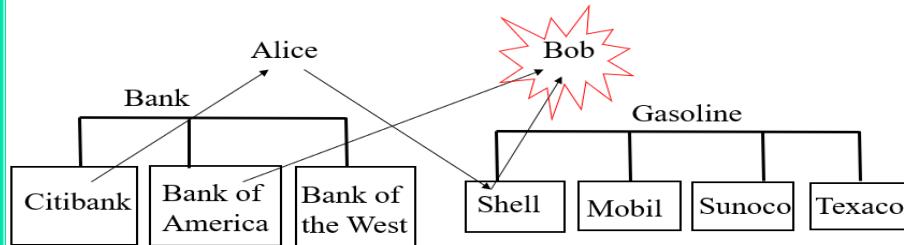
1. There is an object O' such that S has accessed O' and $CD(O') = CD(O)$.
 2. For all objects $O', O' \in PR(S) \Rightarrow COI(O') \neq COI(O)$.
 3. O is a sanitized object.
- Initially, $PR(S) = \emptyset$, so any initial read request is granted.

What about writing?

- Anthony and Susan work in the same trading house.
- Anthony can read objects in Bank of America's CD, and Susan can read objects in Citibank's CD.
- Both can read objects in ARCO's CD.

- If Anthony can also write to objects in ARCO's CD, then he can read information from objects in Bank of America's CD and write to objects in ARCO's CD, and then Susan can read that information.
- So, Susan can indirectly obtain information from Bank of America's CD, causing a conflict of interest.

- Alice reads Citibank's and Shell's CD
 - Bob reads Bank of America's and Shell's CD
 - So Bob must not read Citibank's CD
- If Alice writes what she read from Citibank's to Shell's CD; Bob can then read what Alice wrote



Since two subjects could have access to the same object in one COI and different objects in another COI, we have

- **CW-*-Property:** A subject S may write to an object O if and only if both of the following conditions hold.
 1. The CW-simple security condition permits S to read O .
 2. For all un-sanitized objects O' , S can read $O' \Rightarrow CD(O') = CD(O)$.

Bell-LaPadula and Chinese Wall Models:

- Fundamentally different
- CW has no security labels, B-LP does
- CW has notion of past accesses, B-LP does not
- Bell-LaPadula can capture state at any time
- Bell-LaPadula cannot track changes over time
 - Susan becomes ill, Anna needs to take over
 - ✓ C-W history lets Anna know if she can.
 - ✓ No way for Bell-LaPadula to capture this
 - Access constraints change over time
 - Initially, subjects in C-W can read any object
 - Bell-LaPadula constrains set of objects that a subject can access
 - ✓ Can't clear all subjects for all categories, because this violates CW-simple security condition

Clark-Wilson and Chinese Wall Models

- The Clark-Wilson model deals with many aspects of integrity, such as validation and verification, as well as access control.

- Because the Chinese Wall model deals exclusively with access control, it cannot emulate the Clark-Wilson model fully.
- So, consider only the access control aspects of the Clark-Wilson model.
- If “subjects” and “processes” are interchangeable, a single person could use multiple processes to violate CW-simple security condition
- Would still comply with Clark-Wilson Model
 - If “subject” is a specific person and includes all processes the subject executes, then consistent with Clark-Wilson Model

Clinical Information Systems Security Policy:

- Intended for medical records
- Conflict of interest not critical problem
- Patient confidentiality, authentication of both records and the personnel making entries in those records, and assurance that the records have not been changed erroneously are critical.
- Anderson presents a model for such policies that illuminates the combination of confidentiality and integrity to protect patient privacy and record integrity.
- Anderson defines three types of entities in the policy.
 1. A **patient** is the subject of medical records, or an agent for that person who can give consent for the person to be treated.
 2. **Personal health information** is information about a patient's health or treatment enabling that patient to be identified.
 3. A **clinician** is a health-care professional who has access to personal health information while performing his or her job.

Assumptions and Principles:

- Assumes that personal health information concerns one individual at a time.
- Not always true; obstetrics/gynaecology records contain information about both the father and the mother.
 - Principles derived from medical ethics of various societies, and from practicing clinicians

Access:

- **Access Principle 1:**
 - ✓ Each medical record has an access control list naming the individuals or groups who may read and append information to the record.
 - ✓ The system must restrict access to those identified on the access control list.
 - ✓ Medical ethics require that only clinicians and the patient have access to the patient's medical record.
- **Access Principle 2:**
 - ✓ One of the clinicians on the access control list must have the right to add other clinicians to the access control list.
 - Called the responsible clinician

- **Access Principle 3:**
 - ✓ The responsible clinician must notify the patient of the names on the access control list whenever the patient's medical record is opened.
 - ✓ Except for situations given in statutes, or in cases of emergency, the responsible clinician must obtain the patient's consent.
 - Patient must consent to all treatment, and must know of violations of security

- **Access Principle 4:**

- ✓ The name of the clinician, the date, and the time of the access of a medical record must be recorded. Similar information must be kept for deletions.
 - This is for auditing. Don't delete information; update it (last part is for deletion of records after death, for example, or deletion of information when required by statute). Record information about all accesses

Creation:

Principle:

- A clinician may open a record, with the clinician and the patient on the access control list.
- If a record is opened as a result of a referral, the referring clinician may also be on the access control list.
 - Creating clinician needs access, and patient should get it. If created from a referral, referring clinician needs access to get results of referral.

Deletion:

Principle:

- Clinical information cannot be deleted from a medical record until the appropriate time has passed.
 - This varies with circumstances.

Confinement:

Principle:

- Information from one medical record may be appended to a different medical record if and only if the access control list of the second record is a subset of the access control list of the first.
 - This keeps information from leaking to unauthorized users.
 - All users have to be on the access control list.

Aggregation:

Principle:

- Measures for preventing aggregation of patient data must be effective.
- In particular, a patient must be notified if anyone is to be added to the access control list for the patient's record and if that person has access to a large number of medical records.
 - Fear here is that a corrupt investigator may obtain access to a large number of records, correlate them, and discover private information about individuals which can then be used for nefarious purposes (such as blackmail)

Enforcement:**Principle:**

- Any computer system that handles medical records must have a subsystem that enforces the preceding principles.
- The effectiveness of this enforcement must be subject to evaluation by independent auditors.
- This policy has to be enforced, and the enforcement mechanisms must be auditable (and audited)

Compared to Bell-LaPadula

- Confinement Principle imposes lattice structure on entities in model
- Similar to Bell-LaPadula
- CISS focuses on objects being accessed; BLP on the subjects accessing the objects
- May matter when looking for insiders in the medical environment.

Compared to Clark-Wilson:

- CDIs are medical records
- TPs are functions updating records, access control lists
- IVPs certify:
 1. A person identified as a clinician is a clinician;
 2. A clinician validates, or has validated, information in the medical record;
 3. When someone is to be notified of an event, such notification occurs; and
 4. When someone must give consent, the operation cannot proceed until the consent is obtained
- Auditing (CR4) requirement: make all records append-only, notify patient when access control list changed.

Information & Network Security

Course Code: 16CA301

Module-1

Introduction

What is Security ?

- **Security** is about protection of assets.
- **Prevention**
 - take measures that prevent your assets from being damaged (or stolen)
- **Detection**
 - take measures so that you can detect when, how, and by whom an asset has been damaged
- **Reaction**
 - take measures so that you can recover your assets

Real world example

- **Prevention**
 - locks at doors, window bars, secure the walls around the property, hire a guard
- **Detection**
 - missing items, burglar alarms, closed circuit TV
- **Reaction**
 - attack on burglar (not recommended ☺), call the police, replace stolen items, make an insurance claim

Internet shopping example

- **Prevention**

- encrypt your order and card number, enforce merchants to do some extra checks, using PIN even for Internet transactions.

- **Detection**

- an unauthorized transaction appears on your credit card statement

- **Reaction**

- complain, dispute, ask for a new card number, sue (if you can find of course ☺)
- Or, pay and forget (a glass of cold water) ☺

Information & Network Security

- **Information security** is about protecting information and information systems from unauthorized access, use, disclosure, disruption, modification, perusal, inspection, recording or destruction.
- **Network security** is the generic name for the collection of tools designed to protect data during their transmission.

Overview of Computer Security

- **Computer security** is the protection of computing systems & the data that they store or access.

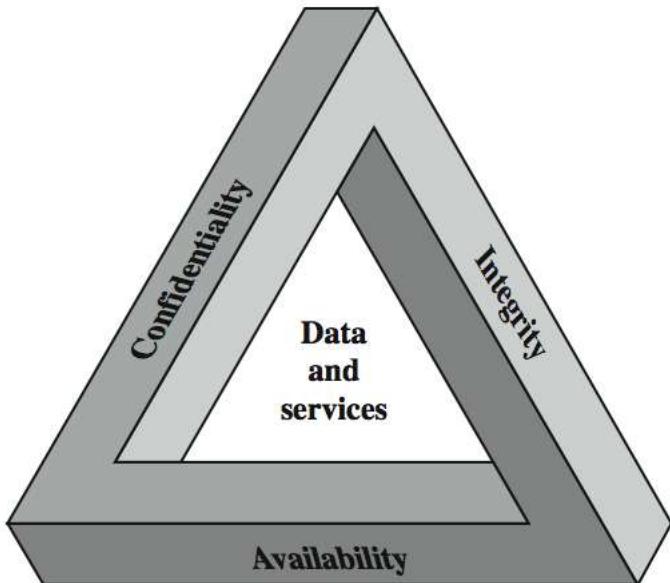


Figure 1.1 The Security Requirements Triad



Computer Security:

The protection afforded to an automated information system in order to attain the applicable objectives of preserving the **integrity**, **availability** and **confidentiality** of information system resources (includes hardware, software, firmware, information/data, and telecommunications)

Security Services

- A **security service** is a service that ensures adequate security of the systems or of data transfers.
- Information security has held Confidentiality, Integrity and Availability (known as the CIA triad) to be the core principles of information security.
- Later, other elements of information security were added to the three classic security attributes of the CIA triad. These elements are:
 - Authentication,
 - Access control,
 - Non-repudiation, and
 - Privacy.

1. Confidentiality

- Confidentiality is the concealment of information or resources.
- Confidentiality refers to the protection of information from disclosure to unauthorized entities (organizations, people, machines, processes).
- No one may read the data except for the specific entity (or entities) intended.
- Access control mechanisms supports confidentiality.

Confidentiality is a requirement :

1. When data is stored on a medium (such as a computer hard drive) that can be read by an unauthorized individual.
 2. When data is backed up onto a device (such as a tape) that can fall into the hands of an unauthorized individual.
 3. When data is transmitted over unprotected networks.
-
- Resource hiding is important aspect of confidentiality.

2. Integrity

- Integrity refers to the trustworthiness of data or resources.
- Integrity includes :
 - **Data integrity** (the content of the information) and
 - **Origin integrity** (the source of the data, often called authentication).
- Data integrity is the protection of data against creation, alteration, deletion, duplication or re-ordering by unauthorized entities (organizations, people, machines, processes).

- Data integrity is the assurance of non-alteration: the data (either in transit or in storage) has not been undetectably altered whether by accident or deliberately malign activity.
- Integrity mechanisms fall into two classes:
 - prevention mechanisms and
 - detection mechanisms.
- Prevention mechanisms seek to maintain the integrity of the data by blocking any unauthorized attempts to change the data or any attempts to change the data in unauthorized ways.

- Detection mechanisms do not try to prevent violations of integrity; they simply report that the data's integrity is no longer trustworthy.

3. Availability

- Availability refers to the ability to use the information or resource desired.
- Availability means having timely access to information.
- For example, a disk crash or denial-of-service attacks both cause a breach of availability.
- An information system that is not available when you need it is at least as bad as none at all.

4. Authentication

- The assurance that the communicating entity is the one that it claims to be.
- Factors of identification:
 1. Something you know – user ID and password
 - User ID identifies you while the password authenticates you
 - Easy to compromise if weak password
 2. Something you have – key or card
 - Can be lost or stolen
 3. Something you are – physical characteristics (i.e., biometrics)
 - Much harder to compromise
- A combination of at least 2 factors is recommended.

➤ **Peer Entity Authentication:**

Used in association with a logical connection to provide confidence in the identity of the entities connected.

➤ **Data Origin Authentication:**

In a connectionless transfer, provides assurance that the source of received data is as claimed.

5. Access Control

- Access control refers to the prevention of unauthorized use of a resource (i.e., this service controls who can have access to certain resources, under what conditions access can occur, and what those accessing the resources are allowed to do).
- To achieve this service, each entity trying to gain access must be first identified, or authenticated, so that access rights can be tailored to the individual.

- **Access control list (ACL)** created for each resource (information)
 - List of users that can read, write, delete or add information
 - Difficult to maintain all the lists
- **Role-based access control (RBAC)**
 - Rather than individual lists
 - Users are assigned to roles
 - Roles define what they can access
 - Simplifies administration

6. Non-repudiation:

- Provides protection against denial by one of the entities involved in a communication of having participated in all or part of the communication.
- It includes:
 1. **Non-repudiation of origin (NRO)**: proof that the message was sent by the specific party.
 2. **Non-repudiation of reception (NRR)**: proof that the message was received by the specific party.
- The typical protection mechanisms are: notarization, timestamp, digital signatures

7. Data Privacy

- Data privacy is the security service that allows an individual to maintain the right to control what information about him is collected, how it is used and who uses it.

On the Internet, privacy, a major concern of users, can be divided into these concerns:

1. What personal information can be shared with whom.
2. Whether messages can be exchanged without anyone else seeing them.
3. Whether and how one can send messages anonymously.

Security Mechanisms

- Security mechanism is a process that implements security services based on a hardware (technical), software (logical), physical or administrative approach.
- Security mechanisms support the security services and execute specific activities for the protection against attacks or attack results.

The mechanisms are divided into:

1. Specific Security Mechanisms
2. Pervasive Security Mechanisms

I. Specific Security Mechanisms

These may be incorporated into the appropriate protocol layer in order to provide some of the OSI security services. Some techniques for realizing security are listed here.

1. Encipherment
2. Digital Signature
3. Access control
4. Data Integrity
5. Authentication Exchange
6. Traffic Padding
7. Routing Control
8. Notarization

1. Encipherment:

- This is the process of using mathematical algorithms to transform data into a form that is not readily intelligible.
- The transformation and subsequent recovery of the data depend on an algorithm and zero or more encryption keys.

2. Digital Signature:

- Data or cryptographic transformation of a data unit is appended to the data, so that the recipient of the data unit is convinced of the source and integrity of the data unit and this can also serve to protect the data against forgery (e.g., by the recipient).

3. Access Control:

A variety of mechanisms are available that enforce access rights to resources.

4. Data Integrity:

A variety of mechanisms may be used to assure the integrity of a data unit or stream of data units.

5. Authentication Exchange:

This is a mechanism intended to ensure the identity of an entity by means of information exchange.

6. Traffic Padding:

- The insertion of bits into gaps in a data stream is called traffic padding.
- This helps to thwart traffic analysis attempts.

7. Routing Control:

Routing control enables selection of particular physically secure routes for certain data transmission and allows routing changes, especially when a breach of security is suspected.

8. Notarization:

This is the use of a trusted third party to assure certain properties of a data exchange.

II. Pervasive Security Mechanisms

These are the mechanisms that are not specific to any particular OSI security service or protocol layer.

1. Trusted Functionality
2. Security Label
3. Event Detection
4. Security Audit Trail
5. Security Recovery

1. Trusted Functionality:

The process that which is perceived to be correct with respect to some criteria (e.g., as established by a security policy).

2. Security Label:

This is the technique of marking of a bound to a resource (which may be a data unit) that names or designates the security attributes of that resource.

3. Event Detection:

Detection of security-relevant events such as forgery, denial of sending or receiving of data, alteration of data etc. is another important essential mechanism.

4. Security Audit Trail:

Data can be collected and potentially used to facilitate a security audit, which is an independent review and examination of system records and activities.

5. Security Recovery:

This deals with requests from mechanisms, such as event handling and management functions, and takes recovery actions.

Relationship between Security Services & Mechanisms

- Single security services may need to be implemented by multiple and different security mechanisms.

Service	Encipherment	Digital Signature	Access Control	Data Integrity	Authentication Exchange	Traffic Padding	Routing Control	Notarization
Peer Entity Authentication	Y	Y			Y			
Data Origin Authentication	Y	Y						
Access Control			Y					
Confidentiality	Y						Y	
Traffic Flow Confidentiality	Y					Y	Y	
Data Integrity	Y	Y		Y				
Nonrepudiation		Y		Y				Y
Availability				Y	Y			

Security Attacks

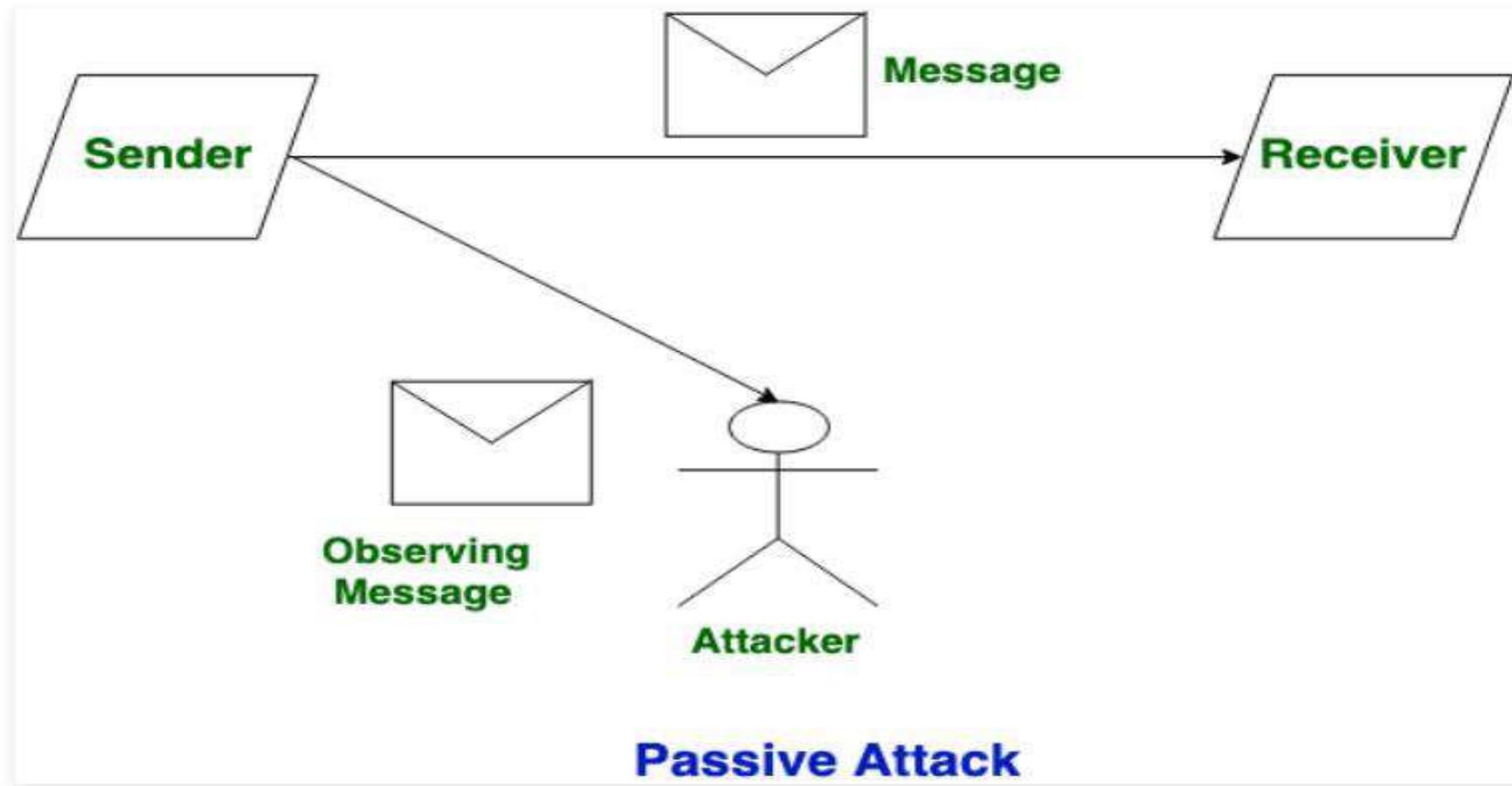
Security attacks are classified as:

1. Passive attacks and
 2. Active attacks.
-
- A **passive attack** attempts to learn or make use of information from the system but does not affect system resources.
 - An **active attack** attempts to alter system resources or affect their operation.

I. Passive Attacks:

- Passive attacks attempt to learn or make use of information from the system but do not affect system resources.
- A passive attack is one where the attacker only monitors the communication channel.
- A passive attacker only threatens the confidentiality of data.
- Passive attacks are in the nature of eavesdropping on, or monitoring of, transmissions.
- The goal of the opponent is to obtain information that is being transmitted.

Contd..



There are two types of passive attacks:

- a) Release of message contents
- b) Traffic Analysis

- Passive attacks are very difficult to detect because they do not involve any alteration of the data.
- Typically, the messages are sent and received in seemingly normal fashion.
- Message encryption is a simple solution to thwart passive attacks.
- The emphasis in dealing with passive attacks is on **prevention** rather than detection.

A) Release of message contents:

- A telephone conversation, an electronic mail message, and a transferred file may contain sensitive or confidential information.
- We would like to prevent an opponent from learning the contents of these transmissions.

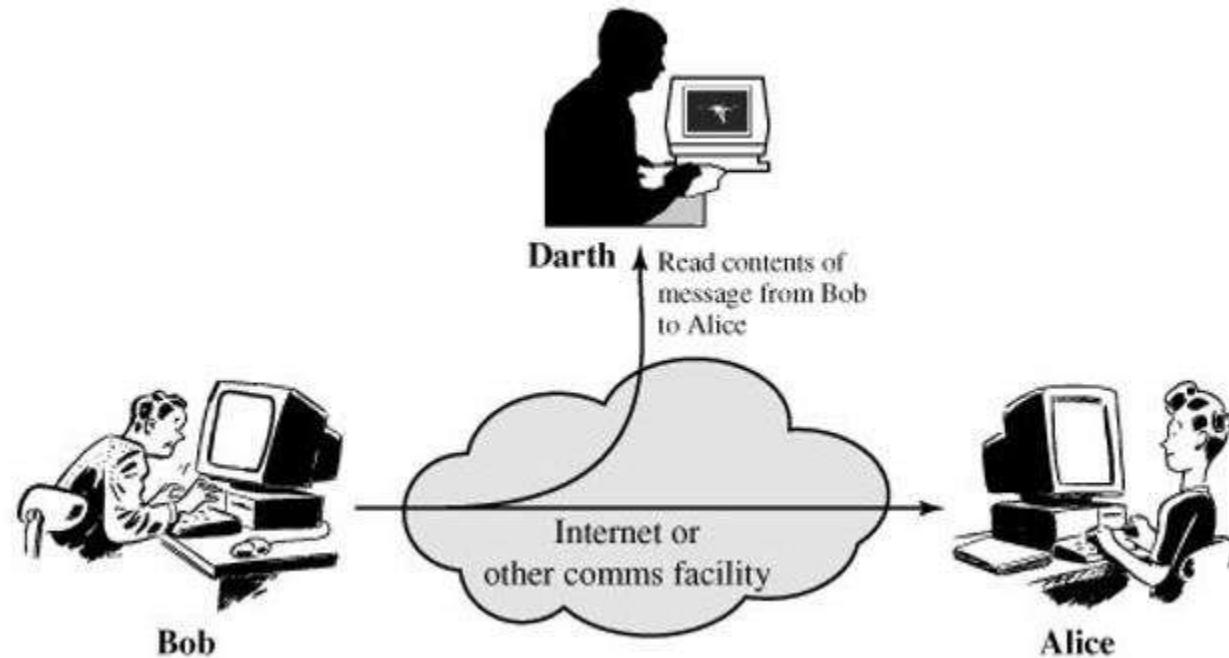
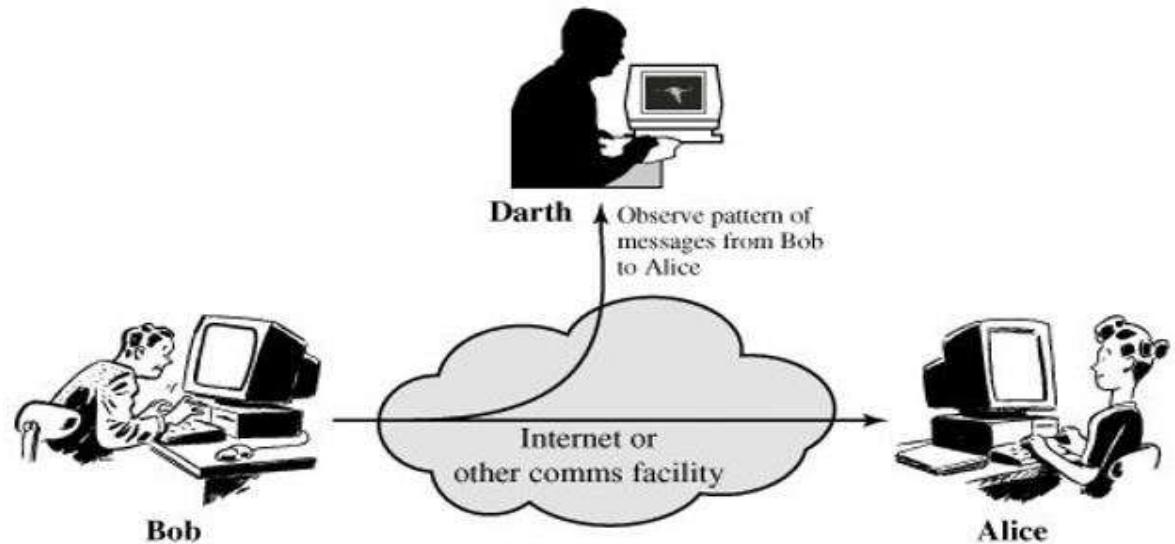


Figure 1.2 Release of message contents

B) Traffic Analysis

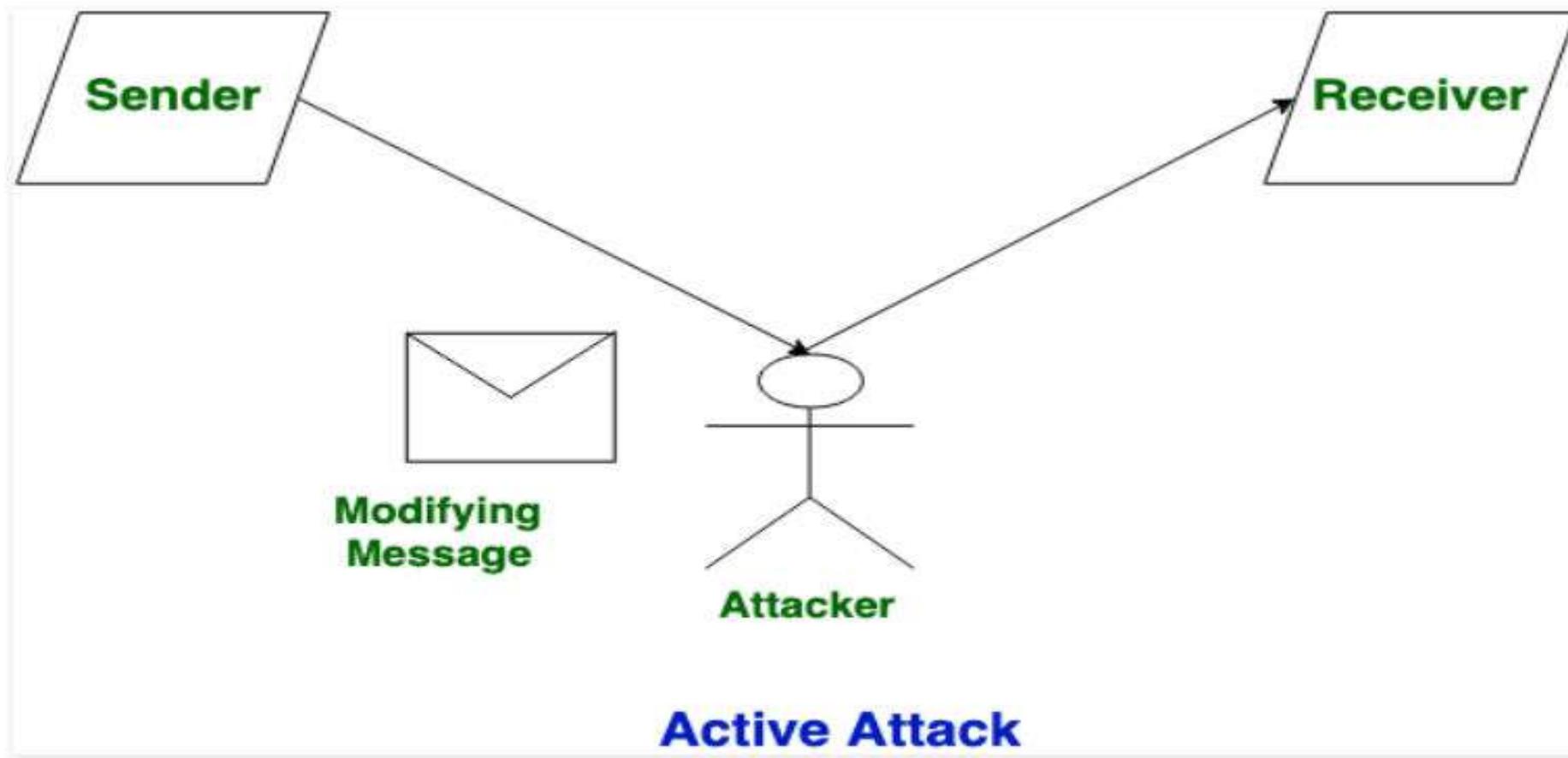
- It refers to the process of intercepting and examining messages in order to deduce information from patterns in communication.
- It can be performed even when the messages are encrypted and cannot be decrypted.
- The greater the number of messages observed, or even intercepted and stored, the more can be inferred from the traffic.



II. Active Attacks:

- Active attacks attempt to alter system resources or affect their operation.
- This type of attack is one where the adversary attempts to delete, add, or in some other way alter the transmission on the channel.
- An active attacker threatens data integrity and authentication as well as confidentiality.
- Active attacks involve some modification of the data stream or the creation of a false stream.

Contd..

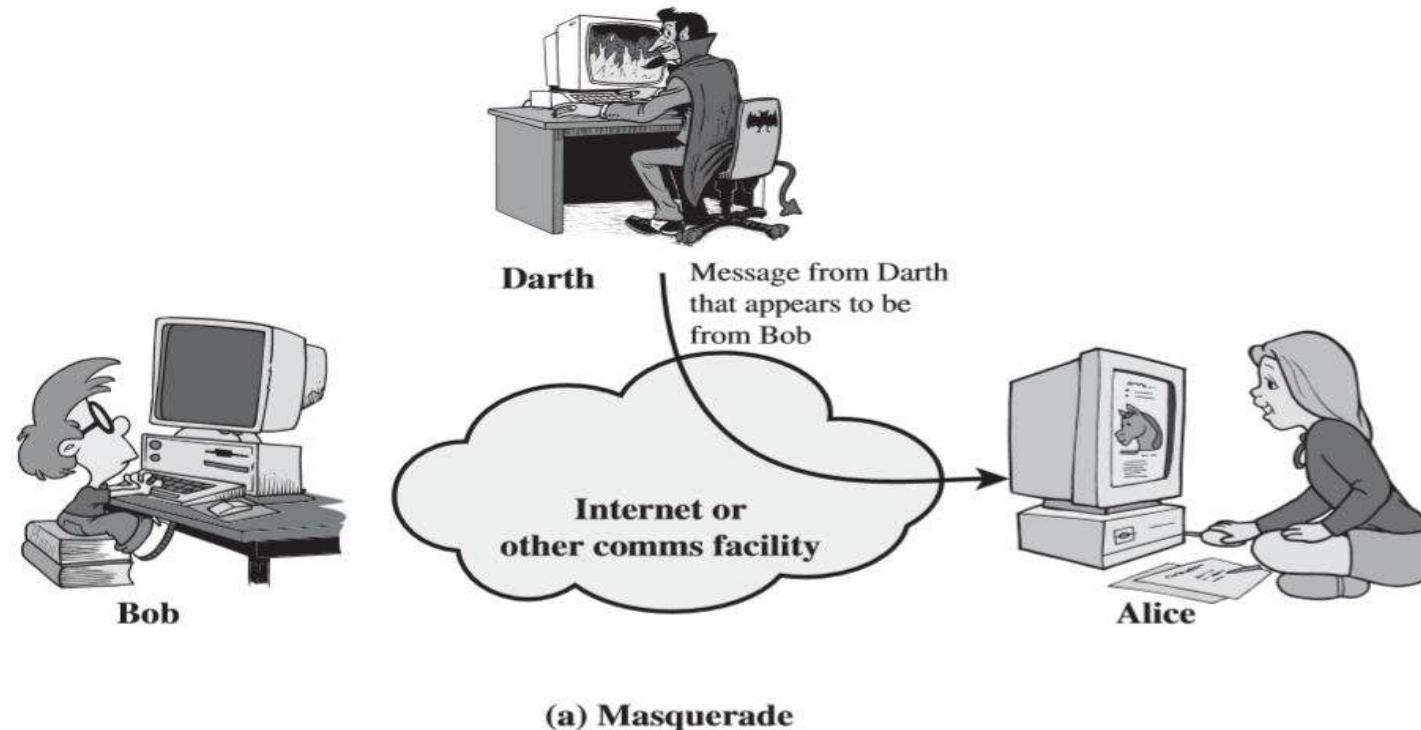


Active attack are subdivided into four categories:

1. Masquerade
2. Replay
3. Modification of messages
4. Denial of service.

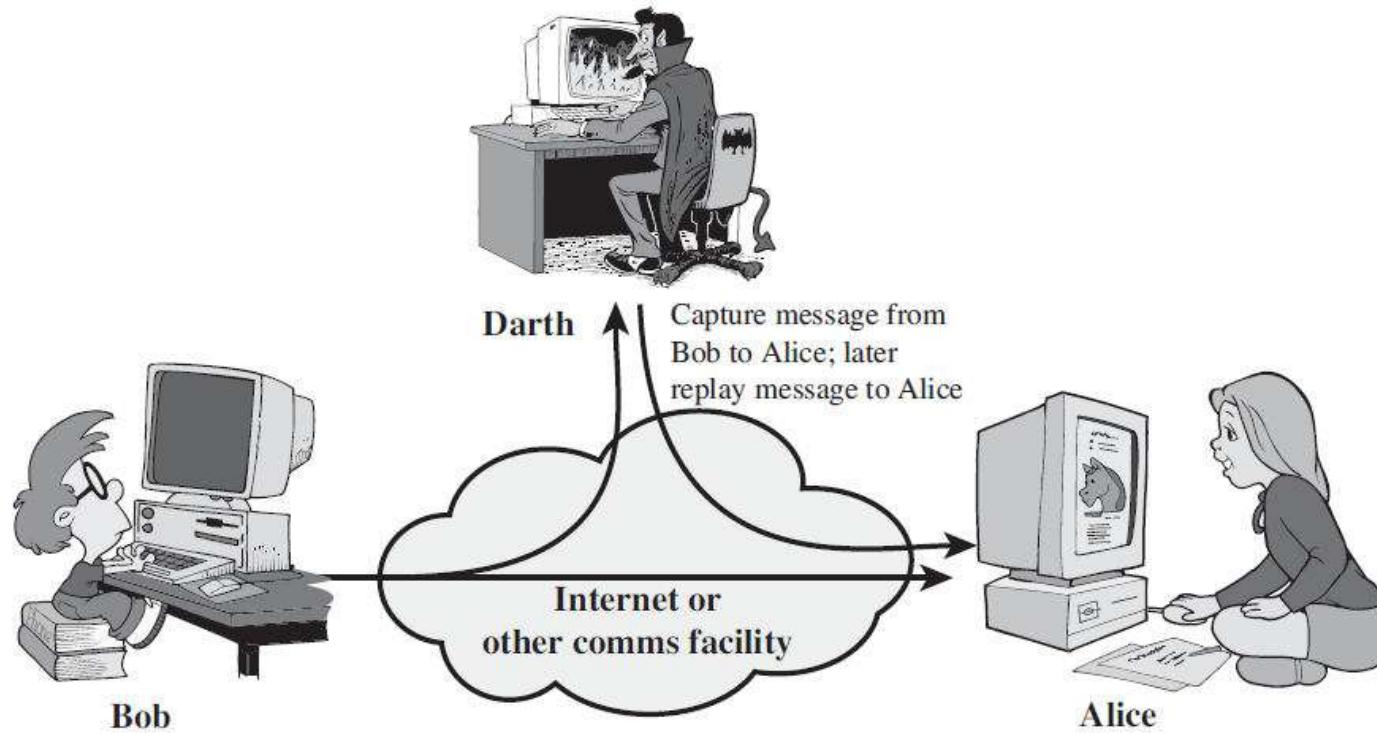
1. Masquerade

- It takes place when one entity pretends to be different entity.
- The attacker pretends to be an authorized user of a system in order to gain access to it or to gain greater privileges than they are authorized for.



2. Replay:

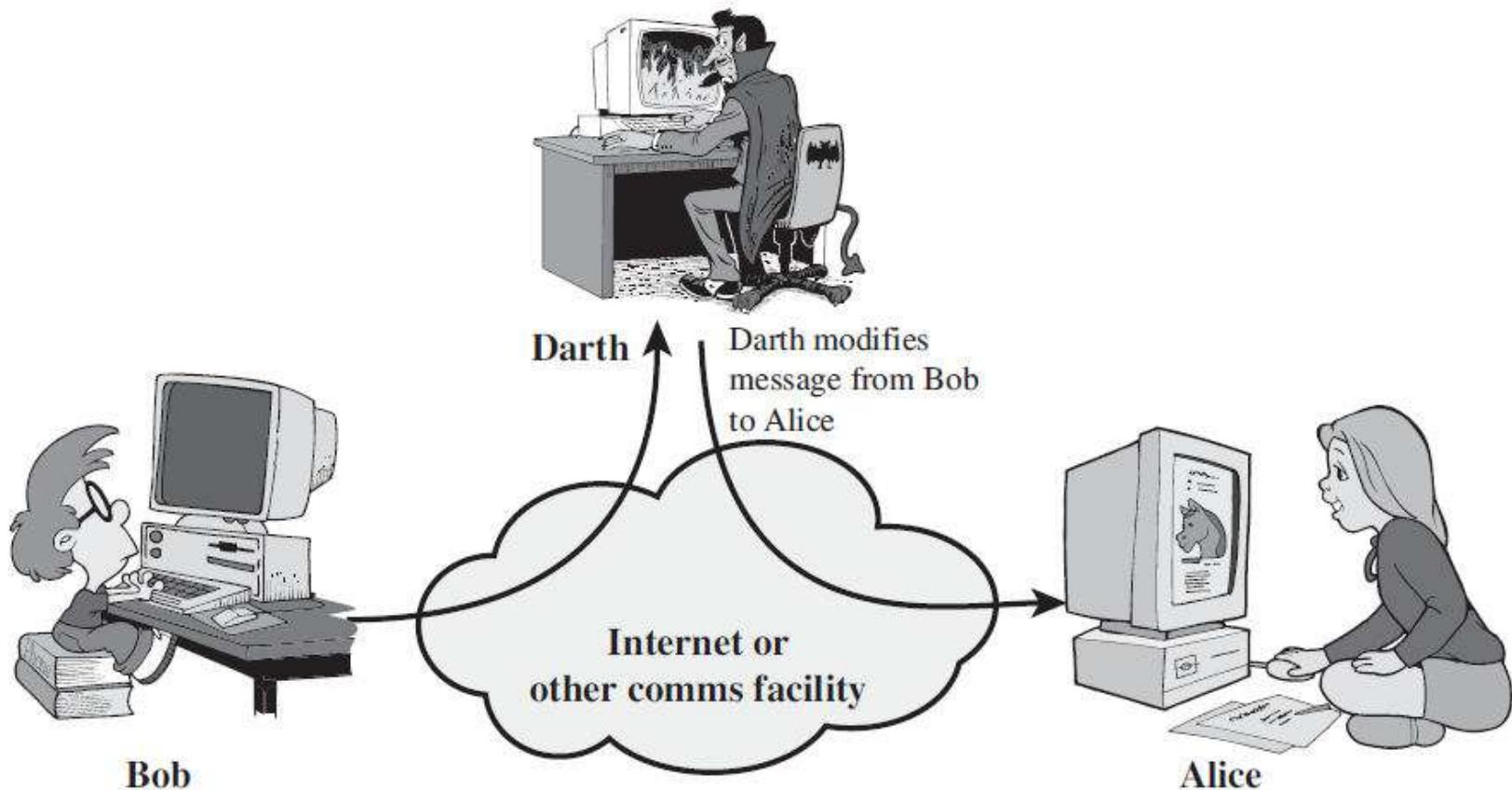
- Replay involves the passive capture of a data unit and its subsequent retransmission to produce an unauthorized effect.



3. Modification of messages :

- Modification of messages simply means that some portion of a legitimate message is altered, or that messages are delayed or reordered, to produce an unauthorized effect.
- For example, a message meaning “Allow John Smith to read confidential file *accounts*” is modified to mean “Allow Fred Brown to read confidential file *accounts*.”

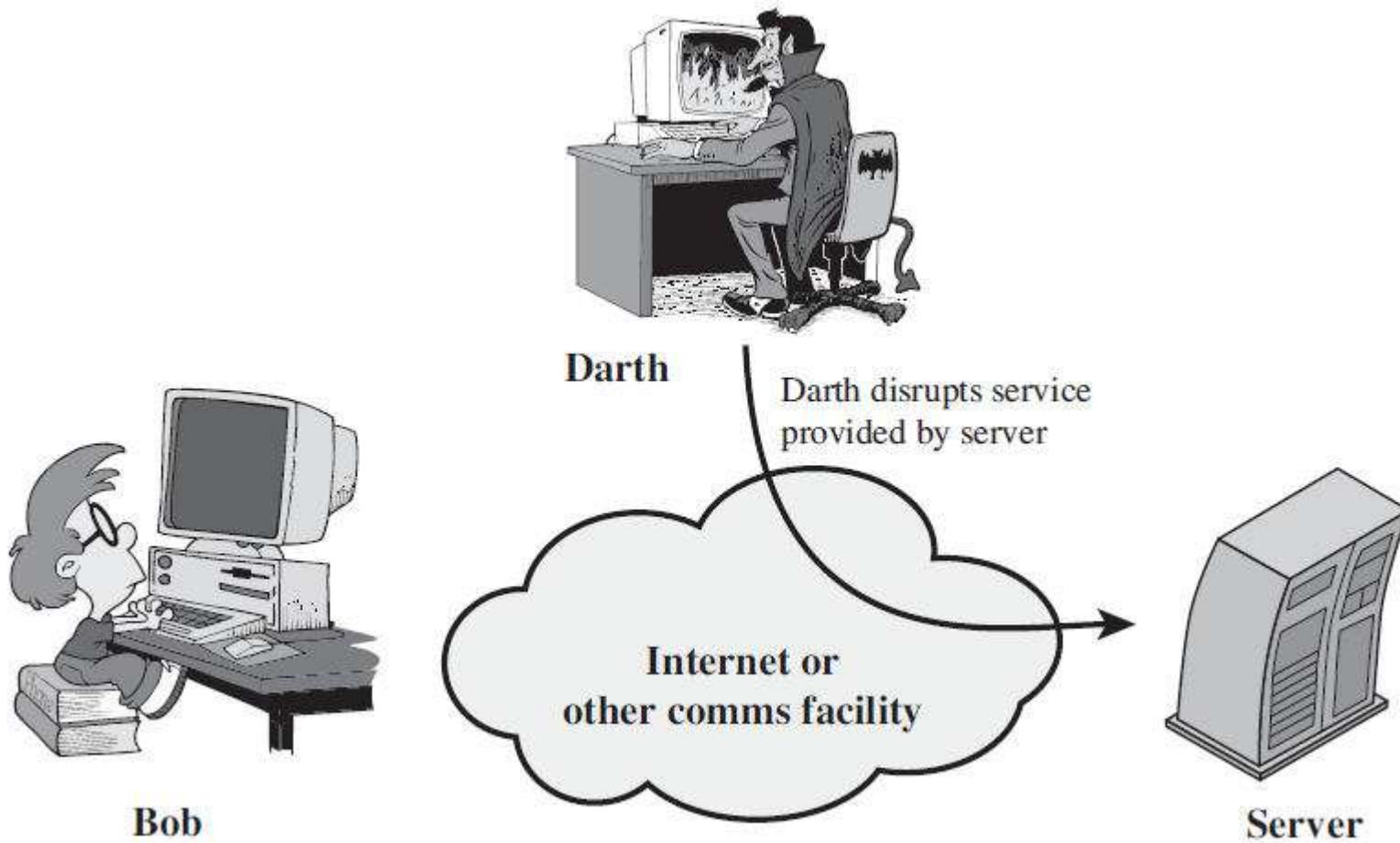
Example: Modification of messages



4. Denial of service

- The denial of service prevents or inhibits the normal use or management of communications facilities.
- This attack may have a specific target.
- Example: an entity may suppress all messages directed to a particular destination (e.g., the security audit service).
- Another form of service denial is the disruption of an entire network, either by disabling the network or by overloading it with messages so as to degrade performance.

Example: Denial of service



Difference between Active & Passive attack:

Sr. No.	Active Attack	Passive Attack
1.	In active attack, Modification in information take place.	While in passive attack, Modification in the information does not take place.
2.	Active Attack is danger for Integrity as well as availability .	Passive Attack is danger for Confidentiality .
3.	In active attack attention is on detection.	While in passive attack attention is on prevention.
4.	Due to active attack system is always damaged.	While due to passive attack, there is no any harm to the system.
5.	In active attack, Victim gets informed about the attack.	While in passive attack, Victim does not get informed about the attack.
6.	In active attack, System resources can be changed.	While in passive attack, System resources are not change.

Access Control Matrix

❖ Protection State:

- The *state* of a system is the collection of the current values of all memory locations, all secondary storage, and all registers and other components of the system.
- The subset of this collection that deals with protection is the *protection state* of the system.
- An *access control matrix* is one tool that can describe the current protection state.

ACM Contd...

- Consider the set of possible protection states P .
- Some subset Q of P consists of exactly those states in which the system is authorized to reside.
- So, whenever the system state is in Q , the system is secure.
- When the current state is in $P - Q$, the system is not secure.
- Characterizing the states in Q is the function of a **security policy**.
- Preventing the system from entering a state in $P - Q$ is the function of a **security mechanism**.
- The *access control matrix model* is the most precise model used to describe a protection state.
- It characterizes the rights of each *subject* (active entity, such as a process) with respect to every other entity.

Access Control Matrix Model

- **Access Control Matrix** or **Access Matrix** is an abstract, formal security model of protection state in computer systems, that characterizes the rights of each subject with respect to every object in the system.
- The set of all protected entities (ie., entities that are relevant to the protection state of the system) is called the set of *objects* O .
- The set of *subjects* S is the set of active objects, such as processes and users.

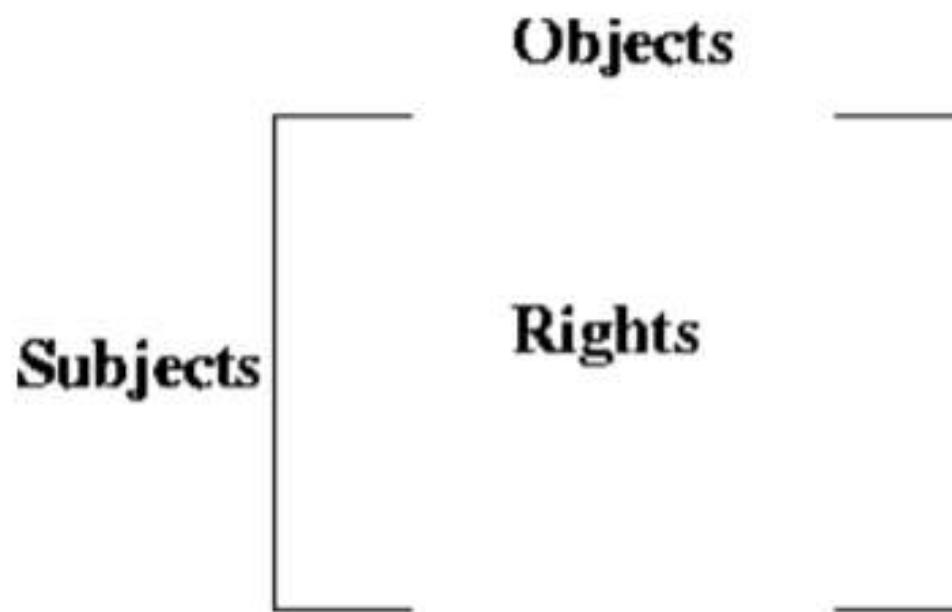
- In the access control matrix model, the relationship between these entities is captured by a matrix A with *rights* drawn from a set of rights R in each entry $a[s, o]$,
 - where $s \in S$,
 - $o \in O$, and
 - $a[s, o] \subseteq R$.
- The subject s has the set of rights $a[s, o]$ over the object o .
- The set of protection *states* of the system is represented by the triple (S, O, A) .

ACM Contd...

Set of objects: O

Set of subjects: S

Set of Rights: R



Example:

	file 1	file 2	process 1	process 2
process 1	read, write, own	read	read, write, execute, own	write
process 2	append	read, own	read	read, write, execute, own

Fig: An access control matrix. The system has two processes and two files. The set of rights is {read, write, execute, append, own}.

Security Policies

Consider a computer system to be a finite-state automation with a set of transition functions that change state. Then:

- A ***security policy*** is a statement that partitions the states of the system into a set of *authorized*, or *secure*, states and a set of *unauthorized*, or *nonsecure*, states.
- A ***secure system*** is a system that starts in an authorized state and cannot enter an unauthorized state.

- Consider the finite-state machine.
- It consists of four states and five transitions.
- The security policy partitions the states into a set of authorized states $A = \{ s1, s2 \}$ and a set of unauthorized states $UA = \{ s3, s4 \}$.

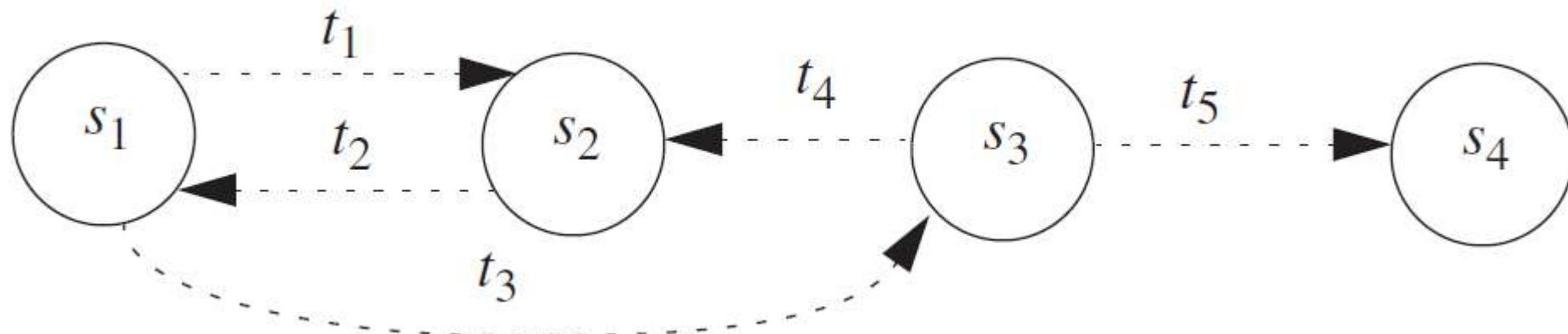


Fig. A simple finite-state machine. In this example, the authorized states are $s1$ and $s2$.

- This system is not secure, because regardless of which authorized state it starts in, it can enter an unauthorized state.
- However, if the edge from s_1 to s_3 were not present, the system would be secure, because it could not enter an unauthorized state from an authorized state.
- A ***breach of security*** occurs when a system enters an unauthorized state.

- Let X be a set of entities and let I be some information. Then I has the property of **confidentiality** with respect to X if no member of X can obtain information about I .
- Let X be a set of entities and let I be some information or a resource. Then I has the property of **integrity** with respect to X if all members of X trust I .
- Let X be a set of entities and let I be a resource. Then I has the property of **availability** with respect to X if all members of X can access I .

- A ***security mechanism*** is an entity or procedure that enforces some part of the security policy.
- A ***security model*** is a model that represents a particular policy or set of policies.

Types of Security Policies

1. A ***military security policy*** (also called a *governmental security policy*) is a security policy developed primarily to provide confidentiality.
 - The name comes from the military's need to keep information, such as the date that a troop ship will sail, secret.

2. A ***commercial security policy*** is a security policy developed primarily to provide integrity.
 - The name comes from the need of commercial firms to prevent tampering with their data, because they could not survive such compromises.

3. A ***confidentiality policy*** is a security policy dealing only with confidentiality.
4. An ***integrity policy*** is a security policy dealing only with integrity.

Types of Access Control

- A security policy may use two types of access controls, alone or in combination.
- In one, access control is left to the discretion of the owner.
- In the other, the operating system controls access, and the owner cannot override the controls.

1. If an individual user can set an access control mechanism to allow or deny access to an object, that mechanism is a ***discretionary access control (DAC)***, also called an ***identity-based access control (IBAC)***.
2. When a system mechanism controls access to an object and an individual user cannot alter that access, the control is a ***mandatory access control (MAC)***, occasionally called a ***rule-based access control***.
3. An ***originator controlled access control (ORCON or ORGCON)*** bases access on the creator of an object (or the information it contains).

Confidentiality Policies

❖ Goals of Confidentiality Policies:

- A confidentiality policy, also called an *information flow policy*, prevents the unauthorized disclosure of information.
- Unauthorized alteration of information is secondary.
- Example: Privacy Act requires that certain personal data be kept confidential. E.g., income tax returns are legally confidential & are only available to Internal Revenue Service or to legal authority with court order. It limits the distribution of documents/information.

The Bell-LaPadula Model

- The Bell-LaPadula Model corresponds to military-style classifications.
- The **Bell-LaPadula Model (BLP)** is a state machine model used for enforcing access control in government and military applications.
- The Bell-LaPadula model is a security method of keeping files confidential.
- The US government uses **classification levels**, which are rated lowest to highest: Unclassified, Confidential, Secret and Top Secret to a file.
- It describes a set of access control rules which use security labels on objects and clearances for subjects.

- The simplest type of confidentiality classification is a set of *security clearances* arranged in a linear (total) ordering.

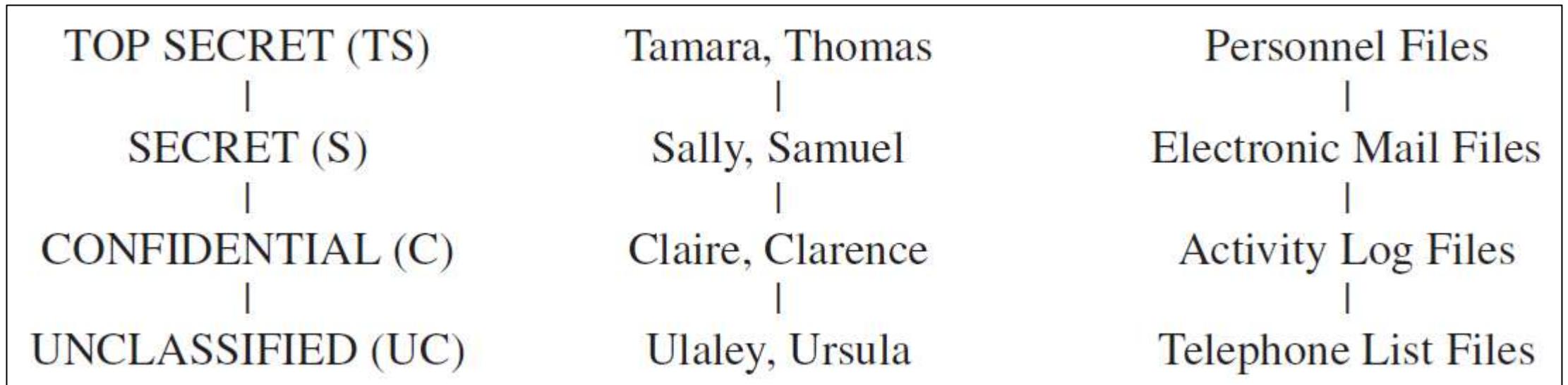


Figure: At the left is the basic confidentiality classification system. The four security levels are arranged with the most sensitive at the top and the least sensitive at the bottom. In the middle are individuals grouped by their security clearances, and at the right is a set of documents grouped by their security levels.

- These clearances represent sensitivity levels.

- The higher the security clearance, the more sensitive the information (and the greater the need to keep it confidential).
- A subject has a *security clearance*.
- The goal of the Bell-LaPadula security model is to prevent read access to objects at a security classification higher than the subject's clearance.
- The Bell-LaPadula security model combines mandatory and discretionary access controls.

The set of access rights given to a subject are the following:

- **Read-Only**: The subject can only read the object.
- **Append** : The subject can only write to the object but it cannot read.
- **Execute** : The subject can execute the object but can neither read nor write.
- **Read-Write**: The subject has both read and write permissions to the object.

Restrictions imposed by the Bell-Lapadula Model:

The following restrictions are imposed by the model:

- 1. reading down:** A subject has only read access to objects whose security level is below the subject's current clearance level. This prevents a subject from getting access to information available in security levels higher than its current clearance level.
- 2. writing up:** A subject has append access to objects whose security level is higher than its current clearance level. This prevents a subject from passing information to levels lower than its current level.

- Let $L(S) = l_s$ be the security clearance of subject S , and
- let $L(O) = l_o$ be the security classification of object O .
- For all security classifications l_i , $i = 0, \dots, k - 1$, $l_i < l_{i+1}$.
- **Simple Security Condition, Preliminary Version:**
 S can read O if and only if $l_o \leq l_s$ and
 S has discretionary read access to O .
- ***-Property (Star property) Preliminary Version :**
 S can write O if and only if $l_s \leq l_o$ and
 S has discretionary write access to O .
- TS (Top secret) can not write documents lower than TS. → Prevent classified information leak.

Basic Security Theorem, Preliminary Version

- Let Σ be a system with secure initial state σ_0
- Let T be the set of state transformations.
- If every element of T preserves the simple security condition, preliminary version, and the $*$ -property, preliminary version,
Then every state σ_i , $i \geq 0$, is secure.

Integrity Policies

- **Problem area:** systems require data to be changed accurately and follow the rules. Disclosure is not a major concern.
- Lipner identifies five requirements for preserving data integrity:
 1. Users will not write their own programs, but will use existing production programs and databases.
 2. Programmers will develop and test programs on a nonproduction system; if they need access to actual data, they will be given production data via a special process, but will use it on their development system.
 3. A special process must be followed to install a program from the development system onto the production system.
 4. The special process in requirement 3 must be controlled and audited.
 5. The managers and auditors must have access to both the system state and the system logs that are generated.

The requirements suggest 3 principles of operation:

1. Separation of duty
2. Separation of function
3. Auditing

I. Separation of duty

- The principle of separation of duty states that if two or more steps are required to perform a critical function, at least two different people should perform the steps.
- Moving a program from the development system to the production system is an example of a critical function.

II. Separation of function

- Developers do not develop new programs on production systems because of the potential threat to production data.
- Similarly, the developers do not process production data on the development systems.
- Depending on the sensitivity of the data, the developers and testers may receive sanitized production data.
- The development environment must be as similar as possible to the actual production environment.

III. Auditing

- Commercial systems emphasize recovery and accountability.
- Auditing is the process of analyzing systems to determine what actions took place and who performed them.

Biba Integrity Model

- In his model, a system consists of a set S of subjects, a set O of objects, and a set I of integrity levels.
- The levels are ordered.
- Data at a higher level is more accurate and/or reliable (with respect to some metric) than data at a lower level.

- Integrity labels, in general, are not also security labels.
- They are assigned and maintained separately, because the reasons behind the labels are different.
- Security labels primarily limit the flow of information; integrity labels primarily inhibit the modification of information.

Biba's model is the dual of the Bell-LaPadula Model. Its rules are as follows.

1. $s \in S$ can read $o \in O$ if and only if $i(s) \leq i(o)$.
2. $s \in S$ can write to $o \in O$ if and only if $i(o) \leq i(s)$.
3. $s_1 \in S$ can execute $s_2 \in S$ if and only if $i(s_2) \leq i(s_1)$.

Note: rules 1 and 2 imply that if both read and write are allowed, $i(s) = i(o)$.

Clark-Wilson Integrity Model

- This model uses transactions as the basic operation, which models many commercial systems more realistically than previous models.
- One main concern of a commercial environment is the **integrity of the data** in the system and of the actions performed on that data.
- The data is said to be *in a consistent state* (or *consistent*) if it satisfies given properties.
- **For example**, let D be the amount of money deposited so far today,
 W the amount of money withdrawn so far today,
 YB the amount of money in all accounts at the end of yesterday, and
 TB the amount of money in all accounts so far today.

- Then the consistency property is:

$$D + YB - W = TB$$

- Before and after each action, the consistency conditions must hold.
- A *well-formed transaction* is a series of operations that transition the system from one consistent state to another consistent state.
- **For example**, if a depositor transfers money from one account to another, the transaction is the transfer; two operations, the deduction from the first account and the addition to the second account, make up this transaction. Each operation may leave the data in an inconsistent state, but the well-formed transaction must preserve consistency.

- The second feature of a commercial environment relevant to an integrity policy is the **integrity of the transactions themselves**.
- Someone must certify that the transactions are implemented correctly.
- The principle of separation of duty requires that the certifier and the implementors be different people.
- In order for the transaction to corrupt the data (either by illicitly changing the data or by leaving the data in an inconsistent state), two different people must either make similar mistakes or collude to certify the well-formed transaction as correct.

The Model:

- The Clark-Wilson model defines data subject to its integrity controls as *constrained data items*, or CDIs.
- Data not subject to the integrity controls are called *unconstrained data items*, or UDIs.
- For example, in a bank, the balances of accounts are CDIs since their integrity is crucial to the operation of the bank, whereas the gifts selected by the account holders when their accounts were opened would be UDIs, because their integrity is not crucial to the operation of the bank.
- The set of CDIs and the set of UDIs partition the set of all data in the system being modeled.

- The model also defines two sets of procedures.
 1. ***Integrity verification procedures***, or IVPs, test that the CDIs conform to the integrity constraints at the time the IVPs are run. In this case, the system is said to be in a *valid state*.
 2. ***Transformation procedures***, or TPs, change the state of the data in the system from one valid state to another; TPs implement well-formed transactions.
- **Example:** The balances in the accounts are CDIs; checking that the accounts are balanced, as described above, is an IVP. Depositing money, withdrawing money, and transferring money between accounts are TPs.

Certification Rules:

- **Certification rule 1 (CR1):** When any IVP is run, it must ensure that all CDIs are in a valid state.
- **Certification rule 2 (CR2):** For some associated set of CDIs, a TP must transform those CDIs in a valid state into a (possibly different) valid state.
- CR2 defines as *certified* a relation that associates a set of CDIs with a particular TP.
- Let C be the certified relation. Then, in the bank example,
 $(\text{balance}, \text{account}1), (\text{balance}, \text{account}2), \dots, (\text{balance}, \text{account}n) \in C$.

- CR2 implies that a TP may corrupt a CDI if it is not certified to work on that CDI.
- **For example**, the TP that invests money in the bank's stock portfolio would corrupt account balances even if the TP were certified to work on the portfolio, because the actions of the TP make no sense on the bank accounts.
 - Hence, the system must prevent TPs from operating on CDIs for which they have not been certified.

This leads to the following *enforcement rule*:

- **Enforcement rule 1 (ER1)**: The system must maintain the *certified* relations, and must ensure that only TPs certified to run on a CDI manipulate that CDI.

- **Enforcement rule 2 (ER2):** The system must associate a user with each TP and set of CDIs.
 - The TP may access those CDIs on behalf of the associated user.
 - If the user is not associated with a particular TP and CDI, then the TP cannot access that CDI on behalf of that user.
- **Certification rule 3 (CR3):** The *allowed* relations must meet the requirements imposed by the principle of separation of duty.
- **Enforcement rule 3 (ER3):** The system must authenticate each user attempting to execute a TP.

- **Certification rule 4 (CR4):** All TPs must append enough information to reconstruct the operation to an append-only CDI.
- **Certification rule 5 (CR5):** Any TP that takes as input a UDI may perform only valid transformations, or no transformations, for all possible values of the UDI. The transformation either rejects the UDI or transforms it into a CDI.
- **Enforcement rule 4 (ER4):** Only the certifier of a TP may change the list of entities associated with that TP. No certifier of a TP, or of an entity associated with that TP, may ever have execute permission with respect to that entity.

Hybrid Policies

❖ Chinese Wall Model:

- Security policy that refers equally to confidentiality and integrity.
- It describes policies that involve a conflict of interest in business.
- **Problem:**
 - Consultant advises Bank1 and Bank2 about investments.
 - Conflict of interest: advice for either bank would affect advice to the other bank
- **Solution:**
 - Consultant can only access objects on his/her side of the wall

Definitions:

1. **Objects** : items of information related to a company.
 2. **Company dataset (CD)**: collection of objects related to a single company.
 3. **Conflict of interest class (COI)**: collection of datasets of companies in competition
-
- Let $COI(O)$ represent the COI class that contains object O , and let $CD(O)$ be the company dataset that contains object O .
 - The model assumes that each object belongs to exactly one COI class.

Example:

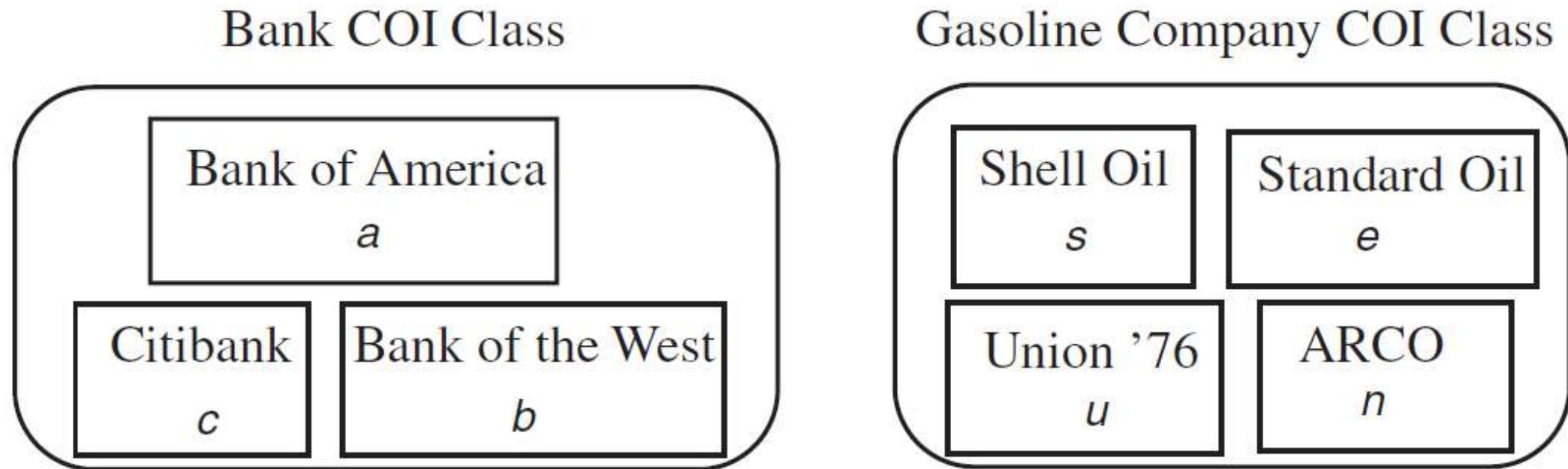


Figure: The Chinese Wall model database. It has two COI classes. The one for banks contains three CDs. The other one, for gasoline companies, contains four CDs. Each (COI, CD) pair is represented by a lowercase letter (for example, (Bank COI, Citibank) is *c*). Susan may have access to no more than one CD in each COI, so she could access Citibank's CD and ARCO's CD, but not Citibank's CD and Bank of America's CD.

Temporal Element:

- Rights depend on access history.
- Initially, a subject can read any object in any CD of any COI.
- If a subject reads an object in a CD in a COI, he can never read an object in another CD in the same COI.
 - Possible that information learned earlier may allow him to make decisions later
- $PR(S)$ denotes the set of objects that a subject S has already read.

- ***CW-Simple Security Condition, Preliminary Version:*** S can read O if and only if either of the following is true.
 1. There is an object O' such that S has accessed O' and $CD(O') = CD(O)$.
 2. For all objects O' , $O' \in PR(S) \Rightarrow COI(O') \neq COI(O)$.

Subject affects:

1. Once a subject reads any object in a COI class, the only other objects in that COI class that the subject can read are in the same CD as the read object.
2. The minimum number of subjects needed to access every object in a COI class is the same as the number of CDs in that COI class. Since most companies have information that is available to all subjects, the model distinguishes between sanitized and unsanitized data by adding condition 3.
3. O is a sanitized object.

Sanitization:

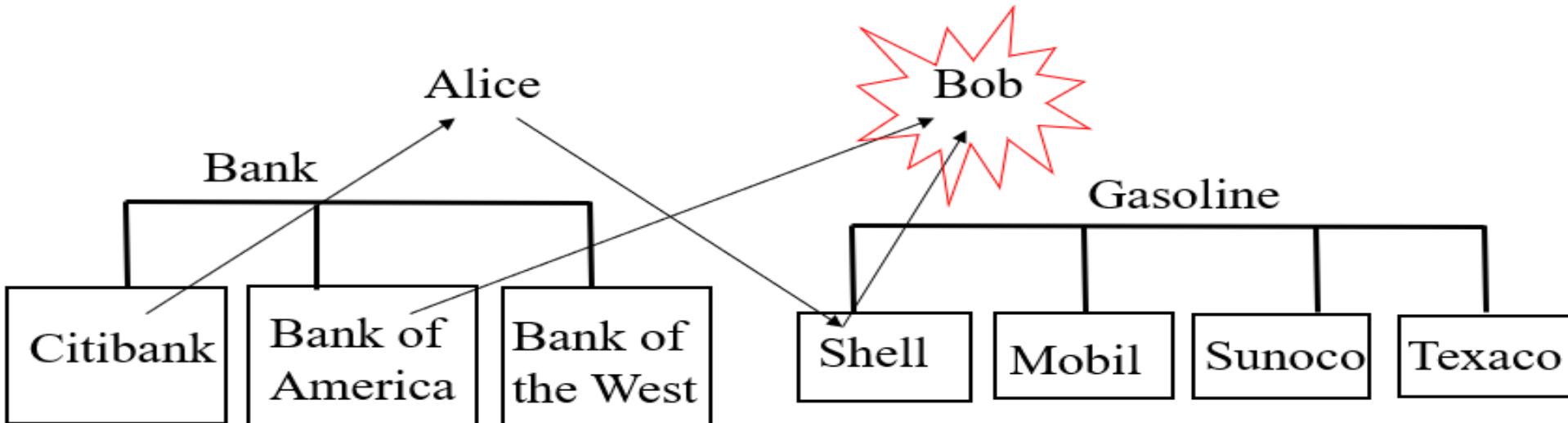
- Public information may belong to a CD.
 - As is publicly available, no conflicts of interest arise.
 - So, should not affect ability of subject to read.
 - Typically, all sensitive data removed from such information before it is released publicly (called sanitization)

- ***CW-Simple Security Condition:*** S can read O if and only if any of the following holds.
 1. There is an object O' such that S has accessed O' and $CD(O') = CD(O)$.
 2. For all objects $O', O' \in PR(S) \Rightarrow COI(O') \neq COI(O)$.
 3. O is a sanitized object.
- Initially, $PR(S) = \emptyset$, so any initial read request is granted.

What about writing ?

- Alice reads Citibank's and Shell' CD
- Bob reads Bank of America's and Shell's CD
 - So Bob must not read Citibank's CD

If Alice writes what she read from Citibank's to Shell' CD; Bob can then read what Alice wrote



Since two subjects could have access to the same object in one COI and different objects in another COI, we have

- ***CW*-Property:*** A subject S may write to an object O if and only if both of the following conditions hold.
 1. The CW-simple security condition permits S to read O .
 2. For all unsanitized objects O' , S can read $O' \Rightarrow CD(O') = CD(O)$.

Bell-LaPadula and Chinese Wall Models:

- Fundamentally different
 - CW has no security labels, B-LP does
 - CW has notion of past accesses, B-LP does not
- Bell-LaPadula can capture state at any time

- Bell-LaPadula cannot track changes over time
 - Susan becomes ill, Anna needs to take over
 - ✓ C-W history lets Anna know if she can.
 - ✓ No way for Bell-LaPadula to capture this
- Access constraints change over time
 - Initially, subjects in C-W can read any object
 - Bell-LaPadula constrains set of objects that a subject can access
 - ✓ Can't clear all subjects for all categories, because this violates CW-simple security condition

Clark-Wilson and Chinese Wall Models

- The Clark-Wilson model deals with many aspects of integrity, such as validation and verification, as well as access control.
- Because the Chinese Wall model deals exclusively with access control, it cannot emulate the Clark-Wilson model fully.
- So, consider only the access control aspects of the Clark-Wilson model.
- If “subjects” and “processes” are interchangeable, a single person could use multiple processes to violate CW-simple security condition
 - Would still comply with Clark-Wilson Model
- If “subject” is a specific person and includes all processes the subject executes, then consistent with Clark-Wilson Model

Clinical Information Systems Security Policy:

- Intended for medical records
 - Conflict of interest not critical problem
 - Patient confidentiality, authentication of both records and the personnel making entries in those records, and assurance that the records have not been changed erroneously are critical.
- Anderson presents a model for such policies that illuminates the combination of confidentiality and integrity to protect patient privacy and record integrity.

- Anderson defines three types of entities in the policy.
 1. A ***patient*** is the subject of medical records, or an agent for that person who can give consent for the person to be treated.
 2. ***Personal health information*** is information about a patient's health or treatment enabling that patient to be identified.
 3. A ***clinician*** is a health-care professional who has access to personal health information while performing his or her job.

Assumptions and Principles:

- Assumes that personal health information concerns one individual at a time.
 - Not always true; obstetrics/gynecology records contain information about both the father and the mother.
- Principles derived from medical ethics of various societies, and from practicing clinicians

Access:

- **Access Principle 1:**
 - ✓ Each medical record has an access control list naming the individuals or groups who may read and append information to the record.
 - ✓ The system must restrict access to those identified on the access control list.
 - ✓ Medical ethics require that only clinicians and the patient have access to the patient's medical record.

- **Access Principle 2:**

- ✓ One of the clinicians on the access control list must have the right to add other clinicians to the access control list.
 - Called the responsible clinician

- **Access Principle 3:**

- ✓ The responsible clinician must notify the patient of the names on the access control list whenever the patient's medical record is opened.
- ✓ Except for situations given in statutes, or in cases of emergency, the responsible clinician must obtain the patient's consent.
 - Patient must consent to all treatment, and must know of violations of security

- **Access Principle 4:**

- ✓ The name of the clinician, the date, and the time of the access of a medical record must be recorded. Similar information must be kept for deletions.
 - This is for auditing. Don't delete information; update it (last part is for deletion of records after death, for example, or deletion of information when required by statute). Record information about all accesses

Creation:

Principle:

- A clinician may open a record, with the clinician and the patient on the access control list.
- If a record is opened as a result of a referral, the referring clinician may also be on the access control list.
 - Creating clinician needs access, and patient should get it. If created from a referral, referring clinician needs access to get results of referral.

Deletion:

Principle:

- Clinical information cannot be deleted from a medical record until the appropriate time has passed.
 - This varies with circumstances.

Confinement:

Principle:

- Information from one medical record may be appended to a different medical record if and only if the access control list of the second record is a subset of the access control list of the first.
 - This keeps information from leaking to unauthorized users.
 - All users have to be on the access control list.

Aggregation:

Principle:

- Measures for preventing aggregation of patient data must be effective.
- In particular, a patient must be notified if anyone is to be added to the access control list for the patient's record and if that person has access to a large number of medical records.
 - Fear here is that a corrupt investigator may obtain access to a large number of records, correlate them, and discover private information about individuals which can then be used for nefarious purposes (such as blackmail)

Enforcement:

Principle:

- Any computer system that handles medical records must have a subsystem that enforces the preceding principles.
- The effectiveness of this enforcement must be subject to evaluation by independent auditors.
 - This policy has to be enforced, and the enforcement mechanisms must be auditable (and audited)

Compare to Bell-LaPadula

- Confinement Principle imposes lattice structure on entities in model
 - Similar to Bell-LaPadula
- CISS focuses on objects being accessed; BLP on the subjects accessing the objects
 - May matter when looking for insiders in the medical environment.

Compare to Clark-Wilson:

- CDIs are medical records
- TPs are functions updating records, access control lists
- IVPs certify:
 1. A person identified as a clinician is a clinician;
 2. A clinician validates, or has validated, information in the medical record;
 3. When someone is to be notified of an event, such notification occurs; and
 4. When someone must give consent, the operation cannot proceed until the consent is obtained
- Auditing (CR4) requirement: make all records append-only, notify patient when access control list changed.

MODULE 2

CRYPTOSYSTEMS & AUTHENTICATION

Objectives

- ▶ To introduce basic concepts & terminology of encryption
- ▶ To prepare us for studying modern cryptography

Overview

- ▶ Cryptography
- ▶ Basic Terminology
- ▶ Classical Cryptography
 - Substitution
 - Transposition
 - Product

What is Cryptography?

- ▶ The idea of storing and transmitting data in a form that only the authorized parties can interpret.
- ▶ Process of making and using codes to secure transmission of information

Cryptography

- ▶ Can be characterized by:
 - type of encryption operations used
 - substitution / transposition / product
 - number of keys used
 - single-key or private / two-key or public
 - way in which plaintext is processed
 - block / stream

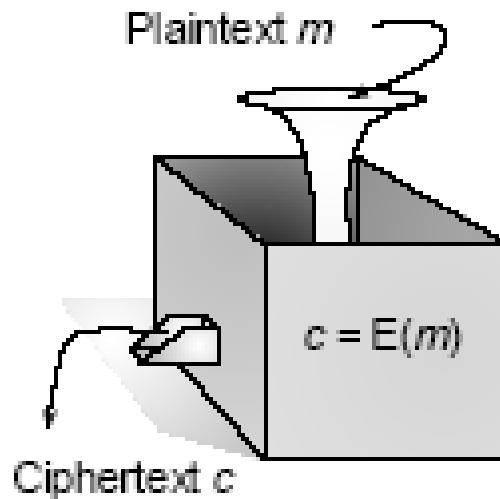
Basic terminology

- ▶ **Plaintext:** original message to be encrypted
 - ▶ **Ciphertext:** the encrypted message
 - ▶ **Enciphering or encryption:** the process of converting plaintext into ciphertext
-
- ▶ **Encryption algorithm:** performs encryption
 - Two inputs: a **plaintext** and a **secret key**

Basic terminology

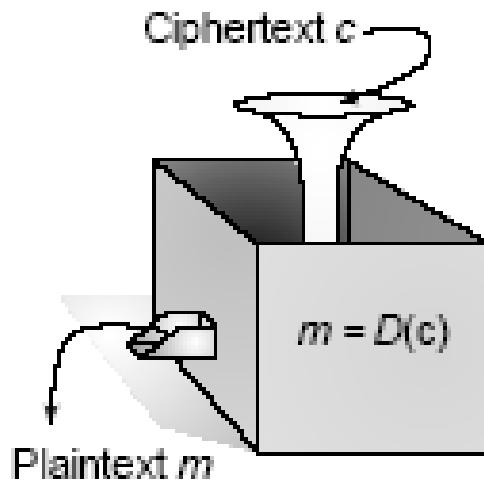
- ▶ Deciphering or decryption: recovering plaintext from ciphertext
- ▶ Decryption algorithm: performs decryption
 - Two inputs: ciphertext and secret key
- ▶ Secret key: same key used for encryption and decryption
 - Also referred to as a symmetric key

Encryption



The conversion of a original message, referred to as *plaintext* or *cleartext*, into a different message known as *ciphertext* (the word cipher comes from an old Arabic word meaning empty or zero), or *cryptogram*.

Decryption



The extraction process by which the intended receiver extracts the plaintext from the ciphertext

Cryptanalysis

(or breaking)



The process by which
an unintended
receiver discovers the
decryption process,
and thereby the
plaintext,

Cryptanalysis

- ▶ Opponent whose goal is to break cryptosystem is the *adversary*
- ▶ Objective: to recover the plaintext of a ciphertext or, more typically, to recover the secret key.
- ▶ **Kerkhoff's principle:** adversary knows algorithm used, but not key
- ▶ Two general approaches:
 - **brute-force** attack
 - **non-brute-force** attack (cryptanalytic attack)

Brute-Force Attack

- Try every key to decipher the ciphertext.
- On average, need to try half of all possible keys
- Time needed proportional to size of **key space**

Key Size (bits)	Number of Alternative Keys	Time required at 1 decryption/μs	Time required at 10^6 decryptions/μs
32	$2^{32} = 4.3 \times 10^9$	$2^{31} \mu\text{s} = 35.8 \text{ minutes}$	2.15 milliseconds
56	$2^{56} = 7.2 \times 10^{16}$	$2^{55} \mu\text{s} = 1142 \text{ years}$	10.01 hours
128	$2^{128} = 3.4 \times 10^{38}$	$2^{127} \mu\text{s} = 5.4 \times 10^{24} \text{ years}$	$5.4 \times 10^{18} \text{ years}$
168	$2^{168} = 3.7 \times 10^{50}$	$2^{167} \mu\text{s} = 5.9 \times 10^{36} \text{ years}$	$5.9 \times 10^{30} \text{ years}$
26 characters (permutation)	$26! = 4 \times 10^{26}$	$2 \times 10^{26} \mu\text{s} = 6.4 \times 10^{12} \text{ years}$	$6.4 \times 10^6 \text{ years}$

Cryptanalytic Attacks

- ▶ Classified by how much information needed by the attacker
- ▶ Different types of attacks:
 - Ciphertext only
 - Known plaintext
 - Chosen plaintext
 - Chosen-ciphertext attack
 - Chosen text

Type of Attack	Known to Cryptanalyst
Ciphertext only	<ul style="list-style-type: none"> • Encryption algorithm • Ciphertext
Known plaintext	<ul style="list-style-type: none"> • Encryption algorithm • Ciphertext • One or more plaintext-ciphertext pairs formed with the secret key
Chosen plaintext	<ul style="list-style-type: none"> • Encryption algorithm • Ciphertext • Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key
Chosen ciphertext	<ul style="list-style-type: none"> • Encryption algorithm • Ciphertext • Purported ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key

Chosen text

- Encryption algorithm
- Ciphertext
- Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key
- Purported ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key

Basis for Attacks

- ▶ Mathematical attacks
 - Based on analysis of underlying mathematics
- ▶ Statistical attacks
 - Make assumptions about the distribution of letters, pairs of letters (di-grams), triplets of letters (trigrams), etc.
 - Called *models of the language*
 - Examine ciphertext, correlate properties with the assumptions.

Statistical Attack

- ▶ Compute frequency of each letter in ciphertext:

G 0.1 H 0.1 K 0.1 O 0.3

R 0.2 U 0.1 Z 0.1

Cryptology

CRYPTOLOGY

CRYPTOGRAPHY

CRYPTANALYSIS

Private Key
(Secret Key)

Public Key

Block Cipher

Stream Cipher

Integer Factorization

Discrete Logarithm

More Definitions

► **Unconditional security**

- no matter how much computer power is available, the cipher cannot be broken since the ciphertext provides insufficient information to uniquely determine the corresponding plaintext

► **Computational security**

- given limited computing resources (eg: time needed for calculations is greater than age of universe), the cipher cannot be broken

Cryptosystem

- ▶ Quintuple (E, D, M, K, C)
 - M set of plaintexts
 - K set of keys
 - C set of ciphertexts
 - E set of encryption functions $e: M \times K \rightarrow C$
 - D set of decryption functions $d: C \times K \rightarrow M$

Example

- ▶ Example: Caesar cipher

- $M = \{ \text{sequences of letters} \}$
- $K = \{ i \mid i \text{ is an integer and } 0 \leq i \leq 25 \}$
- $E = \{ E_k \mid k \in K \text{ and for all letters } m,$

$$E_k(m) = (m + k) \bmod 26 \}$$

- $D = \{ D_k \mid k \in K \text{ and for all letters } c,$

$$D_k(c) = (26 + c - k) \bmod 26 \}$$

- $C = M$

Ciphers

- ▶ **Symmetric cipher:** same key used for encryption and decryption
 - **Block cipher:** encrypts a block of plaintext at a time (typically 64 or 128 bits)
 - **Stream cipher:** encrypts data one bit or one byte at a time
- ▶ **Asymmetric cipher:** different keys used for encryption and decryption

Classical Cryptography

- ▶ Sender, receiver share common key
 - Keys may be the same, or trivial to derive from one another
 - Sometimes called *symmetric cryptography*
- ▶ Two basic types
 - Transposition ciphers
 - Substitution ciphers
 - Combinations are called *product ciphers*

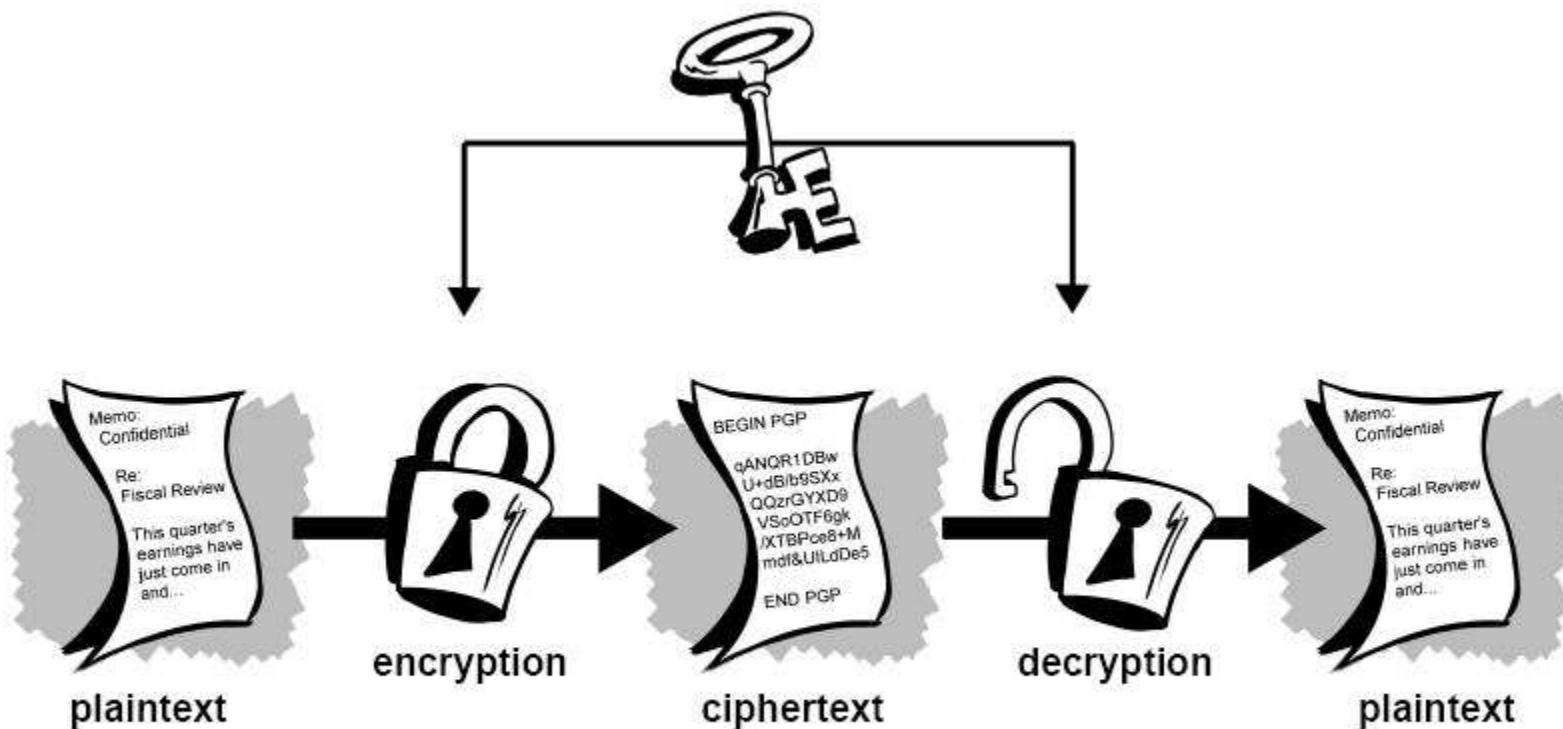


Figure 1-2. Conventional encryption

Classical Ciphers

- ▶ Plaintext is viewed as a sequence of elements (e.g., bits or characters)
- ▶ **Substitution cipher:** replacing each element of the plaintext with another element.
- ▶ **Transposition (or permutation) cipher:** rearranging the order of the elements of the plaintext.
- ▶ **Product cipher:** using multiple stages of substitutions and transpositions

Substitution Ciphers

- ▶ Change characters in plaintext to produce ciphertext
- ▶ Monoalphabetic Substitution
 - Each plaintext character is mapped onto a ***unique*** character of a ciphertext.
- ▶ Polyalphabetic Substitution
 - Each plaintext character can be mapped onto ***m*** alphabetic characters of a ciphertext.

Monoalphabetic Ciphers

- ▶ Shift Cipher(Caesar)
- ▶ Substitution Cipher
- ▶ Playfair Cipher
- ▶ Hill Cipher

Caesar Cipher

- ▶ Earliest known substitution cipher
- ▶ Invented by Julius Caesar
- ▶ Each letter is replaced by the letter three positions further down the alphabet.

Plain: 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

Cipher: D E F G H I J K L M N O P Q R S T U V W X Y Z A B C

- ▶ Example: ohio state → RKL R VWD W H

- ▶ Mathematically, map letters to numbers:
a, b, c, ..., x, y, z
0, 1, 2, ..., 23, 24, 25
- ▶ Then the general Caesar cipher is:
 $c = E_K(p) = (p + k) \text{ mod } 26$
 $p = D_K(c) = (c - k) \text{ mod } 26$
- ▶ Can be generalized with any alphabet.

Cryptanalysis of Caesar Cipher

- ▶ Only have 26 possible ciphers
 - A maps to A,B,..Z
- ▶ Could simply try each in turn
- ▶ A **brute force search**
- ▶ Given ciphertext, just try all shifts of letters
- ▶ Do need to recognize when have plaintext
- ▶ Eg. break ciphertext "GCUA VQ DTGCM"

Mono-alphabetic Cipher

- ▶ Shuffle (jumble) the letters arbitrarily
- ▶ Each plaintext letter maps to a different random cipher text letter
- ▶ Hence key is 26 letters long

Plain letters: abcdefghijklmnopqrstuvwxyz

Cipher letters: dkvqfibjwpescxhtmyauolrgzn

Plaintext: if we wish to replace letters

Ciphertext: wi rf rwaj uh yftsdvfs fuufya

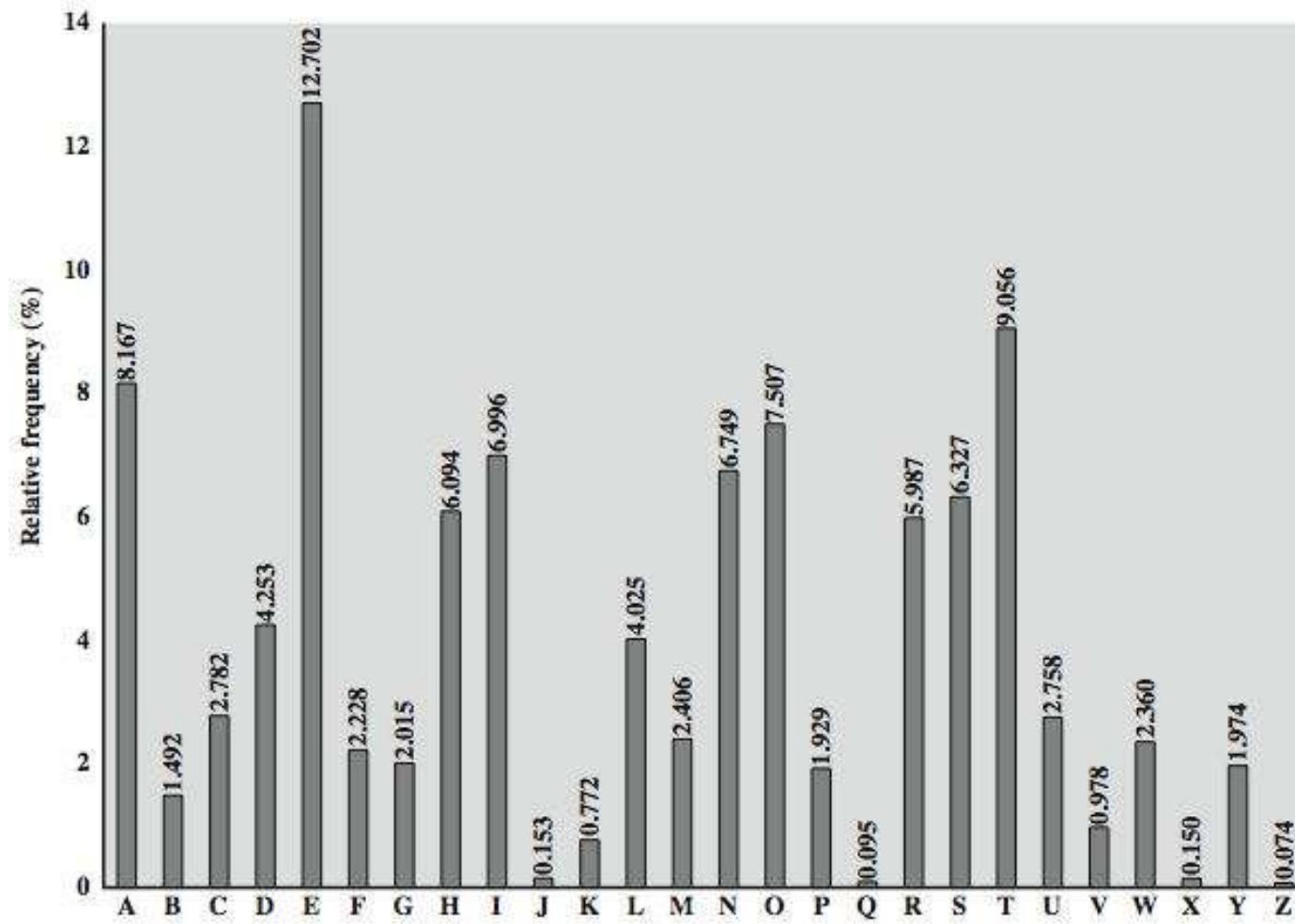
Mono-alphabetic Cipher Security

- ▶ Now have a total of $26! = 4 \times 10^{26}$ keys
- ▶ With so many keys, might think is secure
- ▶ But not secure against some cryptanalytic attacks.
- ▶ Problem is language characteristics

Language Statistics and Cryptanalysis

- ▶ Human languages are not random.
- ▶ Letters are not equally frequently used.
- ▶ In English, E is by far the most common letter, followed by T, R, N, I, O, A, S.
- ▶ Other letters like Z, J, K, Q, X are fairly rare.
- ▶ There are tables of single, double & triple letter frequencies for various languages

English Letter Frequencies



Statistics for double & triple letters

- ▶ In decreasing order of frequency
- ▶ Double letters:
th he an in er re es on, ...
- ▶ Triple letters:
the and ent ion tio for nde, ...

Example Cryptanalysis

- ▶ Given cipher text:

UZ QSO VUOHXMOPV GPOZPEVSG ZWSZ OPFPESX
UDBMETSX AIZ VUEPHZ HMDZSHZO WSFP APPD
TSVP QUZW YM XUZUHSX
EPYEPOPDZSZUFPO MB ZWP FUPZ HMDJ UD
TMOHMQ

- ▶ Count relative letter frequencies

- ▶ Guess P & Z are e and t

- ▶ Guess ZW is th and hence ZWP is the

- ▶ Proceeding with trial and error finally get:

it was disclosed yesterday that several informal
but direct contacts have been made with
~~political~~ representatives of the viet cong in
moscow

Playfair Cipher

- Not even the large number of keys in a monoalphabetic cipher provides security
- One approach to improving security was to encrypt multiple letters
- **Playfair Cipher** was invented by Charles Wheatstone in 1854, but named after his friend Baron Playfair

Playfair Key Matrix

- ▶ A 5X5 matrix of letters based on a keyword
- ▶ Fill in letters of keyword
- ▶ Fill rest of matrix with other letters
- ▶ Eg. using the keyword MONARCHY

M	O	N	A	R
C	H	Y	B	D
E	F	G	IJ	K
L	P	Q	S	T
U	V	W	X	Z

Encrypting and Decrypting

- ▶ Plaintext is encrypted two letters at a time
 1. if a pair is a repeated letter, insert filler like 'X'
 2. if both letters fall in the same row, replace each with letter to right (wrapping back to start from end)
 3. if both letters fall in the same column, replace each with the letter below it (wrapping to top from bottom)
 4. otherwise each letter is replaced by the letter in the same row and in the column of the other letter of the pair

Security of Playfair Cipher

- ▶ Equivalent to a mono-alphabetic cipher with an alphabet of $26 \times 26 = 676$ characters.
- ▶ Security is much improved over the simple mono-alphabetic cipher.
- ▶ Widely used for many years
 - Eg: by US & British military in WW1 and WW2
- ▶ Once thought to be unbreakable.
- ▶ Actually, it **can** be broken, because it still leaves some structure of plaintext intact.

Hill Cipher

- ▶ Takes two or three or more letter combinations to the same size combinations, e.g. “the” → “rqv”
- ▶ Uses simple linear equations
- ▶ An example of a “block” cipher encrypting a block of text at a time
- ▶ Numbered alphabet: a = 0, b = 1, c = 3, etc

Letter to Number Substitution

A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

Modular Inverses of Mod 26

A	1	3	5	7	9	11	15	17	19	21	23	25
A^{-1}	1	9	21	15	3	19	7	23	11	5	17	25

Example – Find the Modular Inverse of 9 for Mod 26

$$9 \cdot 3 = 27$$

$$27 \text{ Mod } 26 = 1$$

3 is the Modular Inverse of 9 Mod 26

Encryption

- ▶ Assign each letter in alphabet a number between 0 and 25
- ▶ Change message into 2×1 letter vectors
- ▶ Change each vector into 2×1 numeric vectors
- ▶ Multiply each numeric vector by encryption matrix
- ▶ Convert product vectors to letters

Change Message to Vectors

Message to encrypt = I

$$\begin{bmatrix} H \\ E \end{bmatrix} = \begin{bmatrix} 7 \\ 4 \end{bmatrix}$$

$$\begin{bmatrix} L \\ L \end{bmatrix} = \begin{bmatrix} 11 \\ 11 \end{bmatrix}$$

$$\begin{bmatrix} O \\ W \end{bmatrix} = \begin{bmatrix} 14 \\ 22 \end{bmatrix}$$

$$\begin{bmatrix} O \\ R \end{bmatrix} = \begin{bmatrix} 14 \\ 17 \end{bmatrix}$$

$$\begin{bmatrix} L \\ D \end{bmatrix} = \begin{bmatrix} 11 \\ 3 \end{bmatrix}$$

$$A = \begin{pmatrix} 2 & 1 \\ 3 & 4 \end{pmatrix}$$

A. A^{-1} = Identity

$$\begin{pmatrix} 2 & 1 \\ 3 & 4 \end{pmatrix} \quad \begin{pmatrix} 4 & -1 \\ -3 & 2 \end{pmatrix}$$

$$2*4 + 1*(-3)$$

$$3*4 + 4*(-3)$$

$$2*(-1) + 1*2$$

$$3*(-1) + 4*2$$

$$\begin{pmatrix} 5 & 0 \\ 0 & 5 \end{pmatrix}$$

Multiply Matrix by Vectors

$$\begin{bmatrix} 2 & 1 \\ 3 & 4 \end{bmatrix} \times \begin{bmatrix} 7 \\ 4 \end{bmatrix} = \begin{bmatrix} 14 + 4 \\ 21 + 16 \end{bmatrix} = \begin{bmatrix} 18 \\ 37 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 1 \\ 3 & 4 \end{bmatrix} \times \begin{bmatrix} 11 \\ 11 \end{bmatrix} = \begin{bmatrix} 22 + 11 \\ 33 + 44 \end{bmatrix} = \begin{bmatrix} 33 \\ 77 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 1 \\ 3 & 4 \end{bmatrix} \times \begin{bmatrix} 14 \\ 22 \end{bmatrix} = \begin{bmatrix} 28 + 22 \\ 42 + 88 \end{bmatrix} = \begin{bmatrix} 50 \\ 130 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 1 \\ 3 & 4 \end{bmatrix} \times \begin{bmatrix} 14 \\ 17 \end{bmatrix} = \begin{bmatrix} 28 + 17 \\ 42 + 68 \end{bmatrix} = \begin{bmatrix} 45 \\ 110 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 1 \\ 3 & 4 \end{bmatrix} \times \begin{bmatrix} 11 \\ 3 \end{bmatrix} = \begin{bmatrix} 22 + 3 \\ 33 + 12 \end{bmatrix} = \begin{bmatrix} 25 \\ 45 \end{bmatrix}$$

Convert to Mod 26

$$\begin{bmatrix} 18 \\ 37 \end{bmatrix} \text{ Mod } 26 = \begin{bmatrix} 18 \\ 11 \end{bmatrix}$$

$$\begin{bmatrix} 33 \\ 77 \end{bmatrix} \text{ Mod } 26 = \begin{bmatrix} 7 \\ 25 \end{bmatrix}$$

$$\begin{bmatrix} 50 \\ 130 \end{bmatrix} \text{ Mod } 26 = \begin{bmatrix} 24 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 45 \\ 110 \end{bmatrix} \text{ Mod } 26 = \begin{bmatrix} 19 \\ 6 \end{bmatrix}$$

$$\begin{bmatrix} 25 \\ 45 \end{bmatrix} \text{ Mod } 26 = \begin{bmatrix} 25 \\ 19 \end{bmatrix}$$

Convert Numbers to Letters

$$\begin{bmatrix} 18 \\ 11 \end{bmatrix} = \begin{bmatrix} S \\ L \end{bmatrix}$$

$$\begin{bmatrix} 7 \\ 25 \end{bmatrix} = \begin{bmatrix} H \\ Z \end{bmatrix}$$

$$\begin{bmatrix} 24 \\ 0 \end{bmatrix} = \begin{bmatrix} Y \\ A \end{bmatrix}$$

$$\begin{bmatrix} 19 \\ 6 \end{bmatrix} = \begin{bmatrix} T \\ G \end{bmatrix}$$

$$\begin{bmatrix} 25 \\ 19 \end{bmatrix} = \begin{bmatrix} Z \\ T \end{bmatrix}$$

“hello world” has been encrypted to **SLHZY ATGZT**

Decryption

- ▶ Change message into 2×1 letter vectors
- ▶ Change each vector into 2×1 numeric vectors
- ▶ Multiply each numeric vector by decryption matrix
- ▶ Convert new vectors to letters

Change Message to Vectors

Ciphertext to decrypt = SLHZYATGZ

$$\begin{bmatrix} S \\ L \end{bmatrix} = \begin{bmatrix} 18 \\ 11 \end{bmatrix}$$

$$\begin{bmatrix} H \\ Z \end{bmatrix} = \begin{bmatrix} 7 \\ 25 \end{bmatrix}$$

$$\begin{bmatrix} Y \\ A \end{bmatrix} = \begin{bmatrix} 24 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} T \\ G \end{bmatrix} = \begin{bmatrix} 19 \\ 6 \end{bmatrix}$$

$$\begin{bmatrix} Z \\ T \end{bmatrix} = \begin{bmatrix} 25 \\ 19 \end{bmatrix}$$

Multiply Matrix by Vectors

$$\begin{bmatrix} 6 & 5 \\ 15 & 16 \end{bmatrix} \times \begin{bmatrix} 18 \\ 11 \end{bmatrix} = \begin{bmatrix} 108 + 55 \\ 270 + 176 \end{bmatrix} = \begin{bmatrix} 163 \\ 446 \end{bmatrix}$$

$$\begin{bmatrix} 6 & 5 \\ 15 & 16 \end{bmatrix} \times \begin{bmatrix} 7 \\ 25 \end{bmatrix} = \begin{bmatrix} 42 + 125 \\ 105 + 400 \end{bmatrix} = \begin{bmatrix} 167 \\ 505 \end{bmatrix}$$

$$\begin{bmatrix} 6 & 5 \\ 15 & 16 \end{bmatrix} \times \begin{bmatrix} 24 \\ 0 \end{bmatrix} = \begin{bmatrix} 144 + 0 \\ 360 + 0 \end{bmatrix} = \begin{bmatrix} 144 \\ 360 \end{bmatrix}$$

$$\begin{bmatrix} 6 & 5 \\ 15 & 16 \end{bmatrix} \times \begin{bmatrix} 19 \\ 6 \end{bmatrix} = \begin{bmatrix} 114 + 30 \\ 285 + 96 \end{bmatrix} = \begin{bmatrix} 144 \\ 381 \end{bmatrix}$$

$$\begin{bmatrix} 6 & 5 \\ 15 & 16 \end{bmatrix} \times \begin{bmatrix} 25 \\ 19 \end{bmatrix} = \begin{bmatrix} 150 + 95 \\ 375 + 304 \end{bmatrix} = \begin{bmatrix} 245 \\ 679 \end{bmatrix}$$

Convert to Mod 26

$$\begin{bmatrix} 163 \\ 446 \end{bmatrix} \text{ Mod } 26 = \begin{bmatrix} 7 \\ 4 \end{bmatrix}$$

$$\begin{bmatrix} 167 \\ 505 \end{bmatrix} \text{ Mod } 26 = \begin{bmatrix} 11 \\ 11 \end{bmatrix}$$

$$\begin{bmatrix} 144 \\ 360 \end{bmatrix} \text{ Mod } 26 = \begin{bmatrix} 14 \\ 22 \end{bmatrix}$$

$$\begin{bmatrix} 144 \\ 381 \end{bmatrix} \text{ Mod } 26 = \begin{bmatrix} 14 \\ 17 \end{bmatrix}$$

$$\begin{bmatrix} 245 \\ 679 \end{bmatrix} \text{ Mod } 26 = \begin{bmatrix} 11 \\ 3 \end{bmatrix}$$

Convert Numbers to Letters

$$\begin{bmatrix} 7 \\ 4 \end{bmatrix} = \begin{bmatrix} H \\ E \end{bmatrix}$$

$$\begin{bmatrix} 11 \\ 11 \end{bmatrix} = \begin{bmatrix} L \\ L \end{bmatrix}$$

SLHZYATGZT has been decrypted to
“hello world”

$$\begin{bmatrix} 14 \\ 22 \end{bmatrix} = \begin{bmatrix} O \\ W \end{bmatrix}$$

$$\begin{bmatrix} 14 \\ 17 \end{bmatrix} = \begin{bmatrix} O \\ R \end{bmatrix}$$

$$\begin{bmatrix} 11 \\ 3 \end{bmatrix} = \begin{bmatrix} L \\ D \end{bmatrix}$$

Polyalphabetic Substitution Ciphers

- ▶ A sequence of monoalphabetic ciphers ($M_1, M_2, M_3, \dots, M_k$) is used in turn to encrypt letters.
- ▶ A key determines which sequence of ciphers to use.
- ▶ Each plaintext letter has multiple corresponding ciphertext letters.
- ▶ This makes cryptanalysis harder since the letter frequency distribution will be flatter.

Vigenère Cipher

- ▶ Simplest polyalphabetic substitution cipher
- ▶ Effectively multiple Caesar ciphers
- ▶ Key is multiple letters long $K = k_1 \ k_2 \dots k_d$
- ▶ i^{th} letter specifies i^{th} alphabet to use
- ▶ Use each alphabet in turn
- ▶ Repeat from start after d letters in message
- ▶ Decryption simply works in reverse

TABLE 8-2 The Vigenère Square

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	
1	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	A
2	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
3	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
4	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
5	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
6	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
7	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
8	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
9	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
10	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
11	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
12	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
13	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
14	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
15	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
16	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
17	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
18	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
19	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
20	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
21	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
22	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
23	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
24	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
25	Z	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
26	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

Example of Vigenère Cipher

- ▶ Keyword: *deceptive*

key: **d**ecept**i**ve**d**ecept**i**ve**d**ecept**i**ve

plaintext: **w**e**a**re**d**is**c**o**v**e**r**e**d**s**a****v**e**y**orself

ciphertext: **Z**I**C**VTW**Q**NG**R**ZGVTWAVZ**H**C**Q**YGLMGJ

Security of Vigenère Ciphers

- ▶ There are multiple ciphertext letters corresponding to each plaintext letter.
- ▶ So, letter frequencies are obscured but not totally lost.
- ▶ To break Vigenere cipher:
 1. Try to guess the key length. How?
 2. If key length is N , the cipher consists of N Caesar ciphers. Plaintext letters at positions k , $N+k$, $2N+k$, $3N+k$, etc., are encoded by the same cipher.
 3. Attack each individual cipher as before.

Guessing the Key Length

- ▶ Main idea: Plaintext words separated by multiples of the key length are encoded in the same way.
- ▶ In our example, if plaintext = “...thexxxxxxthe...” then “the” will be encrypted to the same ciphertext words.
- ▶ So look at the ciphertext for repeated patterns.
- ▶ E.g. repeated “VTW” in the previous example suggests a key length of 3 or 9:

ciphertext: ZIC**VTW**QNGRZG**VTW**AVZHCQYGLMGJ

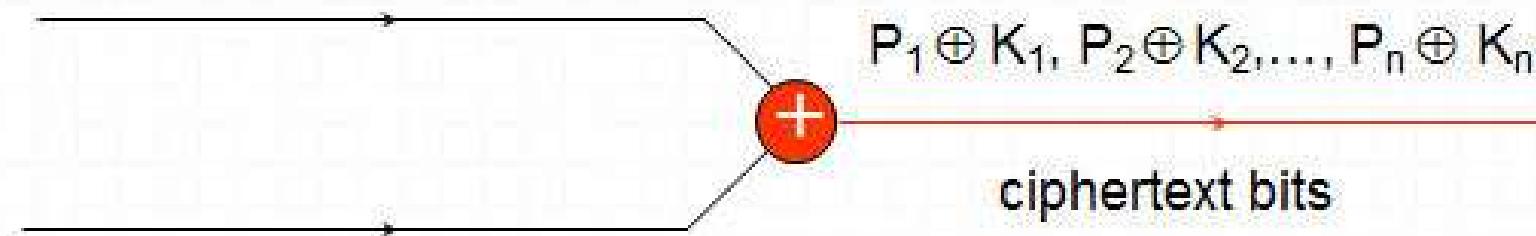
- ▶ Of course, the repetition could be a random fluke.
- ▶ Then attack each monoalphabetic cipher individually using same techniques as before

Vernam Cipher

- ▶ Ultimate defense is to use a key as long as the plaintext with no statistical relationship to it
- ▶ Invented by AT&T engineer Gilbert Vernam in 1918
- ▶ Originally proposed using a very long but eventually repeating key

Vernam cipher

random key bits K_1, K_2, \dots, K_n



plaintext bits P_1, P_2, \dots, P_n

$P_1 \oplus K_1, P_2 \oplus K_2, \dots, P_n \oplus K_n$

ciphertext bits

Vernam Cipher Example

the letters would first be converted to their numeric equivalents, as shown here.

V	E	R	N	A	M	C	I	P	H	E	R
21	4	17	13	0	12	2	8	15	7	4	17

Next, we must generate random numbers to combine with the letter codes. Suppose the following series of random two-digit numbers is generated.

76	48	16	82	44	03	58	11	60	05	48	88
----	----	----	----	----	----	----	----	----	----	----	----

The encoded form of the message is the sum mod 26 of each coded letter with the corresponding random number. The result is then encoded in the usual base-26 alphabet representation.

Plaintext	V	E	R	N	A	M	C	I	P	H	E	R
Numeric Equivalent	21	4	17	13	0	12	2	8	15	7	4	17
+ Random Number	76	48	16	82	44	3	58	11	60	5	48	88
= Sum	97	52	33	95	44	15	60	19	75	12	52	105
= mod 26	19	0	7	17	18	15	8	19	23	12	0	1
Ciphertext	t	a	h	r	s	p	i	t	x	m	a	b

Thus, the message

VERNAM CIPHER

One-Time Pad

- ▶ If a truly random key as long as the message is used, the cipher will be secure
- ▶ Called as One-Time pad
- ▶ It is unbreakable since ciphertext bears no statistical relationship to the plaintext
- ▶ Since for **any plaintext & any ciphertext** there exists a key mapping one to other
- ▶ Can use the key only **once** though
- ▶ Problems in generation & safe distribution of key

- ▶ The message is represented as a binary string (a sequence of 0's and 1's using a coding mechanism such as ASCII coding).
- ▶ The key is a truly random sequence of 0's and 1's of the same length as the message.
- ▶ The encryption is done by adding the key to the message modulo 2, bit by bit. This process is often called *exclusive or*, and is denoted by *XOR*. The symbol \oplus is used

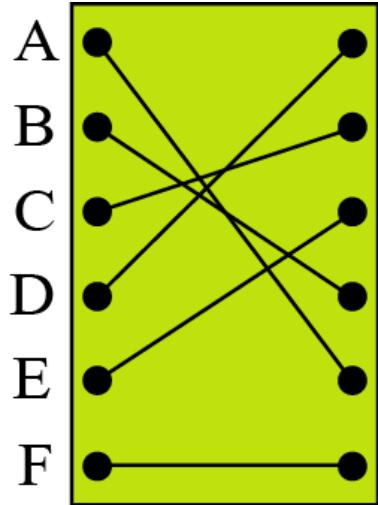
Example

- ▶ message = 'IF'
- ▶ then its ASCII code =(1001001 1000110)
- ▶ key = (1010110 0110001)
- ▶ *Encryption:*
 - 1001001 1000110 plaintext
 - 1010110 0110001 key
 - 0011111 1110110 cipher text
- ▶ *Decryption:*
 - 0011111 1110110 cipher text
 - 1010110 0110001 key
 - 1001001 1000110 plaintext

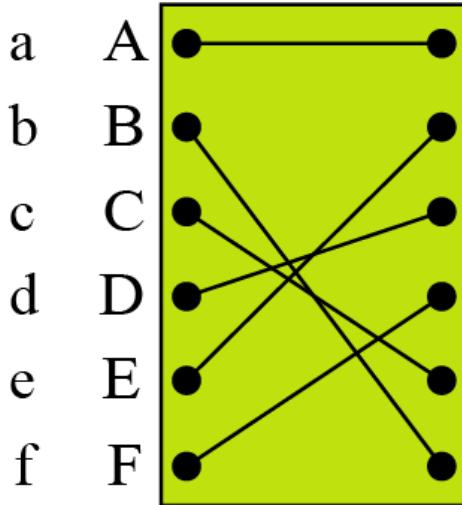
Rotor Cipher

- ▶ Before modern ciphers, rotor machines were most common complex ciphers in use.
- ▶ Widely used in WW2.
- ▶ Used a series of rotating cylinders.
- ▶ Implemented a polyalphabetic substitution cipher of period K.
- ▶ With 3 cylinders, $K = 26^3 = 17,576$.
- ▶ With 5 cylinders, $K = 26^5 = 12 \times 10^6$.

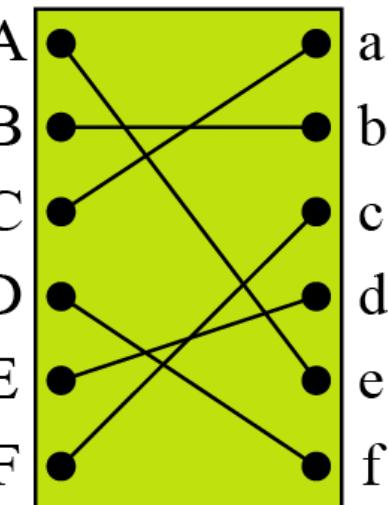
Rotor Cipher



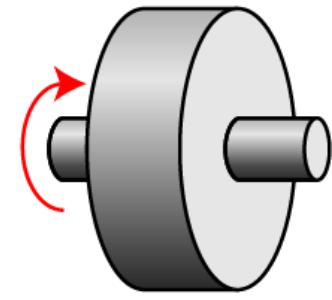
Initial
position



After first
rotation



After second
rotation



A three letter word such as “bee” is encrypted as “BCA”.

Transposition Ciphers

- ▶ Also called **permutation** ciphers.
- ▶ A transposition cipher reorders symbols.
- ▶ Can recognise these since have the same frequency distribution as the original text

Rail Fence cipher

- ▶ Write message letters out diagonally over a number of rows
- ▶ Then read off cipher row by row
- ▶ For example to send the message “Meet me at the park” to Bob, Alice writes

The diagram illustrates the Rail Fence cipher process. It shows the letters of the message "Meet me at the park" arranged in two diagonal rows. The first row starts with 'm', followed by 'e', then 't', then 'e', then 't', then 'h', then 'e', then 'p', then 'r'. The second row starts with 'e', followed by 't', then 'a', then 't', then 'e', then 'p', then 'a', then 'k'. Arrows indicate the flow from left to right within each row.

m	↓	e	↗	e	↓	t	↗	m	↓	e	↗	a	↓	t	↗	t	↓	h	↗	e	↓	p	↗	a	↓	r	↗	k
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

- ▶ Cipher text
- MEMATEAKETETHPR

Row Transposition Ciphers

- ▶ Plaintext is written row by row in a rectangle.
- ▶ Ciphertext: write out the **columns** in an order specified by a key.

Key: 3 4 2 1 5 6 7

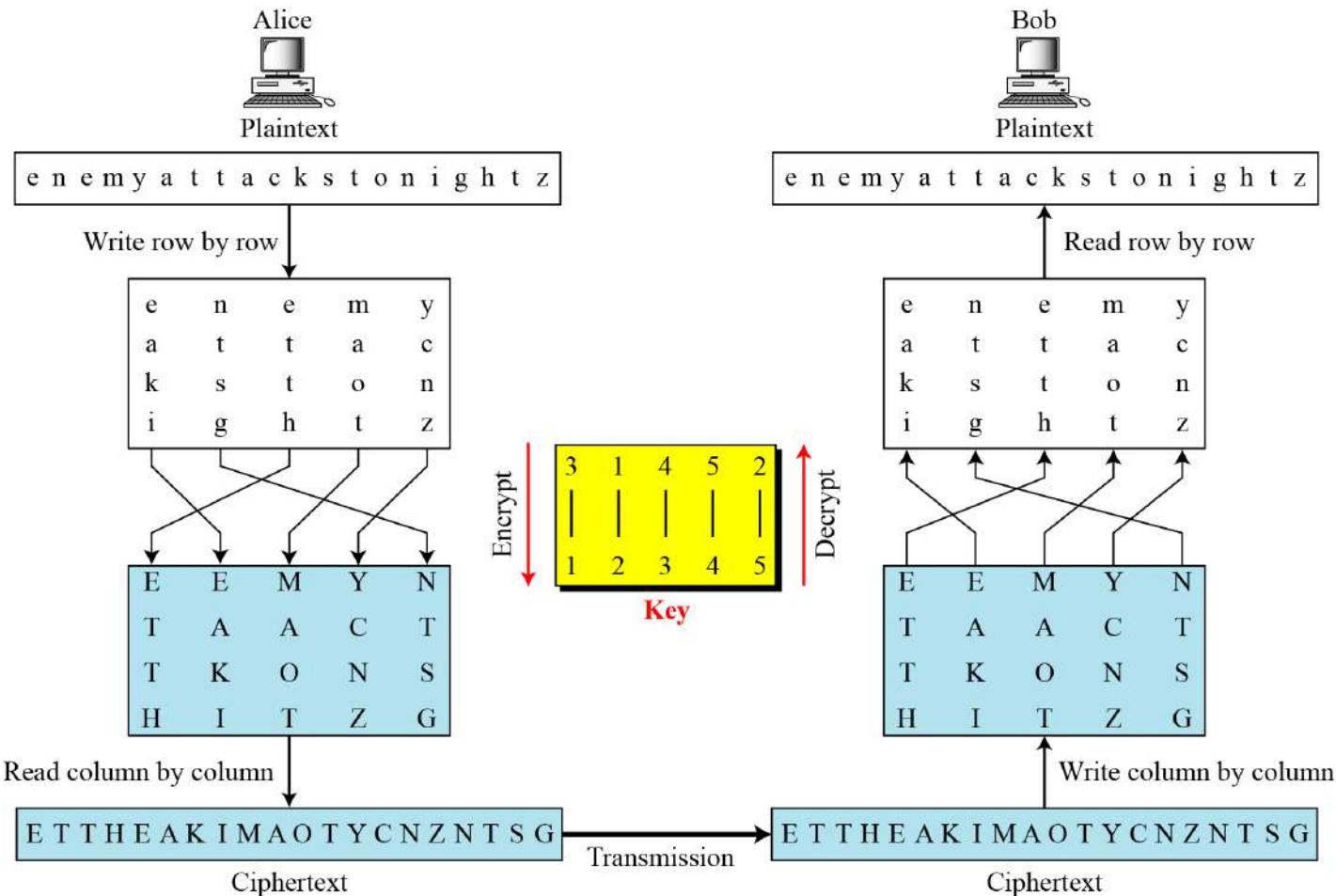
Plaintext:

a	t	t	a	c	k	p
o	s	t	p	o	n	e
d	u	n	t	i		t
w	o	a	m	x	y	z

Ciphertext:

TTNAAPMTSUOAODWCOIXKNLYPETZ

Example 2



Product Ciphers

- ▶ Ciphers using substitutions or transpositions are not secure because of language characteristics
- ▶ Consider using several ciphers in succession to make harder, but:
 - two substitutions make a more complex substitution
 - two transpositions make more complex transposition
- ▶ **Uses a sequence of substitutions and transpositions**
 - Harder to break than just substitutions or transpositions
- ▶ This is a bridge from classical to modern ciphers

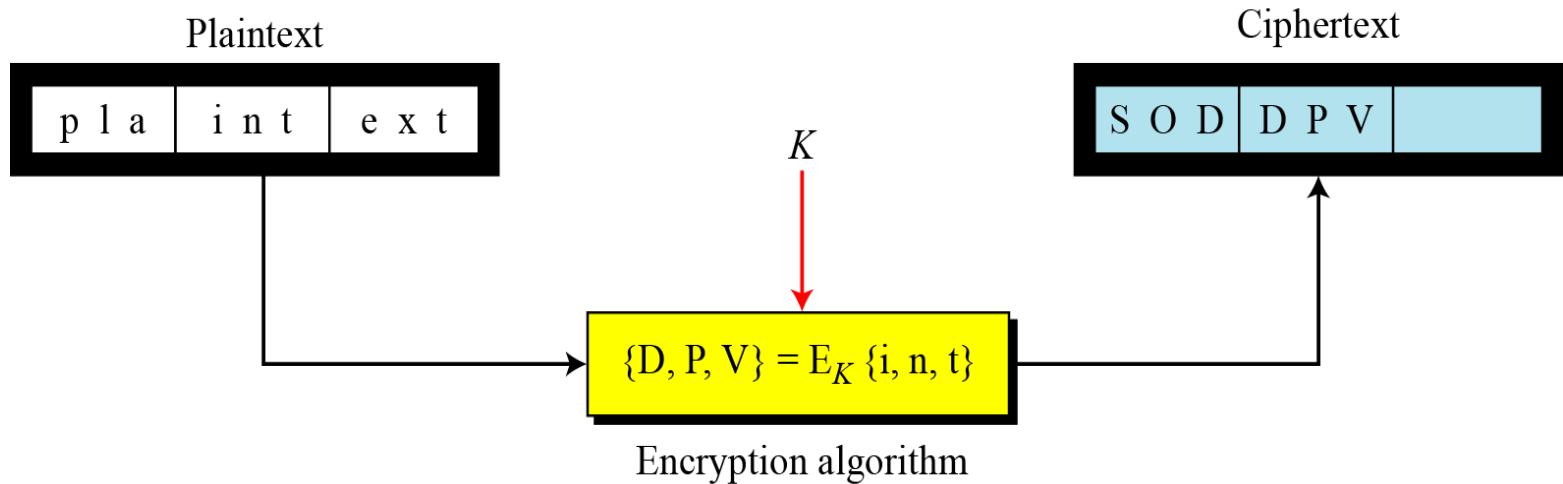
Block Ciphers

- ▶ Traditional Block Cipher Structure
 - Stream and Block Ciphers
 - The Feistel Cipher
- ▶ The Data Encryption Standard
 - DES Details
 - DES Design Issues
 - The Strength of DES
 - Differential and Linear Cryptanalysis

- ▶ A stream cipher is the one that encrypts a digital data stream one bit or one byte at a time.
- ▶ A block cipher is one which a block of plain text is treated as a whole and used to produce a cipher text block of equal length. Typically, a block size of 64 or 128 bits is used.

Block ciphers

- ▶ Encrypt a block of plaintext as a whole to produce same sized ciphertext
- ▶ Typical block sizes are 64 or 128 bits
- ▶ Modes of operation used to apply block ciphers to larger plaintexts



Reversible and Irreversible Mappings

- ▶ n-bit block cipher takes n bit plaintext and produces n bit ciphertext
- ▶ 2^n possible different plaintext blocks
- ▶ Encryption must be reversible (decryption possible)
- ▶ Each plaintext block must produce unique ciphertext block
- ▶ Total transformations is $2^n!$

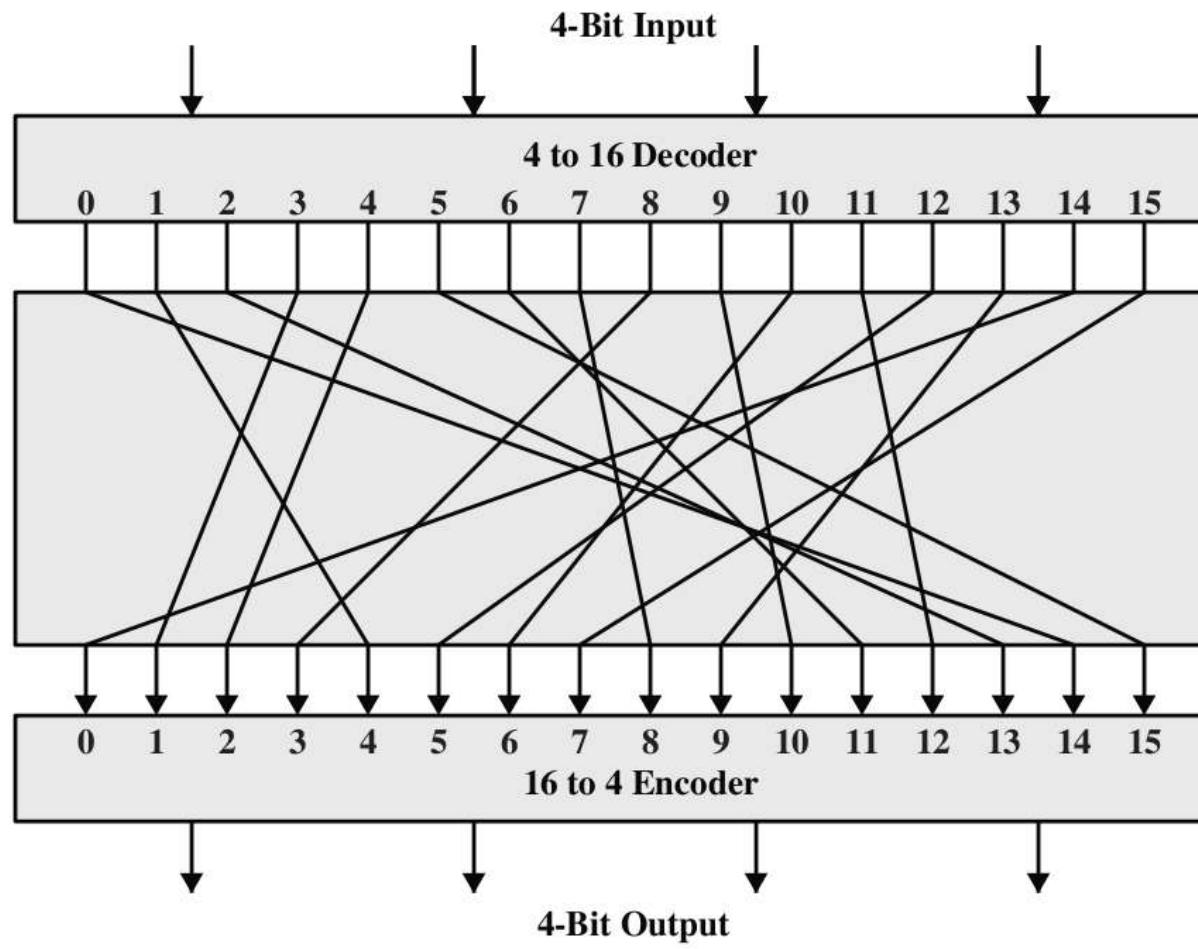
Reversible Mapping

Plaintext	Ciphertext
00	11
01	10
10	00
11	01

Irreversible Mapping

Plaintext	Ciphertext
00	11
01	10
10	01
11	01

General Block Substitution



Encryption/Decryption Tables

Plaintext	Ciphertext
0000	1110
0001	0100
0010	1101
0011	0001
0100	0010
0101	1111
0110	1011
0111	1000
1000	0011
1001	1010
1010	0110
1011	1100
1100	0101
1101	1001
1110	0000
1111	0111

Ciphertext	Plaintext
0000	1110
0001	0011
0010	0100
0011	1000
0100	0001
0101	1100
0110	1010
0111	1111
1000	0111
1001	1101
1010	1001
1011	0110
1100	1011
1101	0010
1110	0000
1111	0101

Ideal Block Cipher

- ▶ n-bit input maps to 2^n possible input states
- ▶ Substitution used to produce 2^n output states
- ▶ Output states map to n-bit output
- ▶ Ideal block cipher allows maximum number of possible encryption mappings from plaintext block
- ▶ Problems with ideal block cipher:
 - Small block size: equivalent to classical substitution cipher; cryptanalysis based on statistical characteristics feasible
 - Large block size: key must be very large; performance/implementation problems

Practical Block Ciphers

- ▶ Modern block ciphers use a key of K bits to specify a **random** subset of 2^K mappings.
- ▶ If $K \approx N$,
 - 2^K is much smaller than $2^N!$
 - But is still very large.
 - If the selection of the 2^K mappings is **random**, the resulting cipher will be a good approximation of the **ideal** block cipher.
- ▶ Horst Feistel, in 1970s, proposed a method to achieve this.

Feistel Structure for Block Ciphers

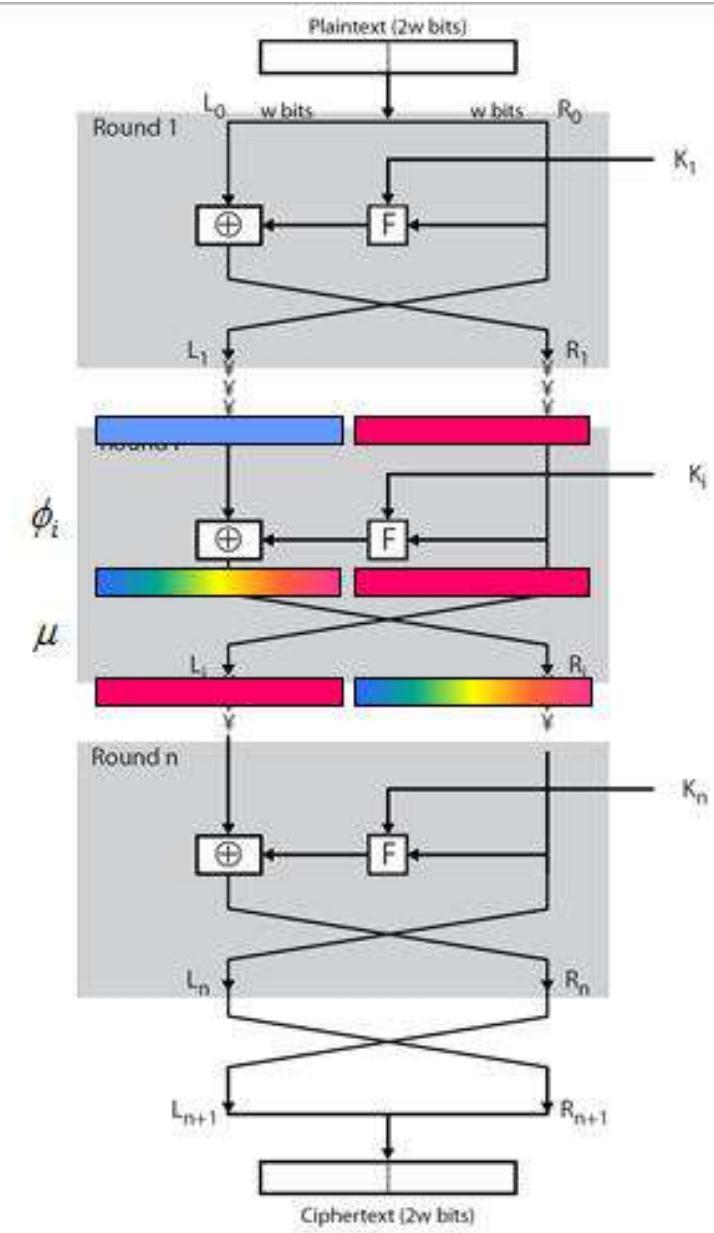
- ▶ Feistel proposed applying two or more simple ciphers in sequence so final result is cryptographically stronger than component ciphers
- ▶ n-bit block length; k-bit key length; 2^k transformations
- ▶ Feistel cipher alternates: substitutions, transpositions(permutations)
- ▶ Applies concepts of diffusion and confusion
- ▶ Applied in many ciphers today

Diffusion and Confusion

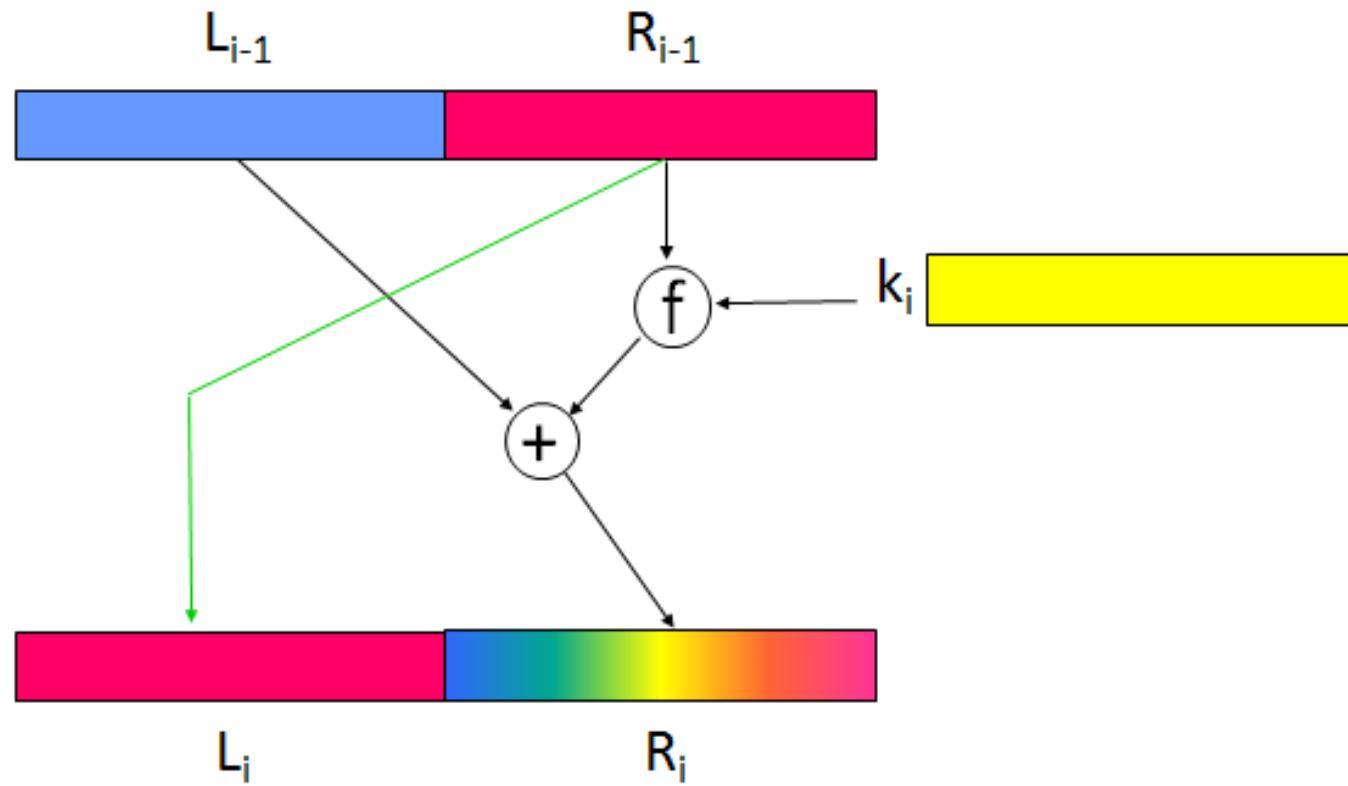
- ▶ Diffusion
 - Statistical nature of plaintext is reduced in cipher text
 - E.g. A plaintext letter affects the value of many cipher text letters
 - How: repeatedly apply permutation (transposition) to data, and then apply function
- ▶ Confusion
 - Make relationship between cipher text and key as complex as possible
 - Even if attacker can find some statistical characteristics of cipher text, still hard to find key
 - How: apply complex (non-linear) substitution algorithm

Feistel Structure for Block Ciphers

- ▶ Approach:
 - Plaintext split into halves
 - Sub keys (or round keys) generated from key
 - Round function, F, applied to right half
 - Apply substitution on left half using XOR
 - Apply permutation: interchange to halves



Round i



Using the Feistel Structure

- ▶ Exact implementation depends on various design features
 - Block size, e.g. 64, 128 bits: larger values leads to more diffusion
 - Key size, e.g. 128 bits: larger values leads to more confusion, resistance against brute force
 - Number of rounds, e.g. 16 rounds
 - Sub key generation algorithm: should be complex
 - Round function F: should be complex
- ▶ Other factors include fast encryption in software and ease of analysis
- ▶ Tradeoff: security vs performance

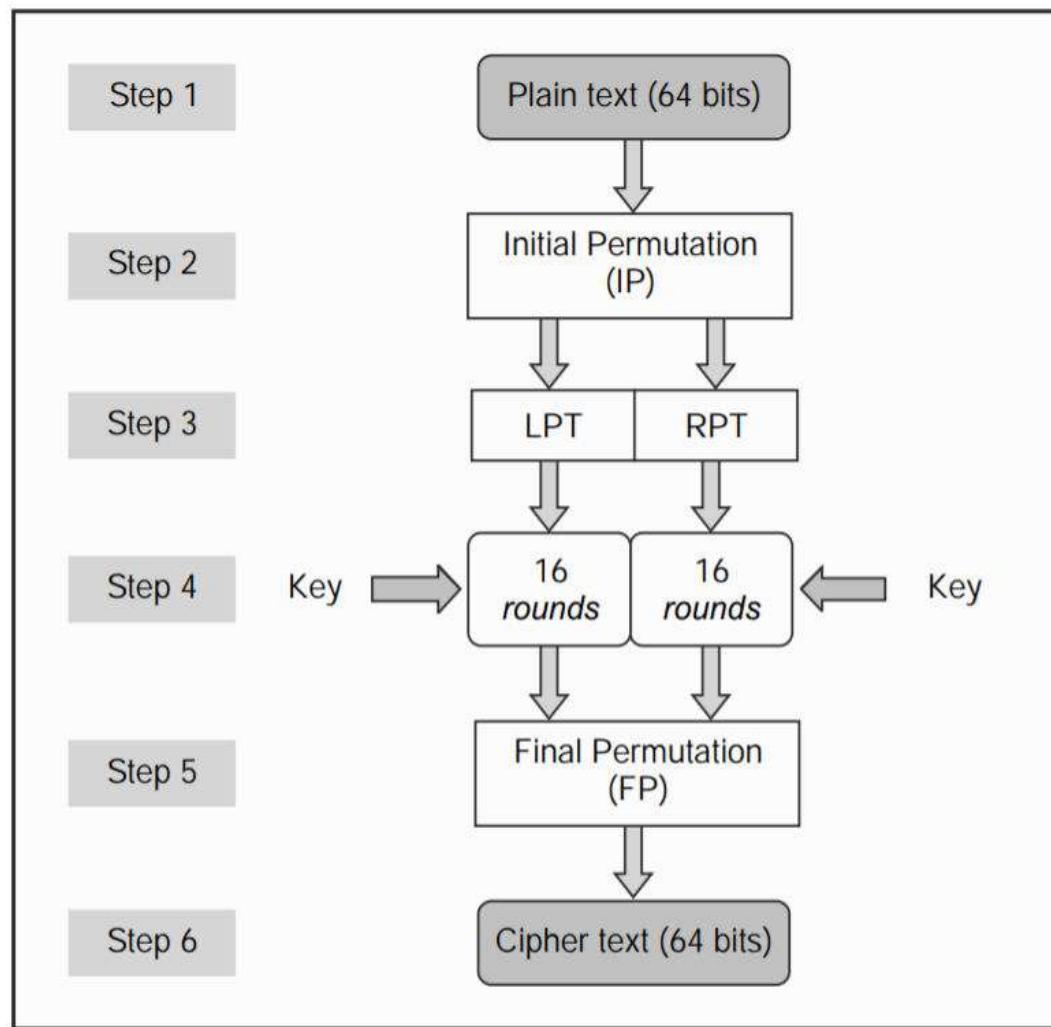
Feistel decryption

- ▶ Same as encryption, except
- ▶ Ciphertext is input
- ▶ Use keys in reverse order
- ▶ At each round the output is equal to the corresponding value of the encryption process with the two halves of the value swapped
- ▶ Final permutation (swap) realigns 2 halves

Data Encryption Standard

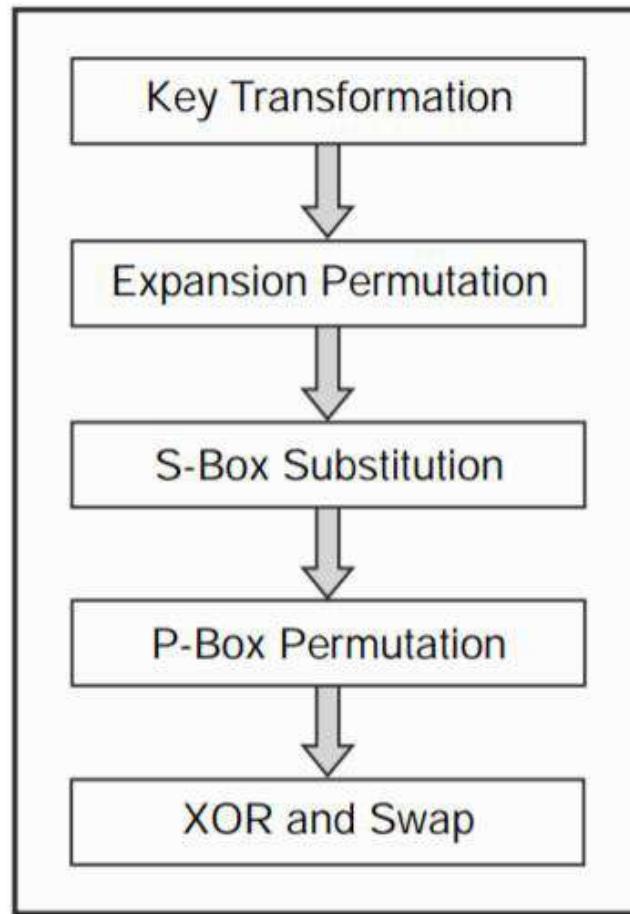
- ▶ DES is a block cipher.
- ▶ It encrypts data in blocks of size of 64 bit each which produces 64 bits of cipher text.
- ▶ The same algorithm and key are used for encryption and decryption, with minor differences.
- ▶ The key length is 56 bits.

Broad level Steps in DES:



- ▶ In the first step, the 64 bit plain text block is handed over to an initial Permutation (IP) function.
- ▶ The initial permutation performed on plain text.
- ▶ Next the initial permutation (IP) produces two halves of the permuted block; says Left Plain Text (LPT) and Right Plain Text (RPT).
- ▶ Now each LPT and RPT to go through 16 rounds of encryption process.
- ▶ In the end, LPT and RPT are rejoined and a Final Permutation (FP) is performed on the combined block
- ▶ The result of this process produces 64 bit cipher text.

Details of one Round in DES:



- ▶ **Step 1: Key transformation**
- ▶ Initial 64-bit key is transformed into a 56-bit key by discarding every 8th bit of the initial key.
- ▶ From this 56-bit key, a different 48-bit Sub Key is generated during each round using a process called as key transformation.

- ▶ **Step 2: Expansion permutation**
- ▶ After Initial Permutation, we had two 32-bit plain text areas, called as Left Plain Text (LPT) and Right Plain Text (RPT).
- ▶ During expansion permutation, the RPT is expanded from 32 bits to 48 bits.
- ▶ Besides increasing the bit size from 32 to 48, the bits are permuted as well, hence the name expansion permutation.

- ▶ **Step 3:S-box substitution**
- ▶ S-box substitution is a process that accepts the 48-bit input from the XOR operation involving the compressed key and expanded RPT and produces a 32-bit output using the substitution technique.

- ▶ **Step 4: P-box permutation**
- ▶ The output of S-box consists of 32 bits. These 32 bits are permuted using a P-box.
- ▶ This straightforward permutation mechanism involves simple permutation (i.e. replacement of each bit with another bit, as specified in the P-box table, without any expansion or compression).
- ▶ This is called as P-box Permutation.

- ▶ **Step 5: XOR and Swap**
- ▶ The left half portion of the initial 64-bit plain text block (i.e. LPT) is XORED with the output produced by P-box permutation.
- ▶ The result of this XOR operation becomes the new right half (i.e. RPT). The old right half (i.e. RPT) becomes the new left half, in a process of swapping.

Modes of operation

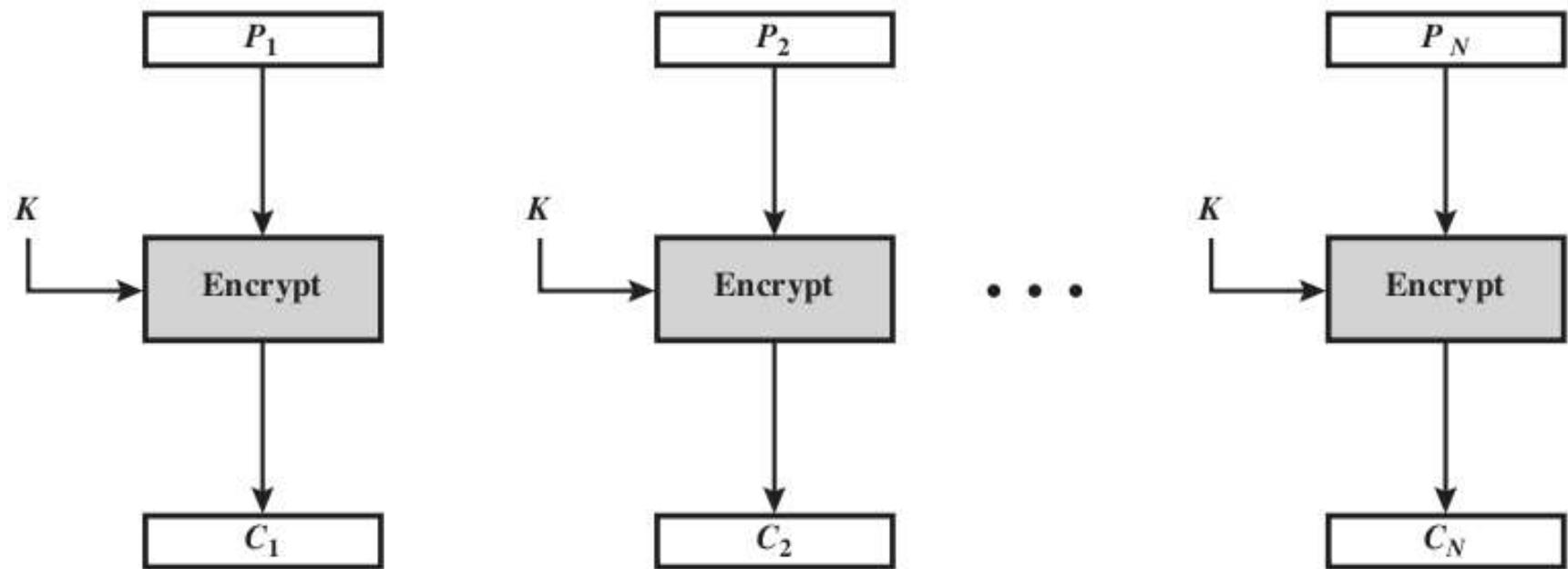
- ▶ A mode of operation is a technique for enhancing the effect of a cryptographic algorithm or adapting the algorithm for an application
- ▶ 5 modes of operation have been defined by NIST

- ▶ Five modes of operation are:
 - Electronic Code Book mode
 - Cipher Block Chaining mode
 - Cipher Feed Back mode
 - Output Feed Back mode
 - Counter mode

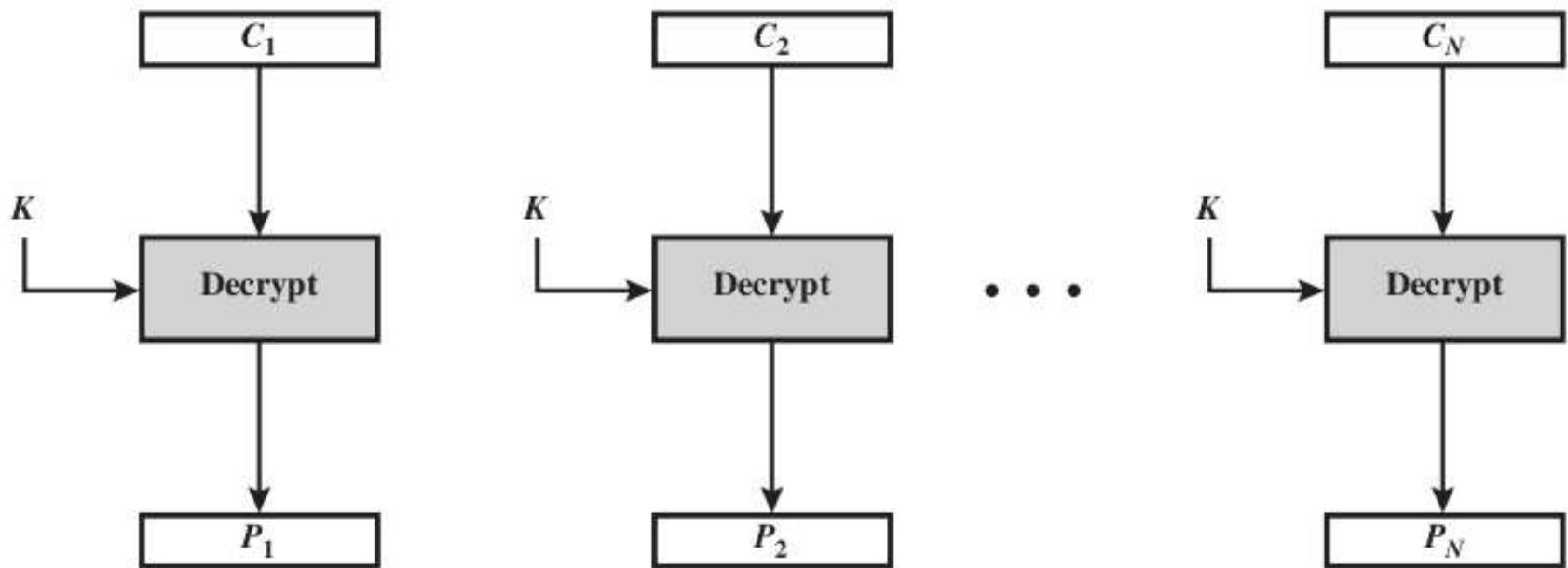
Electronic Code Book

- ▶ Simplest mode of operation
- ▶ Plain text is divided into N blocks
- ▶ Block size is n bits
- ▶ Same key is used to encrypt and decrypt

ECB Encryption



ECB Decryption



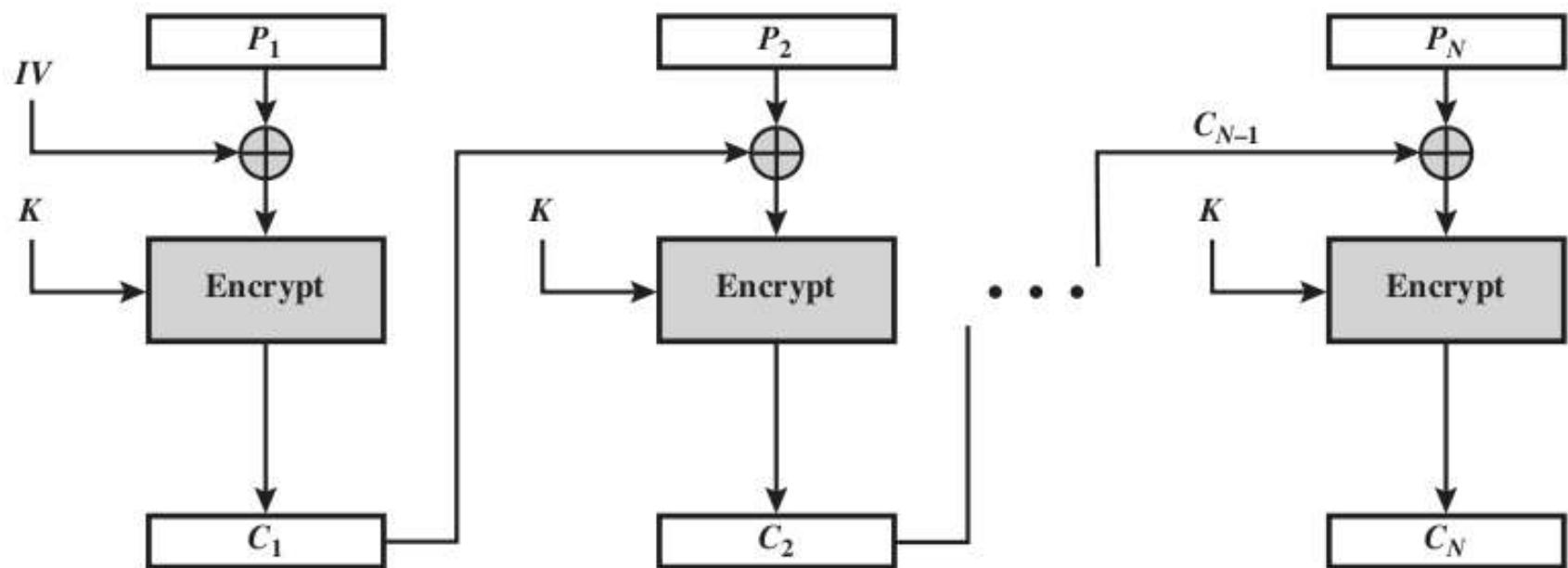
ECB-Pros and Cons

- ▶ Encrypted message blocks are independent
- ▶ Strength – in some applications the independence of message blocks is very useful
 - Databases
 - Parallelizing encryption / decryption
- ▶ Problem: with long message, repetition in plaintext may cause repetition in cipher text
 - ECB mode is rarely used and generally regarded as an insecure mode of operation.

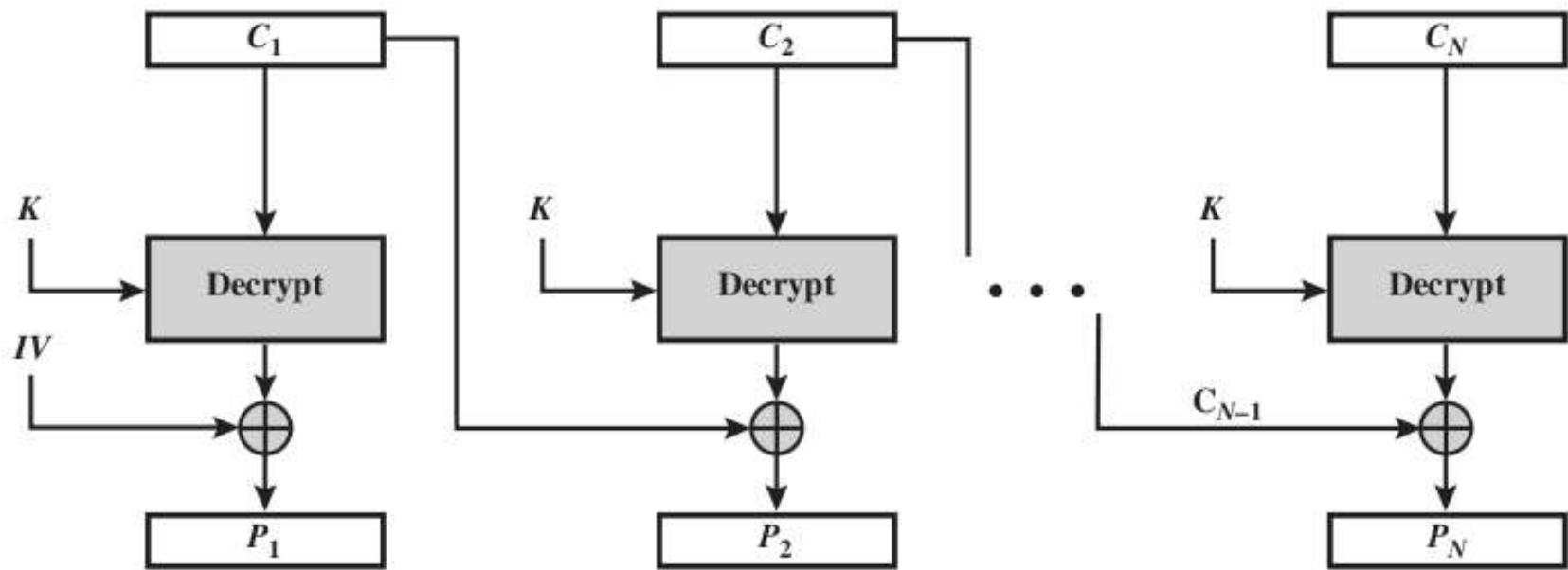
CBC (Cipher Block Chaining Mode)

- ▶ In CBC mode, each block of plain text is XORed with the previous cipher text block before being encrypted
- ▶ This way each cipher text block depends on all plain text blocks processed up to that point. To make each message unique, an initialization vector must be used in the first block.
- ▶ An initialization vector is a block of bits used by several modes to randomize the encryption.

CBC Encryption



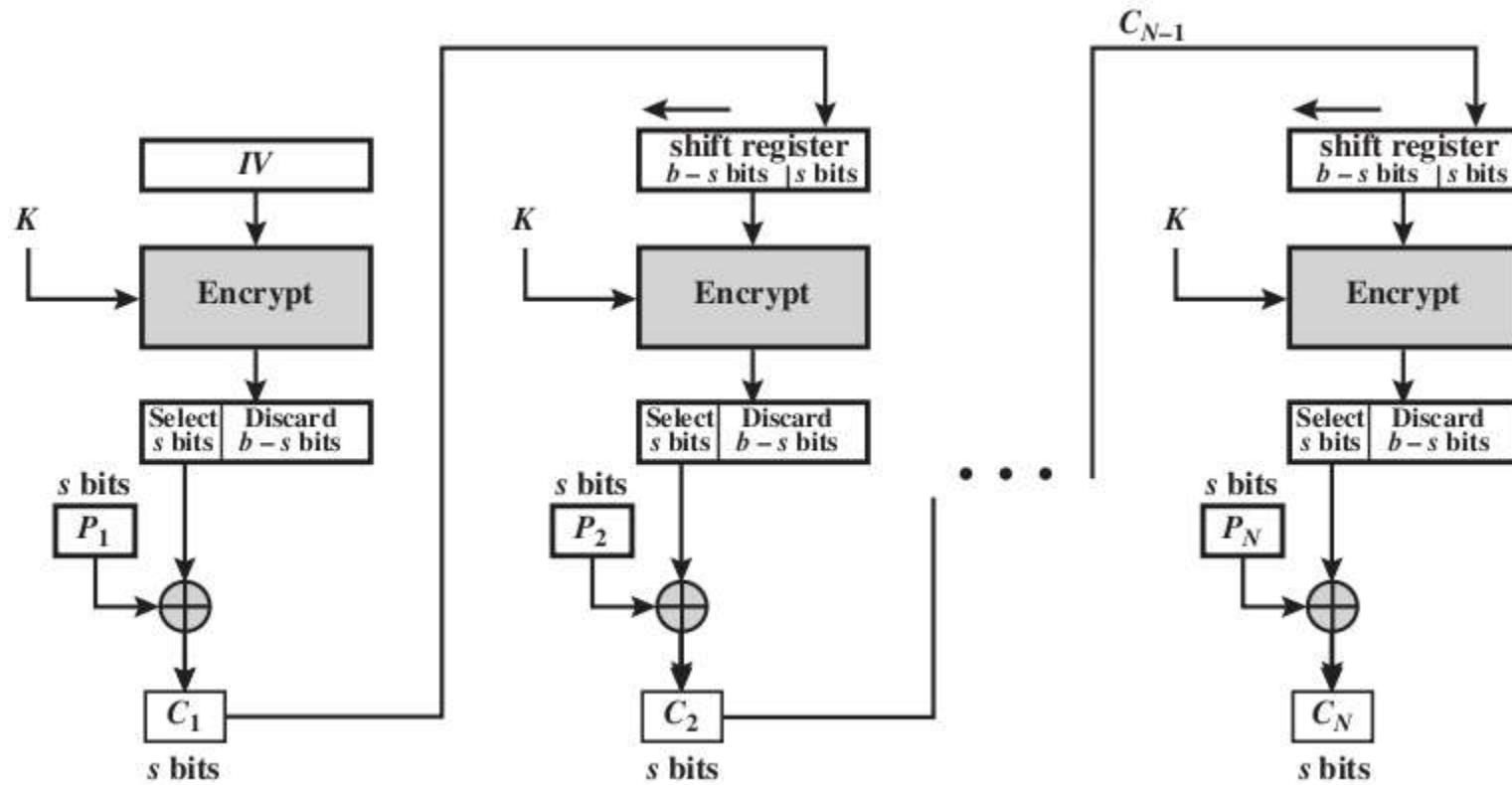
CBC Decryption



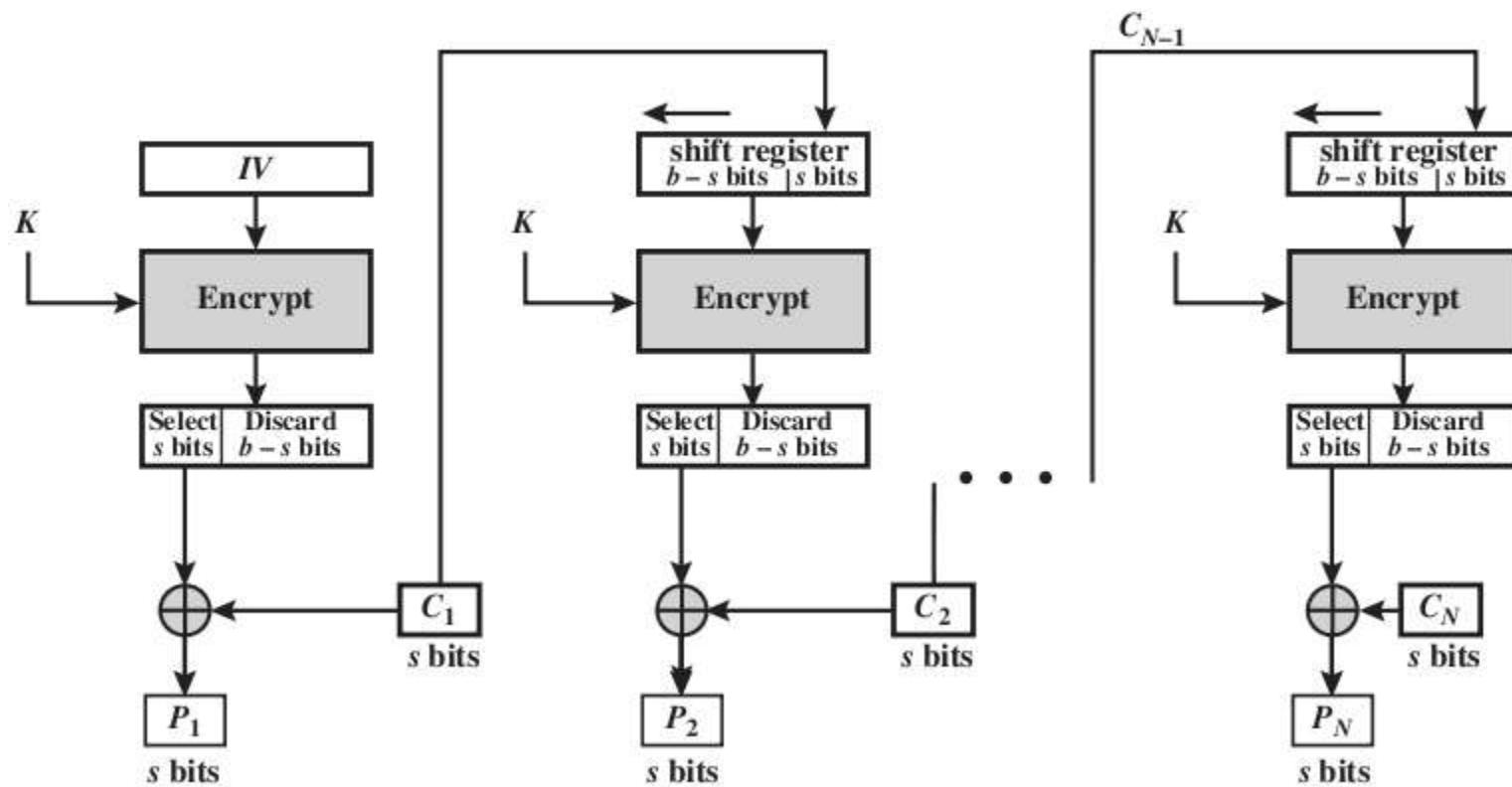
Cipher Feedback Mode (CFB)

- ▶ Message is treated as a stream of bits
- ▶ Added to the output of the block cipher
- ▶ Result is fed back for next stage (hence name)
- ▶ Standard allows any number of bit (1,8 or 64 or whatever) to be feed back
 - denoted CFB-1, CFB-8, CFB-64 etc
- ▶ CFB-64 is used most often (most efficient)
- ▶ In CFB mode you never actually use the encryption algorithm to **decrypt** anything!
- ▶ Applications: stream data encryption, authentication

CFB Encryption



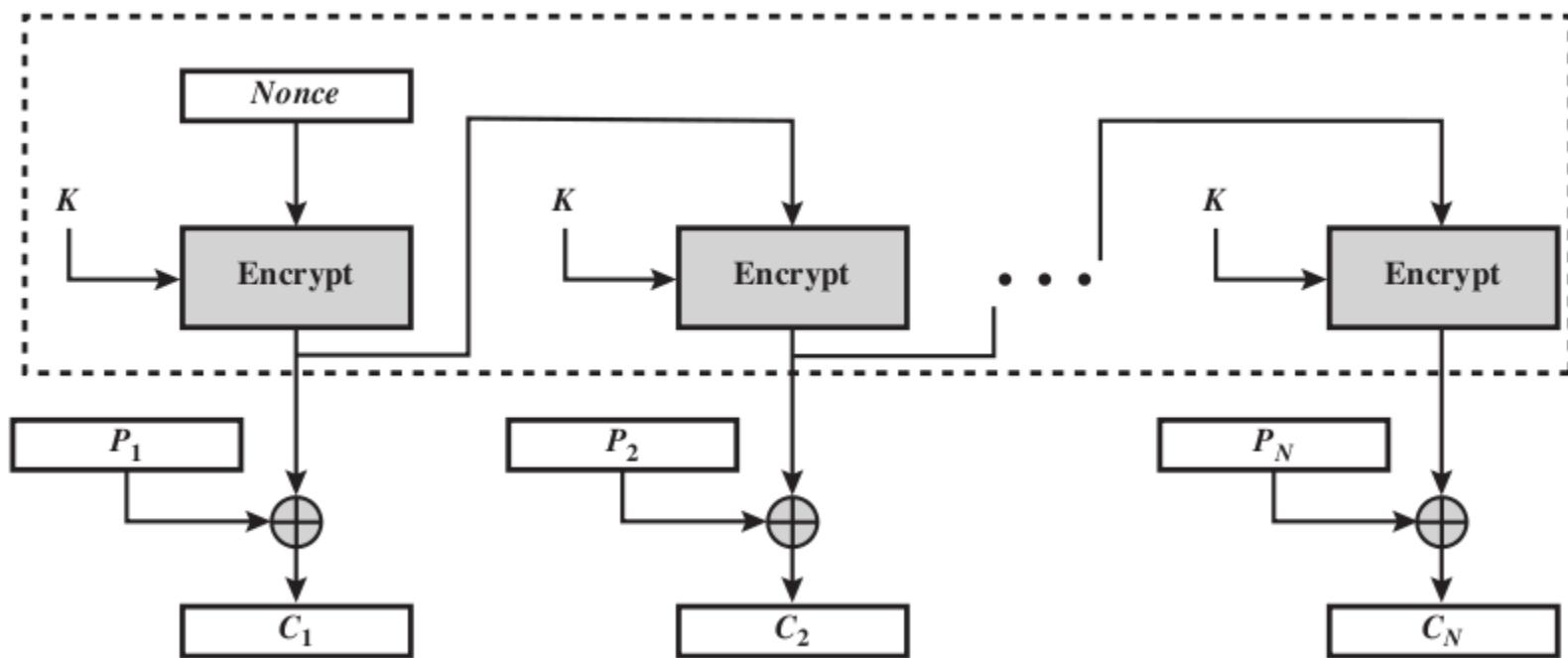
CFB Decryption



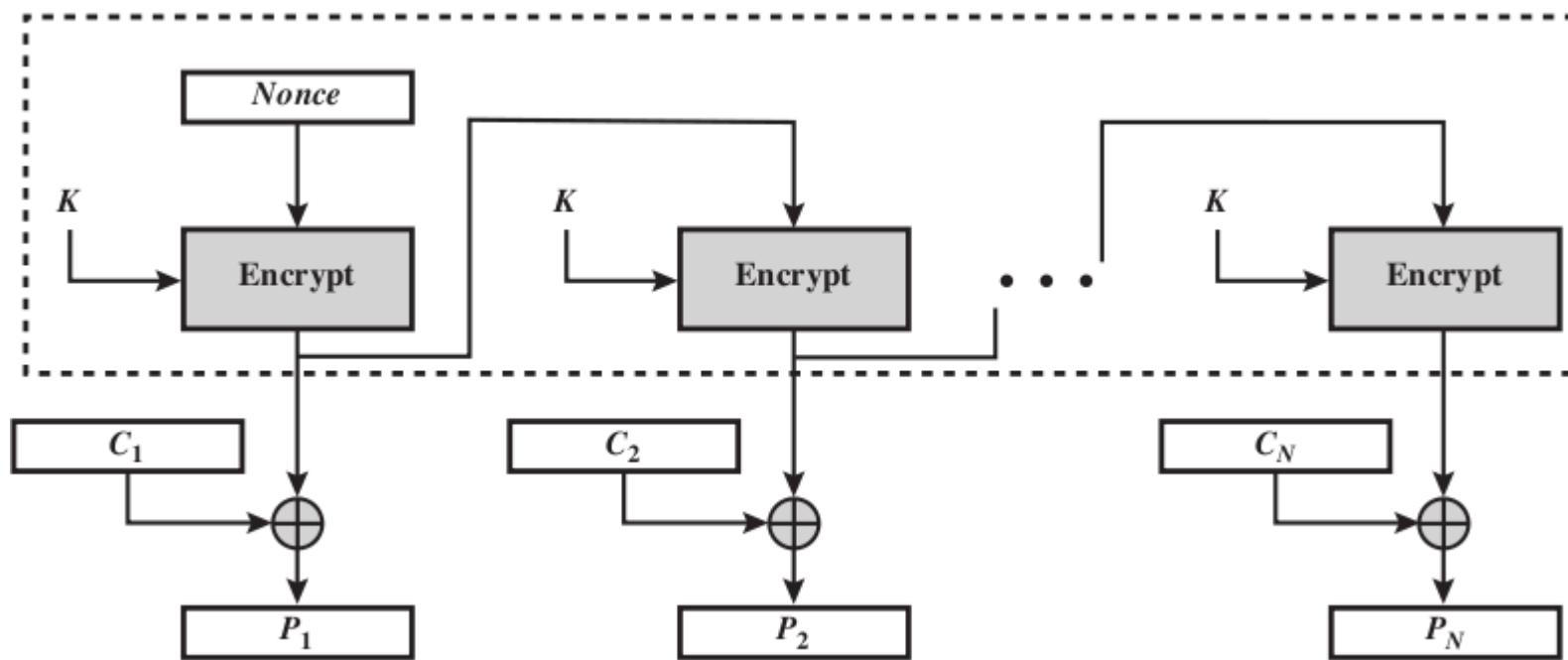
OFB (Output Feedback Mode)

- ▶ Output feedback mode is very similar to CFB mode, with one difference: each bit in the cipher text is independent of the previous bit or previous bits.
- ▶ This avoids error propagation
- ▶ If an error occur in transmission, it does not affect the bits that follow.
- ▶ Like cipher feedback mode, both the sender and the receiver use the encryption algorithm.

OFB Encryption



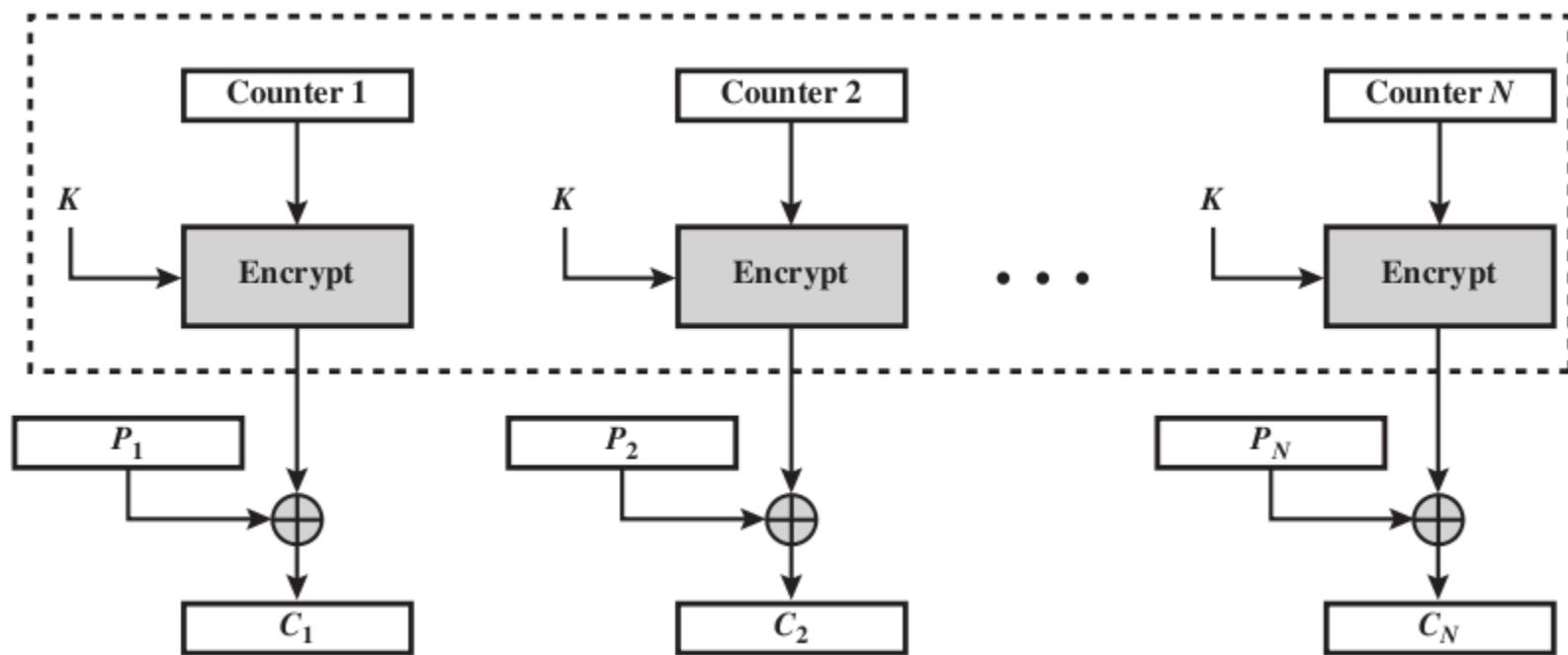
OFB Decryption



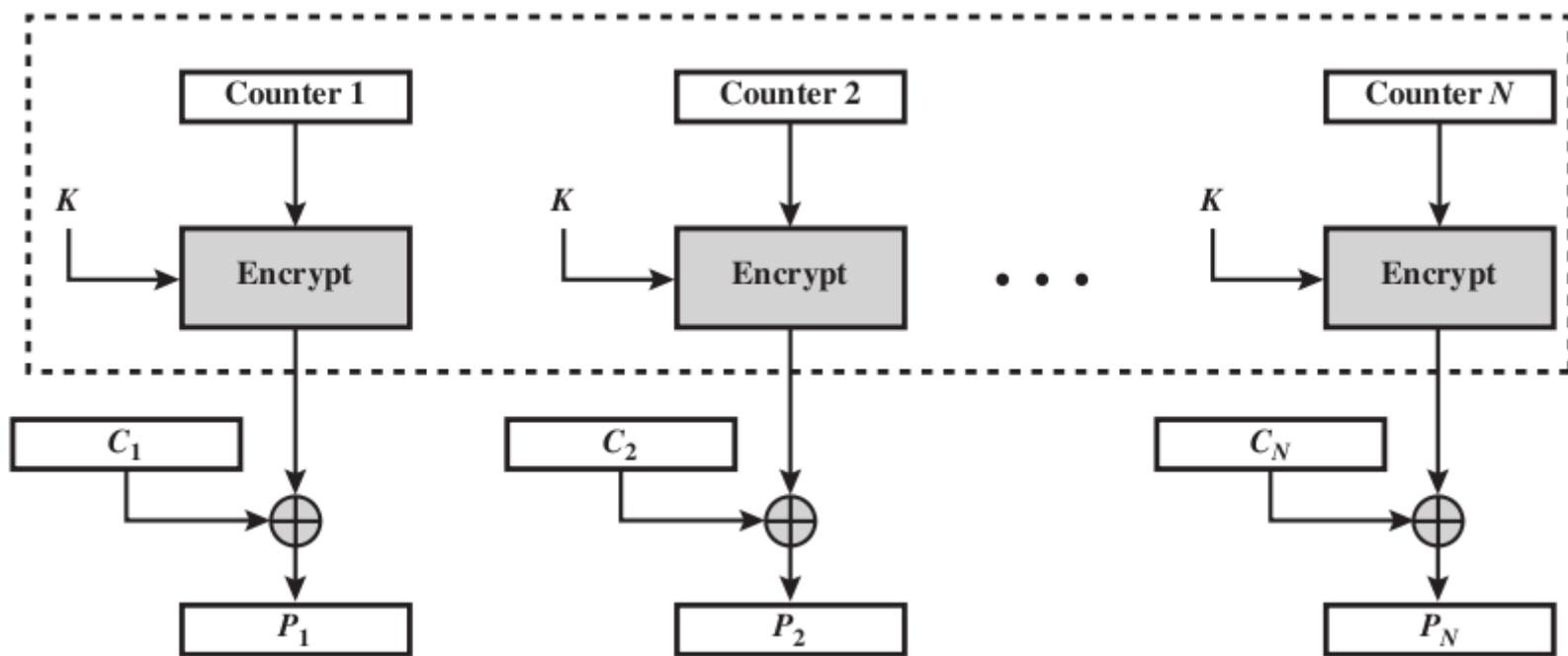
Counter Mode

- ▶ In the counter mode, there is no feedback
- ▶ The randomness in the key streams achieved using a counter
- ▶ The plain text and cipher text block have same block size as the underlying cipher.
- ▶ Both encryption and decryption can be performed fully in parallel on multiple blocks.

CTR Encryption



CTR Decryption



- ▶ The properties of cryptographic algorithms are not only affected by algorithm design, but also by the ways in which the algorithms are used.
- ▶ Different modes of operation can significantly change the properties of a block cipher.

Advanced Encryption Standard

Origin of AES

- ▶ A replacement for DES was needed
 - Key size is too small
- ▶ Can use Triple-DES – but slow, small block

- ▶ It is a symmetric block cipher.
- ▶ Block size = 128 bits.
- ▶ The key length can be 16, 24, or 32 bytes (128, 192, or 256 bits).
- ▶ The algorithm is referred to as AES-128, AES-192, or AES-256, depending on the key length.
- ▶ Number of rounds depends on the key size.
 - ▶ 128 bit key – 10 rounds
 - ▶ 192 bit key – 12 rounds
 - ▶ 256 bit key – 14 rounds

- ▶ The input to the encryption and decryption algorithms is a single 128-bit block.
- ▶ This block is depicted as a $4 * 4$ square matrix of bytes.
- ▶ This block is copied into the **State** array, which is modified at each stage of encryption or decryption.
- ▶ After the final stage, **State** is copied to an output matrix.
- ▶ Similarly, the key is depicted as a square matrix of bytes.
- ▶ This key is then expanded into an array of key schedule words

Algorithm:

- (i) Do the following one-time initialization processes:
 - (a) Expand the 16-byte key to get the actual *Key block* to be used.
 - (b) Do one time initialization of the 16-byte plain text block (called as *State*).
 - (c) XOR the *state* with the *key block*.
- (ii) For each round, do the following:
 - (a) Apply S-box to each of the plain text bytes.
 - (b) Rotate row k of the plain text block (i.e. *state*) by k bytes.
 - (c) Perform a *mix columns* operation.
 - (d) XOR the *state* with the *key block*

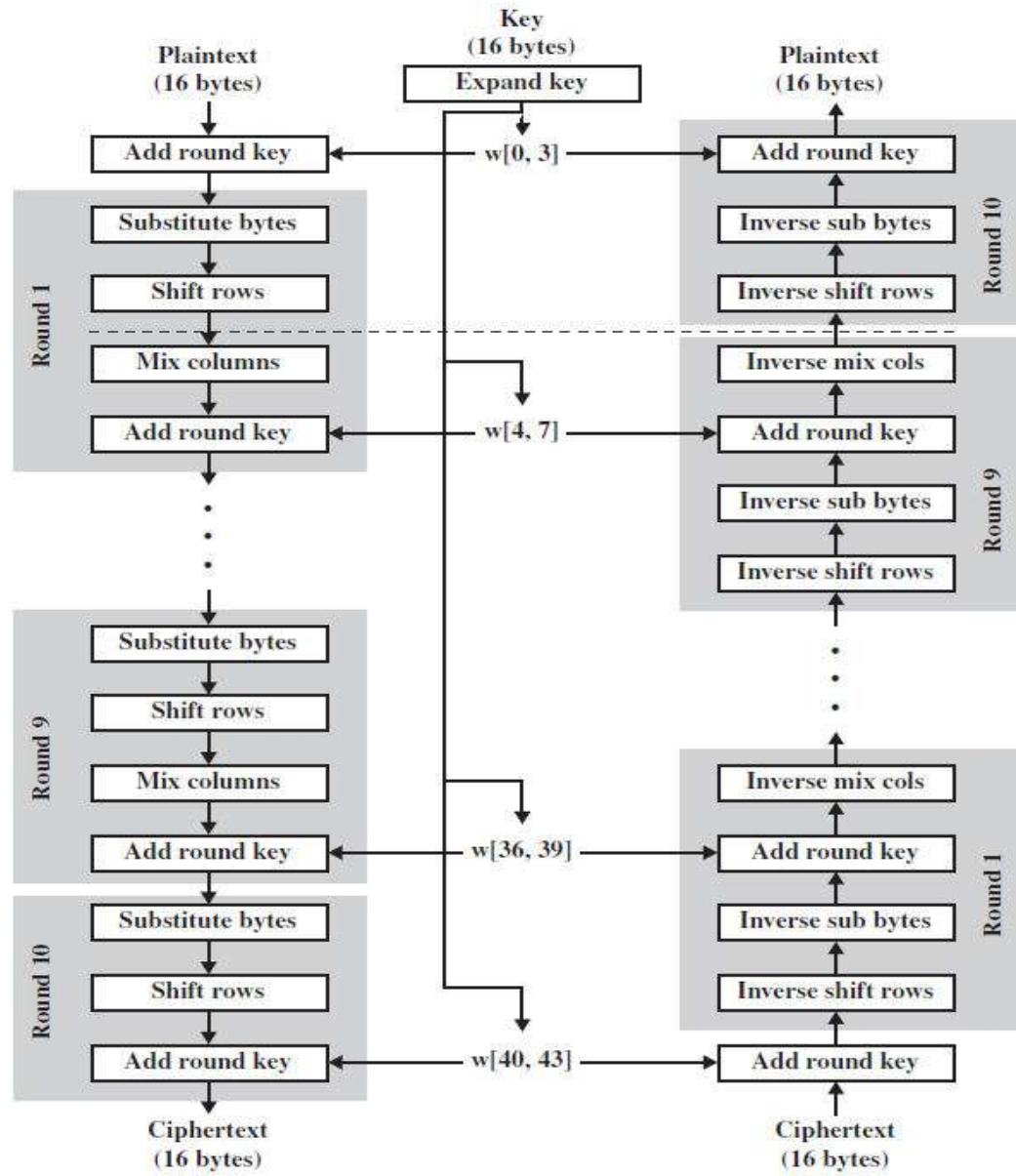


Figure 5.3 AES Encryption and Decryption

AES Structure

- ▶ AES processes the entire data block as a single matrix during each round using substitutions and permutation.
- ▶ The key that is provided as input is expanded into an array of forty four 32 bit words, $w[i]$. Four distinct words (128 bits) serve as a round key for each round.
- ▶ Four different stages are used, one of permutation and three of substitution:
 - Substitute bytes: Uses an S box to perform a byte by byte substitution of the block.

- ShiftRows: A simple permutation
 - MixColumns: A substitution that makes use of arithmetic
 - AddRoundKey: A simple bitwise XOR of the current block with a portion of the expanded key
- The structure is simple. The cipher begins with an AddRoundKey stage followed by 9 rounds that each include all four stages followed by a tenth round of three stages.
- Only AddRoundKey makes use of key.

- ▶ The AddRoundKey is a form of Vernam cipher and may not be powerful. The other three stages provide no security because they do not use the key.
- ▶ Each stage is easily reversible.
- ▶ Decryption makes use of the expanded key in reverse order. However, the decryption algorithm is not identical to the encryption algorithm.
- ▶ Once it is established that all four stages are reversible, it is easy to verify that decryption does recover the plaintext.
- ▶ The final round of both encryption and decryption consists of only three stages.

AES Transformation Function

One-time Initialization Process:

- ***Expand the 16-byte Key to get the Actual Key Block to be Used:***
 - ▶ The inputs to the algorithm are the key and the plain text.
 - ▶ This Step expands this 16-byte key into 11 arrays, each array contains 4 rows and 4 columns.
 - ▶ One of these 11 arrays is used in the initialization process and the other 10 arrays are used in the 10 rounds, one array per round.
 - ▶ A **word** in context of AES means **4 bytes**. Therefore, 16-byte initial key will be expanded into 176-byte key.

- ***Do One Time Initialization of the 16-byte Plain Text Block (Called as State)***
 - ▶ The 16-byte plain text block is copied into a two-dimensional $4 * 4$ array called as state. The order of copying is in the column order.
 - ▶ That is, the first four bytes of the plain text block get copied into the first column of the state array, the next four bytes of the plain text block get copied into the second column of the state array and so on.

- ***XOR the State with the Key block***
 - ▶ The first 16 bytes (i.e. four words $W[0]$, $W[1]$, $W[2]$ and $W[3]$) of the expanded key are XORed into the 16-byte state array (B1 to B16).
 - ▶ Thus, every byte in the state array is replaced by the XOR of itself and the corresponding byte in the expanded key.

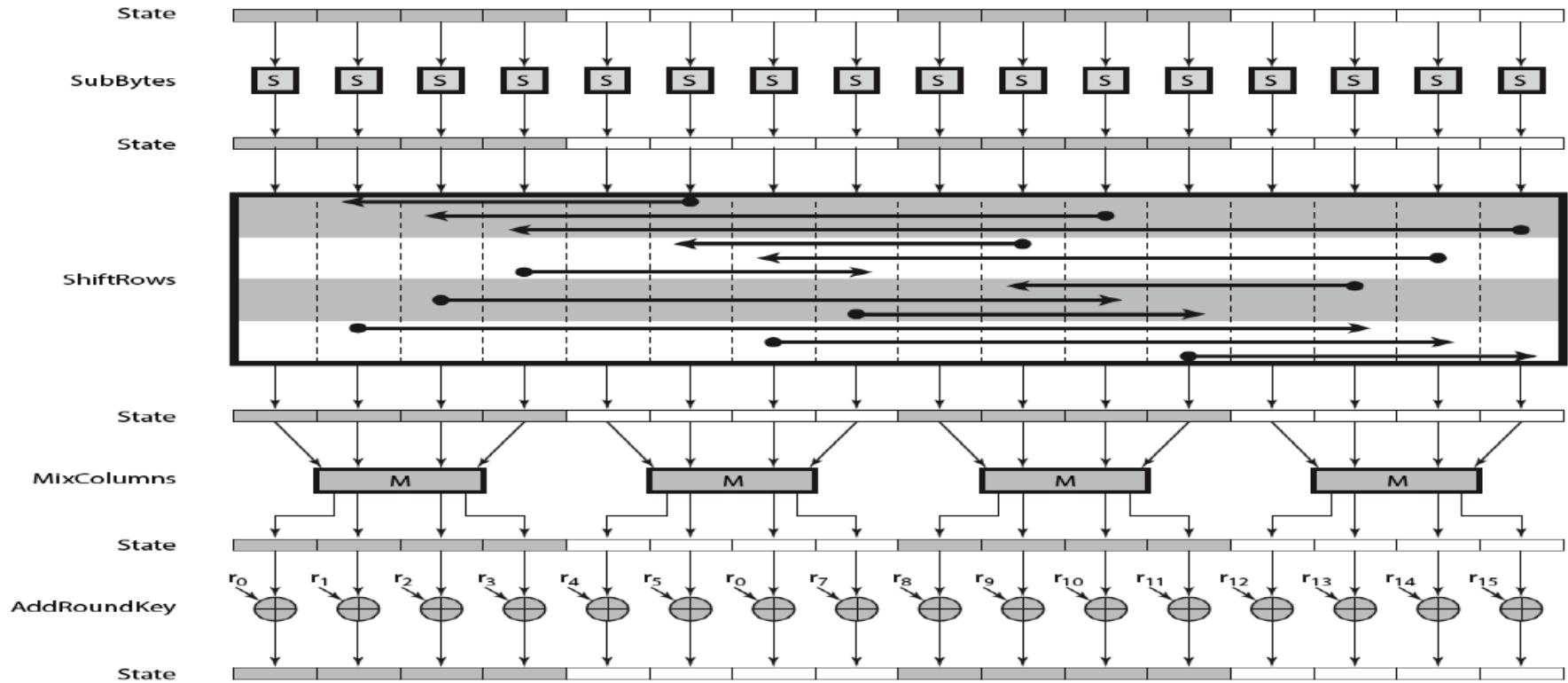
For each round:

- *Apply S-box to Each of the Plain Text Bytes.*
- ▶ Byte by byte substitution is done to replace the contents of the *state* array with the respective entries in the S-box.

- ***Rotate Row k of the Plain Text Block (i.e. state) by k Bytes. (Shift row)***
 - ▶ Here, each of the four rows of the state array are rotated to the left.
 - ▶ Row 0 is rotated 0 bytes (i.e. not rotated at all), row 1 is rotated by 1 byte, row 2 is rotated 2 bytes and row 3 is rotated 3 bytes.
 - ▶ This helps in diffusion of data.

- ***Perform a Mix Columns Operation***
 - ▶ Now, each column is mixed independent of the other.
 - ▶ Matrix multiplication is used. The output of this step is the matrix multiplication of the old values and a constant matrix.

- ***XOR the State with the Key Block.***
 - ▶ This Step XORs the key for this round into the *state* array
- ▶ **Decryption:**
 - ▶ For decryption, the process can be executed in the reverse order.
 - ▶ The same encryption process, run with some different table values, can also perform decryption.



Linear Cryptanalysis vs Differential Cryptanalysis

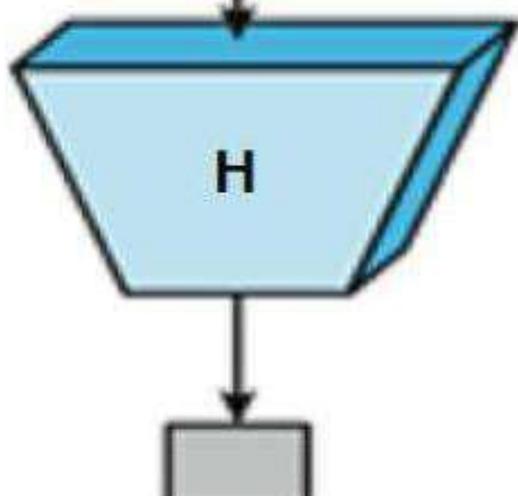
Linear Cryptanalysis	Differential Cryptanalysis
Linear cryptanalysis is a known plaintext attack	Differential cryptanalysis is a chosen plaintext attack
In linear cryptanalysis, the role of cryptanalyst is to identify the linear relation between some bits of the plaintext, some bits of the cipher text and some bits of the unknown key.	Differential cryptanalysis is available obtain clues about some bits of the key, thereby shortening an exhaustive search
The cryptanalyst decrypts each cipher text using all possible sub keys for one round of encryption and studies the resulting intermediate cipher text to analyse the random result.	Cryptanalyst studies changes to the intermediate cipher text obtained between multiple rounds of encryption. The attacks can be combined, which is called differential-linear cryptanalysis.

<p>Linear cryptanalysis focus on statistical analysis against one round of decrypted cipher text</p>	<p>Differential analysis focuses on statistical analysis of two inputs and two outputs of a cryptographic algorithm.</p>
<p>Linear cryptanalysis method can find a DES key given 2^{43} known plaintexts, as compared to 2^{47} chosen plaintexts for differential cryptanalysis.</p>	<p>Differential cryptanalysis is the first published attack that is capable of breaking DES in less than 2^{55} encryptions</p>

Hash Function

- ▶ A hash function H accepts a variable-length block of data M as input and produces a fixed size hash value $h=H(M)$.
- ▶ The kind of hash function needed for security applications is referred to as a cryptographic hash function.

Message M (arbitrary length)



**Hash Value h
(fixed length)**

Applications of Cryptographic Hash Functions:

▶ Message Authentication

- A mechanism or service used to verify the integrity of a message.
- It assures that data received are exactly as sent.
- When a hash function is used to provide message authentication, the hash function value is referred to as a message digest.
- Message authentication is achieved using a Message Authentication Code (MAC) also known as a keyed hash function.

► Digital Signatures

- The operation of the digital signature is similar to that of MAC.
- In the case of the digital signature, the hash value of a message is encrypted with a user's private key.
- Anyone who knows the user's public key can verify the integrity of the message that is associated with the digital signature.
- An attacker who wishes to alter the message would need to know the user's private key.

Secure Hash Algorithm

- ▶ The most widely used hash function has been the Secure Hash Algorithm (SHA).
- ▶ Family of SHA comprise of four SHA algorithms; SHA-0, SHA-1, SHA-2, and SHA-3. Though from same family, there are structurally different.
- ▶ The original version is SHA-0, a 160-bit hash function, was published by the National Institute of Standards and Technology (NIST) in 1993. It had few weaknesses and did not become very popular. Later in 1995, SHA-1 was designed to correct alleged weaknesses of SHA-0.
- ▶ SHA-1 is the most widely used of the existing SHA hash functions. It is employed in several widely used applications and protocols including Secure Socket Layer (SSL) security.

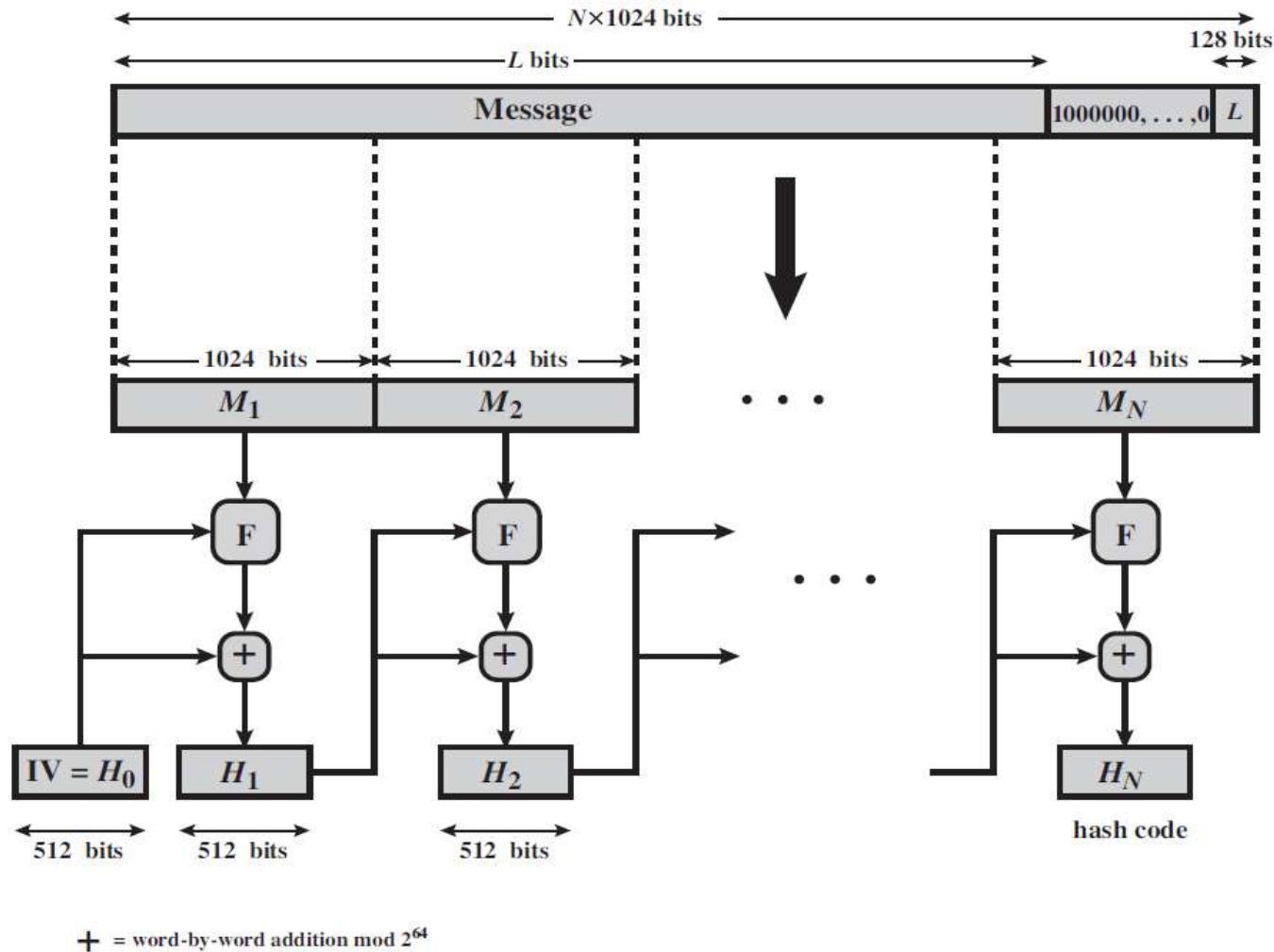
Table 1 Comparison of SHA Parameters

	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
Message Digest Size	160	224	256	384	512
Message Size	$< 2^{64}$	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block Size	512	512	512	1024	1024
Word Size	32	32	32	64	64
Number of Steps	80	64	64	80	80

Note: All sizes are measured in bits.

SHA-512 Logic

- ▶ The algorithm takes as input a message with a maximum length of less than 2^{128} bits and produces as output a 512 bit message digest. The input is processed in 1024 bit blocks.



$+$ = word-by-word addition mod 2^{64}

Figure 11.8 Message Digest Generation Using SHA-512

The processing consists of the following steps:

- 1. Append padding bits.**
- 2. Append length.**
- 3. Initialize hash buffer.**
- 4. Process message in 1024-bit (128-word) blocks.**
- 5. Output.** After all N 1024-bit blocks have been processed, the output from the N th stage is the 512-bit message digest

SHA-512 Processing of a Single 1024-Bit Block

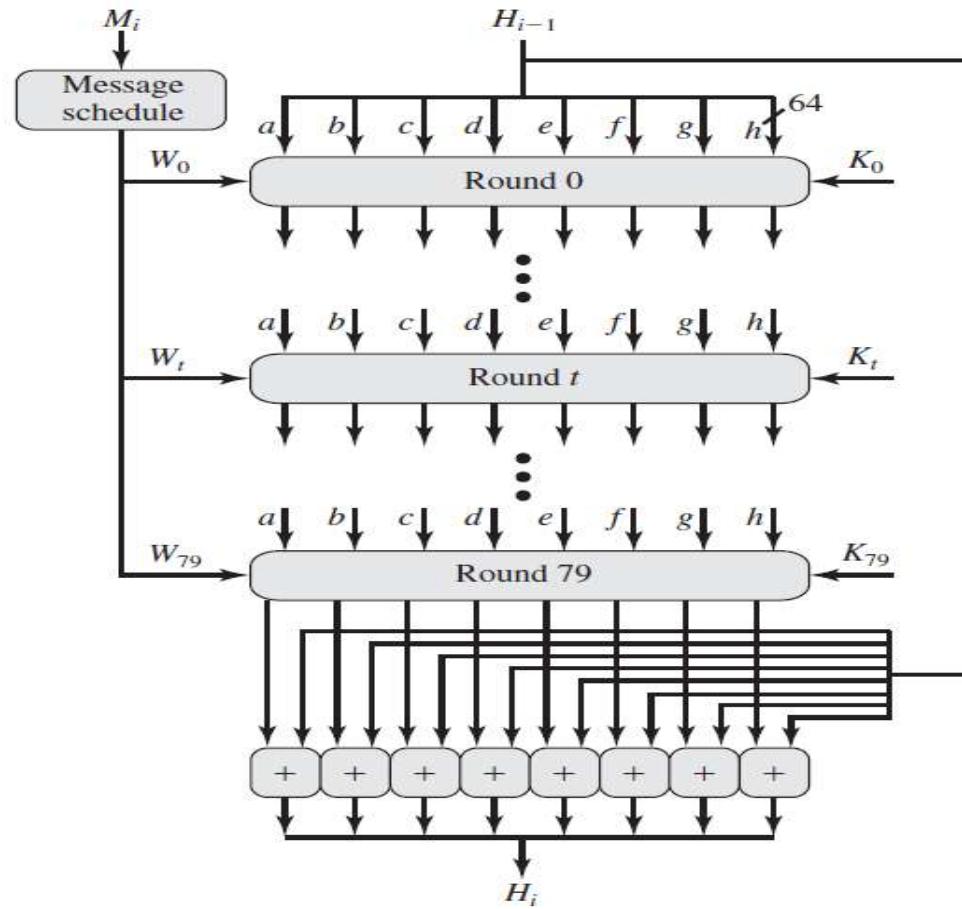
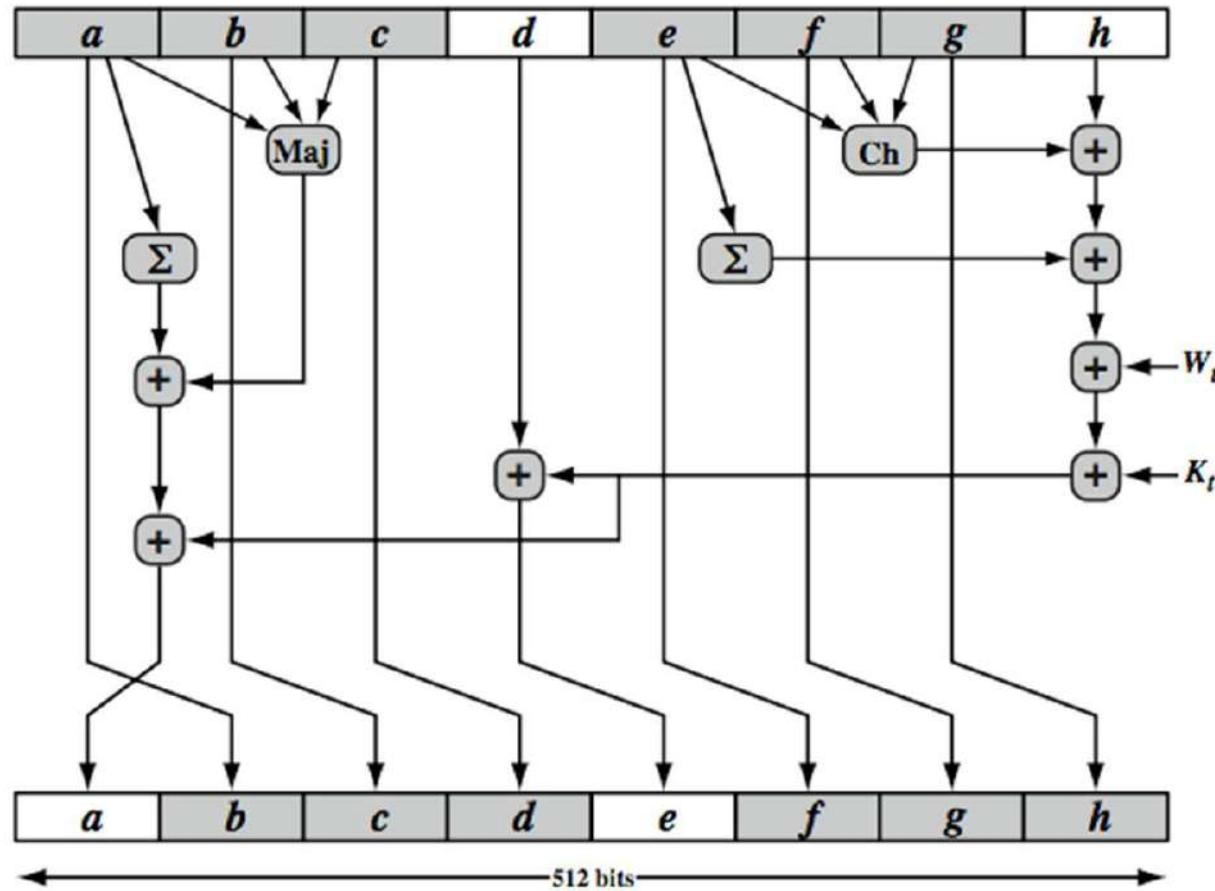


Figure 11.9 SHA-512 Processing of a Single 1024-Bit Block

SHA-512 Compression Function

- ▶ Heart of the algorithm
- ▶ Processing message in 1024-bit blocks
- ▶ Consists of 80 rounds
- ▶ Updating a 512-bit buffer
- ▶ Using a 64-bit value derived from the current message block
- ▶ And a round constant based on cube root of first 80 prime numbers

SHA-512 Operation (single round)



Message Authentication Codes

- ▶ Can also use encryption for secrecy
 - Generally use separate keys for each
 - Can compute MAC either before or after encryption
 - Is generally regarded as better done before
- ▶ Why use a MAC?
 - Sometimes only authentication is needed
 - Sometimes need authentication to persist longer than the encryption (eg. Archival use)
- ▶ Note that a MAC is not a digital signature

Requirements for MACs

- ▶ Taking into account the types of attacks
- ▶ Need the MAC to satisfy the following:
 1. Knowing a message and MAC, is infeasible to find another message with same MAC
 2. MAC should be uniformly distributed
 3. MAC should depend equally on all bits of the message

Why Use MACs?

i.e., why not just use encryption?

- ▶ Clear text stays clear
- ▶ MAC might be cheaper
- ▶ Broadcast
- ▶ Authentication of executable codes
- ▶ Architectural flexibility
- ▶ Separation of authentication check from message use

MACs based on Hash Functions: HMAC

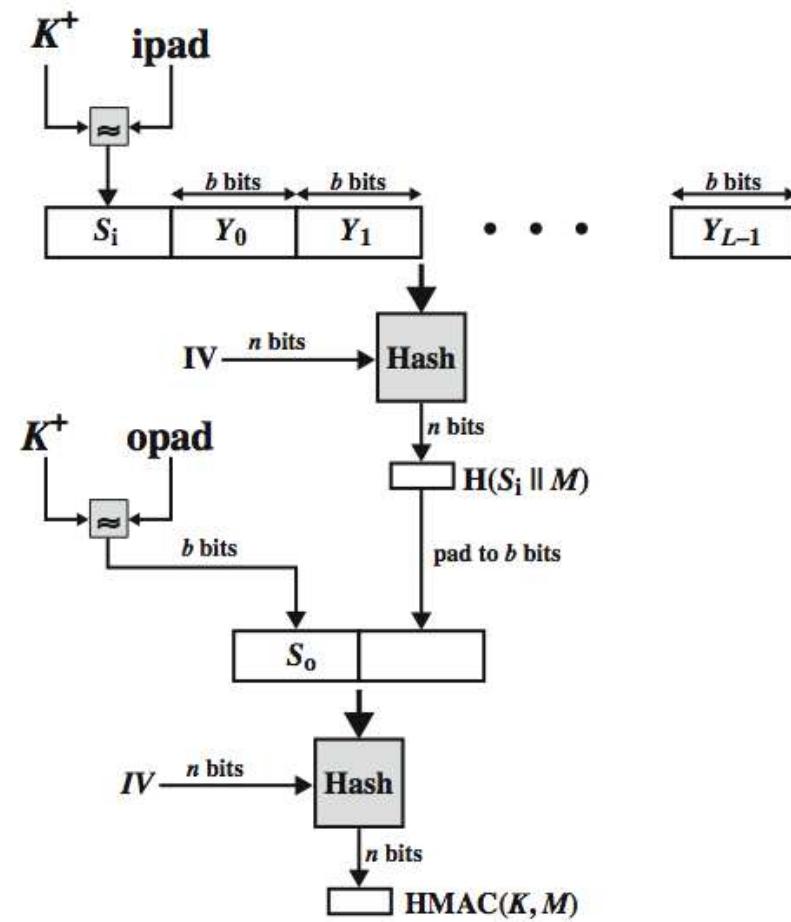
► Design Objectives

- To use, without modifications, available hash functions
- allow for easy replace-ability of embedded hash function
- preserve original performance of hash function without significant degradation
- use and handle keys in a simple way.
- have well understood cryptographic analysis of authentication mechanism strength

Algorithm:

1. Append zeros to the left end of K to create a b-bit string K^+ (eg : if K is of length 160 bits and b=512, then K will be appended with 44 zeroes)
2. XOR K^+ with ipad to produce the b-bit block S_i
3. Append M to S_i
4. Apply H to the stream generated in step 3
5. XOR K^+ with opad to produce the b-bit block S_0 .
6. Append the hash result from step 4 to S_0
7. Apply H to the stream generated in step 6 and output the result.

HMAC Implementation



Public Key Cryptosystems

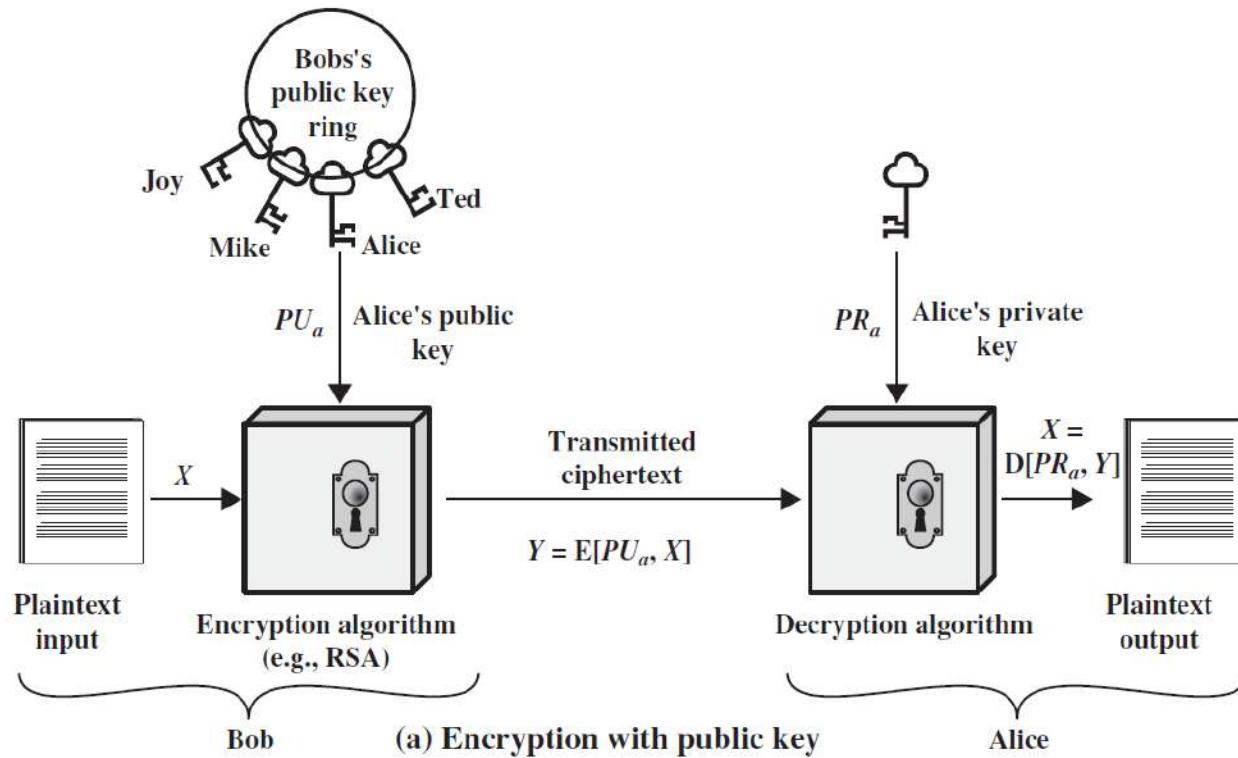
- ▶ Asymmetric algorithms rely on one key for encryption and a different but related key for decryption.
- ▶ These algorithms have the following important characteristic:
- ▶ It is computationally infeasible to determine the decryption key given only knowledge of the cryptographic algorithm and the encryption key.
- ▶ In addition, some algorithms, such as RSA, also exhibit the following characteristic.
- ▶ Either of the two related keys can be used for encryption, with the other used for decryption.

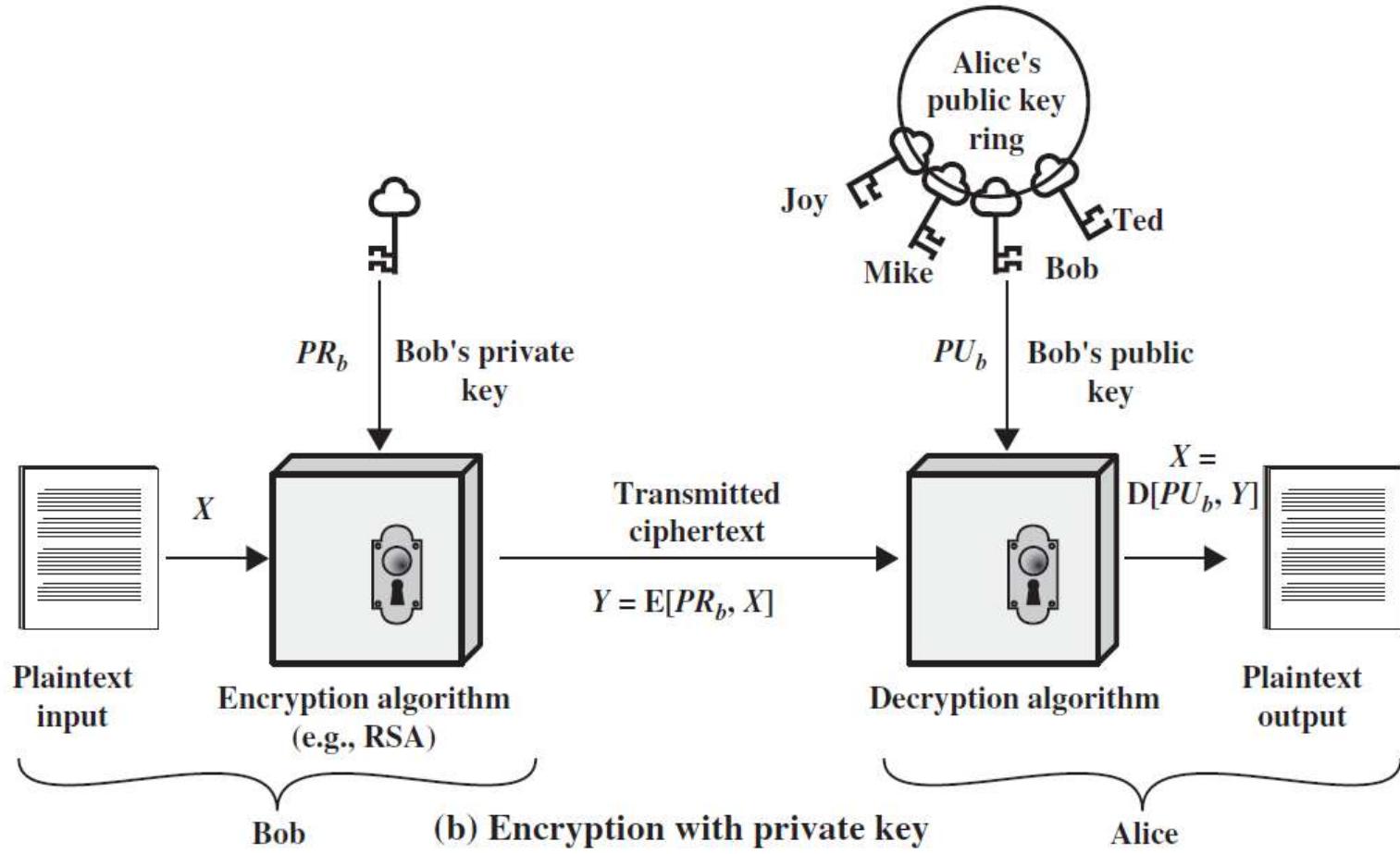
A public-key encryption scheme has six ingredients:

- ▶ **Plaintext:** This is the readable message or data that is fed into the algorithm as input.
- ▶ **Encryption algorithm:** The encryption algorithm performs various transformations on the plaintext.
- ▶ **Public and private keys:** This is a pair of keys that have been selected so that if one is used for encryption, the other is used for decryption. The exact transformations performed by the algorithm depend on the public or private key that is provided as input.

- ▶ **Cipher text:** This is the scrambled message produced as output. It depends on the plaintext and the key. For a given message, two different keys will produce two different cipher texts.
- ▶ **Decryption algorithm:** This algorithm accepts the cipher text and the matching key and produces the original plaintext.

Public-Key Cryptography





- ▶ Each user generates a pair of keys to be used for the encryption and decryption of messages.
- ▶ Each user places one of the two keys in a public register or other accessible file. This is the public key. The companion key is kept private. Figure a. suggests, each user maintains a collection of public keys obtained from others.

- ▶ If Bob wishes to send a confidential message to Alice, Bob encrypts the message using Alice's public key.
- ▶ When Alice receives the message, she decrypts it using her private key. No other recipient can decrypt the message because only Alice knows Alice's private key.

Symmetric Vs Public-Key Encryption

Conventional Encryption	Public-Key Encryption
<p><i>Needed to Work:</i></p> <ol style="list-style-type: none">1. The same algorithm with the same key is used for encryption and decryption.2. The sender and receiver must share the algorithm and the key. <p><i>Needed for Security:</i></p> <ol style="list-style-type: none">1. The key must be kept secret.2. It must be impossible or at least impractical to decipher a message if no other information is available.3. Knowledge of the algorithm plus samples of ciphertext must be insufficient to determine the key.	<p><i>Needed to Work:</i></p> <ol style="list-style-type: none">1. One algorithm is used for encryption and decryption with a pair of keys, one for encryption and one for decryption.2. The sender and receiver must each have one of the matched pair of keys (not the same one). <p><i>Needed for Security:</i></p> <ol style="list-style-type: none">1. One of the two keys must be kept secret.2. It must be impossible or at least impractical to decipher a message if no other information is available.3. Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be insufficient to determine the other key.

- ▶ To discriminate between the symmetric and public key encryption, we refer to the key used in symmetric encryption as a **secret key**.
- ▶ The two keys used for asymmetric encryption are referred to as the **public key** and the **private key**.
- ▶ Invariably, the private key is kept secret, but it is referred to as a private key rather than a secret key to avoid confusion with symmetric encryption.

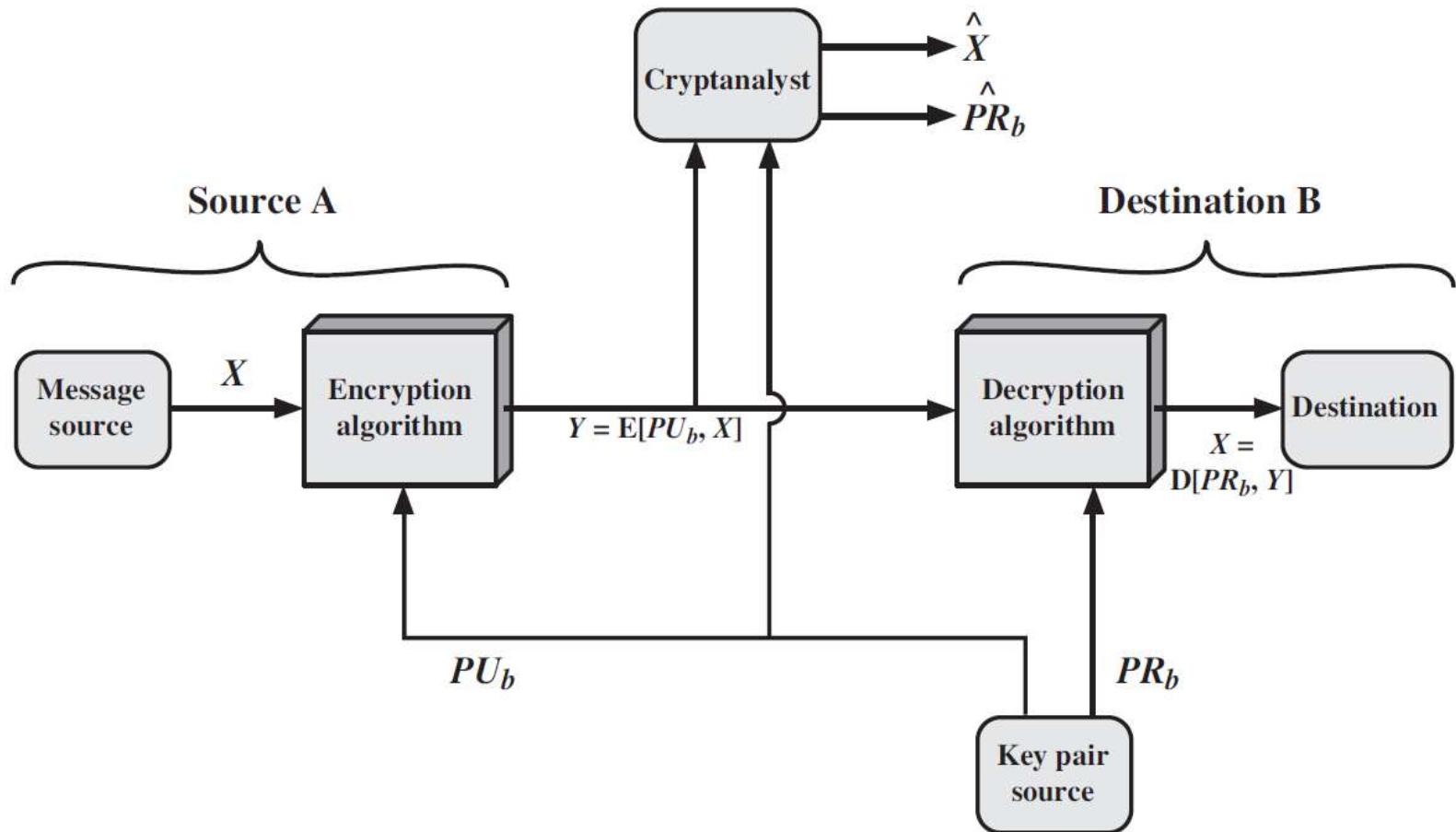
Essential elements of a public-key encryption scheme

- ▶ There is some source A that produces a message in plaintext, $X = [X_1, X_2, \dots, X_M]$.
- ▶ The M elements of X are letters in some finite alphabet.
- ▶ The message is intended for destination B.
- ▶ B generates a related pair of keys: a public key, PU_b , and a private key, PR_b .
- ▶ PR_b is known only to B, whereas PU_b is publicly available and therefore accessible by A.

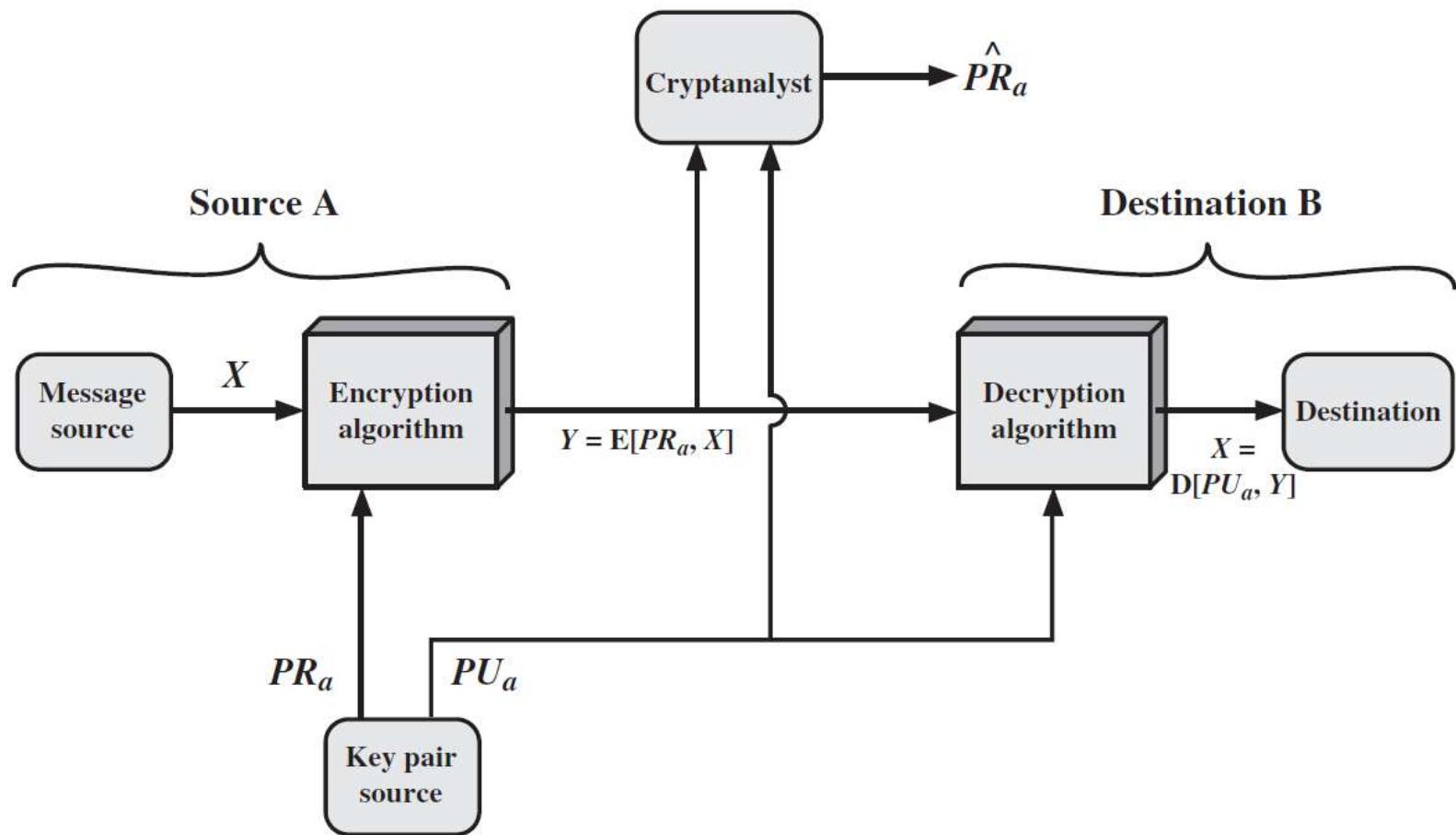
With the message X and the encryption key PU_b as input, A forms the ciphertext $Y = [Y_1, Y_2, \dots, Y_N]$:

$$Y = E(PU_b, X)$$

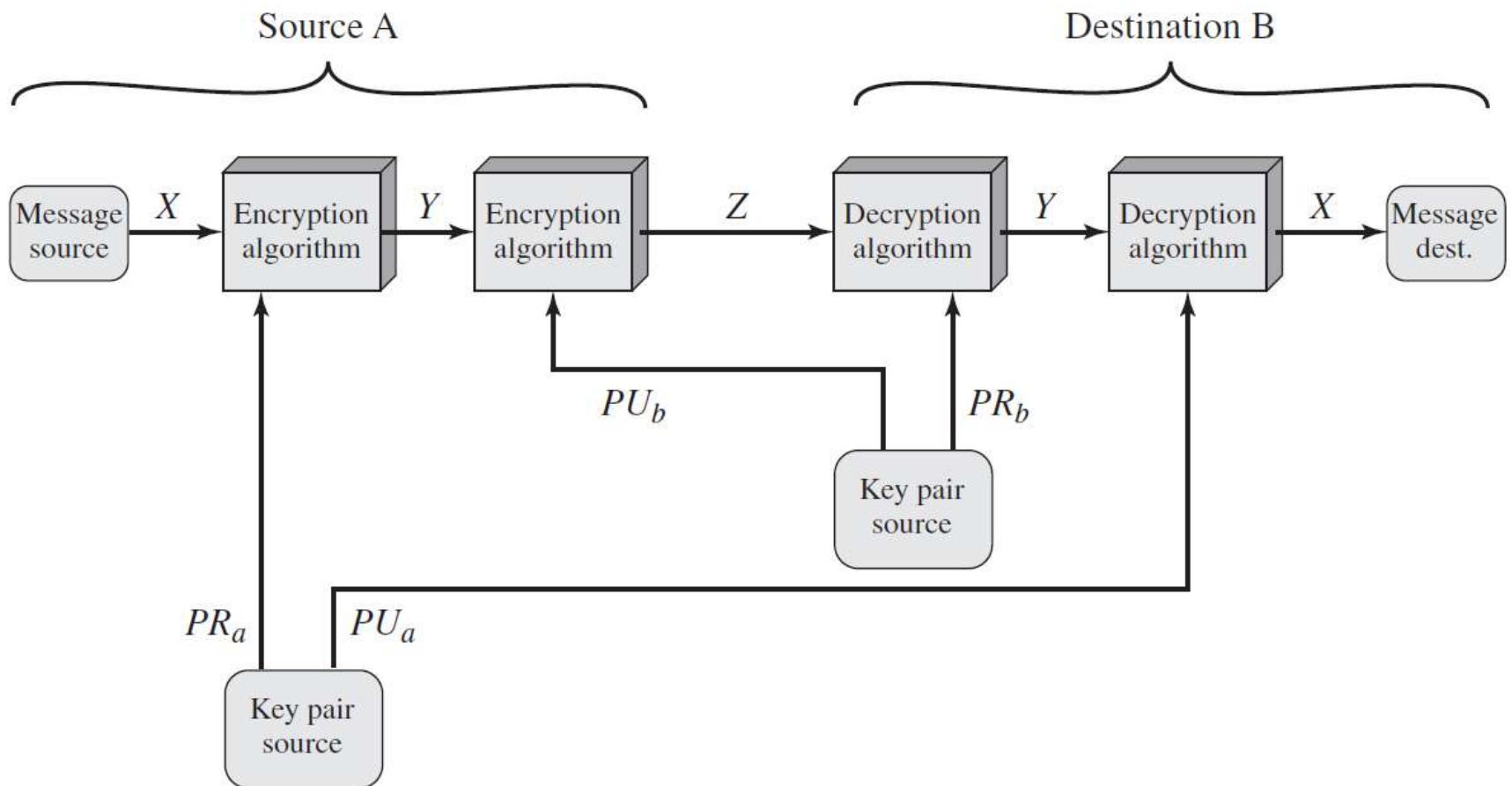
Public-Key Cryptosystem: Secrecy



Public-Key Cryptosystem: Authentication



Public-Key Cryptosystem: Authentication and Secrecy



- ▶ It is, however, possible to provide both the authentication function and confidentiality by a double use of the public-key scheme.

$$Z = E(PU_b, E(PR_a, X))$$

$$X = D(PU_a, D(PR_b, Z))$$

Applications for Public-Key Cryptosystems

Public-key cryptosystems can be classified into three categories:

- **Encryption /decryption:** The sender encrypts a message with the recipient's public key.
- **Digital signature:** The sender “signs” a message with its private key. Signing is achieved by a cryptographic algorithm applied to the message or to a small block of data that is a function of the message.
- **Key exchange:** Two sides cooperate to exchange a session key. Several different approaches are possible, involving the private key(s) of one or both parties.

Applications for Public-Key Cryptosystems

Algorithm	Encryption/Decryption	Digital Signature	Key Exchange
RSA	Yes	Yes	Yes
Elliptic Curve	Yes	Yes	Yes
Diffie-Hellman	No	No	Yes
DSS	No	Yes	No

Requirements for Public-Key Cryptography

- ▶ The cryptosystem depends on a cryptographic algorithm based on two related keys.
- ▶ Conditions that such algorithms must fulfill are as follows:
 1. It is computationally easy for a party B to generate a pair (public key PU_b , private key PR_b).
 2. It is computationally easy for a sender A, knowing the public key and the message to be encrypted, M , to generate the corresponding ciphertext:

$$C = E(PU_b, M)$$

3. It is computationally easy for the receiver B to decrypt the resulting ciphertext using the private key to recover the original message:

$$M = D(PR_b, C) = D[PR_b, E(PU_b, M)]$$

4. It is computationally infeasible for an adversary, knowing the public key, PU_b , to determine the private key, PR_b .
5. It is computationally infeasible for an adversary, knowing the public key, PU_b , and a ciphertext, C , to recover the original message, M .

We can add a sixth requirement that, although useful, is not necessary for all public-key applications:

6. The two keys can be applied in either order:

$$M = D[PU_b, E(PR_b, M)] = D[PR_b, E(PE_b, M)]$$

Number Theory

► Fermat's Theorem:

Fermat's theorem states the following: If p is prime and a is a positive integer not divisible by p , then

$$a^{p-1} \equiv 1 \pmod{p}$$

► Euler's Theorem:

Euler's theorem states that for every a and n that are relatively prime:

$$a^{\phi(n)} \equiv 1 \pmod{n}$$

The RSA Algorithm

- ▶ By Rivest, Shamir & Adleman of MIT in 1977
- ▶ The RSA algorithm is the most popular and proven asymmetric key cryptographic algorithm.
- ▶ Prime numbers, form the basis of the RSA algorithm.

The **RSA** scheme is a block cipher in which the plaintext and ciphertext are integers between 0 and $n - 1$ for some n . A typical size for n is 1024 bits, or 309 decimal digits. That is, n is less than 2^{1024} .

Prime Numbers

- ▶ A prime number is the one that is divisible only by 1 and itself.
- ▶ For instance, 3 is a prime number, because it can be divided only by 1 or 3.
- ▶ However, 4 is not a prime number, because other than by 1 and 4, it can also be divided by 2.
- ▶ Similarly, 5, 7, 11, 13, 17, ... are prime numbers, whereas 6, 8, 9, 10, 12,... are non-prime numbers.
- ▶ A quick observation can also be made that a prime number above 2 must be an odd number (because all even numbers are divisible by 2, therefore, all even numbers from 4 onwards are non-prime).

- ▶ The RSA algorithm is based on the mathematical fact that it is easy to find and multiply large prime numbers together, but it is extremely difficult to factor their product.
- ▶ The private and public keys in RSA are based on very large (made up of 100 or more digits) prime numbers.
- ▶ The algorithm itself is quite simple. However, the real challenge in the case of RSA is the selection and generation of the public and private keys.

Description of the Algorithm

- ▶ RSA makes use of an expression with exponentials.
- ▶ Plaintext is encrypted in blocks, with each block having a binary value less than some number n.
- ▶ Encryption and decryption are of the following form, for some plaintext block M and ciphertext block C.

$$C = M^e \bmod n$$

$$M = C^d \bmod n$$

- ▶ Both sender and receiver must know the value of n .
- ▶ The sender knows the value of e , and only the receiver knows the value of d .
- ▶ Thus, this is a public-key encryption algorithm with a public key of $PU = \{e, n\}$ and a private key of $PR = \{d, n\}$.

RSA Scheme

- ▶ The ingredients are the following:

p, q , two prime numbers

(private, chosen)

$n = pq$

(public, calculated)

e , with $\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$

(public, chosen)

$d \equiv e^{-1} \pmod{\phi(n)}$

(private, calculated)

The private key consists of $\{d, n\}$ and the public key consists of $\{e, n\}$. Suppose that user A has published its public key and that user B wishes to send the message M to A. Then B calculates $C = M^e \pmod{n}$ and transmits C . On receipt of this ciphertext, user A decrypts by calculating $M = C^d \pmod{n}$.

Key Generation:

1. Select two prime numbers, p and q.
2. Calculate $n = p * q$
3. Calculate $\phi(n) = (p - 1)(q - 1)$
4. Select e such that $1 < e < \phi(n)$ and $\text{gcd}(e, \phi(n)) = 1$
(Here e is the public key)
5. Calculate $d = (1 + k\phi(n))/e$

Got by a congruence equation $ed \equiv 1 \pmod{\phi(n)}$

For Encryption consider a message M ,

Calculate cipher $C = M^e \pmod{n}$

For Decryption

Calculate message $M = C^d \pmod{n}$

PROBLEM

In a public-key system using RSA, you intercept the ciphertext $C = 10$ sent to a user whose public key is $e = 5, n = 35$. What is the plaintext M ?

The Security of RSA

Four possible approaches to attacking the RSA algorithm are

- **Brute force:** This involves trying all possible private keys.
- **Mathematical attacks:** There are several approaches, all equivalent in effort to factoring the product of two primes.
- **Timing attacks:** These depend on the running time of the decryption algorithm.
- **Chosen ciphertext attacks:** This type of attack exploits properties of the RSA algorithm.

Attacks on RSA

❖ The Factoring Problem

Three approaches to attacking RSA mathematically are:

1. Factor n into its two prime factors. This enables calculation of $\phi(n) = (p - 1) \times (q - 1)$, which in turn enables determination of $d \equiv e^{-1} \pmod{\phi(n)}$.
2. Determine $\phi(n)$ directly, without first determining p and q . Again, this enables determination of $d \equiv e^{-1} \pmod{\phi(n)}$.
3. Determine d directly, without first determining $\phi(n)$.

Attacks on RSA

❖ **Timing Attacks:**

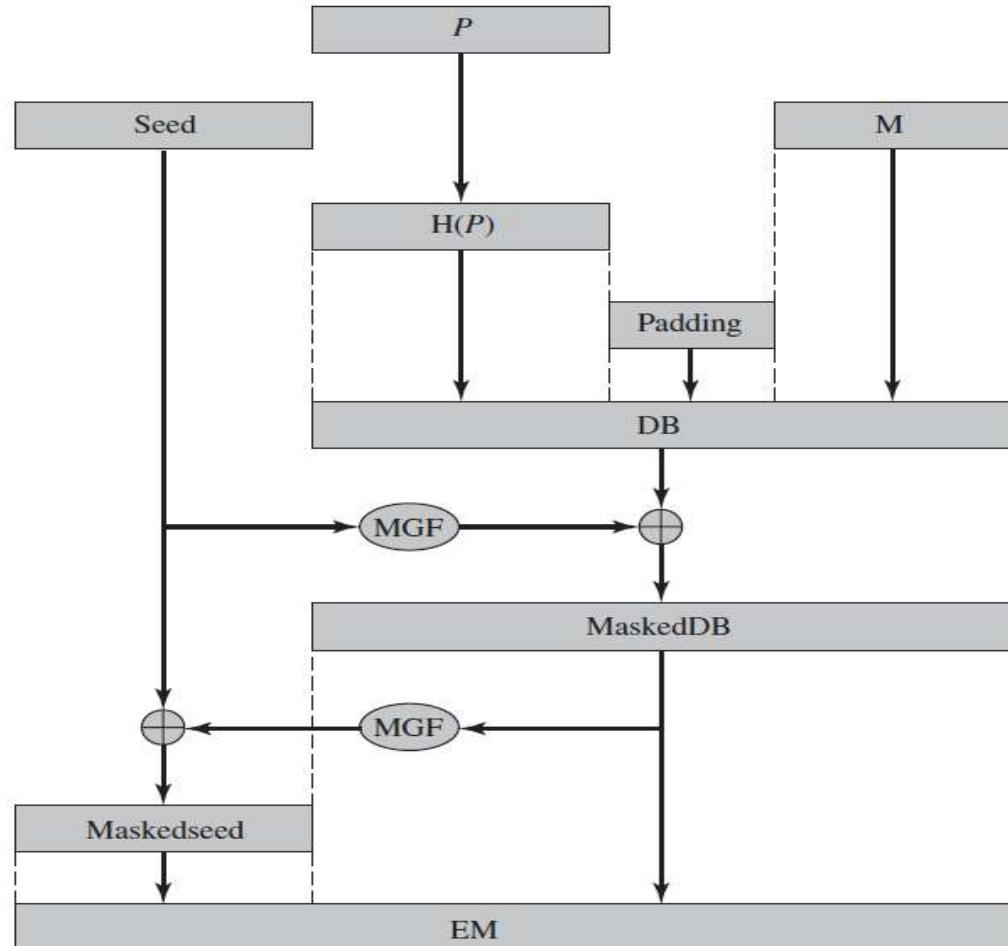
- ▶ A timing attack is a side-channel attack in which the attacker attempts to compromise a cryptosystem by analyzing the time taken to execute cryptographic algorithms.
- ▶ Every logical operation in a computer takes time to execute, and the time can differ based on the input; with precise measurements of the time for each operation, an attacker can work backwards to the input.
- ▶ A timing attack is somewhat analogous to a burglar guessing the combination of a safe by observing how long it takes for someone to turn the dial from number to number.

Although the timing attack is a serious threat, there are simple countermeasures that can be used, including the following.

- ▶ **Constant exponentiation time:** Ensure that all exponentiations take the same amount of time before returning a result. This is a simple fix but does degrade performance.
- ▶ **Random delay:** Better performance could be achieved by adding a random delay to the exponentiation algorithm to confuse the timing attack. Kocher points out that if defenders don't add enough noise, attackers could still succeed by collecting additional measurements to compensate for the random delays.
- ▶ **Blinding:** Multiply the cipher text by a random number before performing exponentiation. This process prevents the attacker from knowing what cipher text bits are being processed inside the computer and therefore prevents the bit-by-bit analysis essential to the timing attack.

- ❖ **Chosen Ciphertext Attack & Optimal Asymmetric Encryption Padding**
 - ▶ The basic RSA algorithm is vulnerable to a chosen ciphertext attack (CCA).
 - ▶ CCA is defined as an attack in which the adversary chooses a number of cipher texts and is then given the corresponding plaintexts, decrypted with the target's private key.
 - ▶ Thus, the adversary could select a plaintext, encrypt it with the target's public key, and then be able to get the plaintext back by having it decrypted with the private key.
 - ▶ This provides the adversary with no new information.
 - ▶ Instead, the adversary exploits properties of RSA and selects blocks of data that, when processed using the target's private key, yield information needed for cryptanalysis.

Encryption Using Optimal Asymmetric Encryption Padding (OAEP)



P = encoding parameters
 M = message to be encoded
 H = hash function

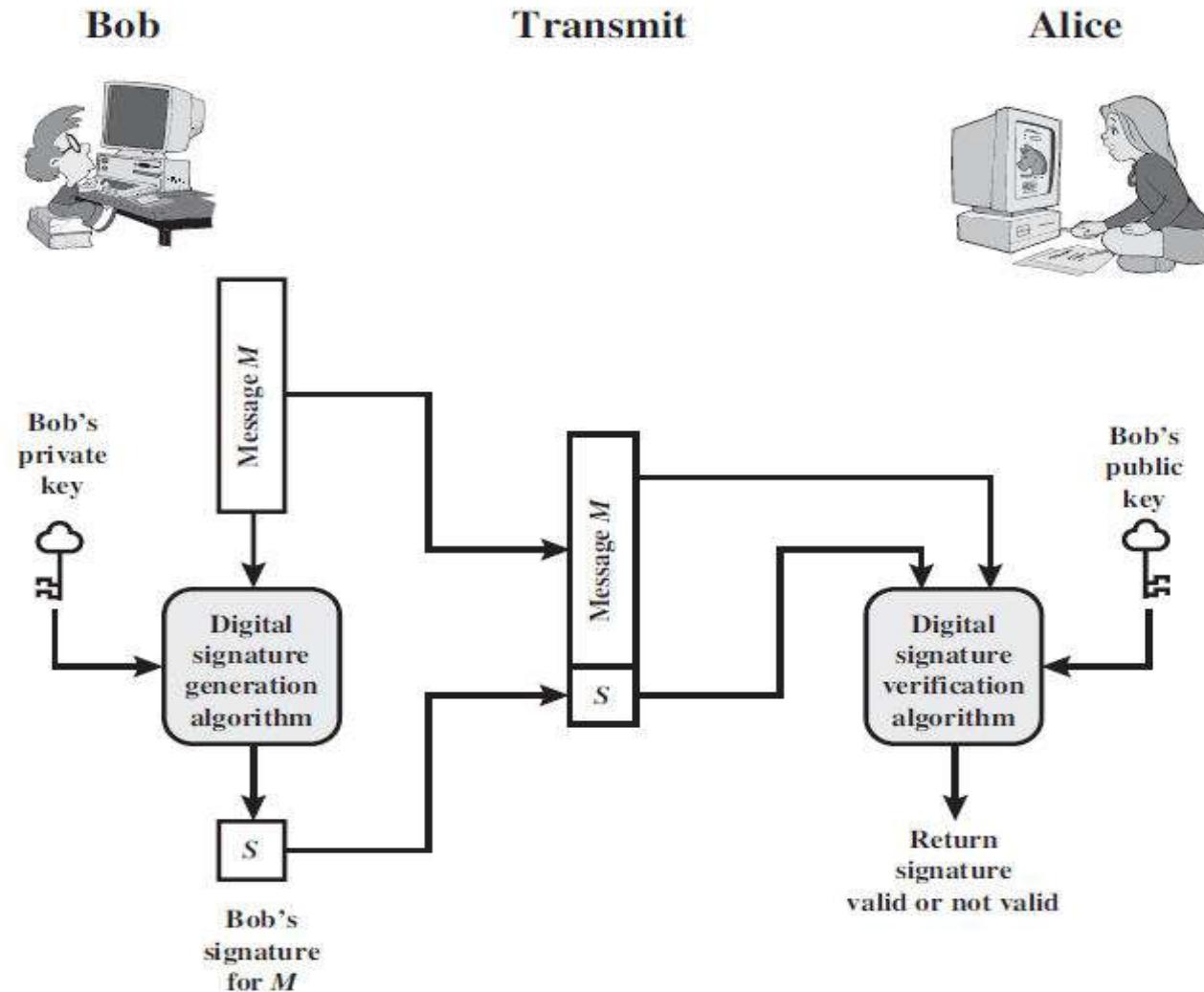
DB = data block
MGF = mask generating function
 EM = encoded message

- ▶ The message M to be encrypted is padded.
- ▶ A set of optional parameters P is passed through a hash function H.
- ▶ Output is then padded with zeros to get the desired length in the data block DB.
- ▶ A random seed is generated and passed through the hash function called Mask Generating Function(MGF)
- ▶ The resulting hash function is XORed bit by bit with DB to get maskedDB
- ▶ The maskedDB is passed through MGF to form a hash that is XORed with seed to produce the maskedseed.
- ▶ The concatenation of the maskedseed and the maskedDB forms the encoded message EM.
- ▶ The EM is then encrypted using RSA

Digital Signature

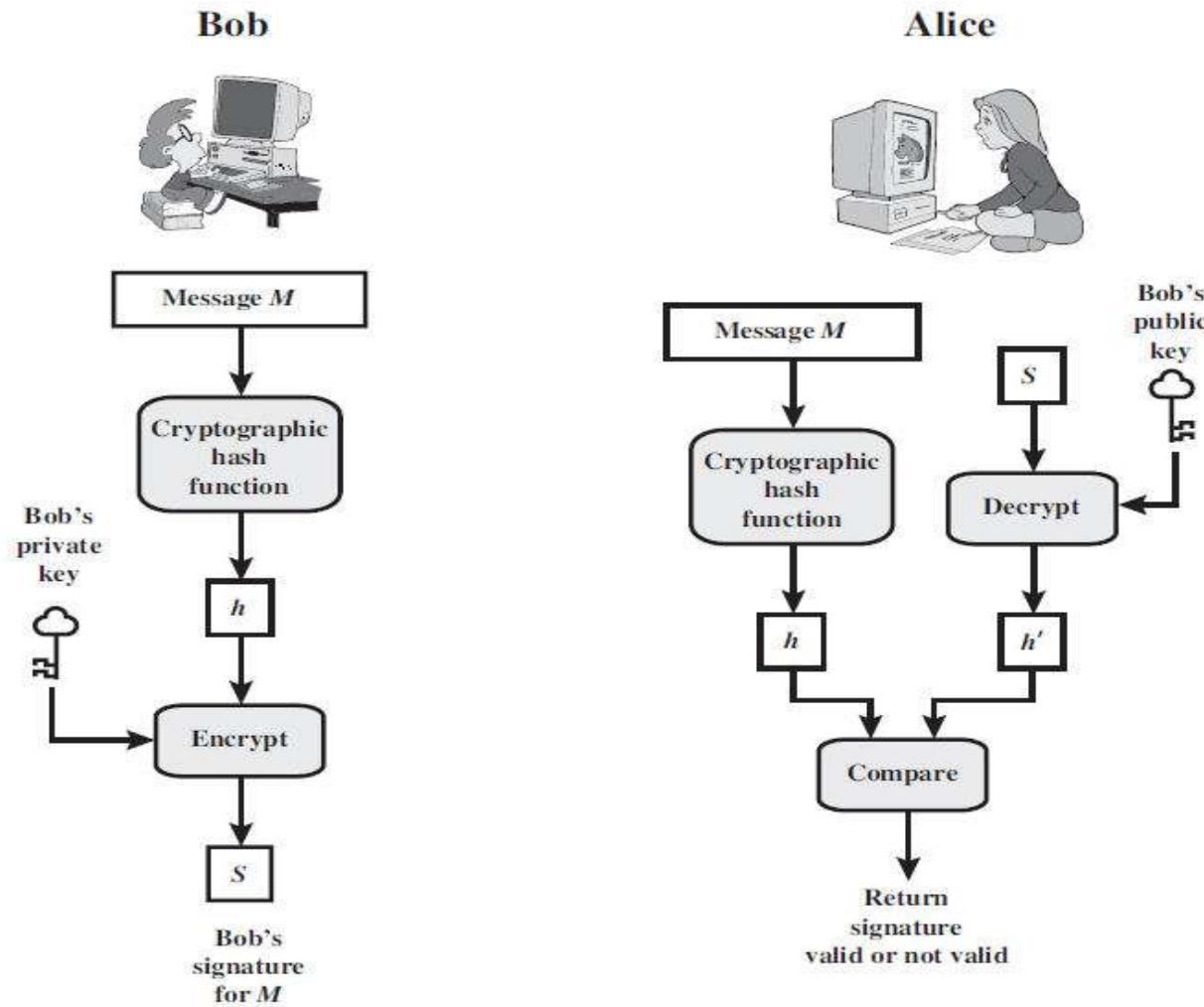
- ▶ The Digital Signature is a technique which is used to validate the authenticity and integrity of the message.
- ▶ A digital signature is an authentication mechanism that enables the creator of a message to attach a code that acts as a signature.
- ▶ Typically the signature is formed by taking the hash of the message and encrypting the message with the creator's private key.
- ▶ The signature guarantees the source and integrity of the message.

Generic Model of Digital Signature Process



- ▶ Bob can send a message using a digital signature generation algorithm.
- ▶ The inputs to the algorithm are the message and Bob's private key.
- ▶ Any other user, say Alice, can verify the signature using a verification algorithm, whose inputs are the message, the signature, and Bob's public key.

Simplified Depiction of Essential Elements of Digital Signature Process



- ▶ In situations where there is no complete trust between sender and receiver, something more than authentication is needed. The most attractive solution to this problem is the digital signature.
- ▶ The digital signature must have the following properties:
 1. It must verify the author and the date and time of the signature.
 2. It must authenticate the contents at the time of the signature.
 3. It must be verifiable by third parties, to resolve disputes.

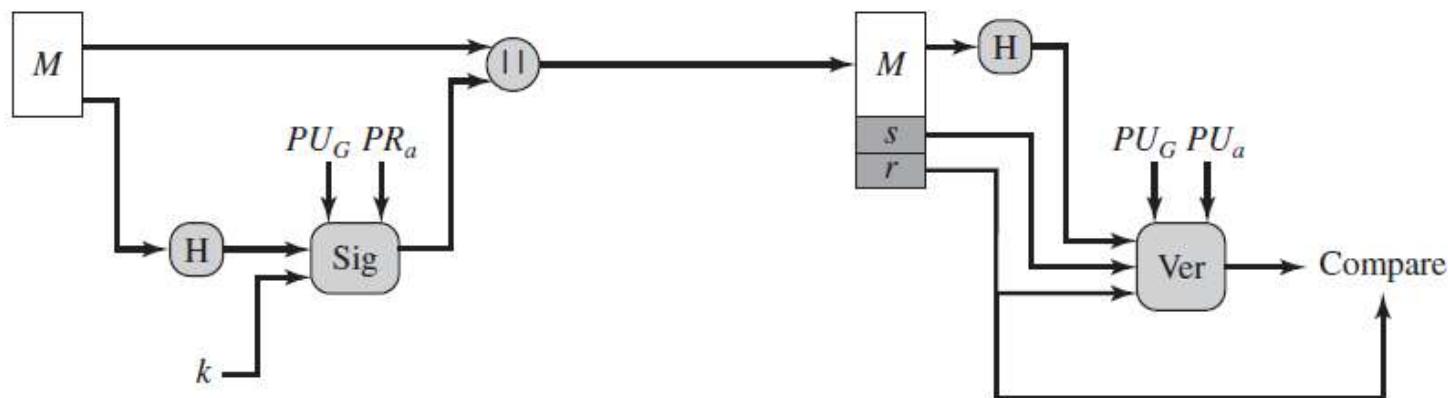
Digital Signature Requirements

1. The signature must be a bit pattern that depends on the message being signed.
2. The signature must use some information unique to the sender to prevent both forgery and denial.
3. It must be relatively easy to produce the digital signature.
4. It must be relatively easy to recognize and verify the digital signature.
5. It must be computationally infeasible to forge a digital signature, either by constructing a new message for an existing digital signature or by constructing a fraudulent digital signature for a given message.
6. It must be practical to retain a copy of the digital signature in storage

Two Approaches to Digital Signatures



(a) RSA approach



(b) DSS approach

Digital Signature Algorithm

Global Public-Key Components

- p prime number where $2^{L-1} < p < 2^L$ for $512 \leq L \leq 1024$ and L a multiple of 64; i.e., bit length of between 512 and 1024 bits in increments of 64 bits
- q prime divisor of $(p - 1)$, where $2^{159} < q < 2^{160}$; i.e., bit length of 160 bits
- g = $h^{(p-1)/q} \pmod{p}$,
where h is any integer with $1 < h < (p - 1)$ such that $h^{(p-1)/q} \pmod{p} > 1$

User's Private Key

- x random or pseudorandom integer with $0 < x < q$

User's Public Key

$$y = g^x \pmod{p}$$

User's Per-Message Secret Number

- k random or pseudorandom integer with $0 < k < q$

Signing

$$\begin{aligned}r &= (g^k \pmod{p}) \pmod{q} \\s &= [k^{-1} (\text{H}(M) + xr)] \pmod{q} \\\text{Signature} &= (r, s)\end{aligned}$$

Verifying

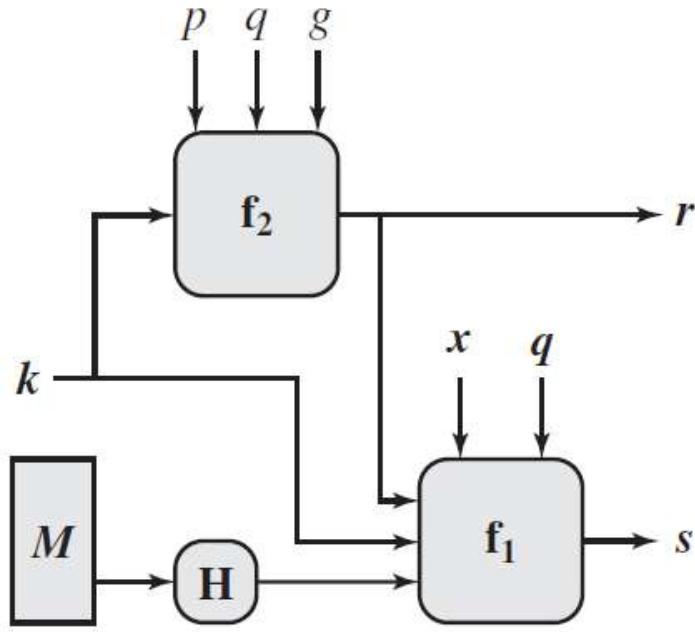
$$\begin{aligned}w &= (s')^{-1} \pmod{q} \\u_1 &= [\text{H}(M')w] \pmod{q} \\u_2 &= (r')w \pmod{q} \\v &= [(g^{u_1} y^{u_2}) \pmod{p}] \pmod{q} \\&\text{TEST: } v = r'\end{aligned}$$

M = message to be signed

$\text{H}(M)$ = hash of M using SHA-1

M', r', s' = received versions of M, r, s

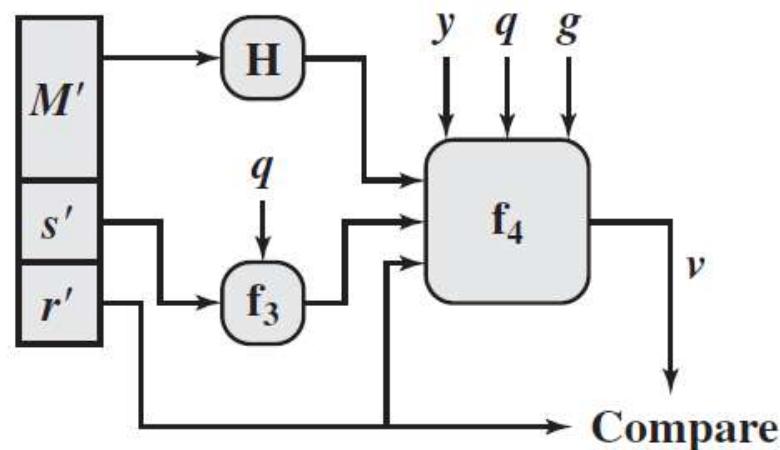
DSS Signing and Verifying



$$s = f_1(H(M), k, x, r, q) = (k^{-1} (H(M) + xr)) \bmod q$$

$$r = f_2(k, p, q, g) = (g^k \bmod p) \bmod q$$

(a) Signing



$$w = f_3(s', q) = (s')^{-1} \bmod q$$

$$v = f_4(y, q, g, H(M'), w, r')$$

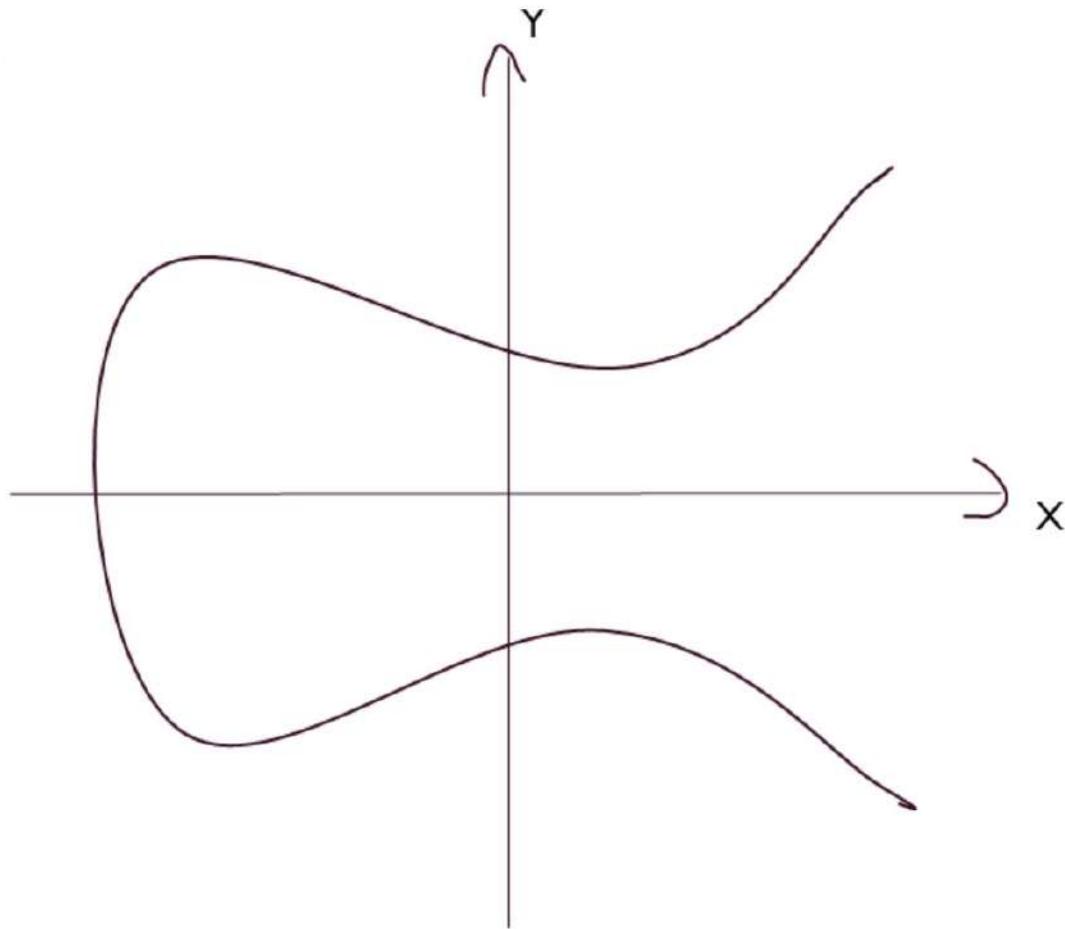
$$= ((g^{(H(M'))w} \bmod q)^{y'^{-1}w} \bmod q) \bmod p \bmod q$$

(b) Verifying

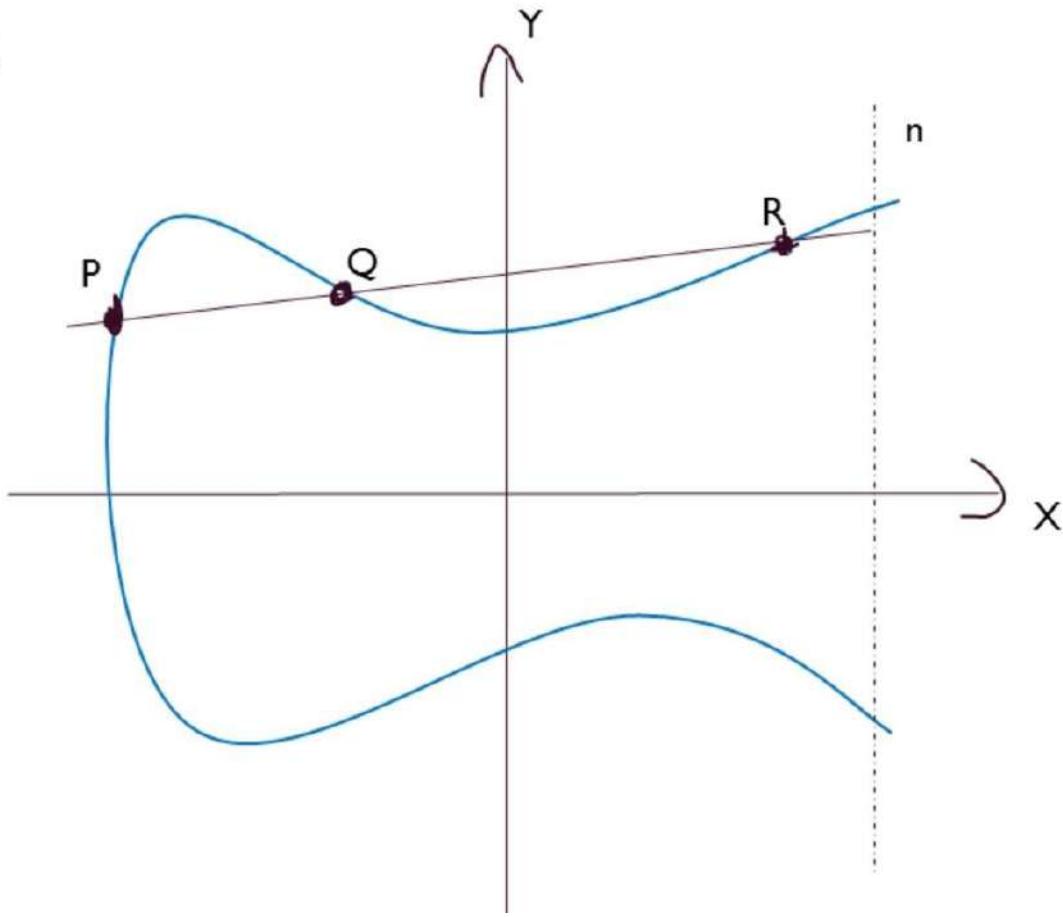
Elliptic Curves Cryptography

- ▶ It is an asymmetric/public key cryptosystem
- ▶ Provides equal security with smaller key size
- ▶ ECC makes use of elliptic curves
- ▶ They are defined by mathematical functions – cubic functions
- ▶ $y^2=x^3+ax+b$
- ▶ Elliptic curves are used as an extension to other current cryptosystems.
- ▶ Elliptic Curve Diffie-Hellman Key Exchange
- ▶ Elliptic Curve Digital Signature Algorithm

An Elliptic curve :



An Elliptic curve :



Elliptic Curve Cryptography

- ▶ Let $E_p(a,b)$ be the elliptic curve
- ▶ Consider equation, $Q=kP$
where $Q,P \in E_p(a,b)$ and $k < n$
- ▶ It should be easy to find Q given k and P
- ▶ But should be extremely difficult to find k given Q and P
- ▶ It is a one way function – trap door function
- ▶ Called the discrete logarithm problem

ECC Diffie-Hellman Key Exchange

Global Public Elements

$E_q(a, b)$ elliptic curve with parameters a, b , and q , where q is a prime or an integer of the form 2^m

G point on elliptic curve whose order is large value n

User A Key Generation

Select private n_A $n_A < n$

Calculate public P_A $P_A = n_A \times G$

User B Key Generation

Select private n_B $n_B < n$

Calculate public P_B $P_B = n_B \times G$

Calculation of Secret Key by User A

$$K = n_A \times P_B$$

Calculation of Secret Key by User B

$$K = n_B \times P_A$$

ECC Encryption

- ▶ Let the message be M
- ▶ First encode the message M into a point on the elliptic curve
- ▶ Let this point be P_m
- ▶ Now this point is encrypted
- ▶ For encrypting choose a random positive integer k
- ▶ Then $C_m = \{kG + P_m + kP_b\}$ where G is the base point

ECC- Encryption & Decryption

To encrypt and send a message P_m to B, A chooses a random positive integer k and produces the ciphertext C_m consisting of the pair of points:

$$C_m = \{kG, P_m + kP_B\}$$

Note that A has used B's public key P_B . To decrypt the ciphertext, B multiplies the first point in the pair by B's secret key and subtracts the result from the second point:

$$P_m + kP_B - n_B(kG) = P_m + k(n_B G) - n_B(kG) = P_m$$

Key Management

- ▶ In cryptography it is a very tedious task to distribute the public and private key between sender and receiver.
- ▶ If key is known to the third party (forger/eavesdropper) then the whole security mechanism becomes worthless.
- ▶ So, there comes the need to secure the exchange of keys.

Symmetric Key Distribution Using Symmetric Encryption

- ▶ For symmetric encryption to work, the two parties to an exchange must share the same key, and that key must be protected from access by others.
- ▶ Frequent key changes are usually desirable to limit the amount of data compromised if an attacker learns the key.
- ▶ Therefore, the strength of any cryptographic system rests with the key distribution technique, a term that refers to the means of delivering a key to two parties who wish to exchange data without allowing others to see the key.

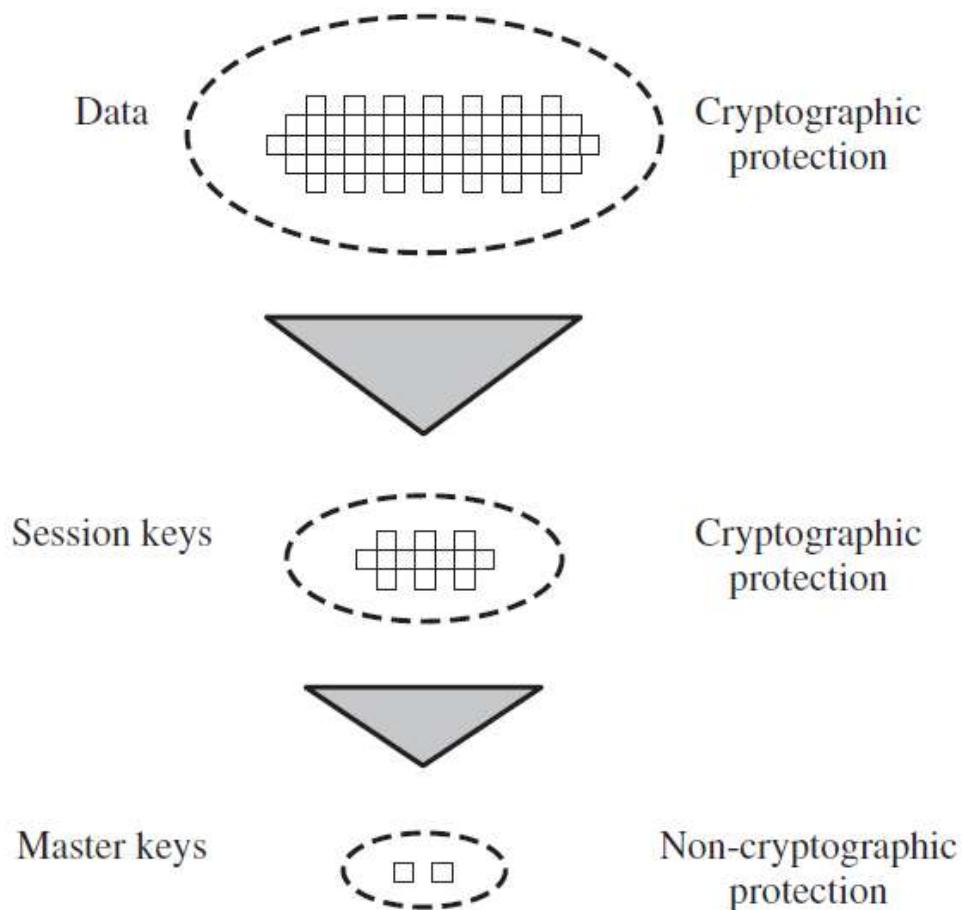
For two parties A and B, key distribution can be achieved in a number of ways, as follows:

1. A can select a key and physically deliver it to B.
2. A third party can select the key and physically deliver it to A and B.
3. If A and B have previously and recently used a key, one party can transmit the new key to the other, encrypted using the old key.
4. If A and B each has an encrypted connection to a third party C, C can deliver a key on the encrypted links to A and B.

- ▶ **Options 1 and 2** call for manual delivery of a key.
- ▶ For link encryption, this is a reasonable requirement, because each link encryption device is going to be exchanging data only with its partner on the other end of the link.
- ▶ However, for end-to-end encryption over a network, manual delivery is awkward.
- ▶ In a distributed system, any given host or terminal may need to engage in exchanges with many other hosts and terminals over time.
- ▶ Thus, each device needs a number of keys supplied dynamically.
- ▶ The problem is especially difficult in a wide-area distributed system.

- ▶ **Option 3** is a possibility for either link encryption or end-to-end encryption, but if an attacker ever succeeds in gaining access to one key, then all subsequent keys will be revealed.
- ▶ For end-to-end encryption, some variation on **Option 4** has been widely adopted. In this scheme, a key distribution center is responsible for distributing keys to pairs of users (hosts, processes, applications) as needed.
- ▶ Each user must share a unique key with the key distribution center for purposes of key distribution.
- ▶ The use of a key distribution center is based on the use of a hierarchy of keys.

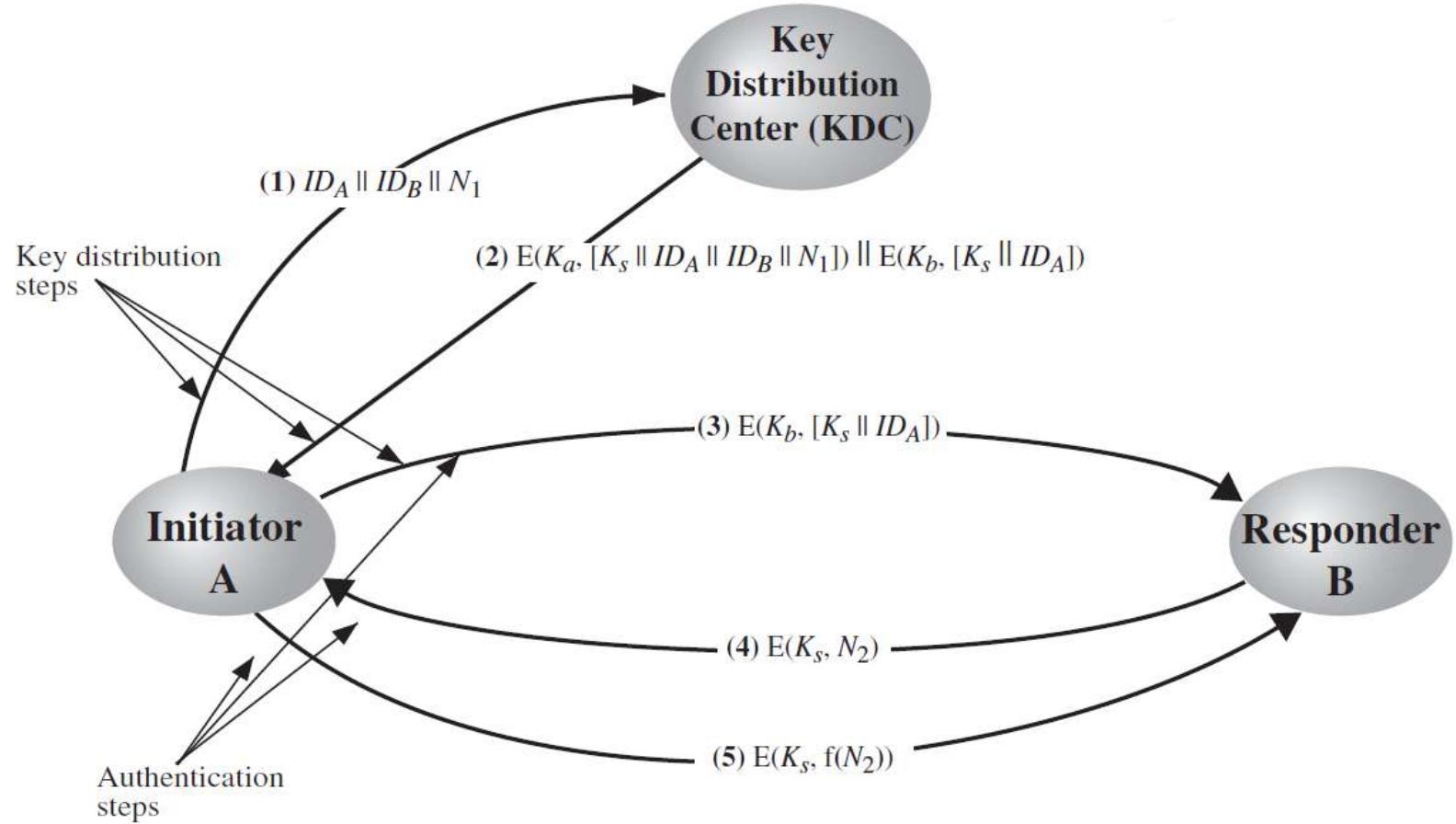
The Use of a Key Hierarchy



- ❖ **Session key**
 - ▶ Temporary key
 - ▶ Used for encryption of data between users
 - ▶ Used for one logical session then discarded

- ❖ **Master key**
 - ▶ Used to encrypt session keys
 - ▶ Shared by user & key distribution center

A Key Distribution Scenario



Symmetric Key Distribution Using Asymmetric Encryption

- Public key cryptosystems are inefficient
 - ▶ so almost never use for direct data encryption
 - ▶ rather use to encrypt secret keys for distribution

Simple Secret Key Distribution

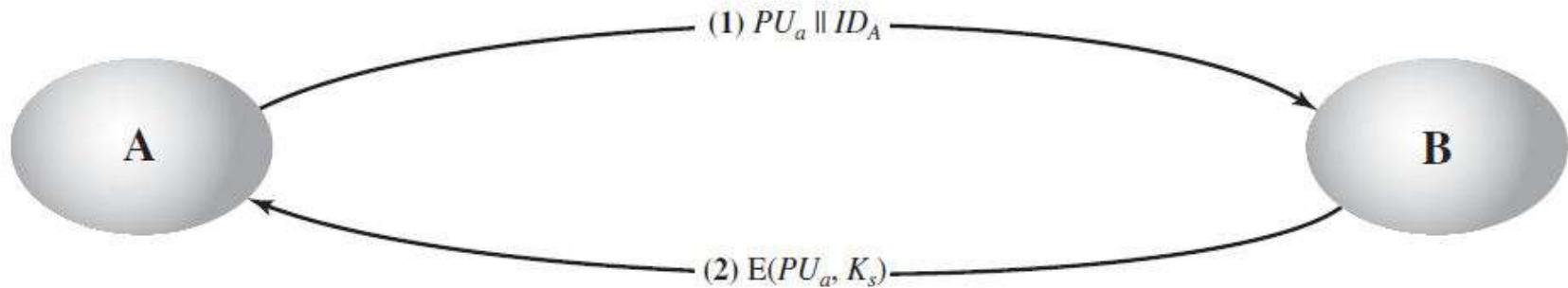
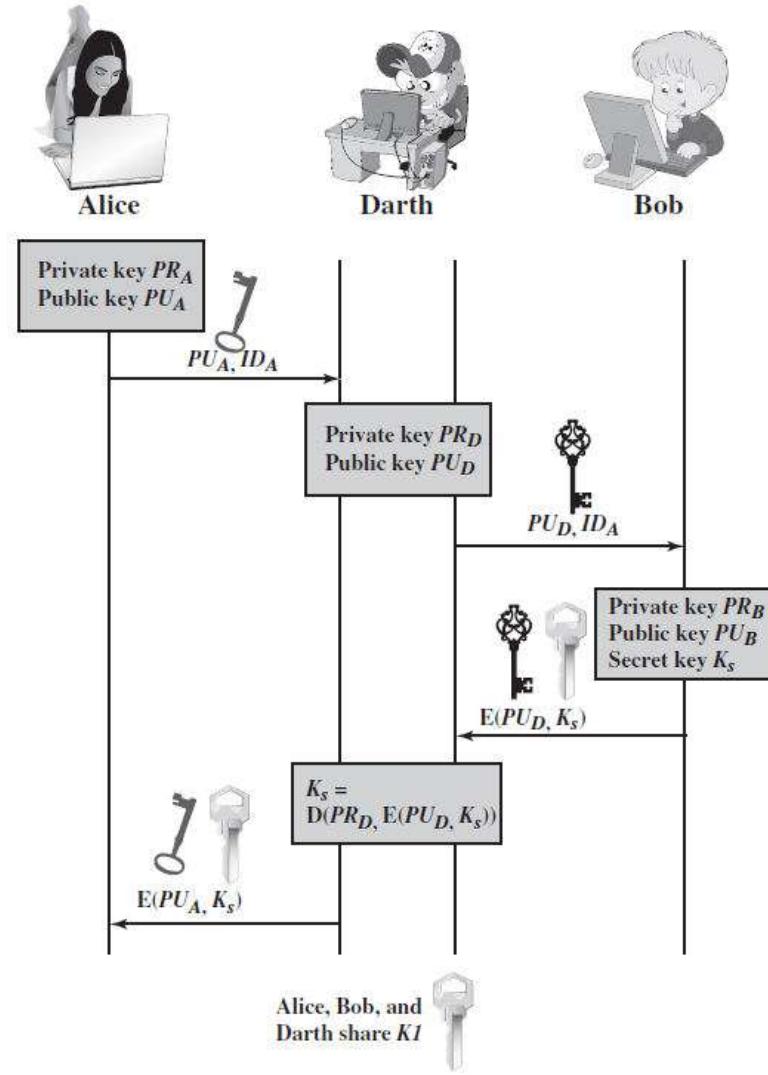


Figure 14.7 Simple Use of Public-Key Encryption to Establish a Session Key

- ▶ If A wishes to communicate with B, the following procedure is employed:
 1. A generates a public/private key pair $\{PU_a, PR_a\}$ and transmits a message to B consisting of PU_a and an identifier of A, ID_A .
 2. B generates a secret key, K_s , and transmits it to A, which is encrypted with A's public key.
 3. A computes $D(PR_a, E(PU_a, K_s))$ to recover the secret key. Because only A can decrypt the message, only A and B will know the identity of K_s .
 4. A discards PU_a and PR_a and B discards PU_a .

A and B can now securely communicate using conventional encryption and the session key K_s . At the completion of the exchange, both A and B discard K_s .

Another Man-in-the-Middle Attack



1. A generates a public/private key pair $\{PU_a, PR_a\}$ and transmits a message intended for B consisting of PU_a and an identifier of A, ID_A .
2. D intercepts the message, creates its own public/private key pair $\{PU_d, PR_d\}$ and transmits $PU_s || ID_A$ to B.
3. B generates a secret key, K_s , and transmits $E(PU_s, K_s)$.
4. D intercepts the message and learns K_s by computing $D(PR_d, E(PU_d, K_s))$.
5. D transmits $E(PU_a, K_s)$ to A.

The Diffie-Hellman Key Exchange

- ▶ The purpose of the algorithm is to enable two users to securely exchange a key that can then be used for subsequent symmetric encryption of messages.
- ▶ The Diffie-Hellman algorithm depends for its effectiveness on the difficulty of computing discrete logarithms.
- ▶ Discrete logarithm is defined in the following way:

A primitive root of a prime number p is one whose powers modulo p generate all the integers from 1 to $p - 1$. That is, if a is a primitive root of the prime number p , then the numbers

$$a \bmod p, a^2 \bmod p, \dots, a^{p-1} \bmod p$$

are distinct and consist of the integers from 1 through $p - 1$ in some permutation.

$a^i \bmod p$

P=7

Primitive roots of 7

0-6

$a^i \bmod p \quad 1 \ (i=1 \text{ to } 6) \ p=7$

1 1 1 1 1 1 1

2 2 4 1 2 4 1

3 3 2 6 4 5 1

4 4 4 2 1 4 2 1

5 5 4 6 2 3 1

6 6 1 6 1 6 1

For any integer b and a primitive root a of prime number p , we can find a unique exponent i such that

$$b \equiv a^i \pmod{p} \quad \text{where } 0 \leq i \leq (p - 1)$$

The exponent i is referred to as the **discrete logarithm** of b for the base a , mod p . We express this value as $\text{dlog}_{a,p}(b)$.

Algorithm:



Alice



Bob

Alice and Bob share a prime number q and an integer α , such that $\alpha < q$ and α is a primitive root of q

Alice generates a private key X_A such that $X_A < q$

Alice calculates a public key $Y_A = \alpha^{X_A} \text{ mod } q$

Alice receives Bob's public key Y_B in plaintext

Alice calculates shared secret key $K = (Y_B)^{X_A} \text{ mod } q$

Alice and Bob share a prime number q and an integer α , such that $\alpha < q$ and α is a primitive root of q

Bob generates a private key X_B such that $X_B < q$

Bob calculates a public key $Y_B = \alpha^{X_B} \text{ mod } q$

Bob receives Alice's public key Y_A in plaintext

Bob calculates shared secret key $K = (Y_A)^{X_B} \text{ mod } q$



- ▶ In Diffie-Hellman Key Exchange algorithm, there are two publicly known numbers: a prime number q and an integer α that is a primitive root of q . Suppose the users A and B wish to create a shared key.

User A selects a random integer $X_A < q$ and computes $Y_A = \alpha^{X_A} \text{ mod } q$. Similarly, user B independently selects a random integer $X_B < q$ and computes $Y_B = \alpha^{X_B} \text{ mod } q$. Each side keeps the X value private and makes the Y value available publicly to the other side. Thus, X_A is A's private key and Y_A is A's corresponding public key, and similarly for B. User A computes the key as $K = (Y_B)^{X_A} \text{ mod } q$ and user B computes the key as $K = (Y_A)^{X_B} \text{ mod } q$.

Example:

- ▶ Key exchange is based on the use of the prime number:

$q = 353$ and a primitive root of 353, in this case $\alpha = 3$. A and B select private keys $X_A = 97$ and $X_B = 233$, respectively. Each computes its public key:

A computes $Y_A = 3^{97} \bmod 353 = 40$.

B computes $Y_B = 3^{233} \bmod 353 = 248$.

After they exchange public keys, each can compute the common secret key:

A computes $K = (Y_B)^{X_A} \bmod 353 = 248^{97} \bmod 353 = 160$.

B computes $K = (Y_A)^{X_B} \bmod 353 = 40^{233} \bmod 353 = 160$.

Module-4

System Implementation

Design Principles

1. Principle of Least Privilege:

- ▶ This principle restricts how privileges are granted.
- ▶ The principle of least privilege states that a subject should be given only those privileges that it needs in order to complete its task.

2. Principle of Fail-Safe Defaults

- ▶ This principle restricts how privileges are initialized when a subject or object is created.
- ▶ The *principle of fail-safe defaults* states that, unless a subject is given explicit access to an object, it should be denied access to that object.

3. Principle of Economy of Mechanism

- ▶ This principle simplifies the design and implementation of security mechanisms.
- ▶ The *principle of economy of mechanism* states that security mechanisms should be as simple as possible.

4. Principle of Complete Mediation

- ▶ This principle restricts the caching of information, which often leads to simpler implementations of mechanisms.
- ▶ The *principle of complete mediation* requires that all accesses to objects be checked to ensure that they are allowed.

5. Principle of Open Design

- ▶ This principle suggests that complexity does not add security.
- ▶ The *principle of open design* states that the security of a mechanism should not depend on the secrecy of its design or implementation.

6. Principle of Separation of Privilege

- ▶ This principle is restrictive because it limits access to system entities.
- ▶ The principle of separation of privilege states that a system should not grant permission based on a single condition.

7. Principle of Least Common Mechanism

- ▶ This principle is restrictive because it limits sharing.
- ▶ The *principle of least common mechanism* states that mechanisms used to access resources should not be shared.

8. Principle of Psychological Acceptability

- ▶ This principle recognizes the human element in computer security.
- ▶ The *principle of psychological acceptability* states that security mechanisms should not make the resource more difficult to access than if the security mechanisms were not present.

Representing Identity

- ❖ **What Is Identity?**
- ▶ Identity is simply a computer's representation of an entity.
- ▶ A *principal* is a unique entity. An *identity* specifies a principal.
- ▶ A principal may be a person, an organization or an object.
- ▶ Identities are used for several purposes. The two main ones are for **accountability** and for **access control**.
- ▶ Accountability requires an identity that tracks principals across actions and changes of other identities, so that the principal taking any action can be unambiguously identified.
- ▶ Access control requires an identity that the access control mechanisms can use to determine if a specific access (or type of access) should be allowed.

Files and Objects:

- ▶ The identity of a file or other entity (here called an “object”) depends on the system that contains the object.
- ▶ Local systems identify objects by assigning names.
- ▶ The name may be intended for human use (such as a file name), for process use (such as a file descriptor or handle), or for kernel use (such as a file allocation table entry).
- ▶ Each name may have different semantics.

Users:

- ▶ A user is an identity tied to a single entity.
- ▶ Specific systems may add additional constraints.
- ▶ Systems represent user identity in a number of different ways.
- ▶ Indeed, the same system may use different representations of identity in different contexts.

Groups and Roles:

- ▶ The “entity” may be a set of entities referred to by a single identifier.
- ▶ The members of the set must be distinguishable, but the set may have an identity separate from any of its elements.
- ▶ Principals often need to share access to files.
- ▶ Most systems allow principals to be grouped into sets called, logically enough, *groups*.
- ▶ Groups are essentially a shorthand tool for assigning rights to a set of principals simultaneously.

- ▶ A ***role*** is a type of group that ties membership to function.
- ▶ When a principal assumes a role, the principal is given certain rights that belong to that role.
- ▶ When the principal leaves the role, those rights are removed.
- ▶ The rights given are consistent with the functionality that the principal needs to perform the tasks expected of members of the role.

Naming and Certificates

- ▶ Certificates are described as a mechanism for binding cryptographic keys to identifiers.
- ▶ The identifier corresponds to a principal.
- ▶ The identifier must uniquely identify the principal to avoid confusion.
- ▶ Suppose the principals are people. The identifiers cannot be names, because many different people may have the same name. (How many people named “John Smith” or “Pierre LeBlanc” are there?) The identifiers must include ancillary information to distinguish the “Matt Bishop” who teaches at UC Davis from the “Matt Bishop” who works at Microsoft Corporation.

- ▶ Certification authorities (CAs) vouch, at some level, for the identity of the principal to which the certificate is issued. Every CA has two policies controlling how it issues certificates.
- ▶ A ***CA authentication policy*** describes the level of authentication required to identify the principal to whom the certificate is to be issued.
- ▶ A ***CA issuance policy*** describes the principals to whom the CA will issue certificates.

Internet Policy Registration Authority (IPRA)

- ▶ It sets policies that all subordinate CAs must follow, and it certifies other CAs called ***policy certification authorities (PCAs)***.
- Each PCA has its own issuance and authentication policies, but those policies must not conflict with the policies set by the IPRA.
- ▶ The PCAs issue certificates to ordinary CAs, which can then issue certificates to organizations or individuals.
- ▶ The IPRA and PCAs do not issue certificates to individuals or organizations.
- ▶ All CAs, PCAs, and the IPRA have unique Distinguished Names.

State and Cookies

Many Internet applications require that the client or server maintain state to simplify the transaction process

- ▶ A **cookie** is a token that contains information about the state of a transaction on a network.
- ▶ The cookies consist of several values.
 1. The **name** and **value** are encoded into the cookie and represent the state. The interpretation is that the *name* has an associated *value*.
 2. The **expires** field indicates when the cookie is valid. Expired cookies are discarded; they are not to be given out. If this field is not present, the cookie will be deleted at the end of the session.

3. The **domain** states the domain for which the cookie is intended. It consists of the last n fields of the domain name of a server. The cookie will be sent to servers in that domain.
4. The **path** further restricts the dissemination of the cookie. When a Web server requests a cookie, it provides a domain (its own). Cookies that match that domain may be sent to the server. If the server specifies a path, the path must be the leading substring of the path specified in the cookie.
5. If the **secure** field is set, the cookie will be sent only over secured connections (that is, to “https” or “http” over SSL).

Access Control Mechanisms

❖ Access Control Lists

Definition:

Let S be the set of subjects, and R the set of rights, of a system. An *access control list* (ACL) l is a set of pairs $l = \{ (s, r) : s \in S, r \subseteq R \}$. Let acl be a function that determines the access control list l associated with a particular object o . The interpretation of the access control list $acl(o) = \{ (s_i, r_i) : 1 \leq i \leq n \}$ is that subject s_i may access o using any right in r_i .

Example: Consider the following access control matrix.

	file 1	file 2	process 1	process 2
process 1	read, write, own	read	read, write, execute, own	write
process 2	append	read, own	read	read, write, execute, own

Figure: An access control matrix. The system has two processes and two files. The set of rights is {read, write, execute, append, own}.

Example Contd..

- ▶ The set of subjects is process 1 and process 2, and the set of objects is file 1, file 2, process 1, and process 2. The corresponding access control lists are:

$acl(\text{file 1}) = \{ (\text{process 1}, \{ \text{read, write, own} \}), (\text{process 2}, \{ \text{append} \}) \}$

$acl(\text{file 2}) = \{ (\text{process 1}, \{ \text{read} \}), (\text{process 2}, \{ \text{read, own} \}) \}$

$acl(\text{process 1}) = \{ (\text{process 1}, \{ \text{read, write, execute, own} \}), (\text{process 2}, \{ \text{read} \}) \}$

$acl(\text{process 2}) = \{ (\text{process 1}, \{ \text{write} \}), (\text{process 2}, \{ \text{read, write, execute, own} \}) \}$

❖ Capabilities:

- ▶ A capability is like the row of an access control matrix.
- ▶ Each subject has associated with it a set of pairs, with each pair containing an object and a set of rights.
- ▶ The subject associated with this list can access the named object in any of the ways indicated by the named rights.

Let O be the set of objects, and R the set of rights, of a system. A *capability list* c is a set of pairs $c = \{ (o, r) : o \in O, r \subseteq R \}$. Let cap be a function that determines the capability list c associated with a particular subject s . The interpretation of the capability list $cap(s) = \{ (o_i, r_i) : 1 \leq i \leq n \}$ is that subject s may access o_i using any right in r_i .

We abbreviate “capability list” as C-List.

Example: Consider the following access control matrix.

	file 1	file 2	process 1	process 2
process 1	read, write, own	read	read, write, execute, own	write
process 2	append	read, own	read	read, write, execute, own

Figure: An access control matrix. The system has two processes and two files. The set of rights is {read, write, execute, append, own}.

Example Contd..

- ▶ The set of subjects is process 1 and process 2. The corresponding capability lists are:

$cap(\text{process 1}) = \{ (\text{file 1}, \{ \text{read, write, own} \}), (\text{file 2}, \{ \text{read} \}), (\text{process 1}, \{ \text{read, write, execute, own} \}), (\text{process 2}, \{ \text{write} \}) \}$

$cap(\text{process 2}) = \{ (\text{file 1}, \{ \text{append} \}), (\text{file 2}, \{ \text{read, own} \}), (\text{process 1}, \{ \text{read} \}), (\text{process 2}, \{ \text{read, write, execute, own} \}) \}$

Information Flow

- ▶ Information flow policies define the way information moves throughout a system.
- ▶ Typically, these policies are designed to preserve confidentiality of data or integrity of data.
- ▶ The policy's goal is to prevent information from flowing to a user not authorized to receive it.
- ▶ Any confidentiality and integrity policy embodies an information flow policy.

Definition: The command sequence c causes a flow of information from x to y if, after execution of c , some information about the value of x before c was executed can be deduced from the value of y after c was executed.

This definition views information flow in terms of the information that the value of y allows one to deduce about the value in x .

- ▶ For example, the statement

$y := x;$

reveals the value of x in the initial state, so information about the value of x in the initial state can be deduced from the value of y after the statement is executed.

- ▶ The statement

$y := x / z;$

reveals some information about x , but not as much as the first statement.

- ▶ The final result of the sequence c must reveal information about the initial value of x for information to flow.
- ▶ The sequence

$\text{tmp} := x;$

$y := \text{tmp};$

has information flowing from x to y because the (unknown) value of x at the beginning of the sequence is revealed when the value of y is determined at the end of the sequence. However, no information flow occurs from tmp to x , because the initial value of tmp cannot be determined at the end of the sequence.

Definition: An implicit flow of information occurs when information flows from x to y without an explicit assignment of the form $y := f(x)$, where:

$f(x)$ is an arithmetic expression with the variable x .

- ▶ The flow of information occurs, not because of an assignment of the value of x , but because of a flow of control based on the value of x .

Information Flow Models and Mechanisms

- ▶ An information flow policy is a security policy that describes the authorized paths along which that information can flow.
- ▶ Each model associates a label, representing a security class, with information and with entities containing that information.
- ▶ Each model has rules about the conditions under which information can move throughout the system.
- ▶ The notation $\underline{x} \leq \underline{y}$ means that information can flow from an element of class x to an element of class y .
- ▶ **Mechanisms:**
 1. Compiler Based Mechanisms
 2. Execution Based Mechanisms

I. Compiler Based Mechanisms

- ▶ **Compiler-based mechanisms** check that information flows throughout a program are authorized.
- ▶ The mechanisms determine if the information flows in a program *could* violate a given information flow policy.
- ▶ This determination is not precise, but it is secure, in that no unauthorized path along which information may flow will be undetected.
- ▶ A set of statements is ***certified*** with respect to an information flow policy if the information flow within that set of statements does not violate the policy.

- ▶ Information can be passed into or out of a procedure through parameters.
- ▶ Parameters are classified as:
 - ***input parameters*** (through which data is passed into the procedure),
 - ***output parameters*** (through which data is passed out of the procedure),and
 - ***input/output parameters*** (through which data is passed into and out of the procedure).

Program Statements:

A program consists of several types of statements. Typically, they are:

1. Assignment statements
2. Compound statements
3. Conditional statements
4. Iterative statements
5. Goto statements
6. Procedure calls
7. Function calls
8. Input/output statements.

II. Execution Based Mechanisms

The goal of an execution-based mechanism is to prevent an information flow that violates policy. Checking the flow requirements of explicit flows achieves this result for statements involving explicit flows. Before the assignment

$$y = f(x_1, \dots, x_n)$$

is executed, the execution-based mechanism verifies that

$$lub(\underline{x}_1, \dots, \underline{x}_n) \leq \underline{y}$$

If the condition is true, the assignment proceeds. If not, it fails. A naïve approach, then, is to check information flow conditions whenever an explicit flow occurs.

Implicit flows complicate checking.

❖ Fenton's Data Mark Machine

- ▶ Fenton created an abstract machine called the Data Mark Machine to study handling of implicit flows at execution time.
- ▶ Each variable in this machine had an associated security class, or tag.
- ▶ Fenton also included a tag for the program counter (PC).

The inclusion of the PC allowed Fenton to treat implicit flows as explicit flows, because branches are merely assignments to the PC. He defined the semantics of the Data Mark Machine. In the following discussion, *skip* means that the instruction is not executed, $push(x, \underline{x})$ means to push the variable x and its security class \underline{x} onto the program stack, and $pop(x, \underline{x})$ means to pop the top value and security class off the program stack and assign them to x and \underline{x} , respectively.

Fenton defined five instructions. The relationships between execution of the instructions and the classes of the variables are as follows:

1. The increment instruction

$x := x + 1$

is equivalent to

if PC \leq x then $x := x + 1$; else skip

2. The conditional instruction

```
if x = 0 then goto n else x := x - 1
```

is equivalent to

```
if x = 0 then { push(PC, PC); PC = lub(PC, x); PC := n; }
else           { if PC ≤ x then { x := x - 1; } else skip }
```

This branches, and pushes the PC and its security class onto the program stack. (As is customary, the PC is incremented so that when it is popped, the instruction following the *if* statement is executed.) This captures the PC containing information from *x* (specifically, that *x* is 0) while following the **goto**.

3. The return

`return`

is equivalent to

`pop(PC, PC);`

This returns control to the statement following the last *if* statement.
Because the flow of control would have arrived at this statement, the PC no longer contains information about *x*, and the old class can be restored.

4. The branch instruction

if' $x = 0$ then goto n else $x := x - 1$

is equivalent to

```
if  $x = 0$  then { if  $x \leq \underline{PC}$  then {  $PC := n;$  } else skip }
else           { if  $\underline{PC} \leq x$  then {  $x := x - 1;$  } else skip }
```

This branches without saving the PC on the stack. If the branch occurs, the PC is in a higher security class than the conditional variable x , so adding information from x to the PC does not change the PC's security class.

5. The halt instruction

halt

is equivalent to

if program stack empty then halt execution

The program stack being empty ensures that the user cannot obtain information by looking at the program stack after the program has halted (for example, to determine which *if* statement was last taken).

EXAMPLE: Consider the following program, in which x initially contains 0 or 1.

1. if $x = 0$ then goto 4 else $x := x - 1$
2. if $z = 0$ then goto 6 else $z := z - 1$
3. halt
4. $z := z + 1$
5. return
6. $y := y + 1$
7. return

This program copies the value of x to y . Suppose that $x = 1$ initially. The following table shows the contents of memory, the security class of the PC at each step, and the corresponding certification check.

x	y	z	PC	<u>PC</u>	stack	<i>certification check</i>
1	0	0	1	<i>Low</i>	—	
0	0	0	2	<i>Low</i>	—	$Low \leq \underline{x}$
0	0	0	6	\underline{x}	(3, <i>Low</i>)	
0	1	0	7	\underline{x}	(3, <i>Low</i>)	<u>PC</u> $\leq \underline{y}$
0	1	0	3	<i>Low</i>	—	

Fenton's machine handles errors by ignoring them. Suppose that, in the program above, $\underline{y} \leq \underline{x}$. Then at the fifth step, the certification check fails (because $\underline{PC} = \underline{x}$). So, the assignment is skipped, and at the end $y = 0$ regardless of the value of x . But if the machine reports errors, the error message informing the user of the failure of the certification check means that the program has attempted to execute step 6. It could do so only if it had taken the branch in step 2, meaning that $z = 0$. If $z = 0$, then the *else* branch of statement 1 could not have been taken, meaning that $x = 0$ initially.

- ▶ To prevent this type of deduction, Fenton's machine continues executing in the face of errors, but ignores the statement that would cause the violation.
- ▶ This satisfies the requirements.
- ▶ Aborting the program, or creating an exception visible to the user, would also cause information to flow against policy.
- ▶ The problem with reporting of errors is that a user with lower clearance than the information causing the error can deduce the information from knowing that there has been an error.
- ▶ If the error is logged in such a way that the entries in the log, and the action of logging, are visible only to those who have adequate clearance, then no violation of policy occurs.

- ▶ But if the clearance of the user is sufficiently high, then the user can see the error without a violation of policy.
- ▶ Thus, the error can be logged for the system administrator (or other appropriate user), even if it cannot be displayed to the user who is running the program.
- ▶ Similar comments apply to any exception action, such as abnormal termination.

Compiler Based Mechanism Vs Execution Based Mechanism

- ▶ A **compiler-based mechanism** assesses the flow of information in a program with respect to a given information flow policy.
 - The mechanism either certifies that the program meets the policy or shows that it fails to meet the policy.
 - It has been shown that if a set of statements meet the information flow policy, their combination (using higher-level language programming constructs) meets the information flow policy.
- ▶ **Execution-based mechanisms** check flows at runtime.
 - Unlike compiler-based mechanisms, execution-based mechanisms either allow the flow to occur (if the flow satisfies the information flow policy) or block it (if the flow violates the policy).

Confinement Problem

- ▶ The **confinement problem** is the problem of preventing a server from leaking information that the user of the service considers confidential.
- ▶ Consider a client and a server. When the client issues a request to the server, the client sends the server some data.
- ▶ The server then uses the data to perform some function and returns a result (or no result) to the client. Access control affects the function of the server in two ways:
 1. The server must ensure that the resources it accesses on behalf of the client include only those resources that the client is authorized to access.
 2. The server must ensure that it does not reveal the client's data to any other entity not authorized to see the client's data.

- ▶ The **first requirement** represents the goal of the **service provider**.
 - That goal is to prevent the client from sending messages to the server that cause it to access, alter, transmit, or consume resources that the client is not authorized to access, alter, transmit, or consume.

- ▶ The **second requirement** represents the goal of the **service user**.
 - That goal is to prevent the server from transmitting confidential information to the service provider.

In both cases, the server must be confined to accessing only a specific set of resources.

EXAMPLE: A server balances accounts for subscribers. The subscribers use a client to transmit the register entries, the current bank balance, and those withdrawals and deposits that have cleared the bank to the server. The server returns the list of outstanding checks and deposits and any discrepancy between the register balance and the bank balance. Subscribers pay a fee for each use.

The service provider requires that the server correctly record who used the service each time it is used. Otherwise, the service provider cannot bill for the use of the service. The threat is that someone may use the service without being detected (and therefore without being charged) or that the user may impersonate another subscriber (resulting in the wrong subscriber being charged). The service provider also does not want the server to transmit billing records or any other unauthorized information to the client. The server should send only the information it derived from the data that the client sent. So the server must be confined to operating only on the data it is sent.

The subscriber expects certain security services from the server. The server must correctly log the user's invocation so that the user is not charged incorrectly. (This matches the need of the service provider.) The server must not record or transmit the data that the subscriber sends to it because the subscriber's data is confidential to the subscriber and is not relevant to the service provider. So the server must be confined to keeping the data to itself and to sending the results only to the subscriber.

❖ **Total Isolation:**

- ▶ One characteristic of processes that do not leak information comes from the observation that a process must store data for later retrieval (the leaking).
- ▶ A process that does not store information cannot leak it.
- ▶ However, in the extreme, such processes also cannot perform any computations, because an analyst could observe the flow of control (or state of the process) and from that flow deduce information about the inputs.
- ▶ This leads to the observation that a process that cannot be observed and cannot communicate with other processes cannot leak information.
- ▶ Lampson calls this **total isolation**.

Covert Channel:

- ▶ A *covert channel* is a path of communication that was not designed to be used for communication.

EXAMPLE:

- Process p is to be confined such that it cannot communicate with process q .
- However, processes p and q share a file system.
- In order for process p to send a message to process q , it creates a file called *send* in a directory that both processes can read.
- Just before process q is to read the information, q deletes the *send* file.
- Process p then transmits a bit by creating a file named *0bit* or *1bit*, as appropriate.
- When q detects either file, it records the bit and deletes the file.
- This continues until p creates a file called *end*, at which point the communication ceases.

- ▶ Confinement is **transitive**.
- ▶ Assume that a process p is confined to prevent leakage.
- ▶ If it invokes a second process q , then q must be similarly confined or q could leak the information that p passes.

The *rule of transitive confinement* states that if a confined process invokes a second process, the second process must be as confined as the caller.

Secure Software Development

Building secure and trusted systems depends on standard software engineering techniques augmented with specific technologies and methodologies.

- ❖ **Life Cycle:**
 - ▶ A life cycle starts when a system is considered for development and use.
 - ▶ The life cycle ends when the system is no longer used.
 - ▶ A life cycle includes a set of processes that define how to perform activities, as well as methods for managing activities.
 - ▶ A typical life cycle process is defined in stages.
 - ▶ Some stages depend on previous stages, whereas others do not.
 - ▶ Each stage describes activities of all the involved disciplines and controls inter discipline interactions.
 - ▶ As work progresses, the project ideally transitions from one stage to the next.

❖ **Conception:**

- ▶ The conception stage starts with an **idea**.
- ▶ Ideas come from anywhere—for example, from customers, engineers, other disciplines, user groups, or others.
- ▶ The organization decision makers may decide to:
 - fund the idea and make it a project,
 - reject the idea, or
 - ask for further information or for a demonstration that the idea has merit.
- ▶ A *proof of concept* is a demonstration that an idea has merit.

Conception Contd...

- ▶ The output of the conception stage must provide sufficient information for all disciplines to begin their tasks in the next stage
- ▶ This information may be an overview of the project; high-level requirements that the project should meet; or schedule, budget, staffing, or planning information.
- ▶ The planning information could be a detailed project plan or more general high-level plans for each of the disciplines involved in the project.
- ▶ The exact nature of the information depends on the size and complexity of the project.

Conception Contd...

- ▶ Security feasibility and high-level requirement analysis should begin during this stage of the life cycle.
- ▶ Before time and resources are invested in development or in proof of concept activities, the following questions should be considered.
 1. What does “secure” mean for this concept?
 2. Is it possible for this concept to meet this meaning of security?
 3. Is the organization willing to support the additional resources required to make this concept meet this meaning of security?
- ▶ Identification of threats comprises another important set of security issues.
- ▶ It is especially important to determine the threats that are visible at the conception stage.
- ▶ Development of assurance considerations is important at this stage.

❖ Manufacture:

- ▶ Once a project has been accepted, funded, approved, and staffed, the manufacturing stage begins.
- ▶ Manufacturing begins with the development of more detailed plans for each of the involved disciplines, which could include:
 - marketing plans,
 - sales training plans,
 - development plans, and
 - test plans.
- ▶ These documents describe the specific tasks for this stage of the life cycle within each discipline.

❖ Manufacture Contd..

- ▶ The software development or engineering process lies in this stage.
- ▶ It includes procedures, tools, and techniques used to develop and maintain the system.
- ▶ Technical work may include design techniques, development standards and guidelines, and testing tools and methods.
- ▶ Management aspects may include planning, scheduling, review processes, documentation guidelines, metrics, and configuration management such as source code control mechanisms and documentation version controls.
- ▶ The output of this stage from each discipline should be the materials necessary to determine whether to proceed.

❖ Deployment

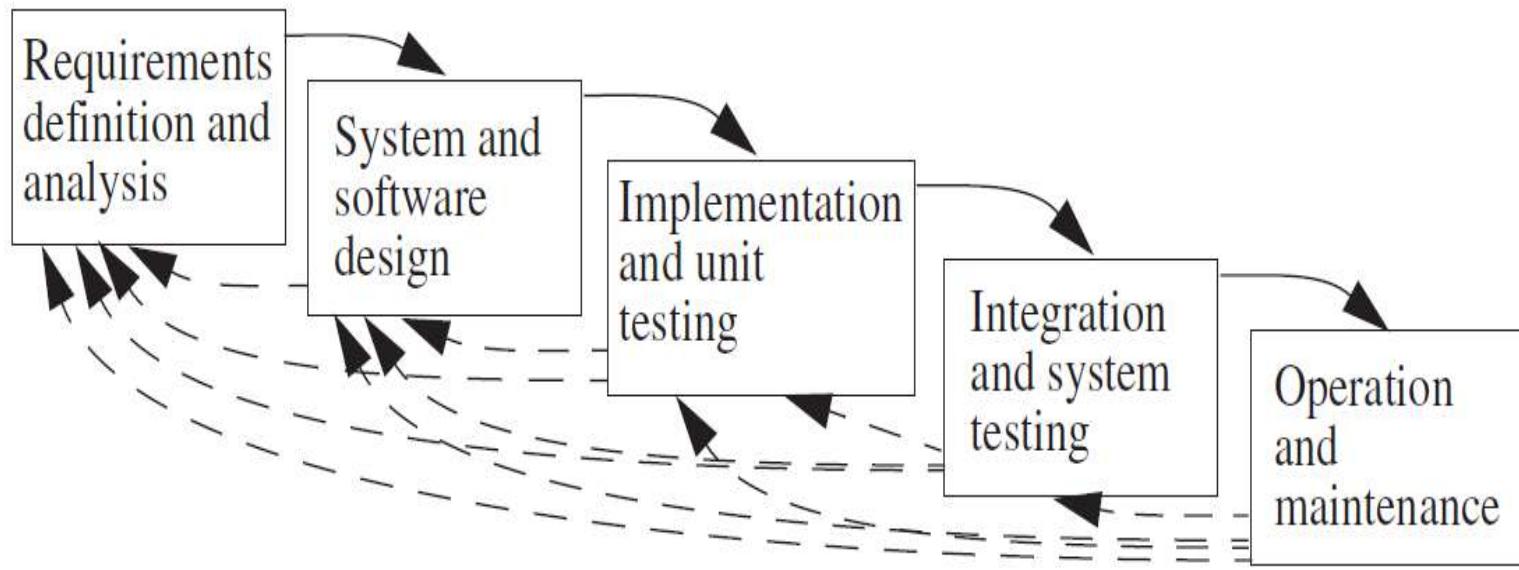
- ▶ Once the system has passed the acceptance criteria in the manufacturing stage, it is ready for deployment.
- ▶ This stage is the process of getting the system out to the customer.
- ▶ It is divided into two substages.
 1. The first substage is the domain of production, distribution, and shipping. The role of the other disciplines (such as engineering and marketing) is to deliver masters to the production staff.
 2. The second substage of deployment is proper installation and configuration of the system in its production setting.

❖ Fielded Product Life

- ▶ The primary tasks of fielded product life are patching or fixing of bugs, maintenance, and customer service.
- ▶ Separate from the product development organization.
 - engineering process must track maintenance and patches.
 - a deployment process must distribute patches and new releases.

The Waterfall Life Cycle Model

- ▶ The *waterfall life cycle model* is the model of building in stages, whereby one stage is completed before the next stage begins.
- ▶ This model is not the only technique for building secure and trusted systems, but it is perhaps the most common.
- ▶ It consists of five stages. The solid arrows show the flow from each stage to the next.



I. Requirements Definition and Analysis:

- ▶ In this phase, the high-level requirements are expanded.
- ▶ Requirements may be functional requirements or nonfunctional requirements.
- Functional requirements describe interactions between the system and its environment.
- Nonfunctional requirements are constraints or restrictions on the system that limit design or implementation choices.
- ▶ Requirements describe what and not how.
- ▶ They should be implementation-independent.
- ▶ Two sets of requirements are defined.
 1. A requirements definition of what the customer can expect the system to do is generally presented in natural language.
 2. A technical description of system characteristics, sometimes called a *requirements specification*, may be presented in a more precise form.
- ▶ The analysis of the requirements may include a feasibility study and may examine whether or not the requirements are correct, consistent, complete, realistic, verifiable, and traceable.

II. System and Software Design

- ▶ Software design partitions the requirements into specific executable programs.
- ▶ At this stage, external functional specifications and internal design specifications are written.
 - The **external functional specifications** describe the inputs, outputs, and constraints on functions that are external to the entity being specified,
 - The **internal design specifications** describe algorithms to be used, data structures, and required internal routines.
- ▶ This stage is sometimes broken into the two phases *system design*, in which the system as a whole is designed, and *program design*, in which the programs of the system are individually designed.

III. Implementation and Unit Testing

- ▶ *Implementation* is the development of software programs based on the software design from the previous step.
- ▶ The work is divided into a set of programs or program units.
- ▶ *Unit testing* is the process of establishing that the unit as implemented meets its specifications.

IV. Integration and System Testing

- ▶ *Integration* is the process of combining all the unit-tested program units into a complete system.
- ▶ Automated tools and guidelines governing the integration process may be in place.
- ▶ *System testing* is the process of ensuring that the system as a whole meets the requirements.
- ▶ System testing is an iterative step because invariably bugs and errors are found that have to be corrected.
- ▶ Typically, the errors are sent back to the development team to be corrected. This requires iteration with the previous step.
- ▶ The corrected code is reintegrated into the system, and system testing is repeated.

V. Operation and Maintenance:

- ▶ Once the system is finished, it is moved into production. This is called *fielding the system*.
- ▶ Maintenance involves correction of errors that have been reported from the field and that have not been corrected at earlier stages.
- ▶ This stage also involves routine maintenance and the release of new versions of the system.
- ▶ Finally, retirement of the system also falls under this phase.

Secured Coding

- ▶ **Secure coding** is the practice of developing computer software in a way that guards against the accidental introduction of security vulnerabilities.
- ▶ Defects, bugs and logic flaws are consistently the primary cause of commonly exploited software vulnerabilities.

OWASP/SANS Top Vulnerabilities

Buffer Overflow

- ▶ A buffer is a sequential section of memory allocated to contain anything from a character string to an array of integers.
- ▶ A buffer overflow, or buffer overrun, occurs when more data is put into a fixed-length buffer than the buffer can handle.
- ▶ The extra information, which has to go somewhere, can overflow into adjacent memory space, corrupting or overwriting the data held in that space.
- ▶ This overflow usually results in a system crash, but it also creates the opportunity for an attacker to run arbitrary code or manipulate the coding errors to prompt malicious actions.

- ▶ Programming languages commonly associated with buffer overflows include C and C++, which provide no built-in protection against accessing or overwriting data in any part of memory and do not automatically check that data written to an array (the built-in buffer type) is within the boundaries of that array.
- ▶ Bounds checking can prevent buffer overflows, but requires additional code and processing time.

Example:

- ▶ A buffer overflow occurs when data written to a buffer also corrupts data values in memory addresses adjacent to the destination buffer due to insufficient bounds checking.
- ▶ This can occur when copying data from one buffer to another without first checking that the data fits within the destination buffer.
- ▶ In the following example expressed in C, a program has two variables which are adjacent in memory: an 8-byte-long string buffer, A, and a two-byte big-endian integer, B.

```
char      A[8]    = "";
unsigned short B   = 1979;
```

Example Contd..

Initially, A contains nothing but zero bytes, and B contains the number 1979.

variable name	A								B
value	[null string]								1979
hex value	00	00	00	00	00	00	00	00	07 BB

Now, the program attempts to store the [null-terminated string](#) "excessive" with [ASCII](#) encoding in the A buffer.

```
strcpy(A, "excessive");
```

"excessive" is 9 characters long and encodes to 10 bytes including the null terminator, but A can take only 8 bytes. By failing to check the length of the string, it also

variable name	A								B
value	'e'	'x'	'c'	'e'	's'	's'	'i'	'v'	25856
hex	65	78	63	65	73	73	69	76	65 00

B's value has now been inadvertently replaced by a number formed from part of the character string. In this example "e" followed by a zero byte would become 25856.

Writing data past the end of allocated memory can sometimes be detected by the operating system to generate a [segmentation fault](#) error that terminates the process.

Example Contd..

- ▶ To prevent the buffer overflow from happening in this example, the call to strcpy could be replaced with strlcpy, which takes the maximum capacity of A (including a null-termination character) as an additional parameter and ensures that no more than this amount of data is written to A:

```
strlcpy(A, "excessive", sizeof(A));
```

- ▶ When available, the strlcpy library function is preferred over strncpy which does not null-terminate the destination buffer if the source string's length is greater than or equal to the size of the buffer (the third argument passed to the function), therefore A may not be null-terminated and cannot be treated as a valid C-style string.

Incomplete Mediation

- ▶ Incomplete mediation occurs when a computer program leaves sensitive data in an exposed, uncontrolled condition.
- ▶ The vulnerability occurs primarily in the form of web URLs that expose data in such a way that user-made alterations to the URL allow the user to manipulate the program or website.

Example: Input data to a web form is often transferred to the server by embedding it in a URL.

- ▶ Suppose the input is validated on the client before constructing the required URL say

www.kt280.com/orders/final&custID=111&num=55A&qty=5&price=60&shipping=4&total=300

- ▶ This URL is interpreted to mean that the customer with ID number 111 has ordered 5 books of item number 55, at a cost of rs60 each, with a rs5 shipping charge, giving a total cost of rs300.
- ▶ Since the input is checked on the client, the developer of the server software believes it would be wasted effort to check it again on the server.

Incomplete Mediation Contd...

- ▶ However, instead of using the client software, Trudy can directly send a URL to the server. Suppose Trudy sends the following URL to the server:

**www.kt280.com/orders/final&custID=111#55A&qty=5
&price=60&shipping=4&total=30**

- ▶ If the server doesn't bother to validate the input, Trudy can obtain the same order as above, but for the bargain basement price of rs30 instead of the legitimate price of rs300.
- ▶ There have been numerous buffer overflows in the Linux kernel, and most of these were due to incomplete mediation.
- ▶ There are tools available to help find likely cases of incomplete mediation, but they are not a cure-all since this problem can be subtle, and therefore difficult to detect automatically.

XSS (Cross Site Scripting)

- ▶ Cross Site Scripting attack is a malicious code injection, which will be executed in the victim's browser.
- ▶ Malicious script can be saved on the web server and executed every time when the user calls the appropriate functionality.
- ▶ It can also be performed with the other methods – without any saved script in the web server.
- ▶ The main purpose of this attack is to steal the other user's identity data – cookies, session tokens and other information.
- ▶ In most of the cases, this attack is being used to steal the other person's cookies.
- ▶ Cookies help us to log in automatically. Therefore with stolen cookies, we can login with the other identities. And this is one of the reasons, why this attack is considered as one of the riskiest attacks.

- ▶ XSS attack is being performed on the client side.
- ▶ It can be performed with different client-side programming languages.
- ▶ However, most often this attack is performed with Javascript and HTML.
- ▶ The main reason for this attack is inappropriate user's input validation, where malicious input can get into the output.
- ▶ A malicious user can enter a script, which will be injected into the website's code.
- ▶ Then the browser is not able to know if the executed code is malicious or not.
- ▶ Therefore malicious script is being executed on the victim's browser or any faked form is being displayed for the users.

Main forms of Cross Site Scripting are as follows:

- ▶ Cross Site Scripting can occur on the malicious script executed at the client side.
- ▶ Fake page or form displayed to the user (where the victim types credentials or clicks a malicious link).
- ▶ On the websites with displayed advertisements.
- ▶ Malicious emails sent to the victim.

This attack occurs when the malicious user finds the vulnerable parts of the website and sends it as appropriate malicious input. Malicious script is being injected into the code and then sent as the output to the final user.

Anti-Cross Site Scripting Libraries

- ▶ AntiXSS helps you to protect your current applications from cross-site scripting attacks, at the same time helping you to protect your legacy application with its Security Runtime Engine.
- ▶ AntiXSS incorporates radically and innovatively rethought features, offering a newer, more powerful weapon against the often employed cross-site scripting (XSS) attack.

AntiXSS gives:

1. Improved Performance. AntiXSS has been completely rewritten with performance in mind, and yet retains the fundamental protection from XSS attacks that you have come to rely on for your applications.
2. Secure Globalization. The web is a global market place, and cross-site scripting is a global issue. An attack can be coded anywhere, and Anti-XSS now protects against XSS attacks coded in dozens of languages.
3. Standards Compliance. AntiXSS is written to comply with modern web standards. You can protect your web application without adversely affecting its UI.

- ▶ The Microsoft Anti-Cross Site Scripting Library V4.3 (AntiXSS V4.3) is an encoding library designed to help developers protect their ASP.NET web-based applications from XSS attacks.
- ▶ It differs from most encoding libraries in that it uses the white-listing technique -- sometimes referred to as the principle of inclusions -- to provide protection against XSS attacks.
- ▶ This approach works by first defining a valid or allowable set of characters, and encodes anything outside this set (invalid characters or potential attacks).
- ▶ The white-listing approach provides several advantages over other encoding schemes.

Canonical Data Format

- ▶ A **canonical model** is a design pattern used to communicate between different data formats.
- ▶ A canonical data model (CDM) is a type of data model that presents data entities and relationships in the simplest possible form.
- ▶ It is generally used in system/database integration processes where data is exchanged between different systems, regardless of the technology used.
- ▶ A canonical data model is also known as a common data model.
- ▶ A canonical data model primarily enables an organization to create and distribute a common definition of its entire data unit.

- ▶ The design of a CDM requires identifying all entities, their attributes and the relationships between them.
- ▶ The importance of a CDM is particularly evident in integration processes where data units are shared between different information system platforms.
- ▶ It utilizes a generalized data format to present/define data that makes it simple to share data among multiple applications.

Command Injection

- ▶ Command injection is an attack in which the goal is execution of arbitrary commands on the host operating system via a vulnerable application.
- ▶ Command injection attacks are possible when an application passes unsafe user supplied data (forms, cookies, HTTP headers etc.) to a system shell.
- ▶ In this attack, the attacker-supplied operating system commands are usually executed with the privileges of the vulnerable application.
- ▶ Command injection attacks are possible largely due to insufficient input validation.

- ▶ This attack differs from Code Injection, in that code injection allows the attacker to add his own code that is then executed by the application.
- ▶ In Command Injection, the attacker extends the default functionality of the application, which execute system commands, without the necessity of injecting code.

Example:

The following PHP code snippet is vulnerable to a command injection attack:

```
<?php  
print("Please specify the name of the file to delete");  
print("<p>");  
$file=$_GET['filename'];  
system("rm $file");  
?>
```

The following request and response is an example of a successful attack:

Request

```
http://127.0.0.1/delete.php?filename=bob.txt;id
```

Response

```
Please specify the name of the file to delete  
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Sanitizing Input

```
Replace or Ban arguments with ";"  
Other shell escapes available  
Example:  
- &&  
- |  
- ...
```

Redirection

- ▶ On a Web site, redirection is a technique for moving visitors to a different Web page than the one they request, usually because the page requested is unavailable.
- ▶ Web users often encounter redirection when they visit the Web site of a company whose name has been changed or which has been acquired by another company.

Open Redirection Vulnerability:

- ▶ An Open Redirection is when a web application or server uses a user-submitted link to redirect the user to a given website or page.
- ▶ Even though it seems like a harmless action, to let a user decide on which page he wants to be redirected to, if exploited such a technique can have a serious impact, especially when combined with other vulnerabilities and tricks.

- ❖ **Exploiting an Open Redirect Vulnerability for a Phishing Attack:**
- ▶ When the user clicks on a link of a legitimate website he often won't be suspicious if suddenly a login prompt shows up.
- ▶ To launch a successful phishing attack the attacker sends the victim a link, for example via email, which exploits the vulnerability on the vulnerable website *example.com*:

<https://example.com/redirect.php?go=http://attacker.com/phish/>

- ▶ By exploiting the open redirect vulnerability on the legitimate website, the attacker is redirecting the victim to, **<http://attacker.com/phish>** which is a phishing page that is similar to the legit website.
- ▶ Once the visitor is on the attacker's malicious website, he enters his credentials on the login form which points to a script that is controlled by the attacker.
- ▶ The script is typically used to save the username and the password that is being typed in by the victim, which attackers typically use at a later stage to impersonate the victim on the legitimate website.

Inference

- ▶ An Inference Attack is a data mining technique performed by analyzing data in order to illegitimately gain knowledge about a subject or database.
- ▶ A subject's sensitive information can be considered as leaked if an adversary can infer its real value with a high confidence.
- ▶ This is an example of breached information security.
- ▶ An Inference attack occurs when a user is able to infer from trivial information more robust information about a database without directly accessing it.
- ▶ The object of Inference attacks is to piece together information at one security level to determine a fact that should be protected at a higher security level.

Application Controls

- ▶ Application control is a security practice that blocks or restricts unauthorized applications from executing in ways that put data at risk.
- ▶ The control functions vary based on the business purpose of the specific application, but the main objective is to help ensure the privacy and security of data used by and transmitted between applications.
- ▶ Application control includes completeness and validity checks, identification, authentication, authorization, input controls, and forensic controls, among others.

- ▶ **Completeness checks** – controls ensure records processing from initiation to completion
- ▶ **Validity checks** – controls ensure only valid data is input or processed
- ▶ **Identification** – controls ensure unique, irrefutable identification of all users
- ▶ **Authentication** – controls provide an application system authentication mechanism
- ▶ **Authorization** – controls ensure access to the application system by approved business users only
- ▶ **Input controls** – controls ensure data integrity feeds into the application system from upstream sources
- ▶ **Forensic controls** – controls ensure scientifically and mathematically correct data, based on inputs and outputs

Key features & Benefits of Application Control

- ▶ Identify and control which applications are in your IT environment and which to add to the IT environment
- ▶ Automatically identify trusted software that has authorization to run
- ▶ Prevent all other, unauthorized applications from executing – they may be malicious, untrusted, or simply unwanted
- ▶ Eliminate unknown and unwanted applications in your network to reduce IT complexity and application risk
- ▶ Reduce the risks and costs associated with malware
- ▶ Improve your overall network stability
- ▶ Identify all applications running within the endpoint environment
- ▶ Protect against exploits of unpatched OS and third-party application vulnerabilities